Microprocessor Interface Design

Microprocessor Interface Design

Digital circuits and concepts

J.D. Nicoud

Director of Laboratoire de Microinformatique (LAMI) and Professor at École Polytechnique Fédérale de Lausanne, Switzerland

CHAPMAN & HALL

London · New York · Tokyo · Melbourne · Madras

Published by Chapman & Hall, 2–6 Boundary Row, London SE1 8HN

Chapman & Hall, 2–6 Boundary Row, London SE1 8HN, UK

Van Nostrand Reinhold Inc., 115 5th Avenue, New York NY10003, USA

Chapman & Hall Japan, Thomson Publishing Japan, Hirakawacho Nemoto Building, 7F, 1–7–11 Hirakawa-cho, Chiyoda-ku, Tokyo 102, Japan

Chapman & Hall Australia, Thomas Nelson Australia, 102 Dodds Street, South Melbourne, Victoria 3205, Australia

Chapman & Hall India, R. Seshadri, 32 Second Main Road, CIT East, Madras 600 035, India

English language edition 1991

© 1991 Chapman & Hall and Masson

Original French language edition – *Circuits Numériques pour Interfaces Microprocesseur* – © 1991 Masson.

ISBN 0 412 45140 9 0 442 31513 9 (USA)

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the UK Copyright Designs and Patents Act, 1988, this publication may not be reproduced, stored, or transmitted, in any form or by any means, without the prior permission in writing of the publishers, or in the case of reprographic reproduction only in accordance with the terms of the licences issued by the Copyright Licensing Agency in the UK, or in accordance with the terms of licences issued by the appropriate Reproduction Rights Organization outside the UK. Enquiries concerning reproduction outside the terms stated here should be sent to the publishers at the London address printed on this page.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

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

Library of Congress Cataloging-in-Publication data available

Published with the support of the DIST
Ministère de la Recherche et de la Technologie (France)
« Programme d'aide à l'édition d'ouvrages scientifiques et techniques »

Contents

1 Preface *xiii*

1 Integrated circuit technologies 1

1.1 Electronic elements 1
1.1.1 Passive components 1
1.1.2 Bipolar transistors 3
1.1.3 MOS transistors 4
1.1.4 Diode logic 5
1.1.5 Static and dynamic characteristics 5
1.1.6 Parameter dispersion 6

1.2 Major logic families 6
1.2.1 The "74" family 6
1.2.2 Evolution 8
1.2.3 TTL circuits 8
1.2.4 Open collector and emitter 10
1.2.5 Three-state outputs 11
1.2.6 CMOS circuits 12
1.2.7 Open and three-state CMOS outputs 13
1.2.8 CMOS-TTL compatibility 13

1.3 Other technologies 14
1.3.1 NMOS circuits 14
1.3.2 Dynamic logic 14
1.3.3 ECL circuits 15
1.3.4 GaAs circuits 15
1.3.5 Fluid and optical logic 15
1.3.6 Special interfaces 16

1.4 Implementation 17
1.4.1 Correct working conditions 17
1.4.2 Thyristor effect 17
1.4.3 Electrostatic problems 18

1.5 Physical devices 19
1.5.1 Chips 19
1.5.2 Packages 19
1.5.3 Printed circuits 22

2 Combinational logic 23

2.1 Signals and logic functions 23
2.1.1 Logic signals 23
2.1.2 Positive and negative logic 24
2.1.3 Decomposition of a logic system 25
2.1.4 Logic gates 25
2.1.5 Equivalences 26
2.1.6 Importance of drawing rules 27
2.1.7 Rules for inverting circles 28
2.1.8 Rise and propagation times 30

2.2 Implementing a combinational system 31
2.2.1 Logic expression 31
2.2.2 Implementation using transistors 31
2.2.3 Standard VLSI library 33
2.2.4 Gate arrays 33
2.2.5 The "74" family 34
2.2.6 Programmable logic 35
2.2.7 PROMs 36
2.3 Transformation of logic functions 37
2.3.1 Karnaugh maps 37
2.3.2 Example 38
2.4 Standard combinational circuits 39
2.4.1 Available functions 41
2.5 Simple gates 41
2.5.1 Gate catalogue 42
2.5.2 Exercise 43
2.5.3 Special gates 43
2.6 Special input gates 44
2.6.1 Low charge inputs 44
2.6.2 Hysteresis threshold inputs 44
2.6.3 Available Schmitt trigger gates 44
2.7 Special output gates 45
2.7.1 Buffered outputs 45
2.7.2 Single state outputs 45
2.7.3 Applications of open collector gates 46
2.7.4 Open collector drivers 46
2.7.5 Three-state outputs 46
2.7.6 Bus termination 47
2.7.7 Three-state buffers and drivers 48
2.7.8 Line drivers 49
2.7.9 Analog switches 50
2.8 Switches and encoders 50
2.8.1 Multiplexors 50
2.8.2 Decoders 51
2.8.3 Display decoders 52
2.8.4 Priority encoders 53
2.8.5 Parity generators 54
2.8.6 Error detection and correction 54
2.9 Comparators and arithmetic circuits 55
2.9.1 Parallel comparators 55
2.9.2 Serial comparison 56
2.9.3 Adders 56
2.9.4 Arithmetic and logic units 56
2.9.5 Anticipated report 57
2.9.6 Multipliers 57
2.9.7 Barrel shifters 57
2.9.8 Other circuits 58

3 Sequential logic systems 59
3.1 Asynchronous sequential systems 59
3.1.1 Oscillators 59
3.1.2 Bistable flip-flops 59
3.1.3 Elementary arbiters 61
3.1.4 Metastable states 61
3.1.5 Latches 62
3.1.6 Addressable latches 63
3.1.7 Designing asynchronous systems 64
3.1.8 Static glitches 64
3.1.9 Example 65
3.2 Synchronous flip-flops 66
3.2.1 The clock pulse 66
3.2.2 Note 66
3.2.3 Multi-phase clock 67
3.2.4 D flip-flops 68
3.2.5 Output conventions 68
3.2.6 Working conditions 69
3.2.7 Metastability probability 69
3.2.8 Double synchronization 71
3.2.9 Standard D flip-flops 71
3.2.10 JK flip-flops 71
3.2.11 Standard JK flip-flops 73
3.2.12 Dynamic SR flip-flops 73
3.2.13 T flip-flops 73
3.2.14 Standard T flip-flops 74
3.2.15 Universality of D and JK flip-flops 74
3.2.16 One's catchers 75
3.2.17 Double flip-flops 75
3.2.18 Pinpointing an operation 76
3.2.19 Cycle samplers 77
3.2.20 Pulse sampler 77
3.3 Synthesis of sequential systems 78
3.3.1 Synthesis of a semi-synchronous system 78
3.3.2 Synthesis of a synchronous system 78
3.3.3 Principles of the method 79
3.3.4 Conclusion 80
3.4 Complex sequential circuits 80
3.4.1 Standard sequential functions 80
3.4.2 Inhibiting the clock 81
3.4.3 Asynchronous and synchronous reset 83
3.4.4 Asynchronous and synchronous loading 83
3.5 Registers 84
3.5.1 Parallel registers and latches 84
3.5.2 Read-back registers 85
3.5.3 Serial-parallel registers 85
3.5.4 Universal registers 86
3.5.5 Case analysis 86

3.6 Counters 88
3.6.1 Asynchronous counters 88
3.6.2 Divide by 2^n counter 88
3.6.3 Synchronous counters 89
3.6.4 Programmable dividers 90
3.6.5 Counters with registers 91
3.6.6 Pseudo-random counters 91
3.6.7 Rate multipliers 92
3.7 Delays and pulses 93
3.7.1 Delay lines 94
3.7.2 RC networks delays 94
3.7.3 Quantified delays 94
3.7.4 Analog oscillators 95
3.7.5 Crystal oscillators 96
3.7.6 Monostables 97
3.7.7 Power-up reset 98
3.8 Standardized logic symbols 99
3.8.1 Main standards 99
3.8.2 IEC-617/AIEE-91 standard 101
3.8.3 Drawing conventions 102
4 Memory circuits 105
4.1 Introduction 105
4.2 Read only memories 106
4.2.1 Principles 106
4.2.2 Types of ROMs 107
4.2.4 Standard PROMs 108
4.2.3 Standard EPROMs 109
4.2.5 E^2PROMs 110
4.3 Read Write memories 111
4.3.1 Static RAMs 111
4.3.2 Serial memories 113
4.3.3 Dual ported memories 113
4.3.4 Dynamic RAMs 113
4.3.5 Video RAMs 115
4.3.6 FIFO memories 116
4.3.7 LIFO memories 117
4.3.8 Associative memories 117
5 Programmable logic 117
5.1 Regular logic 117
5.1.1 A simple example 117
5.1.2 Solution based on a decoder 118
5.1.3 Completely programmable solution 119
5.1.4 Partially programmable solution 119
5.1.5 PROM logic 120
5.1.6 PLA logic 121

5.2 PLD logic 121
5.2.1 Simple PLDs 122
5.2.2 Simplifying logic equations for PLDs 123
5.2.3 Correcting existing PLDs 124
5.2.4 Application example: 68008 decoder 125
5.2.5 Asynchronous sequential logic 126
5.2.6 Example: boot flip-flop for 68008 126
5.2.7 Three-state output PLDs 127
5.2.8 Application example: resetting the 68000 128
5.3 Registered PLDs 130
5.3.1 Basic structure 130
5.3.2 Binary counter. 131
5.3.3 Gray and special counters 132
5.3.4 Independent clocks 133
5.3.5 Tricks 134
5.3.6 Buried registers 135
5.3.7 Macrocells, PLDs and GALs 135
5.3.8 Existing PLDs 136
5.3.9 Future PLDs 137

6 Input-output interfaces 139
6.1 Logic inputs 139
6.1.1 Filtering and galvanic insulation 139
6.1.2 Digital filtering 140
6.1.3 Reading an analog sensor 141
6.1.4 Suppressing contact bounce 142
6.1.5 Grouped inputs 143
6.1.6 Angular encoder 144
6.1.7 Quadrature encoder 145
6.1.8 Analog angular encoder 146
6.2 Analog inputs 146
6.2.1 Operational amplifiers 146
6.2.2 Ideal model 148
6.2.3 Basic schematics 148
6.2.4 Comparators 149
6.2.5 Instrumentation amplifiers 149
6.2.6 A/D converters 150
6.2.7 Serial interface 151
6.2.8 Analog sensors 151
6.2.9 Voltage-frequency conversion 152
6.3 Output interfaces 152
6.3.1 Microcontroller outputs 152
6.3.2 D/A converters 153
6.3.3 Display controllers 153
6.3.4 Power amplifiers 154
6.3.5 Relays 155
6.3.6 Driving motors 155

6.3.7 Proportional control 157
6.3.8 Collectorless motor 158
6.3.9 Step by step motor 158
6.3.10 Regulated power supplies 159

7 Testing circuits 161
7.1 Types of tests 161
7.1.1 Debugging 161
7.1.2 Manufacturing 161
7.1.3 Maintenance 162
7.2 Test aids 162
7.2.1 Logic indicators 162
7.2.2 Oscilloscopes 162
7.2.3 Logic analyzers 162
7.2.4 Function generator 163
7.2.5 Signature analyzers 163
7.2.6 Industrial tester 163
7.3 Test theory 164
7.3.1 Break-downs and reliability 164
7.3.2 Testability 165
7.3.3 Test coverage 165
7.3.4 Test methods 165
7.4 Design for testability 166
7.4.1 Simulation 166
7.4.2 Observability 167
7.4.3 Controllability 167
7.4.4 Initial state 168
7.4.5 Test points 168
7.4.6 Multiplexing 168
7.4.7 Scan paths 168
7.4.8 JTAG standard 169
7.4.9 Signature analysis 172
7.5 Debugging prototypes 172
7.5.1 Testing integrated circuits 172
7.5.2 Testing printed circuit boards 173
7.5.3 Special techniques 174
7.5.4 Memory tests 175
7.5.5 Interferences and disturbances 176
7.5.6 Repairing 177
7.5.7 Conclusion 177

8 Design examples 179
8.1 Simple sequencer 179
8.1.1 Pulse generators 179
8.1.2 Solution using decoders 180
8.1.3 Getting rid of glitches 180
8.1.4 Optimized solutions 182
8.1.5 General solution 183

8.2 Frequency divider 184
8.2.1 Rate multipliers 184
8.2.2 Other solutions 185
8.2.3 VLSI implementation 186
8.3 Sequencer for repetitive resets 188
8.3.1 Problem specifications 188
8.3.2 Possible solutions 189
8.3.3 Simplified solutions 189
8.3.4 Complete solution 191
8.3.5 Optimizing 192
8.4 Double buffers 196
8.4.1 Data and options 196
8.4.2 Problem specifications 197
8.4.3 First synchronous solution 198
8.4.4 Second asynchronous solution 200
8.4.5 Purely asynchronous solution 201
8.4.6 Synchronous synthesis 202
8.4.7 Detailed synthesis 203
8.4.8 Quasi synchronous solution 205
8.5 Multiplexing in a microprocessor interface 206
8.5.1 Registers and latches 206
8.5.2 Multiplexers and demultiplexers 207
8.5.3 Demultiplexing outputs 208
8.5.4 Demultiplexing using a shift register 208
8.5.5 Demultiplexing using a decoder or a PAL 209
8.5.6 Demultiplexing using an addressable latch 210
8.5.7 Demultiplexing using a parallel register 210
8.5.8 Multiplexing the inputs 212
8.5.9 Multiplexing using switches 213
8.5.10 Multiplexing using registers 213
8.5.11 Multiplexing using a three-state driver 214
8.5.12 Multiplexing using a matrix 214
8.5.13 Multiplexing input-outputs 215
8.5.14 Scanned keyboards 215
8.5.15 Conclusion 216

References 217

Index 219

Appendix: Integrated circuit catalogue 225

Preface

In order to come up with a reasonably optimized result when designing a hardware interface for a microprocessor or a specialized application, a designer must be quite knowledgeable and constantly keep abreast of new developments. Not only must he master traditional methods used for logic systems but he must also be familiar with a wide range of solutions and a technology which never stops progressing.

The CMOS and TTL integrated circuit families include the basic bricks used to build an interface, although prefabricated modules of various sizes may improve the performance/price ratio. Programmable networks (PLD, PAL®, GAL®, etc.) are becoming more and more diversified and gate arrays can justifiably be used for smaller and smaller series. When signals work on low frequencies, they can be read or activated by an appropriately programmed microcontroller, which may work out cheaper than wired logic. Therefore, engineers must never stop improving their knowledge and questioning the developing techniques which they learnt to master and which have proved reliable.

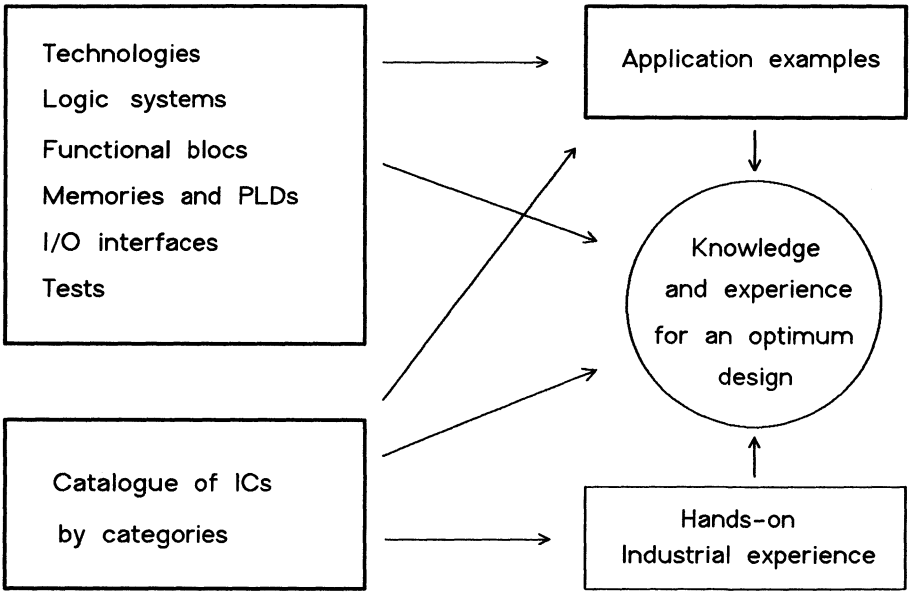
This book does not only deal with digital circuits, although the basic idea is to interconnect functional blocks only through logic connections. The word interface here really refers to an interface between a microprocessor, its support circuits and the outside world. It encompasses various aspects i.e. technological and packaging problems, how to choose from a large catalogue, the various methods used for representing and synthesizing as well as astute solutions. The main aspects of these problems are discussed here.

This book is aimed at showing young students as well as experienced engineers who wish to update their knowledge the main aspects of integrated system design. It is very dense and a large part of the information is to be found in the numerous figures which should be studied carefully. The aim is to impart enough knowledge so that the designer can then increase it by collaborating with component representatives and CAD salespeople, while keeping an objective attitude when faced with the large number of possible solutions. This book will also be of use to experienced designers, since this is an area where it is extremely difficult to be familiar with all the technological possibilities and variations of a solution.

The first three chapters include basic knowledge about the technology as well as about combinatorial and sequential logic circuits. When describing the catalogue of available circuits, I have tried to separate functional principles, which do not depend on any technology, from the characteristics specific to each implementation.

The next three chapters discuss memories, programmable logic devices (PLDs) and input-output interfaces. These fields offer a large number of basic solutions and their associated circuits; therefore, it is important to become familiar with them in order to choose suitable architectures and components. Testing is dealt with in chapter 7. Interface designers should benefit from a global outlook and be better prepared to use appropriate tools.

The last chapter describes some specific solutions and tries to share some of the experience which comes from years of practice. It is also aimed at developing the reader's imagination, a must when seeking original and efficient solutions to interface problems which all differ in time and space.



Book and appendix organization

The appendix lists most of the circuits used by digital interface designers nowadays. The advantage of this guide, compared with the manufacturers' specifications, is that it uses clear functional symbols and stresses which circuits should not be used either because they are obsolete or because they never spread through industry. Drawing rules for logic symbols in this appendix have been chosen so as to afford maximum mnemonic and visual help. The rules laid down when the first graphic stations appeared favour rectangles and are too verbose to really act as a reminder of the function. Progress in interactive systems and printing devices has now resulted in a better interface with the human brain; it would be a pity not to make the most of it.

Studying integrated circuits is not a read only activity. Serveral projects have to be done in order to gain real knowledge. The course given by the author at the Swiss Institute of Technology in Lausanne (Ecole Polytechnique Fédérale de Lausanne) uses this and includes a large proportion of labs [Nicoud91]. Thanks to Logidules, students can get the experience of several small projects in a few hours.

Several people helped me improve this book and make the text, the figures and the appendixes clearer. I would especially like to mention R. Beuchat and H. Kirrmann, whose comments were very precious. Many figures from the appendix have been drawn by D. Lavanchy and checked by H. de Malm. Special thanks must also go to Judith Gruet who did the translation, my assistants, to the EPFL and to the companies which enabled me to acquire hardware interface experience while developing systems of varying complexity. I am also thankful to my students for having helped me understand some learning mechanisms of a discipline which is extremely rich and sometimes calls for knowledge students are not normally supposed to have acquired. This book is dedicated to them in the hope that it will form the basis of many successful industrial projects.

1 Integrated circuit technologies

This chapter summarizes the knowledge needed by an engineer working on microprocessor interfaces and presents it in a practical way. For more information, the reader should refer to the few books that cover the same topic in more detail [Buchanan90] [Wakerly90].

1.1 Electronic elements

An interface designer models the electronic elements according to his application; he uses a simple version of the necessary components. An elementary knowledge of electricity is sufficient when trying to understand a logic chip; only few high frequency effects require more sophisticated analysis tools such as Kirchhoff and Maxwell models.

1.1.1 Passive components

Resistors (figure 1.1a) are mostly used to set a voltage level close to the power supply voltage (state "1") or to the reference voltage (state "0"). They are respectively referred to as a *pull-up* or a *pull-down resistor*. A resistor connected in series can be used either to limit a static current or to absorb high frequency reflections by adapting the impedance [DelCorso86]. Thanks to Ohm's law $U=R \cdot I$, where U is the voltage (in V), R the resistance (in KΩ) and I the current (in mA), steady currents and voltages can be determined.

When connected in parallel, *capacitors* (figure 1.1b), are integrating elements which prevent a sudden voltage change. A capacitor connected in series is a differentiating element which enables the generation of short impulses.

In the case of a line having an explicit resistor in series or with one resulting from the internal resistance of the active components, a capacitor introduces a delay equal to $t=k \cdot R \cdot C$, where R is the resistance (in kΩ), C the capacity (in nF) and t the time (in μs). The constant k is usually worth 0.5 to 0.8. The fact that the capacitor is linked to the voltage supply or to the ground hardly changes anything since the power supply impendance is low. Capacitors provide simple solutions for generating delays and pulses

(section 3.7) but their use is discouraged for they change too much according to the temperature and take up too much space on an integrated circuit.

Capacitors are used as decoupling elements on the supply lines of integrated circuits (ceramic capacitor of 0.1 μF for 2-3 circuits) in order to absorb high frequency current. A larger capacitor (100 μF to 4000 μF) is used at the power supply input on a printed circuit board. Capacitors connected in parallel on the lines are omnipresent (parasitic capacitors) but they should be minimized, except in certain cases where they act as a memory element (sections 4.3.4 and 6.2.6). The capacity of an integrated circuit input or of a few centimetres of track varies between 5 and 10pF.

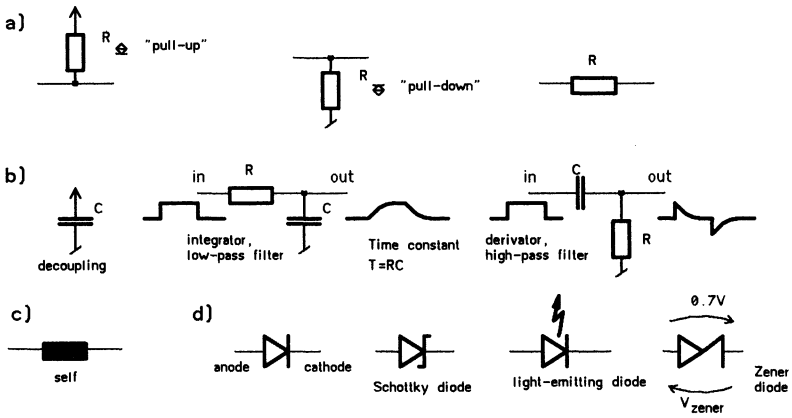


Fig. 1.1 Passive components: resistors, capacitors, inductances and diodes

Inductances (figure 1.1c) are mostly used for decoupling. The parasitic inductance on each link wire sometimes plays a part at high frequency.

Diodes (figure 1.1d) only let the current flow in one direction with a voltage drop of 0.7V for normal silicon diodes, 0.3V for germanium diodes and some 2V for light emitting diodes (leds). *Schottky diodes* have the best speed characteristics and their voltage drop is a low 0.2 to 0.3V. *Zener diodes* have a reverse breakdown voltage which is defined during manufacturing; they are used as voltage reference elements and for limiting voltage surges.

Miniature resistors and capacitors are marked using a *colour code* which is generally that of figure 1.2. Precision resistors carry extra rings.

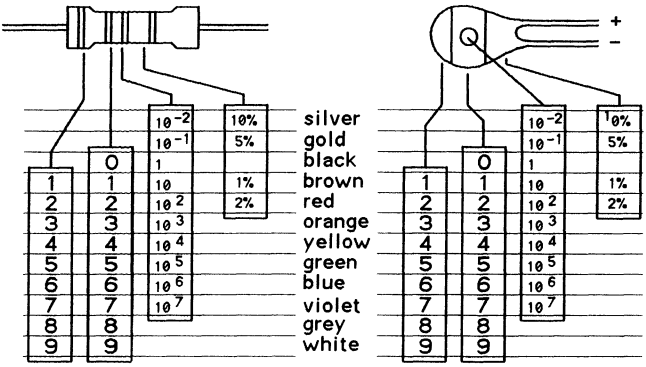


Fig. 1.2 Colour code

1.1.2 Bipolar transistors

A *bipolar transistor* is a current amplifier. In the unsaturated field, a base-emitter current i_b authorizes the presence of a collector-emitter current $i_c = \beta \cdot i_b$, where β is the current gain of the transistor (20 to 500). The transistors are either of the *NPN* or the *PNP* type (figure 1.3a); they may have multiple emitters. In general, a resistor connected in series with the base enables the collector current to be controlled by a voltage on this resistor; a load resistor generates an output voltage and it is possible to draw the transfer function of the resulting *inverter* (figure 1.3b). When dealing with logic, the aim is to cross the linear fields rapidly in order to work in non conductive or saturated fields. *Schottky diodes* limit saturation and accelerate switch-over.

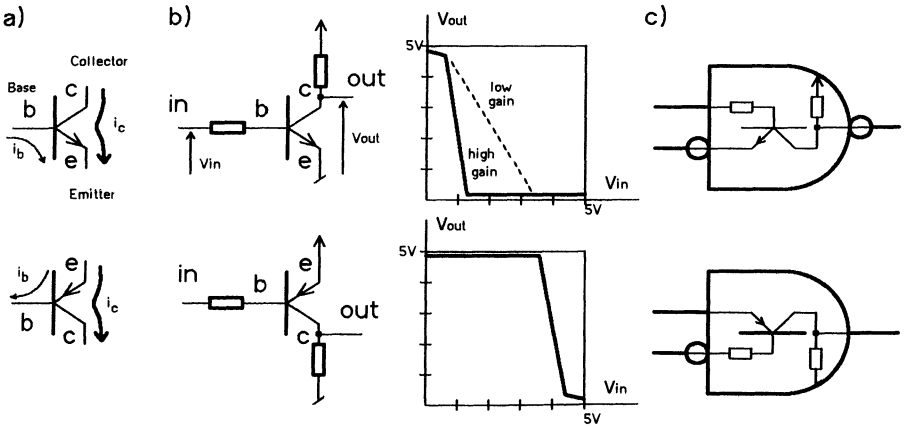


Fig. 1.3 Bipolar transistors (N (PNP) and P (NPN) types) and inverters

In reality, one transistor and two resistors perform the logical NAND function with one inverted input (figure 1.3c). However, it is obviously impossible to cascade a number of these gates without degrading the signals proportionally to the number of levels, which is not what occurs with an inverter (figure 1.3b).

1.1.3 MOS transistors

A *MOS transistor* becomes conductive if the voltage at the gate is sufficiently higher than that of the substrate. A current can then flow in both directions between the drain and the source. The MOS transistor is symmetrical and very similar to a mechanical switch controlled by a power supply. It is available in two polarities, N and P (figure 1.4a). A *MOS inverter* can be built using a MOS transistor and a charge resistor, like a bipolar inverter (figure 1.17). The resistor can be a specially doped transistor (*MOS technology*).

An easier and more efficient solution is to associate two complementary transistors as in figure 1.4b (*CMOS technology*). When the levels are correct (input voltage lower than 2V or higher than 3V), only one of the transistors is conductive and the power consumption is almost zero. The transfer function is a symmetric one. During switch-over, there is a brief short-circuit around 2.5V, since the N and P transistors are conductive simultaneously. In addition, the charge and discharge current of the parasitic capacitors leads to a power consumption which is proportional to the frequency.

A *transfer gate* (figure 1.4c), which is functionally equivalent to a digital switch, can be built using two complementary transistors in order to improve the linearity of the resistor throughout the voltage range. It is impossible to connect several transfer gates in series because the signal becomes more degraded at each level.

Any logical function can be realized by placing the network of N transistors corresponding to the direct function near the ground and the network of P transistors corresponding to the dual function near the +5V [Piguet90] [Annaratone86] [Weste85]. For example, a NAND gate with 2 inputs uses 4 transistors (figure 1.4d).

The delay introduced by an active element (bipolar transistor or MOS) largely depends on the parasitic capacitors. The geometrical scaling down due to progress in manufacturing is the main factor which increases performances.

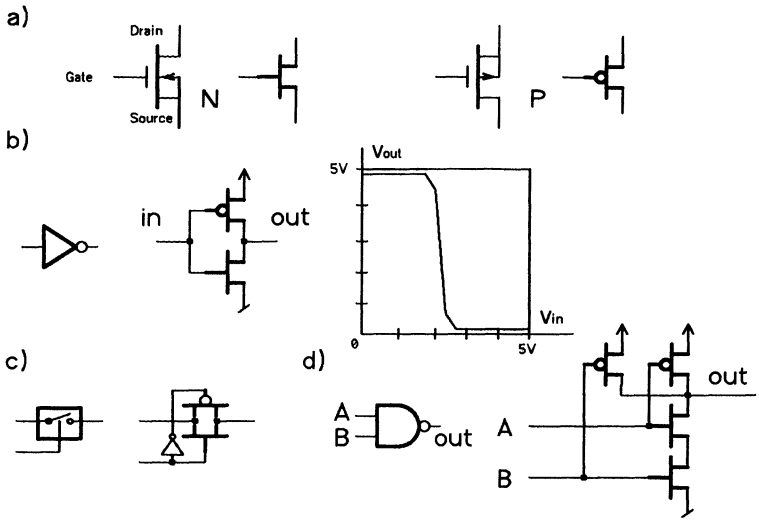


Fig. 1.4 N and P type MOS transistors, inverter and NAND gate

1.1.4 Diode logic

Two or more diodes connected in parallel with a common anode and a pull-up resistor perform the AND function, but the output signal is degraded (figure 1.5a). The OR function requires common cathodes and a pull-down resistor (figure 1.5b). These possibilities are frequently used inside VLSI circuits, such as PLDs.

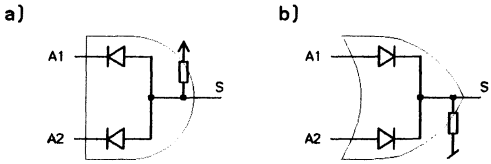


Fig. 1.5 Diode logic

1.1.5 Static and dynamic characteristics

A logic system functions correctly as long as a few static and dynamic characteristics are respected. The transfer function expresses the *static characteristics* (figure 1.4b). Integrated circuits are designed in such a manner that a certain number of circuit inputs can be connected to an output (section 1.2.3). *Dynamic characteristics* are defined by a set of parameters, the most important of which are the *rise* and *fall times* t_r and t_f (figure 1.6a) as well as the *propagation time* t_{pHL} ("high" to "low" time) and t_{pLH} ("low" to "high" time). The impulse response of an inverter can be seen in figure 1.6b.

By ensuring levels corresponding to 10% and 90% of the nominal signal voltage or fixing a threshold (1.5V), it is possible to avoid taking into account overshooting and rounding off caused by real signal reflection.

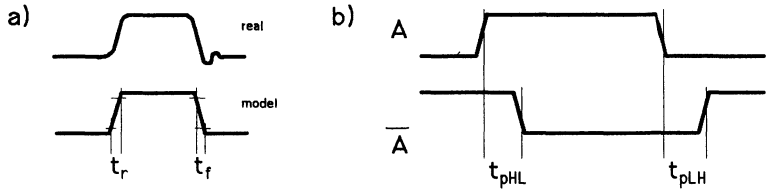


Fig. 1.6 Impulse response of a inverter

1.1.6 Parameter dispersion

Propagation times and voltage levels vary significantly according to the load, the temperature, the voltage and the manufactured series. The load must be specified in order to guarantee maximum times. Therefore, the manufacturer documents an equivalent schematic consisting of resistors, capacitors and diodes. The voltage is specified to 5V ± 5% and the temperature must stay between 0 and 75°C for commercial circuits. With such a typical load, the manufacturer guarantees the highest values (*worst case*). Typical values are sometimes given. Minimal values are never given since the manufacturer wishes to leave himself the possibility of improving technology in the future.

1.2 Major logic families

The first integrated circuits capable of carrying out simple logical functions (gate, counter, decoder) came out in 1964 in an *RTL* (resistor-transistor-logic) technology derived from that of figure 1.3b, later followed by *DTL* (diode-transistor-logic). From 1969 onwards, *TTL* (transistor-transistor-logic) technology took over because of its greater speed. Over the years, the range has become very wide. The "74" family now comprises more than 800 circuits, documented in the manufacturer's catalogues and in the appendix.

1.2.1 The "74" family

Technological evolution has resulted in many bipolar and CMOS generations, as shown in the table in figure 1.7. The changing parameters are speed (the propagation time of a inverter is used as a reference), the current consumption of an input or *fan-in* (multiples of 1.6mA, section 1.3.2) and the output current or *fan-out* (multiples of 1.6mA, section 1.3.2). With CMOS circuits, which have no static load, a high fan-out is required to charge and discharge the parasitic capacitors.

Type	Speed ns	Fan-in	Fan-out	Life	Family size
74xxx	15	1	10	65-78	150
74Hxxx	7	1,5	15	68-74	30
74Lxxx	40	0,1	2	68-74	30
74Sxxx	5	1,5	15	71-83	150
74LSxxx	12	0,2	5	73-	300
74Fxxx	4	0,4	12	80-	100
74ALSxxx	5	0,2	4	81-	200
74ASxxx	2	1,2	15	82-	50
74Cxxx	100	0	1	76-84	50
74HCxxx	20	0	2	83-	200
74HCTxxx	20	0	2	84-	100
74ACxxx	10	0	15	87-	200
74ACTxxx	10	0	15	87-	200
74AC11xxx	10	0	15	88-	20
74ACT11xxx	10	0	15	88-	20
74BCTxxx	7	0	40?	89	10
74ABTxxx	5	0	20?	91	5

Fig. 1.7 Families inside the "74" family

Many families are already "dead" : there is no point in using them for new developments even if the circuits are still available. The *"74" family* has three strong branches [Wakerly90] which will stay alive for many years:

- the widely used *LS (Low-power Schottky)* and *ALS (Advanced Low-power Schottky)* one
- the *F (fast)* and *AS (Advanced Schottky)* one which is suitable for applications requiring greater speed
- the *HC (High speed CMOS)* and *AC (Advanced CMOS)* one for low power consumption at low frequencies and, in the case of AC, a very interesting speed.

It is possible to mix TTL families as long as the load constraints are respected. Mixing TTL/HC-MOS sometimes implies the use of pull-up resistors towards the +5V (section 1.2.8)

Bipolar circuits have voltages of 5V ± 5% (4.75 to 5.25V). Degradation is progressive outside this area. Correct working order generally ceases at 3.5V and it is unwise to go higher than 6V (overheating, breakdowns). HCMOS circuits work from 2 to 6V. Certain manufacturers offer (quite incomplete) families guaranteed between 3 and 15V.

1.2.2 Evolution

Thanks to technological progress, electron mobility has been increased and geometrical scale has been reduced. The ratio between speed (propagation time) and dissipated power therefore improves for every new generation. Moreover, speed can be increased by reducing the value of the internal resistors, i.e by acting on the power; the LS, S, ALS and AS families are in this case (figure 1.8).

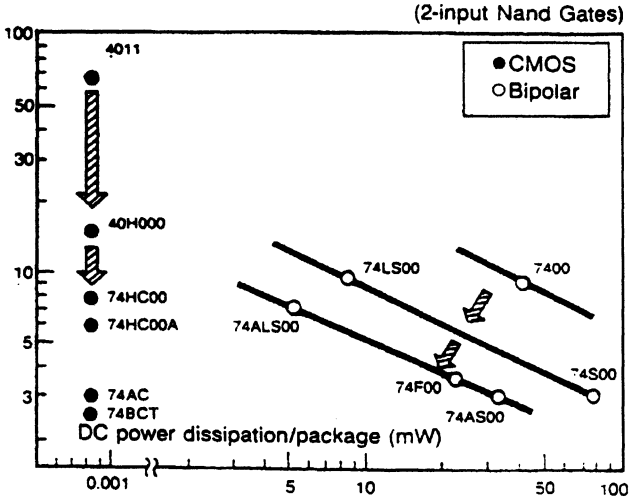


Fig. 1.8 Ratio between dissipated power and propagation times (document Texas Instruments)

This evolution is not over since the new AC (Fairchild FACT) and ACL (Texas Instruments) families are in competition, the latter circuit being equipped with a different pin-out (the 74AC11000 is equipped with power supplies located in its centre and not in its corners). BICMOS circuits (74BCTxx) combine both technologies: complementary MOS is used for the internal logic and bipolar technology, better suited for controlling a bus, is used for the output amplifier.

1.2.3 TTL circuits

The interface designer cannot go into all the electronic subtleties which characterize the 74, 74LS, etc. families. He can only examine inputs and outputs and reason in terms of currents and voltages guaranteeing the logical states "0" and "1". In order to maintain a TTL-LS input in the logical state "0", it is necessary to impose a voltage lower than 0.4V and to absorb a current of approximately 0.4mA. For state "1", a very low positive current is all that is needed in order to ensure a voltage greater than 2.7V (figure 1.10). The output amplifier is designed to absorb more current in state "0" than what it can supply in state "1". Currents relating to normal TTL are called *fan-in* and *fan-out* and are given in the table of figure 1.7.

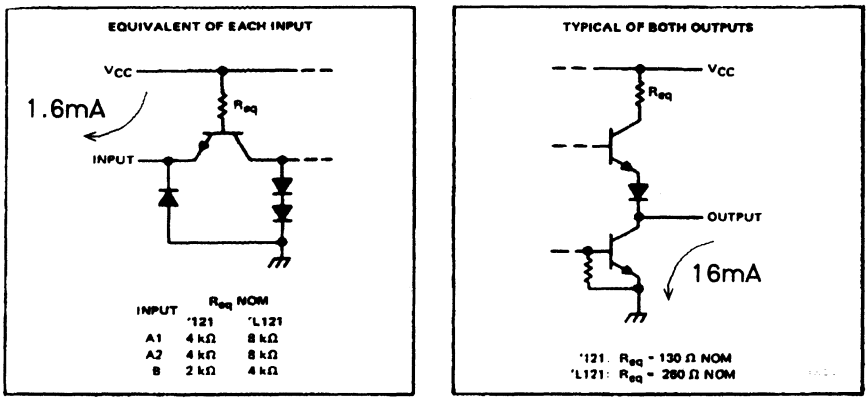


Fig. 1.9 Diagram showing inputs and outputs of a TTL gate (document Texas Instruments)

TTL essentially has dissymmetrical operating points, with as an input a transition from state "0" to state "1" for a voltage close to 1 V. At the output stage, the voltage at state "1" is greater than 2.8 V because of the transistor and the diode (figure 1.9). Due to technological evolution, these voltages have been slightly increased and the current necessary to impose state "0" reduced. Two TTL outputs cannot be linked to one another because a destructive current can be generated if one of the gates tries to force a "1" and the other a "0".

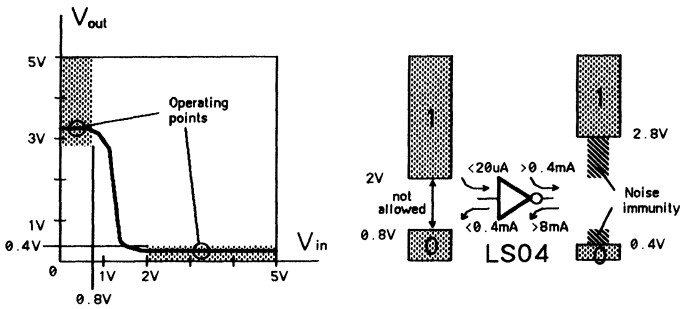


Fig. 1.10 Characteristics of TTL families

When the input of a TTL gate reaches a voltage of approximately 1.2 V, both transistors are slightly conductive and a relatively high current flows between the plus and the minus of the voltage supply during the short time necessary for switch-over. In order to avoid this current propagating along the power lines of the printed circuit, *decoupling capacitors* are strategically placed on the printed circuit, as a rule one per one to four TTL circuits (section 1.5.3).

1.2.4 Open collector and emitter

If in the output schematic of a TTL gate (figure 1.9) the upper transistor is missing, the gate is known as an *open collector* (figure 1.11a). The AND function is performed when two open collector outputs are connected together (figure 1.11b) using a pull-up resistor to set state "1". This is referred to as *wired AND*. Open collector gates (*OC*) are common in TTL; these outputs are signalled by a diamond marked with a dash on the side of the active state (figure 1.11b). Open collectors are often used to perform the OR function in accordance with deMorgan's theorem (section 2.1.4). Figure 1.11e,f show these equivalences which must be clearly understood and shown by symbols indicating in a precise manner the function they perform. Wired OR is often used in microprocessor buses to transmit a request whose source may be varied (interrupt, acknowledge).

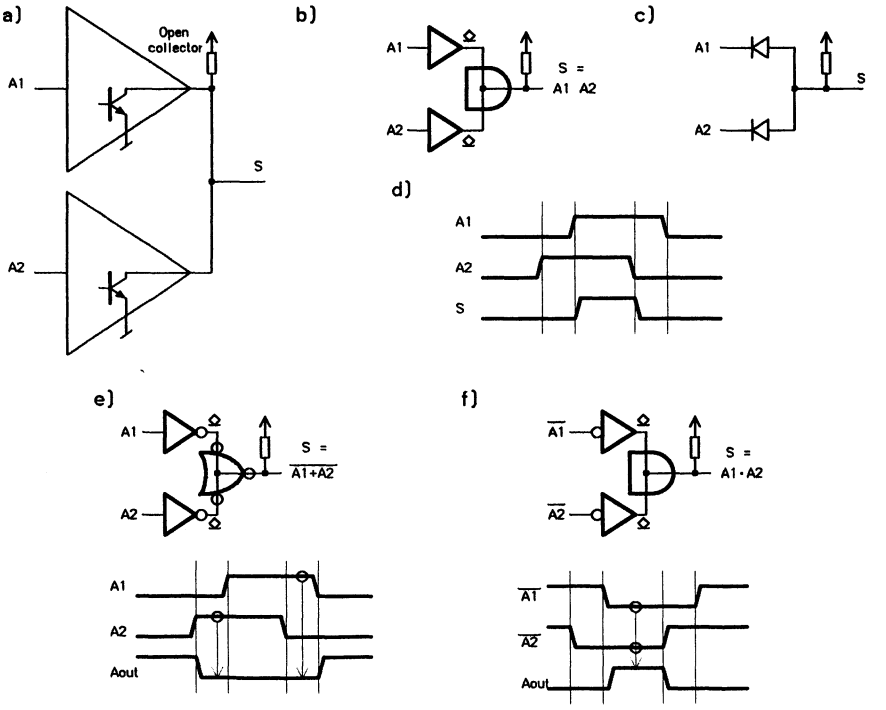


Fig. 1.11 Open collector gates and wired AND

If the lower transistor is missing (open emitter), the OR function is performed when the outputs are connected together and a pull-down resistor is used (figure 1.12a). This is referred to as a *wired OR*. This type of output is signalled by a diamond carrrying a dash on the side of the active state (figure 1.12b). TTL catalogues do not contain open emitter gates. When needed, they are made up using transistors or three state gates (figure 1.13).

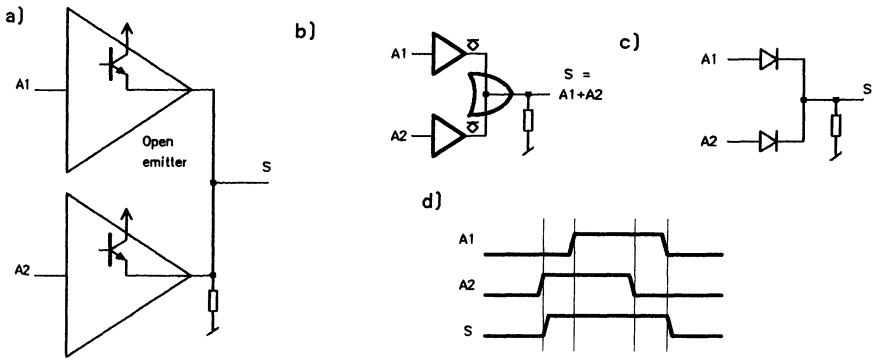


Fig. 1.12 Open emitter gate and wired OR

1.2.5 Three-state outputs

If none of the output transistors of figure 1.9 is activated, the output is in a floating 3rd state. Hence these gates are called *three-state* (tri-state is a registered trade-mark). A control variable decides whether the output is active or not (figure 1.13a). It is interesting to note that the diagram presented in figure 1.13a tolerates a mixture of functional elements (AND gates) and physical ones (transistors), all of them schematic ones; the true schematic is more complex and the aim of such a diagram is only to make the functionality easier to understand. A particular symbol (half-moon) is used in this book to characterize the activation variable (figure 1.13c). Three-state gates are mostly used for data buses (section 2.7.5) with only one output selected at a given moment in time (figure 1.13d).

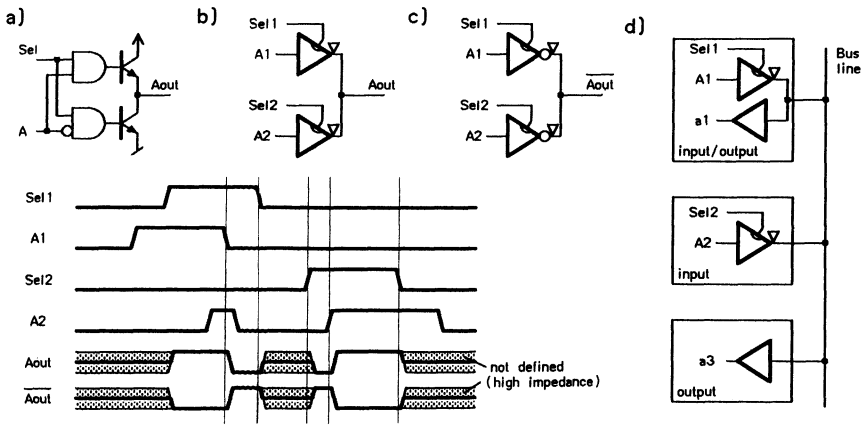


Fig. 1.13 Three-state gates and bus

1.2.6 CMOS circuits

The development of complementary MOS circuits has been slowed down by greater manufacturing difficulties. Present progress of this family probably means the disappearance of the numeric bipolar-only technology before the end of the century.

The 74C, 74HC, 74HCT, 74AC, 74ACT, 74AC11 and 74ACT11 families are based on the principle of complementary MOS with a static power consumption which is almost nil (figure 1.4). The letter T guarantees TTL compatibility, which is only obtained to the detriment of noise immunity. It is interesting to note that the first CMOS circuits were marketed by RCA with numbers starting at 4000. Motorola followed suit with numbers starting at 14500. The circuits belonging to this *"4000" family* which do not duplicate those of the 74 family have been taken over by Texas Instruments and other manufacturers (Siemens, Philips, Toshiba, Mitsubishi, etc.) and appear in "74HC" catalogues.

CMOS circuits are interesting because of their symmetric characteristic which guarantees a high *noise immunity* (figure 1.14). They can function (at various speeds) at between 3V and 15V.

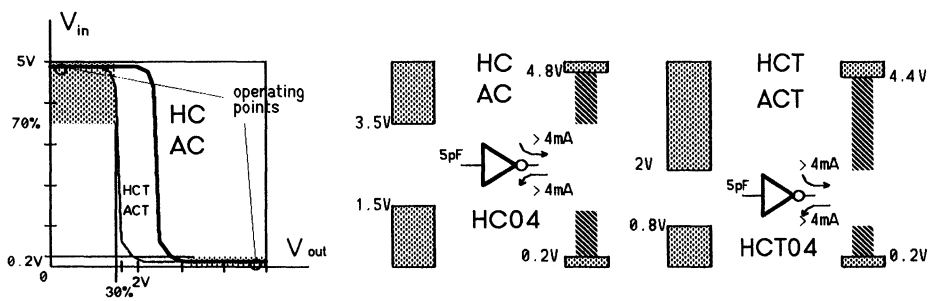


Fig. 1.14 Characteristics of the CMOS families

CMOS is not compatible with TTL since the minimum input for state "1" is higher than 3V. Adequate doping of P transistors lowers the thresholds and provides the HCT and ACT families with TTL compatibility without any change in cost and performance, although noise immunity decreases. Another compatibility solution is described in section 1.2.8.

The main advantage of CMOS is that its static power consumption is almost nil. However, dynamic consumption increases with frequency and equals that of bipolar families at frequencies higher than 10-20 MHz (figure 1.15). In the case of the new rapid AC CMOS families, decoupling must be as carried out as carefully as for the LS and F TTL families.

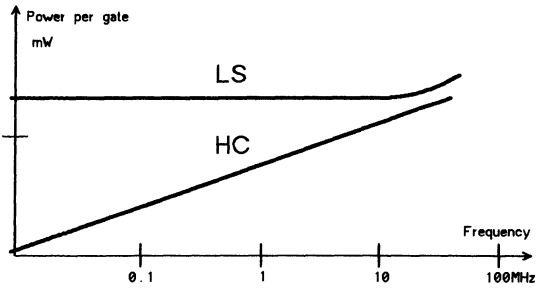


Fig. 1.15 Power consumption

1.2.7 Open and three-state CMOS outputs

CMOS outputs (figure 1.4b) may be incomplete. An *open drain* occurs when the lower transistor is missing and an *open source* when the upper transistor is missing. Both types of gates are seldom found in CMOS. On the other hand, three-state gates are frequent and do not present any functional difference when compared to three-state TTL gates.

1.2.8 CMOS-TTL compatibility

It is obvious when comparing figures 1.10 and 1.14 that a normal CMOS, HCMOS or ACMOS input cannot be controlled by a normal TTL output. If it is impossible to act on the CMOS side by replacing the circuit with HCT or ACT, the solution is to use a pull-up resistor (figure 1.16a) whose value generally varies between 1 and 10KΩ. In the case of a normal output, when the switch-over from state 0 to state 1 occurs, the TTL gate immediately (taking into account its propagation time) imposes a voltage of approximately 3.5V. The resistor then lets the voltage rise to around 5V, the time constant depending on the value of the resistor used and on the parasitic capacitor. If an open collector gate is used, the rise is exponential from the start (figure 1.16b). There are TTL open collector gates whose transistor can stand a 30V supply, which allows TTL signals to be converted into signals between 0 and 30V.

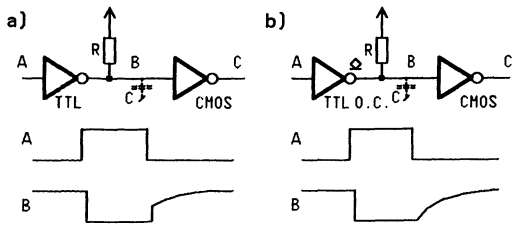


Fig. 1.16 TTL-CMOS interface

1.3 Other technologies

Several other integrated circuit technologies have been and will continue to be developed. Their life-span varies between 5 and 10 years, sometimes more when standard products are marketed in large quantities.

1.3.1 NMOS circuits

NMOS technology has never led to simple standard circuits. On the other hand, all the complex circuits from 1974 to 1986 were built using this technology, which has now been almost completely replaced by CMOS. NMOS logic is simple since it only uses N transistors, some of which play the part of a load resistor (figure 1.17). Because of these resistors, power consumption is quite great and speed is low.

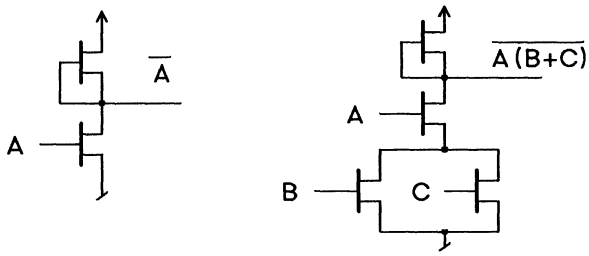


Fig. 1.17 NMOS inverter and logic function

1.3.2 Dynamic logic

Dynamic logic is often used to reduce power consumption or measurements. The basic idea is to charge a capacitor and isolate it; it thus keeps its information which can be used to control another part of the logic during a later phase. The parasitic capacitor of the base is large enough to stock the information for a few milliseconds.

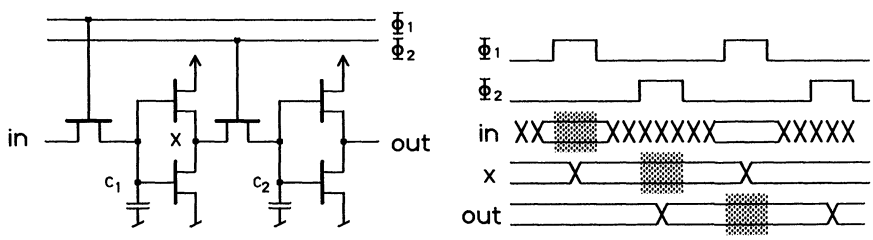


Fig. 1.18 Dynamic register

For example, a shift register element can be formed using 6 transistors whereas a D flip-flop requires 14 (figure 1.18). When clock Φ_1 is active, the "in" input is recopied onto capacitor C1 and state C2 is recopied after amplification onto the "out" output. When clock Φ_2 is active, the state of C1 is copied onto C2. The two clocks must not overlap.

1.3.3 ECL circuits

ECL technology uses non saturated transistors whose switch-over is therefore much faster [Blood82]. Logical levels are negative, between -0.8V and -1.6V. The catalog containing the "10'000" and "100'000" ECL families (Motorola, Fairchild) is quite limited compared to that of the "74" family. The ECL family is used for high speed applications in areas which must work at frequencies greater than 100MHz. Individual gate propagation times are around 2ns. Interface circuits connecting ECL to TTL - or the reverse - are available.

ECL logic uses pull-down resistors and implies respecting the electric characteristics of the transmission lines in order to avoid reflections which reduce performances and may create erroneous conditions. The end of the lines must be matched and, if several elements are to be linked, the designer must decide which is the best method for adapting line impedance (figure 1.19). ECL gates naturally generate a direct output and an inverted output. This can be put to use for wired logic.

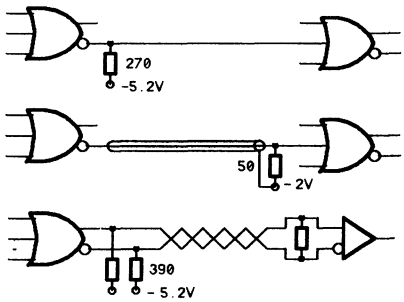


Fig. 1.19 Impedance adaptation in ECL

1.3.4 GaAs circuits

Greater electron mobility in *GaAs* (Gallium arsenide) leads to frequencies which are five times greater than those of the fastest TTL or AC-MOS. This technology took a long time to master and now provides some standard components such as dividers and fast serialisers for transmission by optical fibers. GaAs circuits work at frequencies greater than one GigaHertz. GaAs-ECL interface circuits are available.

1.3.5 Fluid and optical logic

Fluid logic is used in environments which are hostile to electronics (nuclear radiations, vibrations, electrical noise). It takes up a lot of space and only lends itself to simple combinational functions with propagation times of a few hundredths of a second.

Optical logic is still in the research stage. Using optical interferences, it promises great speeds, but the approach is radically different from traditional logic [Hunsperger85].

1.3.6 Special interfaces

The environment of many microprocessor systems includes electromechanical devices controlled by direct or alternate voltages or currents which are high when compared to the +5V of the interface. Static or mechanical relays are used when high voltages are present (section 6.3.5). Control of mechanical relays sometimes implies a relatively high current which can be obtained by connecting two gates in parallel and, if necessary, using resistors in series in order to equalize the voltages (figure 1.20a). *Protective diodes* are always required with an inductive load. A fast Zener diode protects both from negative voltages and from overloading. Open collector TTL gates, whose output voltage is higher, can control a 24V load (figure 1.20b), The same gates are used to convert a logic voltage (0-3V) into a high control voltage (0-30V). A transistor (figure 1.20c) plays the same part and allows for higher voltages and heavy currents when using transistors in a Darlington connection (figure 1.20d).

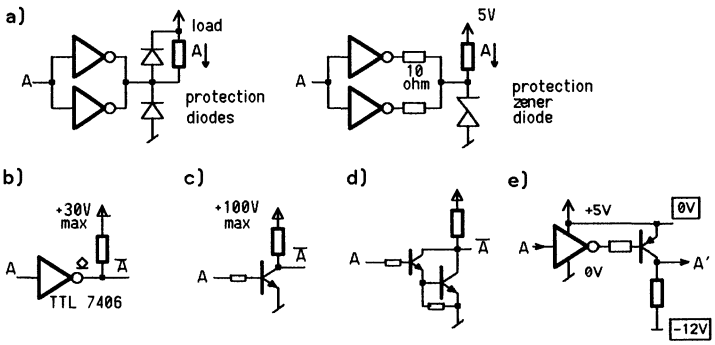


Fig. 1.20 Examples of current and voltage adaptors

In order to control negative voltages, solutions must be thought up for each individual case using transistors (figure 1.20e) or an operational amplifier (section 6.2.1) which takes into account the switch-over thresholds of the control variables, the output currents and voltages as well as the switch-over speed rates. A few power interface circuits are available and in each special logic family manufacturers offer solutions for interfacing with TTL logic.

1.4 Implementation

When comparing theoretical and practical behaviour, several effects are often difficult to master and can endanger the success of the project. Optimum behaviour is guaranteed more by experience than by calculation.

1.4.1 Correct working conditions

If the rules relating to charges, propagation times and voltages are respected, a logically correct circuit should function. In fact, it will work if serious mistakes at the electrical and electrodynamic level are avoided :

- ohmic losses on power supplies
- crosstalk due to several parallel lines switching-over at the same time
- excessively long clock lines, reflections and oscillations on synchronization signals

Generally, the conducting wires used for the power supply are as wide as possible and associated with decoupling capacitors (section 1.1.1). Ground planes (4 layer circuits figure 1.26) are recommended. Crosstalk is negligible if the conducting wires are parallel for less than a few cm and if the frequencies are lower than 20MHz. The voltage drop on the information lines is negligible. High frequency signals must be routed near power supply lines or layouts in order to restrict the impedance value of the line.

Each manufacturer sets his own standards for drawing printed circuits. They must be more or less respected according to the frequencies and power of the circuits used. The most important thing is to avoid taking risks, especially where grounding is concerned.

1.4.2 Thyristor effect

CMOS circuits are prone to a latch-up phenomenon. Following either a brief parasitic overloading/voltage drop or a faulty voltage set-up sequence on the inputs and on the power supply during start-up, parasitic thyristors are set off and produce internal short circuits which lead to partial or total destruction of the circuit through overheating. Recent circuits are better protected against this effect. Figure 1.21, which shows a cross section of the inverter of figure 1.4b, gives an inkling of the complexity of driven-in layers and the difficulty of a modelling all of the logic components.

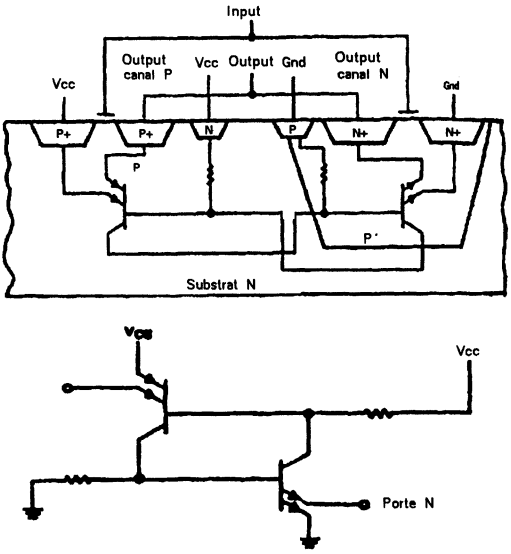


Fig. 1.21 Latch-up model

1.4.3 Electrostatic problems

Due to the high impendance of MOS and CMOS inputs, the circuits are sensitive to *electrostatic discharges* which produce breakdowns on the input transistors. The circuits are provided embedded on rubber conductive foam which short-circuits the leads. When the air is dry, the person handling the circuits should not wear synthetic materials and must be connected to the ground by a bracelet (figure 1.22).

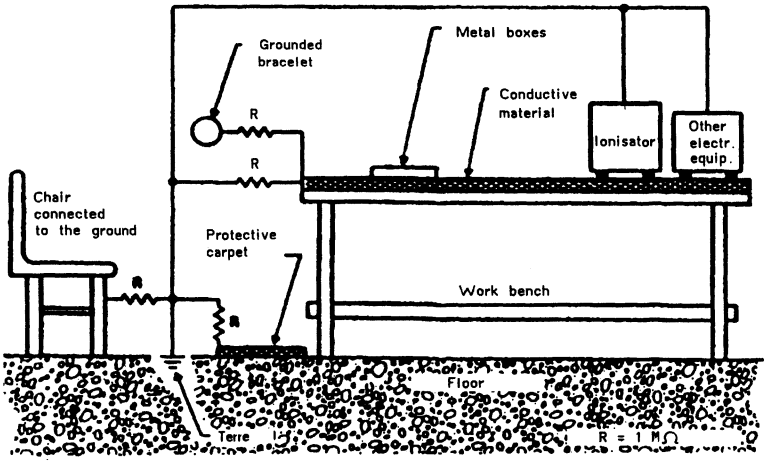


Fig. 1.22 Manipulation environment of CMOS circuits

1.5 Physical devices

There are many ways of building and implementing integrated circuits when designing an operational interface. New techniques are constantly being developed and now tend to rely more and more on miniaturization.

1.5.1 Chips

Integrated chips measure between 2 and 10 mm square (figure 1.23) and are stuck on a lead frame and encapsulated. Packages are made of ceramic in the -55 to +125°C military series and of plastic in the 0 to 70°C commercial range. They can be fixed directly on the printed circuit (*COB Chip On Board*).

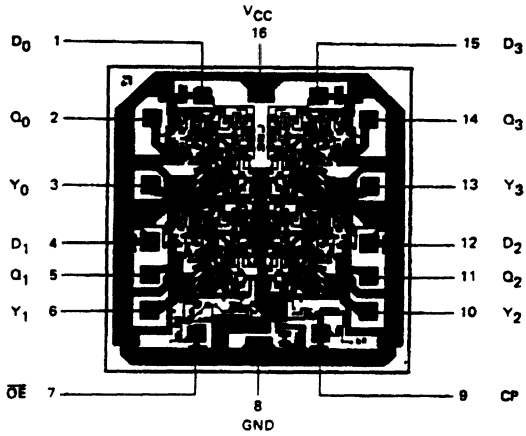


Fig. 1.23 Integrated circuit chip, ≈ 3mm square (document AmD)

1.5.2 Packages

Inside integrated circuit packages, the silicium chip is linked to the metallic components using gold or aluminium wires (figure 1.24). With the *TAB* technology (*tape automated bonding*) shown in figure 1.25g, bumpers on the circuit are used to link it to its support film by simple ultrasound pressure.

Figure 1.25 shows the variety of integrated circuit packages which are in frequent use nowadays. The most common are the *DIL* type (*dual in line*, figure 1.25a). For memories, *ZIL* packages (*zig-zag in line*, figure 1.25b) will become generalized.

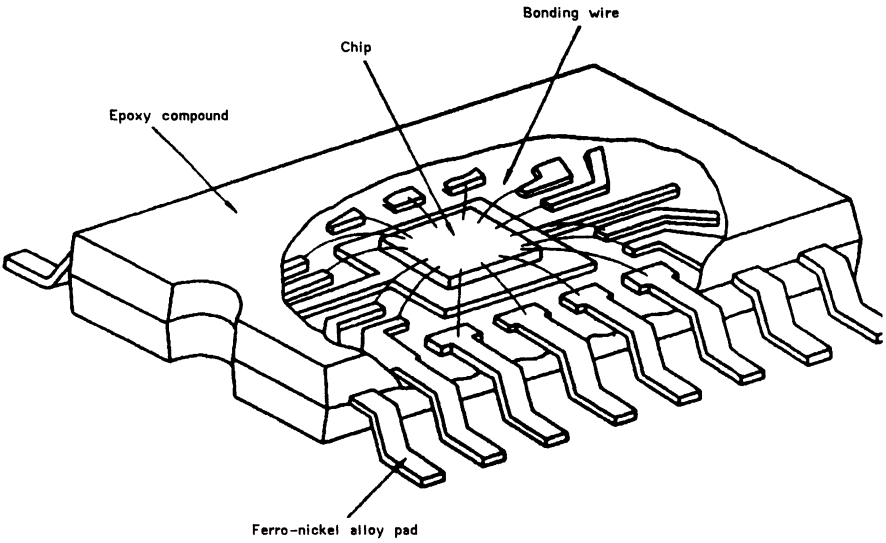


Fig. 1.24 Inside view of an SO package (Toshiba document)

Packages with *pin grids* (figure 1.25d) allow an increase in component density. In order to lower costs and reduce size, there is a tendency to resort to surface mounting (*SMD*) using flat packages (figure 1.25c,e,f and figure 1.24) equipped with pins set very close to one another. Soldering is carried out in vapor phase or by using a special heating head. These packages can also be put on sockets with side leads, and even wave- or immersion-soldered on classic printed circuits (maximum 5 seconds at 250°C).

Most components are available in different packages. Letters, which vary with the manufacturer, follow the number and indicate the type of package or the various finishes.

Power supplies are often located in the corners (*corner pining*) which makes it easier to build two-layer printed circuits but is not recommended for fast circuits. Nowadays, *center pining* is becoming the norm.

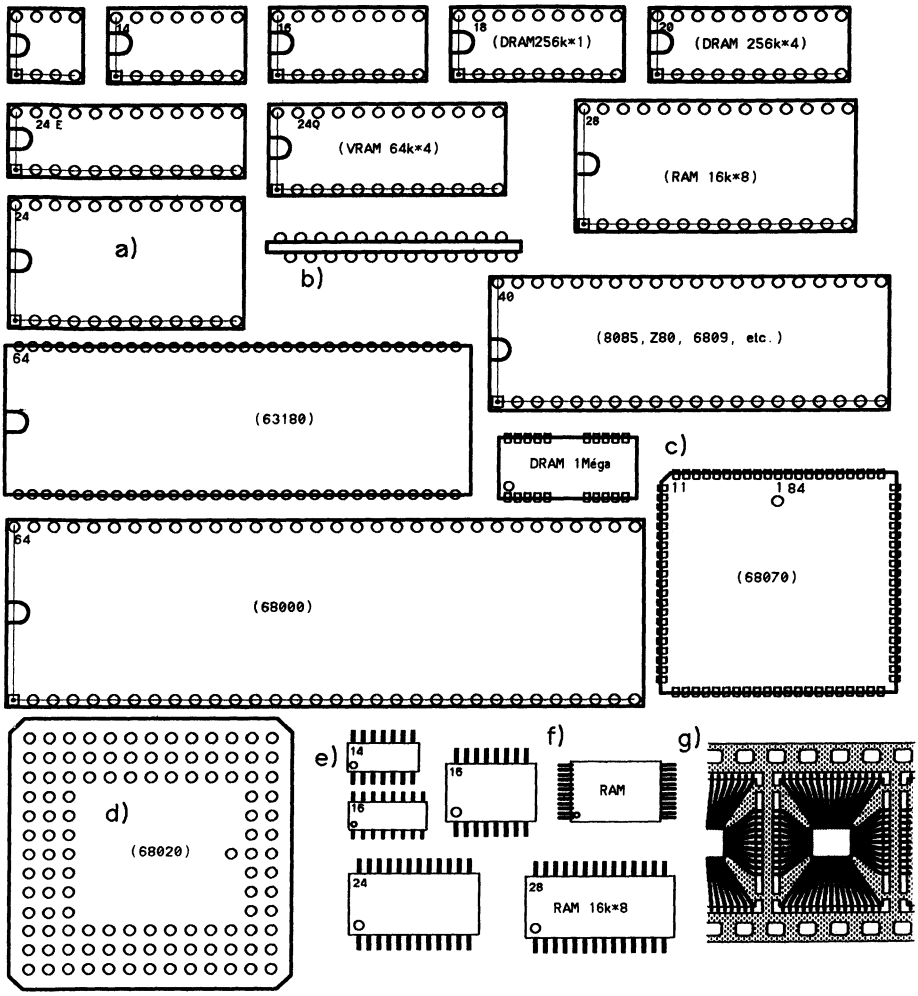


Fig. 1.25 Integrated circuit packages on a 1.1 scale

a) DIL (Dual in line)
b) ZIL (Zig-zag in line)
c) PLCC (Plastic chip carrier)
d) Pin grid
e) SOP-SMD (Small outline-Surface mounted devices)
f) VSOP (Very small outline package)
g) TAB (Tape automatic bonding)

1.5.3 Printed circuits

Printed circuits carry both the discrete components (resistors, capacitors, quartz crystals, diodes, switches) and the integrated circuits. TTL, dynamic memory and complex circuit power supplies must be decoupled by placing beside each circuit polyester capacitors of at least 0.1μF. Moreover, they must be connected by short power supplies whose section is appropriate. Circuits carrying 32 bit microprocessors and/or dynamic memories require both a ground and a power supply layer (4 layer circuit). This has three advantages which compensate the increase in price:

- high current density for power supply
- better defined conductor impedance and reduced cross-talk
- improved layout

In the case of high component density, routing requires 6 layers or more.

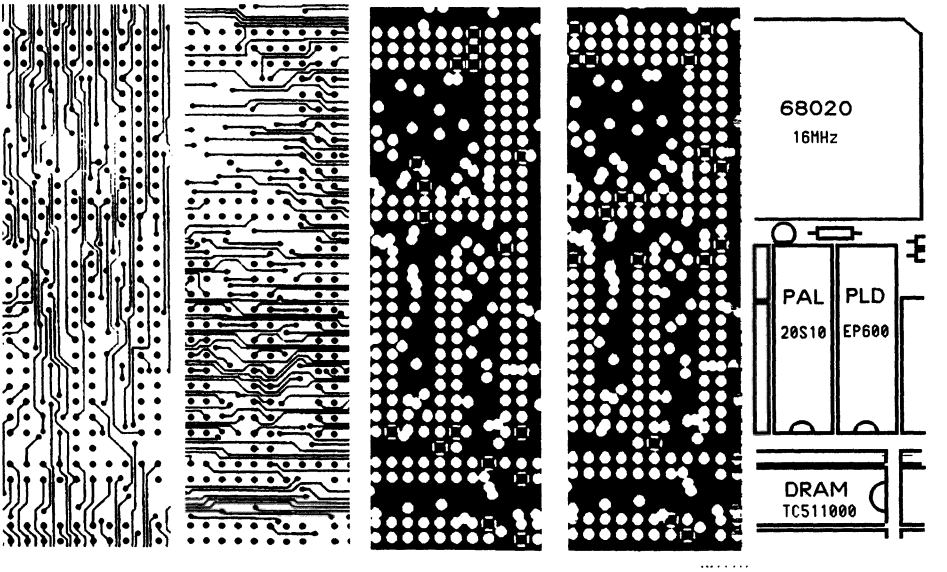


Fig. 1.26 Masks for a four-layer printed circuit

2 Combinational logic

Mastery of logic systems forms the basis of the knowledge needed for designing interfaces. The reader is assumed to have acquired the necessary basic knowledge of logic functions and their simplification using the Karnaugh method [Fletcher80]. Some experience in developing interfaces will ensure that the reminders and complements contained in this chapter are fully profitable. Wakerly's book [Wakerly90] provides a more detailed exposure of the major circuits.

2.1 Signals and logic functions

The aim of this section is to recall a few terms with which the reader is already familiar and stress which important points must be respected in order to follow a correct design method.

2.1.1 Logic signals

A logic system includes inputs and outputs which are logic variables that can only appear in one of two states or *binary values* referred to as "0" and "1" (figure 2.1).

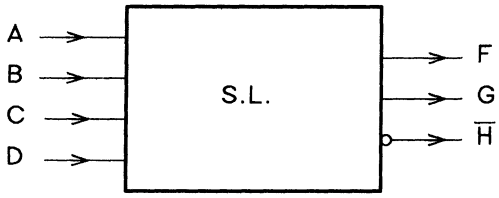


Fig. 2.1 Signals associated with a logic system

In practice, a binary signal is transported by a *line* carrying two levels known as "L" (*low*) and "H" (*high*) and corresponding to two distinct voltage levels, with level "H" more positive than level "L". With TTL logic (section 1.2.3), the "L" state is a voltage level which varies from 0 to 0.4 V whereas the "H" state varies between 2.4 and 5V. In *positive logic* a "0" is represented by state "L". The opposite is true in *negative logic* which was often used in the past.

Lines are given a name corresponding to their function. The signal is known as a *direct* one when it performs its function in state "H". It is in an *active* (*asserted*) state when the signal is "H" and an in an *inactive* (*deasserted*) state when the signal is "L". The opposite is true when dealing with an *inverted* signal. The latter is either overlined or completed by a prefix (often a slash /) or by a suffix (star * or -L).

The name given to the signal must define its active state in an unambiguous manner. For example, the read/write signal of a memory circuit must be referred to as WR and not as R/W ($\overline{R}$/W could maybe be acceptable).

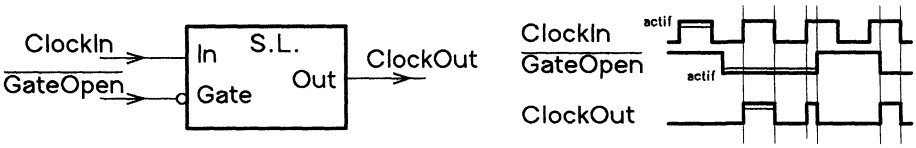


Fig. 2.2 Direct and inverted signal convention

In a logic system, a combination of input signals corresponds to a specific state of the output signals. When signals are inverted, an *inverting circle* must be represented. The functional meaning of the logic system's input or output must always correspond to the name given to the signal. The name inside the rectangle of the symbol is part of the symbol and characterizes the function it performs. The name outside the rectangle is chosen according to the application (figure 2.2).

2.1.2 Positive and negative logic

The inverting circle convention is only true with positive logic, the only one in use. Formally, one should distinguish the "inverting" function, for which a triangular sign has been reserved (figure 2.3a), and the the effect of that function as regards positive (figure 2.3b) and negative (figure 2.3c) logic. That distinction is normal, but when the inverting triangle was invented around 1980, use of the inverting circle was too widespread for industry to change-over without difficulty.

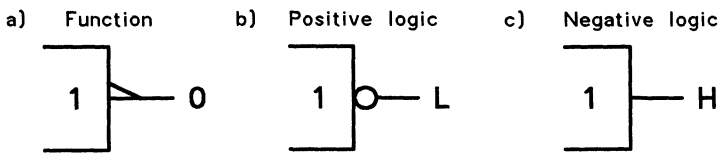


Fig. 2.3 Inverting triangles and circles

In this book, we will distinguish carefully between:

signals which are	inactive	and	active
lines with			
- logic states	"0"	and	"1"
- physical states	V_{IL}(0 to 0.4V)		V_{IH}(2.8 to 5V)

2.1.3 Decomposition of a logic system

A logic system is a *combinational* one if to every combination of the input signal states there corresponds a single equilibrium state of the output signals. A logic system with several outputs can be decomposed into several logic systems with one output (figure 2.4b). The decomposition can be carried out in various ways by defining intermediate signals (figure 2.4c). When dealing with interfaces, the art of logic systems is to single out the decompositions leading to implementations which make the most of the technology used while heeding the constraints related to power consumption, load and application speed.

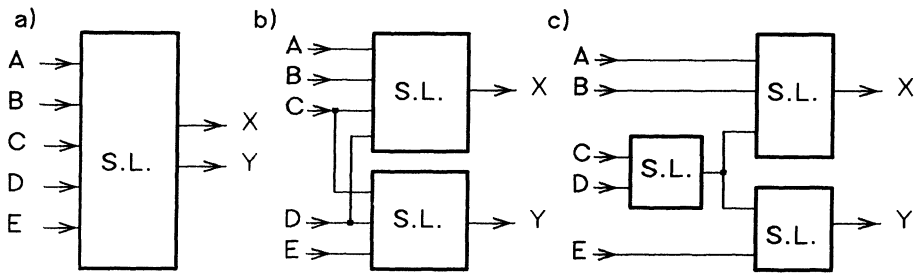


Fig. 2.4 Decomposition of a logic system

The reader is assumed to be familiar with logic systems. The following reminders are meant to clarify the terminology and symbols used in this book as well as to insist on some important points.

2.1.4 Logic gates

A *gate* carries out an elementary function. *DeMorgan's theorem* leads us to define two symbols for equivalent logic functions. Figure 2.5 summarizes the elementary logic functions with two inputs.

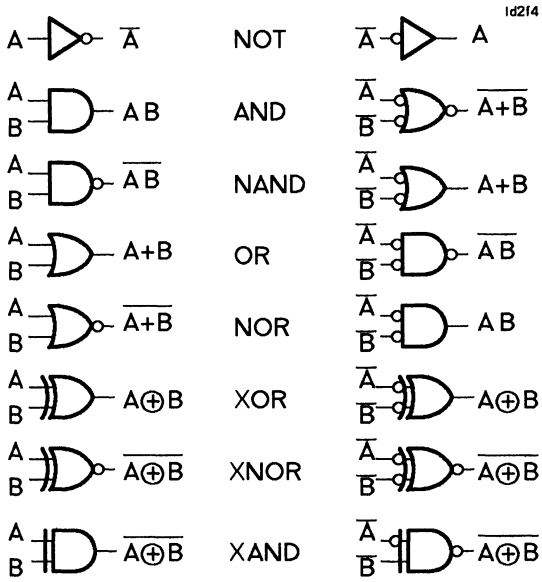


Fig. 2.5 Logic gates with 1 and 2 inputs

It is important to understand all the aspects of deMorgan's theorem. Figure 2.6 shows the equivalence through a timing diagram for an *AND* gate.

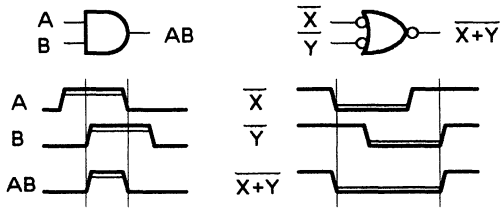


Fig. 2.6 Demonstration of deMorgan's theorem using a timing diagram.

For those who still have difficulty distinguishing between AND and OR symbols, an efficient mnemonic trick is given in figure 2.7

Fig. 2.7 Mnemonic trick for AND and OR symbols

2.1.5 Equivalences

Logic functions with more than two inputs can be built with dual input gates (figure 2.8).

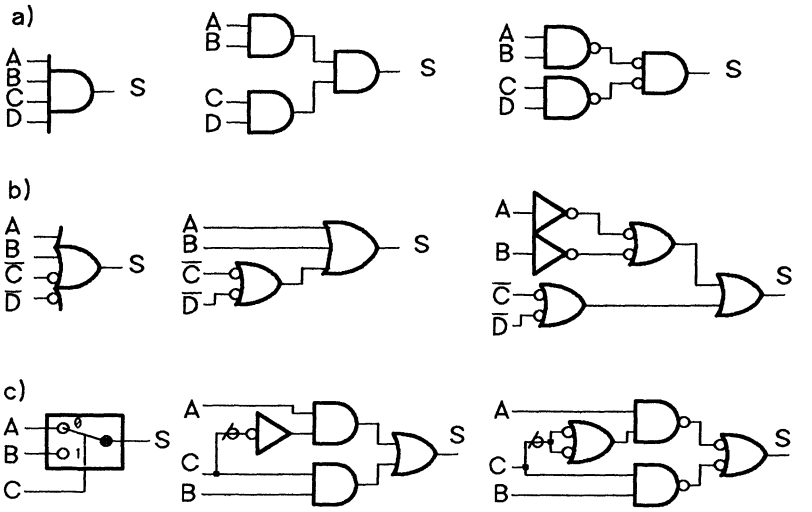


Fig. 2.8 Example of equivalent logic diagrams

Careful thought about the choice of signals naturally leads to choosing the correct symbol. An inverting circle must be placed at the end of any inverted signal (active at 0) and the name of this signal must be overlined.

2.1.6 Importance of drawing rules

The function performed by a circuit and represented by its implementation diagram must remain clear. The draughtsman must clearly distinguish between functional signals (active/inactive) and lines (direct/inverted). According to deMorgan's theorem, a *NAND* gate for example can play the part of an AND gate or of an OR one. The diagram must highlight the function it performs, i.e. the part played by the gate.

Even if both symbols are not documented for a given circuit and even if some CAD systems do not tolerate this double definition, it is necessary to try and respect correct functional symbols.

Figure 2.9 shows the example of output S which must be active low except if A is active or if B is active while C is inactive. The corresponding equation is therefore $S = \overline{A+(B\cdot\overline{C})}$.

The functional logic diagram offers an exact translation of this function. During implementation (figure 2.9b) every effort is made to reduce the number of integrated circuits, the grouping of close-by gates in close-by circuits (thus simplifying and increasing the reliability of the printed circuit) and propagation times. Use of the wrong symbol is a serious language mistake which complicates comprehension (figure 2.9c).

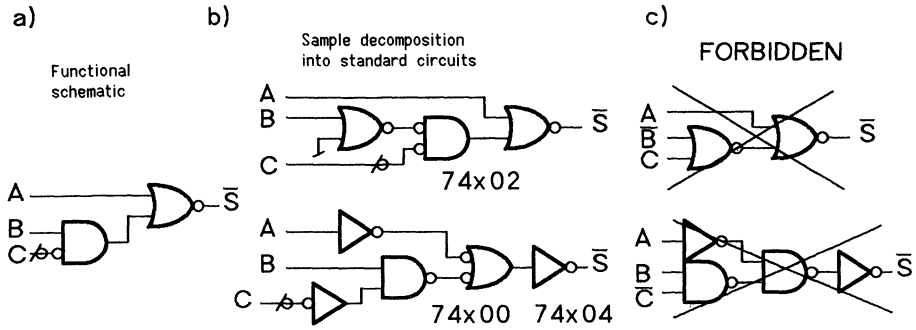


Fig. 2.9 Functional and implementation logic diagram

Some manufacturers tend to use graphic language rather erratically in their diagrams. Figure 2.10 gives an example taken from an application note as well as the corrected version.

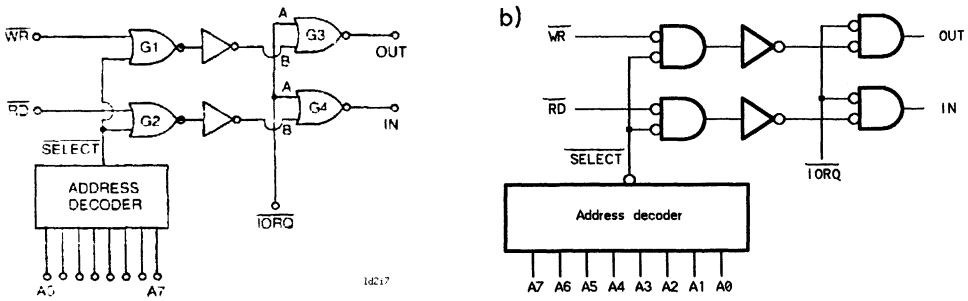


Fig. 2.10 Diagrams which do not respect the rules applied in this book and corrected diagram

2.1.7 Rules for inverting circles

In a combinational diagram, a link corresponds to a signal which is either active at 1 (direct) or active at 0 (inverted). An inverting circle must always appear on the input or on the output of a logic gate when the signal is an inverted one. This rule is respected in figure 2.10b: signals $\overline{WR}$, $\overline{RD}$, $\overline{SELECT}$ and $\overline{IORQ}$ are inverted whereas OUT and IN are direct.

Cases in which this rule cannot be respected must be explicitly signalled using a *functional pseudo-inverting* symbol. For example, the diagram of figure 2.7c, which is repeated in figure 2.11a, was drawn in order to highlight the fact that there are really two command variables C and C' = $\overline{C}$ which respectively let through B and A. Line $\overline{C}$ has a different meaning at each end; a sign must be inserted showing that such is the designer's wish (figure 2.11a). The crossed out inverting circle used on the Valid CAD system is used here.

Far too few designers - and no manufacturers - have realized how important it is to be coherent with inverting circles. Fletcher's book [Fletcher80] is the only one to insist on this notion referred to as inhibition and authorization and noted using as a symbol a triangle whose base faces up or down. The Valid notation is less ponderous and can stand up to automatic verification.

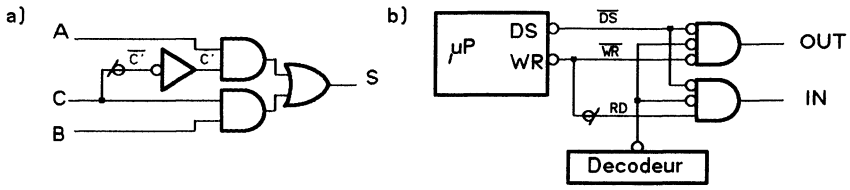


Fig. 2.11 Use of functional pseudo-inverting symbol

Figure 2.11 shows two typical examples of the use of this rule. Figure 2.11a is the 2-input switch already seen in section 2.1.5. The case of figure 2.11b is common with a microprocessor: the read/write signal is perceived both as a read command or as a write command. The functional signals DS (Data Strobe) and WR (Write) are declared inside the rectangle which characterizes the microprocessor. DS is active when the processor is trying to transfer information. Moreover, WR is active if this transfer is a write operation. For technological reasons, both these functional signals are inverted at the output; the output lines of the processor are therefore called $\overline{DS}$ and $\overline{WR}$. When the $\overline{WR}$ line is active (state "L"), the OUT output can be selected. If $\overline{WR}$ is inactive, it can be considered as a functional RD (Read) signal active at "H". The new name on the other side of a functional inverting circle clarifies the designer's thought process. Figure 2.12 emphasizes yet again the double meaning each signal can convey.

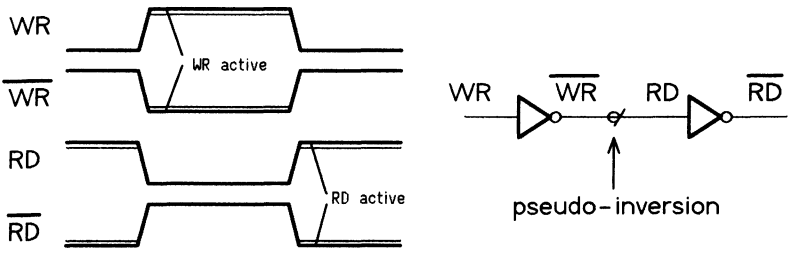


Fig. 2.12 Signals active at "1" and active at "0"

Another example is described in figure 2.14 where the $\overline{U}$ signal is delayed and plays the part of an inhibitor. Not being able to express clearly why the inverting rule is not satisfied helps to pin-point logic synthesis mistakes. In the case of figure 2.14 function $S=A\cdot\overline{B}$ is performed; when it is wired directly, the hazards which appear in figure 2.14 disappear.

2.1.8 Rise and propagation times

Signal transmission is not instantaneous due both to *propagation times* on the wires and in the logic elements and to signal *rise time*. figure 2.13 shows the difference between a *logic signal* and a *real signal* in the case of a simple inverter. A real circuit is characterized by a set of static and dynamic parameters given by the manufacturer. It is interesting to note that manufacturers almost never give minimum values and that typical values must not be used in a *worst-case design*. When no unit is given, time is expressed in nanoseconds.

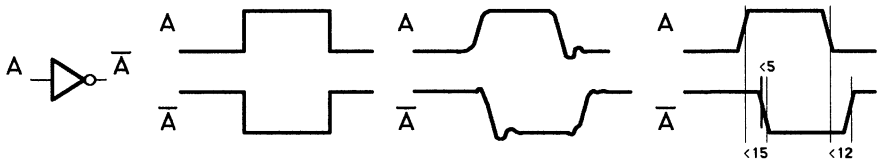


Fig. 2.13 Logic, physical and model of signals

Delays of logic elements cause *glitches* which are more or less troublesome according to the application (figure 2.14).

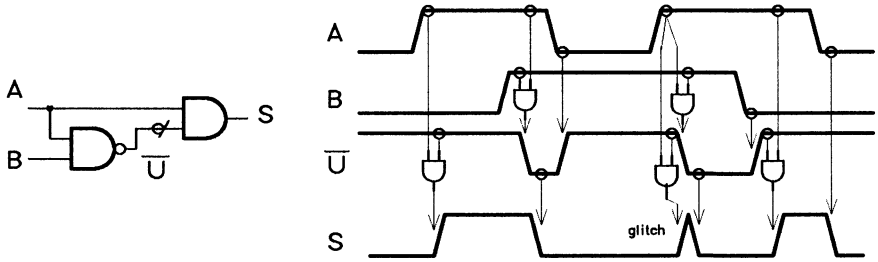


Fig. 2.14 Examples of glitches

Glitches due to the propagation time of a single gate are usually partly filtered by the propagation times of the neighbouring gates. They may only be troublesome in a small number of manufactured elements, i.e. a long time can pass before the problem is detected.

Problems due to *crosstalk*, *reflection on long lines* and *faulty voltage supply* can amplify logic glitches or create new types of glitches and faulty working conditions.

Glitches are sometimes created intentionally with minimum values guaranteed by the use of delays, although this technique should be used with care. The delay introduced by a 74LSxxx gate is typically of 10ns; the manufacturer guarantees a maximum value (for example 13ns) but never a minimum one so as to avoid having to get rid of circuits which are improving every year.

2.2 Implementing a combinational system

Combinational logic is based on a simple and well known theory, but each function may have numerous implementations. The diversity of these implementations will increase in time because of new technologies and development tools. The simple example that follows emphasizes that diversity.

2.2.1 Logic expression

The combinational logic function performed by the output of a logic system is mostly a "logic" one, i.e it can be expressed in terms of logic conditions with which the designer is familiar.

For example, in order to build an interface for a microprocessor, the negative write select impulse on a register must be activated when the hexadecimal addresses 28 to 2F are activated. The diagram and the equation are a logical consequence of this description (figure 2.15) and make a complete truth table unnecessary.

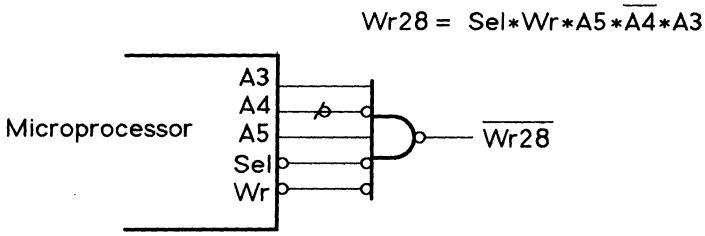


Fig. 2.15 Example of a simple logic function

Detailed implementation can be carried out with various simple elements or with one complex element whose function is a specialized one. The following sections illustrate a few typical examples of implementation.

2.2.2 Implementation using transistors

MOS transistors are the basic elements used when designing a *very large scale integrated circuit* (VLSI) [Annaratone86] [Weste85]. Propagation times are determined by simulation according to the size and characteristics of the transistors and of the parasitic capacitors. The solution of figure 2.16 should be avoided for the 5 transistors connected in series either provide too high a resistance or require the use of excessively large models. The equivalent solution of figure 2.16b is preferable.

Both types of transistors are found in P and N wells. Many topological alternatives are possible. A first step towards ideal topology is a *stick drawing* which does not respect the exact size of the transistors (figure 2.17).

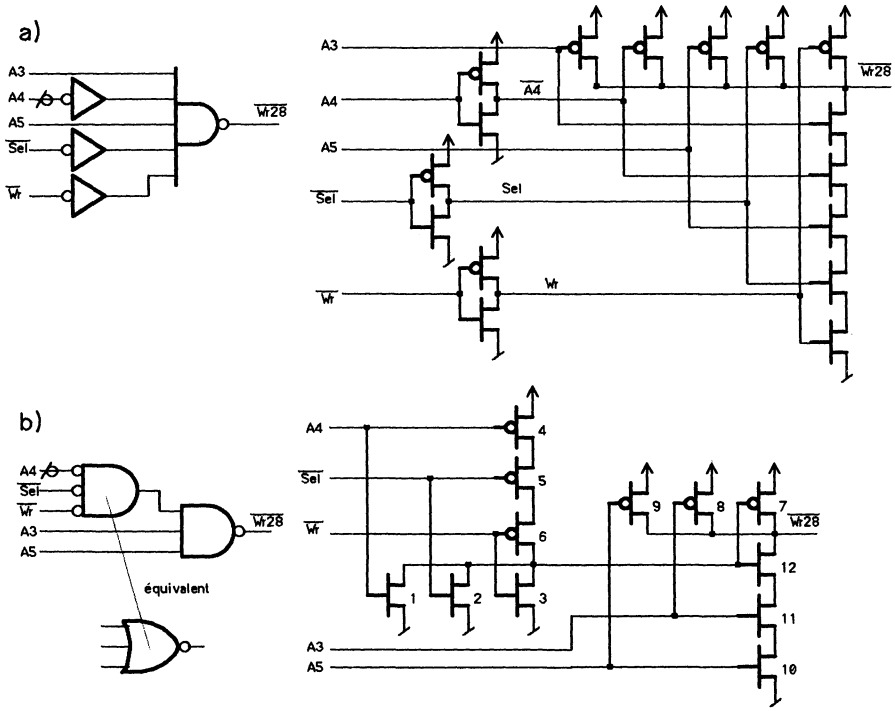


Fig. 2.16 VLSI diagram (the transistor numbers refer to the following figure).

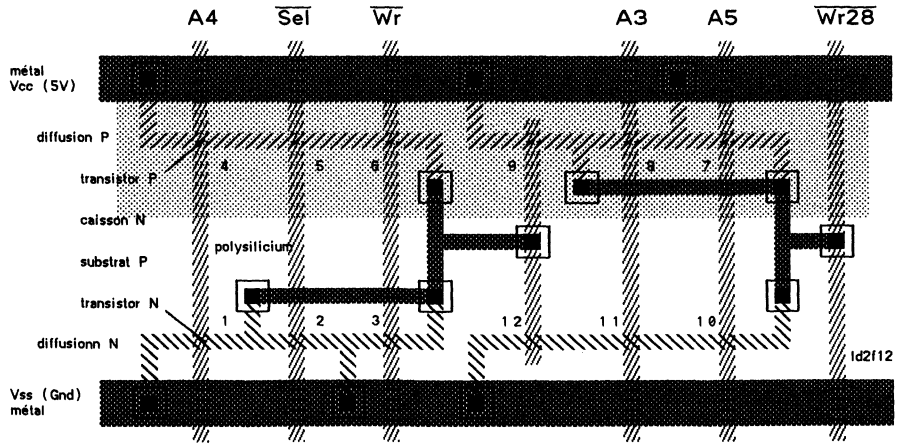


Fig. 2.17 Symbolic layout of the diagram in figure 2.16b (sticks)

2.2.3 Standard VLSI library

When optimization of the silicium surface is not required, the designer uses standard library elements which are *precharacterized*, i.e. elements whose performances with a given technology are known. The logic diagram is drawn using a CAD system and referring to the inverters and gates of the proposed standard catalogue. Automatic routing leads to a solution resembling that of figure 2.18, whose surface is approximately twice as great as what can be obtained with an optimized routing derived from figure 2.17.

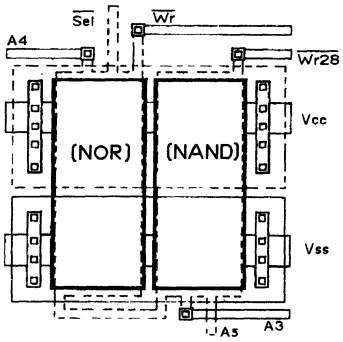


Fig. 2.18 Routing given by standard VLSI library

Building a VLSI circuit with files produced by CAD software implies creating something like ten masks as well as numerous photographic and chemical steps. Manufacturing time exceeds 2 months with a basic cost of over 500.- dollars per mm^2.

2.2.4 Gate arrays

The idea of *gate arrays* is to give the user an integrated circuit equipped with a grid of identical gates, e.g. with 4 inputs. The desired function is defined through a final layer of connections. Figure 2.19 shows one possible implementation with yet again the same logic function. In this case, circuit manufacturing time is greatly reduced for there is only one mask to produce and one single metallization step to carry out (in a single layer when standard cross-connections are available).

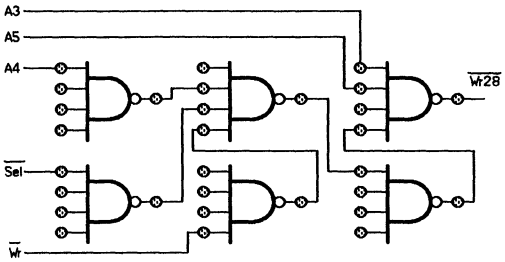


Fig. 2.19 Routing of 2nd solution on gate array

In classic gate arrays, links are implemented using reserved zones (channels) between the gates. With the *sea of gates* concept, routing occurs above the gates, which increases density but implies mastering 3 or 4 superimposed metallization layers, the final one or two of which are at the client's discretion.

The *application specific integrated circuits* or *ASICs* described above contrast with the very low cost standard mass-produced circuits available from various manufacturers.

2.2.5 The "74" family

The use of TTL or HC-MOS circuits also leads to various solutions. The aim is generally to optimize either the number of packages or the speed. Figure 2.20 shows the standard way and three non trivial solutions which only require one package. The first solution needs two low cost integrated circuits which have been available for the last 20 years (figure 2.20a). The second (figure 2.20b) uses the gates of a mixed circuit only available in HC-MOS (section 2.5.2), the third (figure 2.20c) an equality comparator (section 2.9.1) and the last one (figure 2.20d) a decoder (section 2.8.2). The latter solution is the most satisfactory one; moreover it provides reserves for generating other decoding functions which may also be needed.

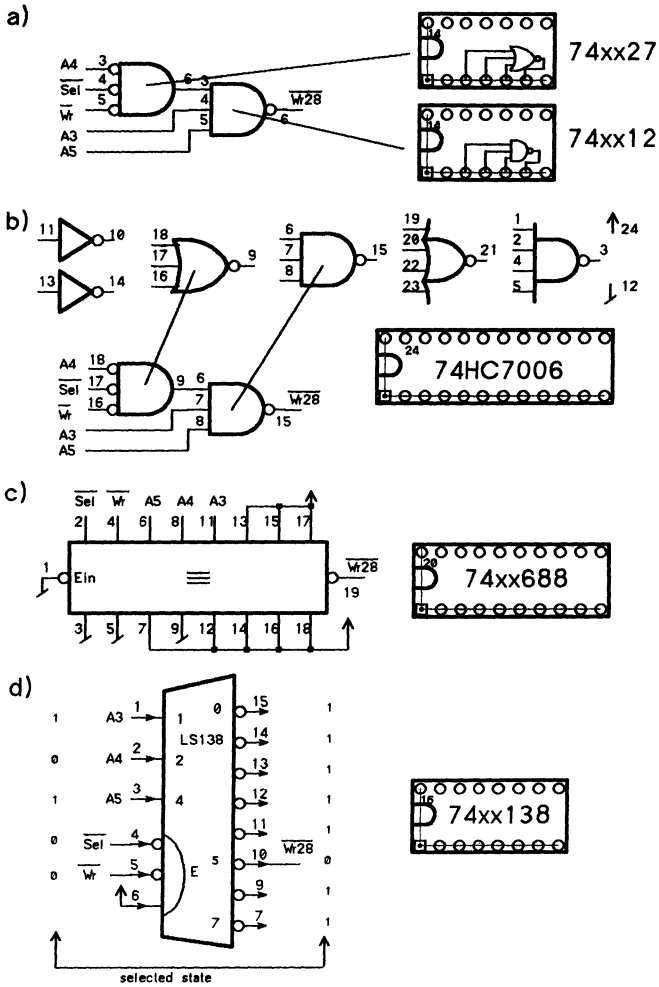


Fig. 2.20 Solutions with standard TTL or HC-MOS circuits

2.2.6 Programmable logic

Using logic equations (figure 2.21), it is possible to program a *programmable logic device* (*PLD*) whose detailed implementation will be explained in chapter 5. The solution using a PLD costs more money and current but it reduces the number of components and requires less investment than gate arrays or special VLSI circuits.

```
.NAME   PAL10L8

.PINS

VCC   = 20
GND   = 10

/Sel  = 1     ; Input declarations
/Wr   = 2
A5    = 3
A4    = 4
A3    = 5

Wr28  = 19    ; and outputs

.EQUATION

Wr28  = SEL * WR * A5 * /A4 * A3

.END
```

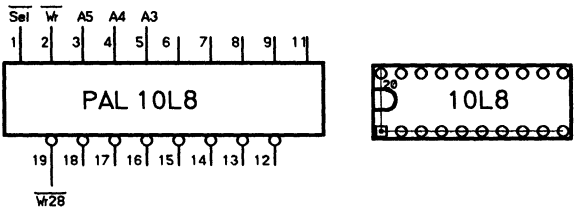


Fig. 2.21 Solution using a 10L8 type PLD (10 inputs, 8 outputs)

2.2.7 PROMs

The truth table of the function to be performed can be used to program a *programmable read-only-memory PROM* (section 4.2.2).

Figure 2.22 gives an example with a small fusible PROM of the TBP18S030 type (formerly known as 74S288), holding 32 8-bit words. Input variables are connected to PROM address lines. For each address at which the output must be active, a "1" must be programmed inside the EPROM. In figure 2.22, the source code of the PROM content indicates, in microprocessor assembly syntax, that the PROM is clear, except for address 00101 whose state corresponds to the output select. This solution is very suitable when dealing with very complex equations.

```
.RADIX2
.LOC   0              ; Addresses: bits 543210 = Sel Wr A5 A4 A3
.FILL  10'32,00000000 ; Zeros everywhere exept for

.LOC   00101          ; the selected address
.8     00000001       ; Data for activating D0 bit

.END
```

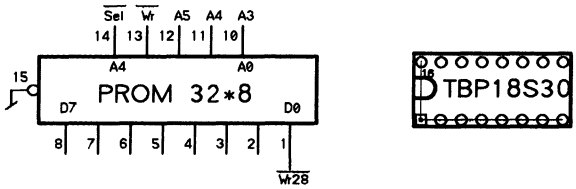


Fig. 2.22 Solution using one 32 x 8 PROM

2.3 Transformation of logic functions

The logic functions to be generated are initially defined either in a truth table or through a description which can easily be translated into equations. The simple example of the previous section shows that it is necessary to be able to switch from one representation to the other and to transform the equations according to the available technology and standard modules. Books on logic systems list all the simplification methods. Common sense and familiarity with *Karnaugh maps* are sufficient to solve all problems without the help of computers; for the difficulty lies more in correct analysis and comprehension of the problem and its decomposition than in the implementation of the chosen solution. Hardware or software simulation is useful when checking that specifications have been heeded as well as for testing alternative solutions.

2.3.1 Karnaugh maps

The reader is assumed to be familiar with the principle of Karnaugh maps [Fletcher80]. The aim of this section is to recall their most practical disposition and to present an example. Tables with 3, 4 and 5 variables are easy to represent (figure 2.23). In the case of 5 variables, the reader should try and imagine that both blocks of 16 cells are superimposed in figure 2.23c and mirrored in figure 2.23d. Users usually prefer the first method they are shown.

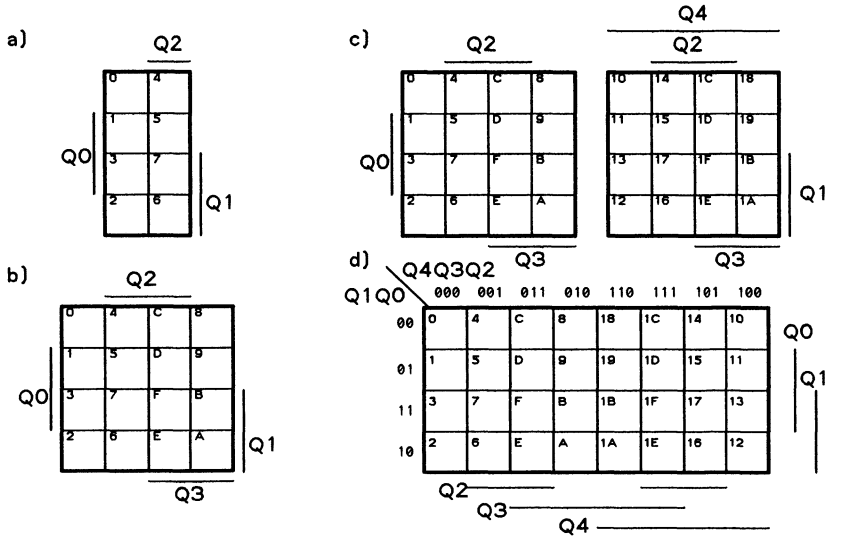


Fig. 2.23 Karnaugh maps with 3, 4 and 5 variables

2.3.2 Example

Karnaugh maps can generally be filled in directly without using a complete truth table. For example, let us take the case of a decoder with 5 inputs and one output and whose specifications are as follows:

SHEN is active if REF is active and Q4 inactive
or if VID is active and Q2 inactive
or if AS is active

Moreover, due to the application, REF, VID and AS are never simultaneous and Q2 always comes before Q4.

The equation $SHEN = REF \cdot \overline{Q4} + VID \cdot \overline{Q2} + AS$ is directly derived from this formulation. The corresponding Karnaugh map appears in figure 2.24b where dashes refer to excluded conditions.

It is sometimes necessary to simplify the reverse function in order to reduce complexity or to avoid an inverting level which creates delays; cover of zeros (figure 2.24c) leads to the equation:

$$\overline{SHEN} = \overline{AS} \cdot \overline{REF} \cdot \overline{VID} + \overline{AS} \cdot Q2 \cdot Q4 + \overline{AS} \cdot \overline{REF} \cdot Q2$$

Other syntaxes for logic equations will be discussed in chapter 7.

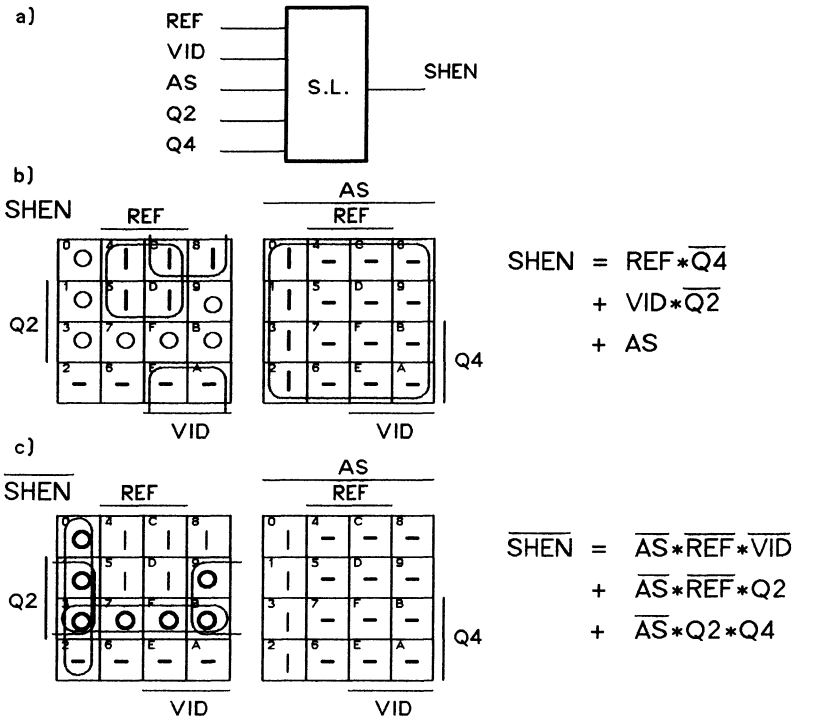


Fig. 2.24 Simplification example for direct and inverted function

2.4 Standard combinational circuits

Apart from the microprocessors, programmable interfaces, PLDs and increasing amounts of gate arrays, the circuits found in personal computers, in their interfaces and peripherals as well as in industrial command systems are circuits of the TTL and HC-MOS families. The choice afforded by the manufacturers is great, sufficiently complete and relatively coherent. The aim here is to condense the knowledge found in dozens of different manufacturers' catalogues into a small number of explanations and summary tables in order to speed up the mastery of existing circuits and make it easier to choose components for a given application.

The catalogue offers a preliminary choice of components providing good functionality. It is necessary to refer to the detailed documentation of the chosen circuit in order to check electrical characteristics and compatibility with linked circuits.

Reasonable documentation about standard logic circuits can be obtained from the distributors of the main manufacturers, whose basic catalogues are very similar except for a few specialities. According to the company and the geographical area, one manufacturer may have preference over all others. The specialities of the other manufacturers are thus ignored or even forbidden in order to avoid excessively diversifying the company's internal stocks. In large companies, each new component must be validated by a special department before it can be used by a designer. This department does not only deal with buying and managing stocks; it also specifies reliability tests, which cost a great amount for each new component.

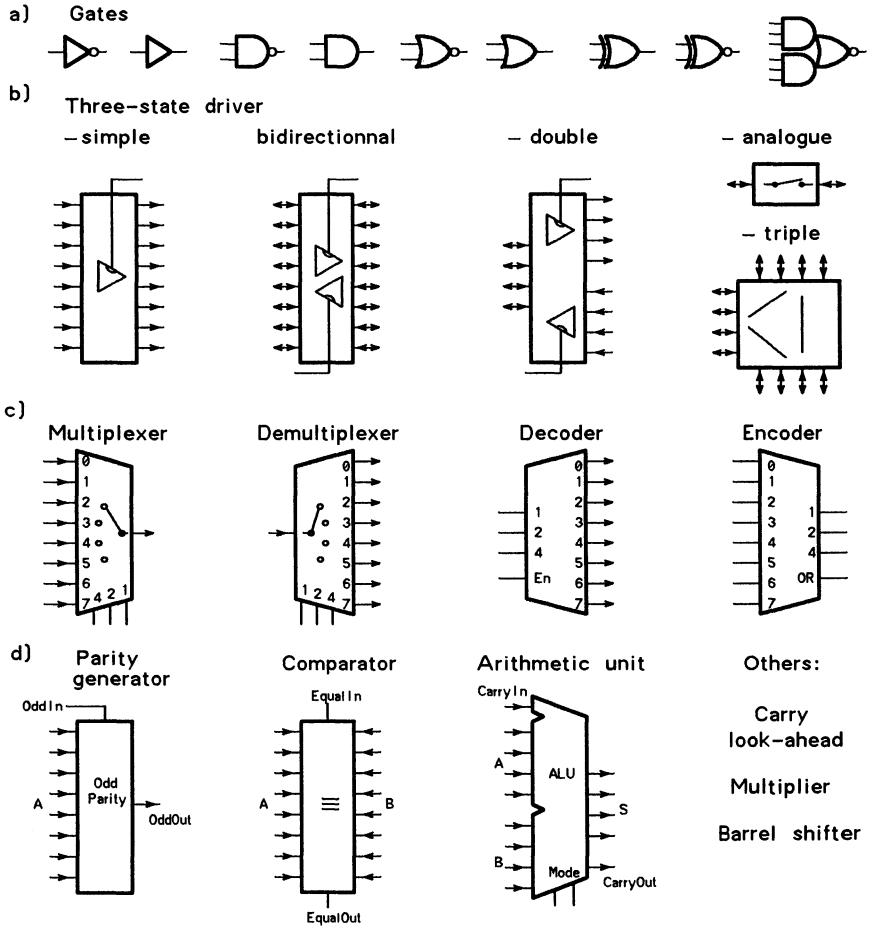


Fig. 2.25 Standard combinational systems

2.4.1 Available functions

Alongside gates equipped with a variable number of inputs (figure 2.25a), a few complex logic functions exist as standard circuits in order to help build computing units and interfaces. *Drivers* (figure 2.25b) can be used to link two or more buses. Existing circuits will be discussed in later sections. Let us note that a driver can be made up using an analog transfer gate (figure 1.4c), which is only found at the internal level of the circuit or in special applications.

Multiplexors are used to select one of many inputs. *Demultiplexors* do the opposite; they are interchangeable with *decoders* and the select input corresponds to the *enable input* of the decoder outputs. *Encoders* give the address of the first active input while giving priority e.g. to the high level inputs, which is precisely why they are known as priority encoders. All the circuits of figure 2.25b have various alternatives with three-state outputs, input registers, etc. and will be discussed in the following sections. There is one last group of arithmetic circuits which includes *parity generators*, *comparators*, *adders* and *ALUs (arithmetic logic unit)* (figure 2.25d). Some other special arithmetic circuits were brought out a few years ago for developing minicomputers based on standard integrated circuits.

Let us note that the outputs of all these elements are often inhibited by a command variable which acts either on the output of an AND gate (figure 2.26a) or on a three-state gate (figure 2.26b). In this book, we will clearly distinguish between these two cases and use the words *gate* and *enable* more consistently than circuit manufacturers.

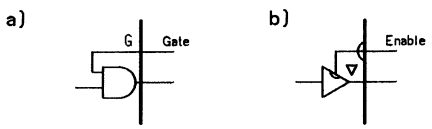


Fig. 2.26 Output inhibition

2.5 Simple gates

Elementary gates have already been discussed in section 2.1.4. Manufacturers only ever give one of the two equivalent symbols which describe each gate according to deMorgan's theorem (figure 2.5).

2.5.1 Gate catalogue

The gate catalogue is quite complete but for each family there are a few gaps and specialities (figure 2.27). Moreover, a gate in a given family may offer special characteristics (higher voltage, special thresholds, etc.). The table highlights the most common circuits. The circuits indicated in small print are either too old (74N and 4000 families) or new i.e. not produced by enough manufacturers. In both cases, careful enquiries should be made before the component is used for mass production.

	TTL (LS,ALS,AS,F)						CMOS (C,HC,HCT,AC,ACT,ACL)					
	1-2	3	4	5-8	>8	comb.	1-2	3	4	5-8	>8	comb.
6x	04 1004						04 4069 901 903					7006 7074
	34 1034 63						4010 4050 4504					
	265											
2x 4x 6x	8003 00 37 804	10	40 20 140	30	133 134	265 31	40107 00 4011 804	10 4023	20 4012	30 4068		7006 4501 4572
4x 6x	28 128 08 808	11	21				08 4081 808	11 4073	21 4082			
4x 6x	02 805	27	23	260			02 4001 7002 805	27 4025	4002	4078		4000
4x 6x	32 832						32 7032 4071 832	4075	4072	4078		
XOR 4x	86 386					135	86 386 4070					
XNOR 4x	810						4077 7266 810					
1x 2x	54 51	54	55 64				4085 4086	58				4506

Fig. 2.27 Main gates available in bipolar 74 families and CMOS (normal outputs)

The above table only includes circuits used for logic. There are large numbers of other circuits whose voltage or current levels are input and/or output incompatible (75 family, line, LED, LCD, plasma display amplifiers, etc.). Pinout of standard circuits is not always coherent from one circuit to the other. Even worse is the fact that two functionally identical circuits carrying the same number may be equipped with two different pinouts according to the sub-family (figure 2.28).

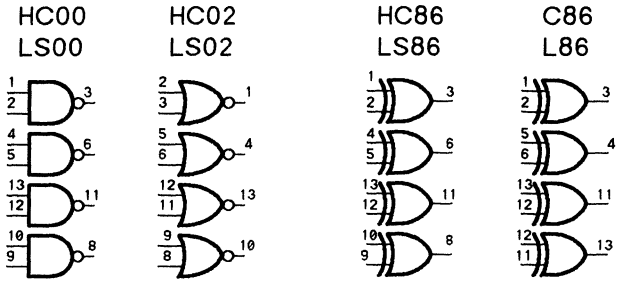


Fig. 2.28 Examples of pinout anomalies

2.5.2 Exercise

Texas Instruments has a 74HC versatile circuit. It is described in figure 2.29a using the symbols adopted throughout this book and in figure 2.29b with the AIEE symbols which will be seen in section 3.8.2. Show how this circuit can be used to build all the AND, NAND, OR, NOR gates having up to 9 inputs as well as the XOR and XNOR ones with 2 inputs.

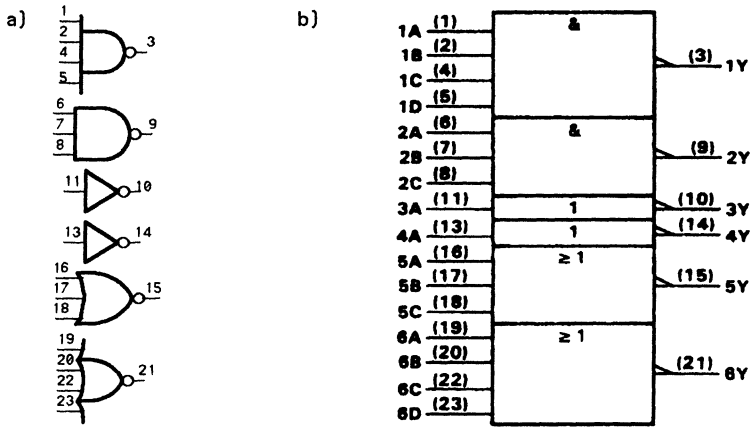


Fig. 2.29 74H7006 gate

2.5.3 Special gates

Catalogues contain some combined gates which perform the inverted AND-OR function (figure 2.30a) in one single level with the propagation time of a simple gate. Such gates are used when they are suitable for the diagram or because there is no other choice (gate arrays). Some old gates have direct links with the internal amplification level so that logic can be added without including a new amplifier, e.g. an extra delay (figure 2.30b).

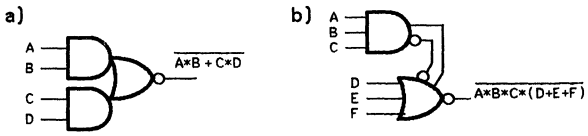


Fig. 2.30 Combined AND-OR gates

2.6 Special input gates

2.6.1 Low charge inputs

The static charge (input current) of a gate is a function of the chosen technology; it reaches its maximum in 74S and is nil in HC-MOS. It is sometimes reduced inside a single family, which is useful when connecting several gates to the same bus line. Power drivers, comparators and decoders (sections 2.7 and 2.8) sometimes have this type of *low load input*.

The capacitive load varies from 7 to 20pF per input for all technologies.

2.6.2 Hysteresis threshold inputs

Schmitt trigger inputs are advantageous because of their definite change of state, even if the input evolves slowly or shows amplitude oscillations which are lower than the hysteresis value (figure 2.31).

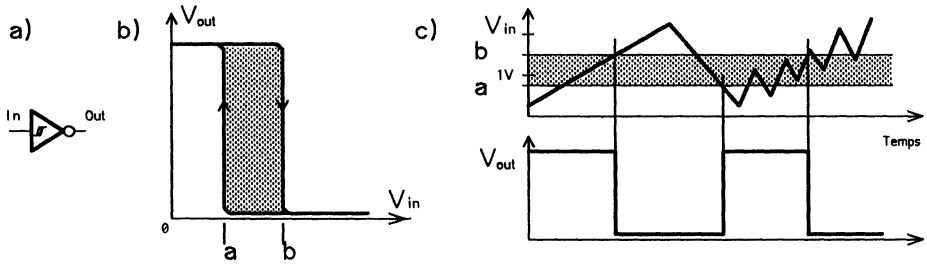


Fig. 2.31 Schmitt trigger inverter

2.6.3 Available Schmitt trigger gates

Only a few logic functions are available in Schmitt trigger gates. They are mostly not used for logic but as inputs/outputs i.e. for line filtering. The propagation time of a Schmitt trigger gate is about twice that of a normal gate in the same technology. Switch-over thresholds depend on the logic family and on the type of circuit. They can even vary from one manufacturer to another. This influences applications in which the input changes slowly.

2.7 Special output gates

2.7.1 Buffered outputs

When controlling heavily loaded lines such as a clock or a bus line, it is necessary to use *amplified output* gates called *buffers* (*power drivers*). They are characterized by the current they can absorb while guaranteeing a good "0" state (this state is the most difficult one to guarantee in TTL) and a good "1" state.

Among the inverters and NAND, NOR, etc. gates, there are quite a few power drivers like the LS37, LS40 and the ALS10xx and ALS18xx. FAST or AS technology gates are often used to increase the output current, although they also increase the input charge. This is not the case for the new BCT family, which will surely be very successful Power drivers often have single or three-state outputs because they are used for bus interfaces. Groups of 4 or 8 power drivers are found in a package; they will be discussed in section 2.7.7.

2.7.2 Single state outputs

A logic gate can be built so as to only impose the electrical "1" or "0" state and remain neutral when faced with the 2nd state. This state is then imposed by a load resistor (section 1.2.4). A *pull-up resistor* is required if the gate is an *open collector* one or an *open source* one in CMOS. Figure 2.32a shows an open output gate with an external resistor. A *pull-down resistor* is required for *open emitter* or *open drain* gates. Some complex circuits include the resistor internally in order to impose the passive state; specific graphics highlight this case (figure 2.32b). Open outputs show degraded temporal characteristics because the resistor charges the parasitic capacitors relatively slowly (figure 2.32c). Speed is improved by reducing resistance but power consumption rises and the logic level becomes degraded.

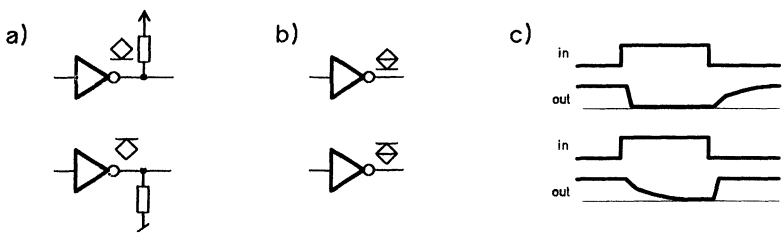


Fig. 2.32 Open collector and emitter gates

The catalogue of open collector or open drain gates is not a very orthogonal one. It is interesting to note that the OR function is lacking for no reason.

2.7.3 Applications of open collector gates

Open collector gates are only exceptionally used for logic (figure 2.49). However, they are often used for microprocessor buses and for controlling lamps, electromagnets and motors (section 6.3.6). Open collector circuits with a high output voltage are suitable for interfacing with 12 or 24 V logic.

2.7.4 Open collector drivers

Open collector drivers associated with pull-up resistors play the same part as three-state drivers which will be discussed in detail in the following section. Manufacturers can easily obtain the open collector version of a circuit by removing the upper output transistor on the masks. Open collector and three-state circuits are therefore often described in the same pages of the catalogue. In order to obtain correct performances, the impedances of an open collector bus must be suitably adapted [DelCorso86] [Texas88a]. Simple microprocessor buses only seldom use open collector drivers.

2.7.5 Three-state outputs

Three-state gates are equipped with an extra command input which activates the output of a logic element. This signal *enables* (*selects*) the transfer of the output signal onto the physical output.

This can be used to connect several output signals while only selecting a single signal at a time (figure 2.33). Three-state gates are preferable to open collector gates when dealing with medium performance buses, which need no adaptation resistors as long as they are only short (10 cm).

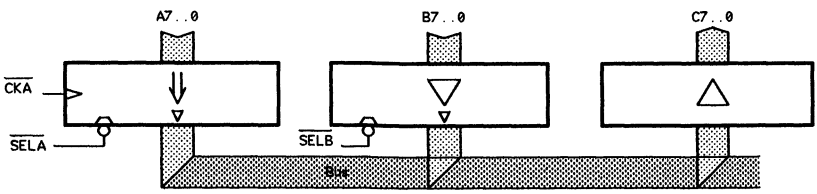


Fig. 2.33 Three-state bus

Isolated three-state gates with independent command inputs can be wired as single state gates (figure 2.34).

Any three-state gate obviously becomes a two-state gate when the transfer authorization input is activated.

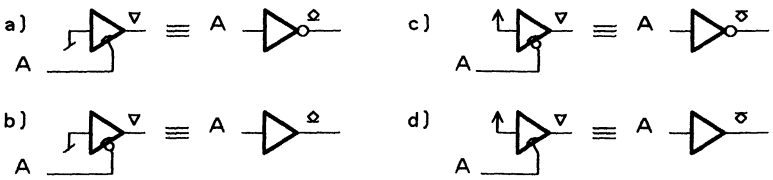


Fig. 2.34 Single state gates made up using three-state gates

2.7.6 Bus termination

If bus length exceeds a dozen centimetres, the end of each bus line must be adapted to the characteristic bus impedance. This adaptation usually determines bus rest potential. Proper adaptation conditions how well the bus works, while ensuring maximum speed and a minimum amount of glitches. Many different techniques are used [DelCorso86], which can trigger endless discussions when defining a bus. Practically, a 20 cm backpanel bus which is connected to fifteen boards offers three solutions. A Thevenin circuit equipped with an equivalent 120Ω resistor is usually used for TTL buses (figure 2.35a). More efficient buses require this resistor to be brought down to 30Ω as well as the addition of special drivers. The ratio between the two resistors defines the voltage at state "1", which is around 3V, and limits the voltage when the output is at "0".

With CMOS circuits, in order to avoid high power consumption at state "0", a capacitor can be connected in series: it does not modify dynamic characteristics (figure 2.35b). The pull-up resistor contributes somewhat towards adapting impendance but its main role is to define the state as long as the drivers are at a high impendance. When dealing with HC-MOS drivers, the equivalent ending resistor must be increased even if adaptation becomes imperfect, since output voltages are insufficient.

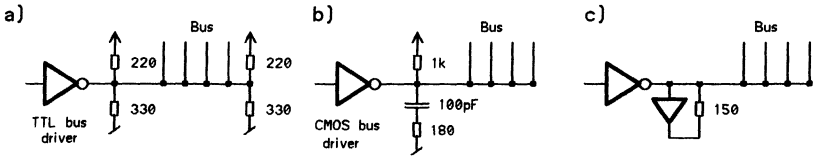


Fig. 2.35 Example of bus termination

An active ending is ideal but rather expensive. The adaptation resistor is linked to an amplifier which follows the voltage on the line (figure 2.35c). Therefore, there is no static current. This solution is often adopted inside integrated circuits, more because it avoids floating buses than because it adapts impendance.

Amplifiers are not strong enough to fight drivers; they memorize the last state of the bus as soon as it stops being active. Designers usually use charge resistors on printed circuits, preferably pull-up ones for TTL circuits.

2.7.7 Three-state buffers and drivers

The various types of three-state drivers are differentiated by their size, function, pinout, output voltage and by their inputs (voltage, Schmitt triggers). Moreover, the defects inherent in certain circuits (sensitivity to parasitic oscillations, poor characteristics as regards high capacitive loads) have encouraged manufacturers to design enhanced versions, which either carry the same number followed by the letter A or a new one. Drivers are generally buffers so that the gates of a bus can be charged with its impendance adaptation resistors (section 7.1.7).

A first category of three-state circuits corresponds to *unidirectional drivers* whose drivers are independent or in clusters. Inverters can be found on the data and on the commands, an AND gate can control the selection and the parity can be computed (figure 2.36a). Functionally identical circuits exist under different numbers because of pinout differences (figure 2.36b,c,d). it is usually easier to draw the printed circuit when transverse pinout is used (figure 2.36c).

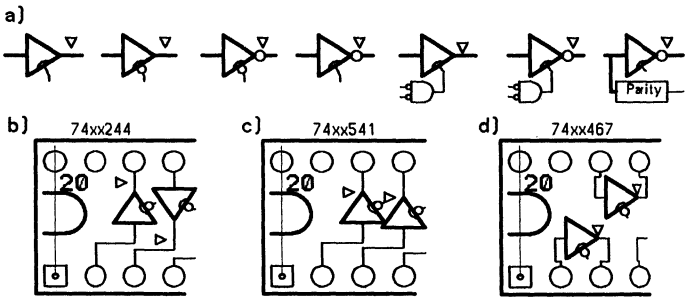


Fig. 2.36 Types of monodirectional drivers and pinout differences

There are some strange mixtures. The 4+4 bit type 74xx241 driver has both direct and inverted command lines for each group; historically, this enabled designers to build 4 bit bidirectional drivers using one of these 4+4 bit drivers before 8 bit bidirectional drivers had been developed (figure 2.37). Another anomaly appears in the 74LS230 circuit which is equipped with a group of 4 direct drivers and a group of 4 inverted ones.

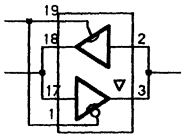


Fig. 2.37 Example of 74xx241 circuit wiring

Bidirectional drivers can be direct, inverted or crossed. They use direct or inverted command variables (figure 2.38) and can include mono or bidirectional parity logic. As for monodirectional drivers, the same functionality is performed using different pinouts.

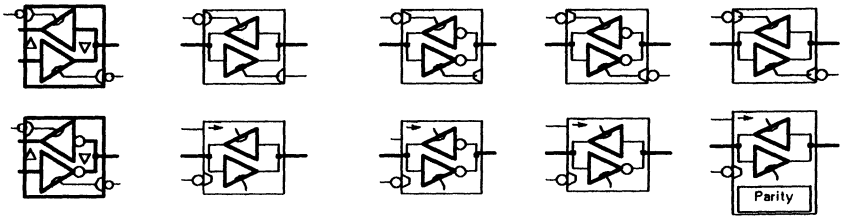


Fig. 2.38 Types of bidirectional drivers

Drivers are sometimes equipped with a pull-up resistor (20kΩ) on their input and a resistor connected in series (25Ω) on their output in order to adapt the impedance in the case of highly capacitive loads like those found in a memory bank.

There are a few *tridirectional drivers* which are rarely used. Several other circuits which will be discussed later (multiplexors, transparent latches) are also used as drivers.

2.7.8 Line drivers

In order to pilot buses with a properly adapted impedance (at least 50Ω), it is necessary to use buffers which can absorb more than 40 mA instead of the 8 mA usually found in TTL-LS. Alternative circuits carrying a -A mark have been developed for such cases. Drivers which are particularly suitable from the electrical and functional point of view are still being developed. Drivers corresponding to the IEEE896 standard (Futurebus) have output levels on the bus side which are not TTL, with a signal amplitude of only 1V and trapezoidal wave fronts. They carry high currents of more than 100 mA, thus guaranteeing a rapid switch-over on very busy lines. A new generation of 16 bit drivers in ACL technology with miniature packages makes interfacing with 32 bit buses easier.

Drivers for the IEEE488 standard are characterized by grouped control lines due to the bus definition.

We will not deal here with RS232 and long line command circuits, which send information through a coaxial or twisted cable while heeding its characteristics.

2.7.9 Analog switches

Analog switches are based on the bidirectional transmission gate of figure 1.4c. They are often used in analog assemblies to change an amplification gain (figure 2.39a) or to let modulated signals flow conditionally from one level to another. A simple normal transistor or one with a field action can also have a logic (figure 2.39b) or analog (figure 2.39c) function. These assemblies are only suitable for low frequencies.

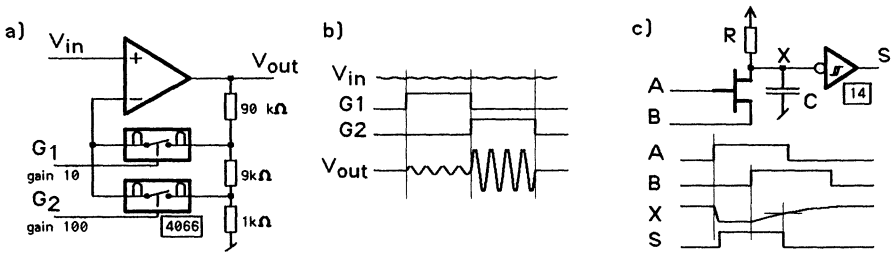


Fig. 2.39 Application examples for an analog switch and FET transistor.

2.8 Switches and encoders

2.8.1 Multiplexors

Multiplexors (*data selectors*) select one of 2^n inputs according to n input selections and possibly to one general activation input (figure 2.40a). Packages with 16 pins are well suited for multiplexors with 2, 4 and 8 inputs.

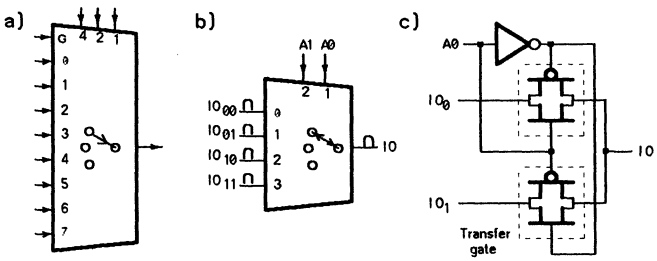


Fig. 2.40 Multiplexor examples

Bidirectional circuits which are both multiplexors and demultiplexors (figure 2.40b) can be developed in CMOS. A bidirectional analog switch can be built (figure 2.40c) using the transfer gates shown in figure 1.3c it can be used in special applications or to send out modulated signals, e.g. along telephone lines.

Multiplexors can easily be cascaded (figure 2.41a). They often have a three-state output which can send information on a bus; a decoder selects the active multiplexor (figure 2.41b). A two position multiplexor is often built using three-state drivers (figure 2.41c). The catalogue also shows multiplexors equipped with a register or latch on the select addresses.

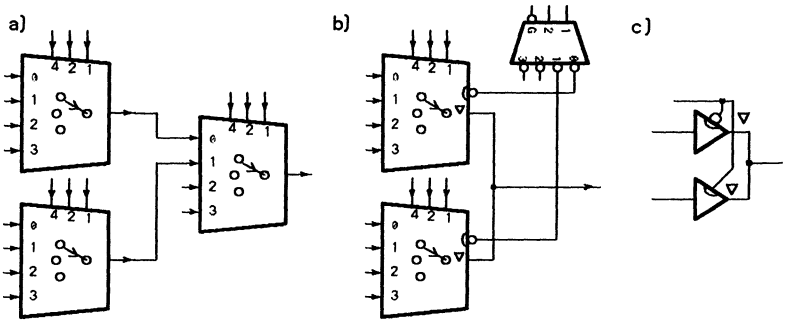


Fig. 2.41 Increasing the number of inputs on a multiplexer

2.8.2 Decoders

Decoders or *demultiplexors* activate one of 2^n outputs corresponding to the code placed on the n inputs (figure 2.42a). An extra enable input activates the output. On some decoders, the polarity of the outputs can be programmed (figure 2.42b).

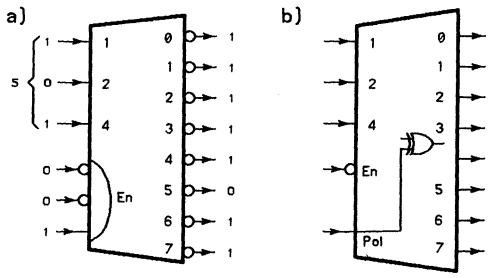


Fig. 2.42 Decoder examples

The decoder catalogue is quite a complete one. Some decoders have a register at the input. Decoders are often made up using PLDs (chapter 5) especially when there is a need for a large number of inputs and relatively few outputs. Decoders are easy to cascade but propagation times (figure 2.43a) are increased. The signal whose delay is most critical (for example in a microprocessor the $\overline{AS}$ or $\overline{RDP}$ signal is more critical than the addresses) must come in on the last level of the decoder (figure 2.43b).

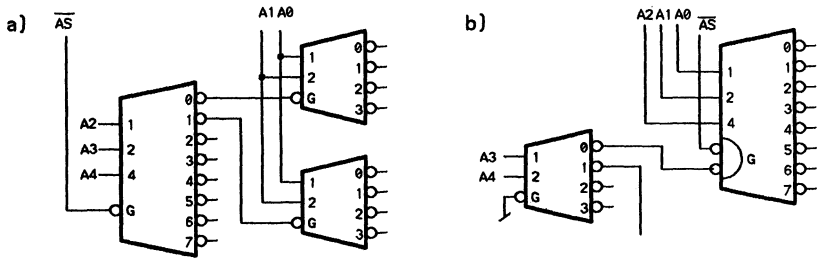


Fig. 2.43 Increasing the number of inputs on a decoder

2.8.3 Display decoders

One-out-of-ten decoders used to be used with lamp or independent filament displays. Seven segment decoders are still sometimes used; they do not all display coherent values for numbers above 9 (figure 2.44). It is possible to force the selection of all segments with an *LT lamp test*. On some decoders, the 0 number is only displayed conditionally to a *Ripple Blanking Input* (RBI). A *Ripple Blanking Output* (RBO) signals that the zero is erased and that therefore the neighbour on the right can also erase its non significant zero. Displays have not been built this way for many years now. Computing and display logic is wired or programmed using either special circuits with a multiplexed display or a simple microprocessor.

The decoder catalogue is far too extensive. Choice is mainly a function of the display technology (voltage, current) and it is wise to follow the display manufacturers' recommendations.

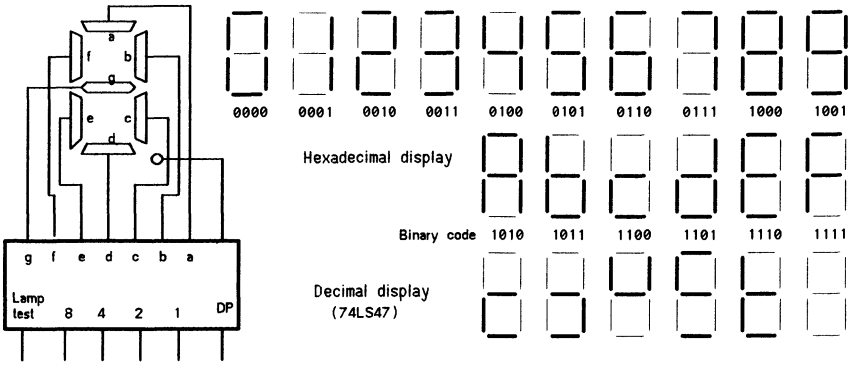
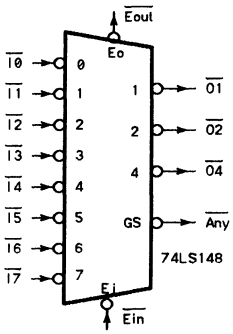


Fig. 2.44 Seven segment decoder display circuit

2.8.4 Priority encoders

A 2^n input encoder sends the binary code of the active input onto n outputs (figure 2.45). Since there may be many active inputs, a priority is wired (in favour of the higher indice inputs), hence their name of *priority encoders*. Several encoders can be cascaded: if the first encoder of the chain has no active input, the Eout output is activated and lets the next circuit work. The manufacturer often sets up a truth table to explain his circuit, as can be seen in figure 2.45, a method which usually does not help to single out the general functionality. It is often necessary to find or to reconstitute a detailed block diagram or even a detailed schematic in order to understand the circuit.



	Entrées									Sorties				
$\overline{Ein}$	$\overline{I0}$	$\overline{I1}$	$\overline{I2}$	$\overline{I3}$	$\overline{I4}$	$\overline{I5}$	$\overline{I6}$	$\overline{I7}$	$\overline{Any}$	$\overline{O4}$	$\overline{O2}$	$\overline{O1}$	$\overline{Eout}$
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	0	1	1	1	1
0	x	0	1	1	1	1	1	1	0	1	1	0	1
0	x	x	0	1	1	1	1	1	0	1	0	1	1
0	x	x	x	0	1	1	1	1	0	1	0	0	1
0	x	x	x	x	0	1	1	1	0	0	1	1	1
0	x	x	x	x	x	0	1	1	0	0	1	0	1
0	x	x	x	x	x	x	0	1	0	0	0	1	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1

Fig. 2.45 Priority encoder and truth table

An input and an output are provided to cascade several encoders: if the first encoder of the chain has no active inputs, Eout is activated and enables next circuit of the chain (figure 2.46). Encoders are used to differentiate between interrupt and bus access requests.

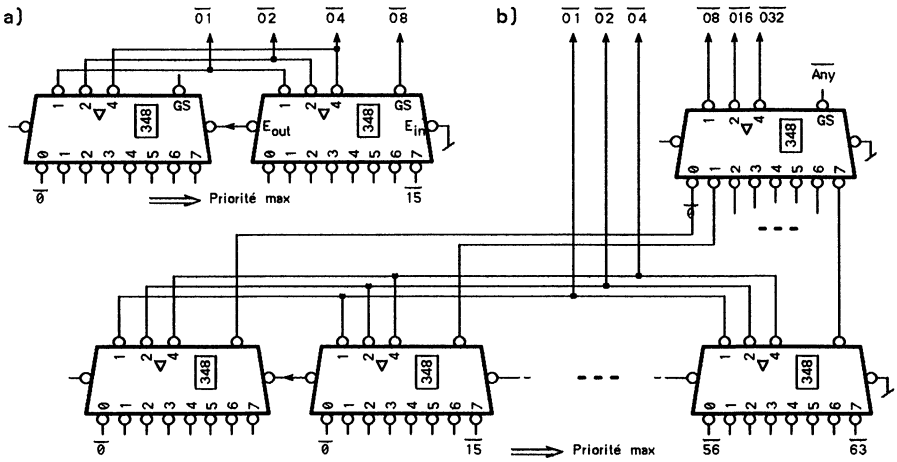


Fig. 2.1 Priority encoder with 16 and 64 inputs

2.8.5 Parity generators

Parity generator circuits can be used when computing the parity of an 8 bit word. They can be cascaded with or without an outside circuit to compute with words of 16 bits or more (figure 2.2). Mono or bidirectional parity drivers are available.

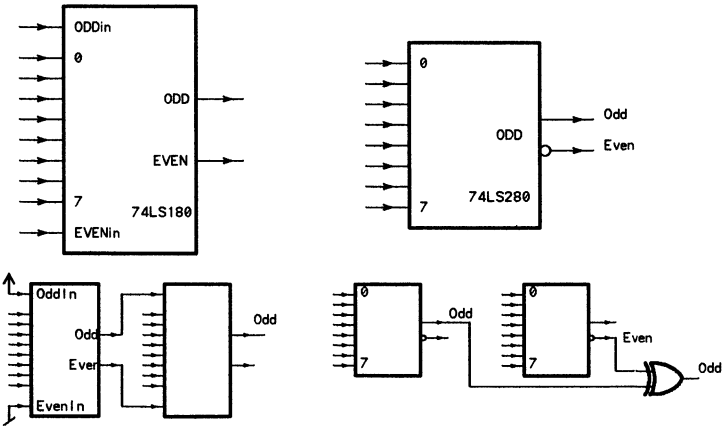


Fig. 2.2 Parity generator circuit and cascading examples

2.8.6 Error detection and correction

Hamming codes [DelCorso86] can be used to detect and to correct errors on a group of lines (generally a memory word). Circuits built for this purpose (*EDAC error detection and correction*) are equipped with a large number of pins since the modular approach used for parity is not possible.

2.9 Comparators and arithmetic circuits

2.9.1 Parallel comparators

Comparators are mainly used to detect equality, e.g. in an address decoder. They are equipped with cascading input-outputs (figure 2.48a,b,c). When dealing with equalities, the order of the sets of bits on the comparators is not relevant; this is not the case when dealing with inequalities where the comparison must start with the least significant bits (figure 2.48d).

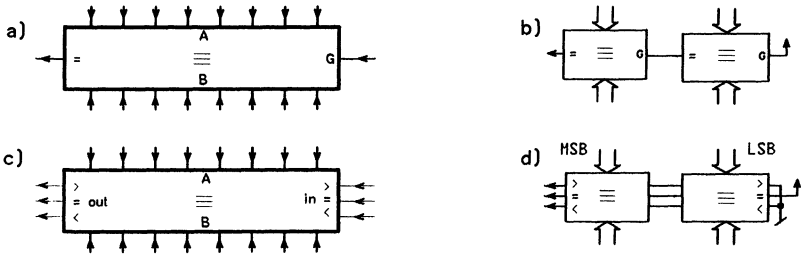


Fig. 2.48 Comparator examples and cascading examples

TTL and CMOS catalogues include a good choice of 4 and 8 bit comparators for equality or inequality. Address comparators have a special encoding which limits their applications; a programmable logic circuits is usually preferred instead (chapter 5).

An equality comparator is based on XNOR (or XAND) and AND gates (figure 2.49a). The AND function can be wired-or (figure 2.49b). The drawback of this schematic is that it switches to state "1", but open collector gates are often found at the output of equality comparators in order to reduce the wiring of multiprecision comparators (figure 2.49c).

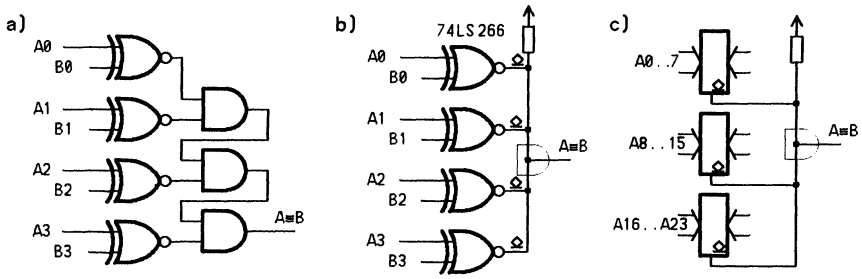


Fig. 2.49 Wired AND comparator

2.9.2 Serial comparison

The equality comparison of two binary words can easily be carried out in series by dealing with the bits in any order. A D flip-flop used as a "one's catcher" (figure 2.50a) or a JK flip-flop (figure 2.50b) is cleared at the beginning of each cycle. The first different bit modifies the state of the flip-flop until the next reset.

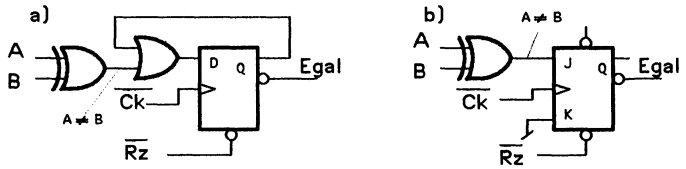


Fig. 2.50 Comparator connected in series

2.9.3 Adders

Adders and *subtractors* exist mainly for binary operations. In BCD the choice is very poor. Adders often have a distinct shape, as in figure 2.51a, and operate on 4 bit words with an input to *carry in* the report of the previous level and a *carry out* report (figure 2.51b).

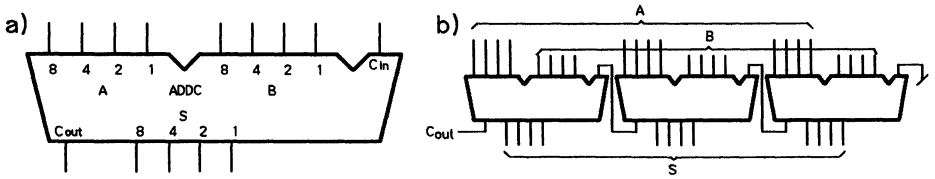
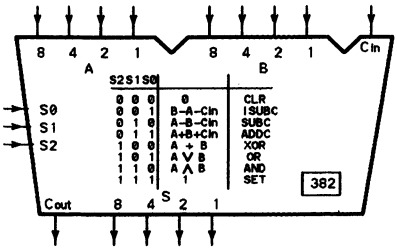


Fig. 2.51 4 and 12 bit adder

2.9.4 Arithmetic and logic units

Arithmetic logic units (ALU) have command inputs which define the function they perform (figure 2.52). ALUs are sometimes documented with inverted inputs and outputs. There is a certain equivalence if one permutes the functions which are performed.



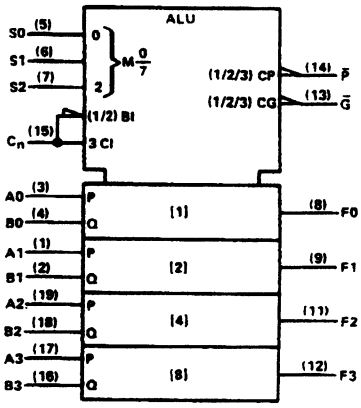


Fig. 2.52 ALU circuit examples

2.9.5 Anticipated report

Operations are slowed down when the carry from one adder is propagated to the next one. It is possible to save time during operations by using *carry look-ahead* circuits which are connected in parallel with the adder and anticipate the build-up of a report. Two signals P (propagate) and G (generate) are created in place of the carry. This increases the parallelism in the report propagation [Muller89]. A 64 bit adder in 74AS technology (8 adders and 2 anticipation circuits) can thus perform an operation in 60ns. The adders included in modern 32 bit microprocessors give better performances.

2.9.6 Multipliers

Cascadable 4 bit *multiplier* circuits in the TTL catalogue have lost all interest. Rapid ECL multipliers (with TTL compatible inputs and outputs) can go up to 16x16 bits with a computation time of 38ns (Am 29516A). Both thermic dissipation and price, however, are great.

2.9.7 Barrel shifters

The most rapid method for shifting a binary word is to use a set of multiple shifters known as *barrel shifters*. They can be made up without too much difficulty using available circuits.

2.9.8 Other circuits

Simple arithmetic circuits are used quite rarely. Microprocessors carry out computations in all applications where speed constraints are reasonable. When making up specialized processors, it is possible to use families of circuits which are available among others from Amd and enable the designer to build very efficient microprogrammed systems. However, when synthesizing VLSI circuits, a good mastery of arithmetic algorithms and circuits is a must [Schwartzlander90].

3 Sequential logic systems

Sequential systems are very rich since they do not only depend on the state of their inputs but also on prior states and on the time interval elapsed since the previous change-over. This section will recall the characteristics of the most simple systems as well as the basic principles of the synthesis methods described in more detail in specialized books [Lewin85] [Fletcher80].

3.1 Asynchronous sequential systems

Pure asynchronous systems change constantly according to their inputs. The variation of one input can therefore modify the state of a series of outputs and lead to new output modifications, etc.. Only simple asynchronous systems can be mastered, due to their complex transient mode.

3.1.1 Oscillators

The most simple sequential system is an inverter which is closed in on itself (figure 3.1). Such a system is an unstable one but only oscillates when a delay occurs in the retroaction loop, e.g. one created by an even number of inverters connected in series.

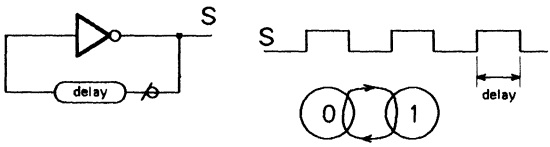


Fig. 3.1 Oscillator

3.1.2 Bistable flip-flops

Two inverters closed in on themselves form a bistable element known as a *flip-flop* (figure 3.2a). If output short-circuits are not used, additional inputs are required to control the change-over (figure 6.5d). In figure 3.2b, an input which is active, even temporarily, causes a change-over to its side. The two outputs are active when both the *set* signal S (or *preset*) and the *reset* signal R (or *clear*) are used (figure 3.2c). This condition is sometimes illegal but does not indicate faulty working conditions.

Faulty working conditions are caused by a critical sequence whose evolution is unforseeable, e.g. if the set pulse is too short or follows the reset pulse too closely. The times shown in figure 3.2d have minimum values which must be respected in order to guarantee good working conditions. The *static SR flip-flop* is often made up using two NAND gates; in such a case, the inputs are active low (figure 3.2e).

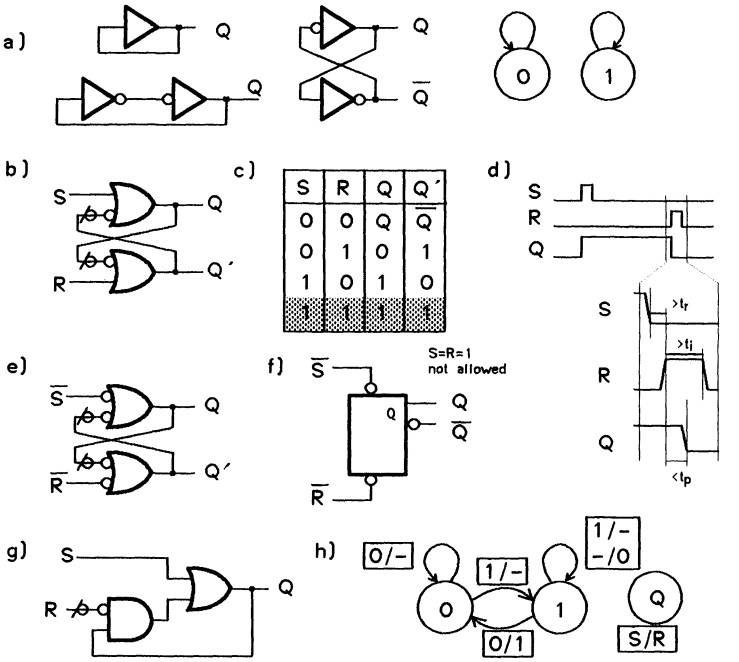


Fig. 3.2 Bistable flip-flops

Excluding the condition S = R = 1 (or $\overline{S}$=$\overline{R}$=0 if the inputs are active low), the relation Q' = $\overline{Q}$ is always true, i.e. both the flip-flop outputs are complementary. A single signal is associated with the output and the representation of figure 3.2f is preferable.

Figure 3.2g shows yet another equivalent representation of the flip-flop in figure 3.2b and highlights the fact that the active S input forces the Q output to 1 and that inactive R enables this "1" to be memorized. Figure 3.3h shows the transition graph of this flip-flop: the circles contain the state and the rectangles the transition conditions.

A frequent application of SR flip-flops is removing contact bounce. In this case, it is necessary to use a toggle switch (figure 3.3). Other solutions are mentioned in section 6.1.4. Another application of the SR flip-flop is the "one's catcher" function (section 3.2.16).

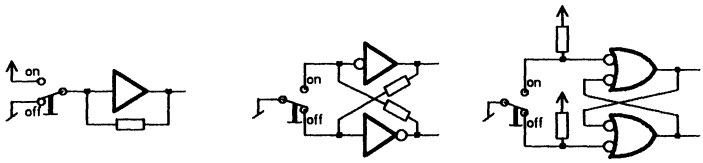
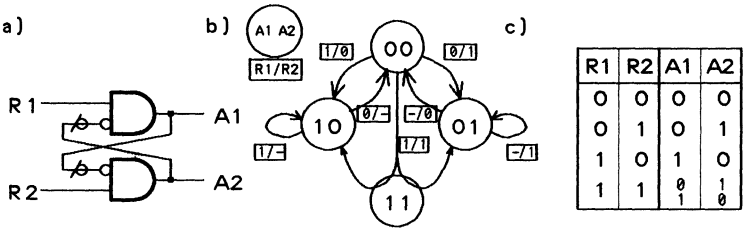


Fig. 3.3 Debounced switches

Catalogues of standard integrated circuits do not contain SR flip-flops, except for a few old circuits which are seldom used. However, the SR flip-flop is a basic element of standard libraries both for gate arrays and standard cells.

3.1.3 Elementary arbiters

The *anticoincidence flip-flop* or *elementary arbiter* (figure 3.4) accepts only one of two requests which cannot be serviced simultaneously. This flop-flop is not really different from an SR flip-flop, used from its illegal state.



R1	R2	A1	A2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0/1	1/0

Fig. 3.4 Elementary arbiter

When at rest (R1 and R2 inactive) there is no request and both outputs are at zero. The first active request (R1 or R2 active) is accepted (A1 or A2 activated) and stops the second request being serviced as long as it is active. The diagram in figure 3.4b shows this evolution: the particular case of a 1/1 transition is described in the following section. As a rule, state 11 is impossible and the name of the circuit is due to the fact that only one output is active in a stable manner.

3.1.4 Metastable states

An interesting problem can occur with the arbiter in figure 3.4 when two requests are perfectly simultaneous. The circuit hesitates for a time before stabilizing in one or other of the two stable states (figure 3.5). This calls to mind a marble sent towards a hill without sufficient momentum (figure 3.6) and which can therefore dither for an indeterminate length of time at the top of the hill. The life of the *metastable state* follows a statistical law. Times lasting several dozen microseconds have been observed with TTL circuits.

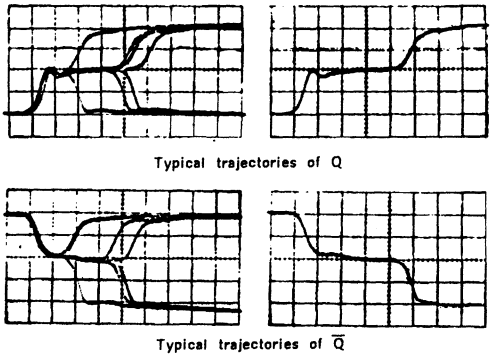


Fig. 3.5 Metastable state

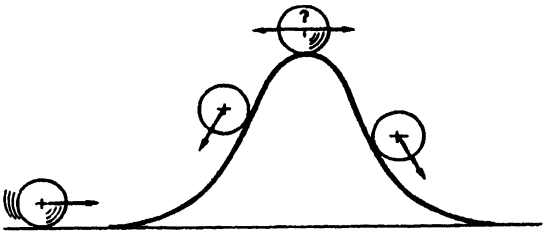


Fig. 3.6 Analogy

By adding analog circuits, overly long metastable states can be detected and the output action delayed (figure 3.7). An analog comparator inhibits the outputs as long as the intermediate states are not correct. The delays are necessary for this comparator to start acting. Metastability is an important problem which will be discussed further in section 3.2.7.

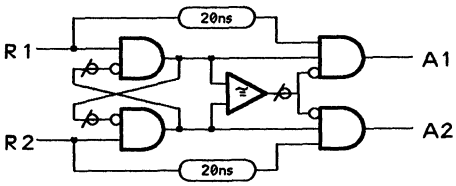


Fig. 3.7 Arbiter devoid of metastable states

3.1.5 Latches

A *latch* is a flip-flop equipped with two gates for copying the input signal onto the output when the *load* input is active (figure 3.8). This input must not be called a "clock" because it is static: as long as LD is active, input D is copied onto to output Q. Good working conditions for a latch require the LD load pulse to be sufficiently long and the data to be stable when the latch "closes" (figure 3.8e).

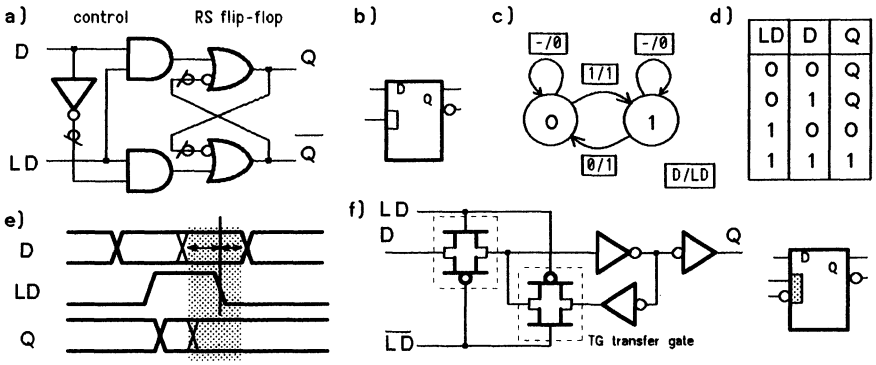


Fig. 3.8 Latch

At the VLSI level, latches are frequently built using the transfer gates seen in figure 1.4c because of the smaller area required (figure 3.8f). The clock inversion is frequently implemented for several latches simultaneously and sometimes includes non overlapping conditions (section 3.2.3).

3.1.6 Addressable latches

An addressable latch is made up of a series of flip-flops (generally latches) whose D inputs are connected together. The load pulse is directed towards one of the flip-flops (figure 3.9a). Figure 3.9b shows the symbol adopted throughout of this book, and figure 3.9c is the AIEE symbol (section 3.8.2).

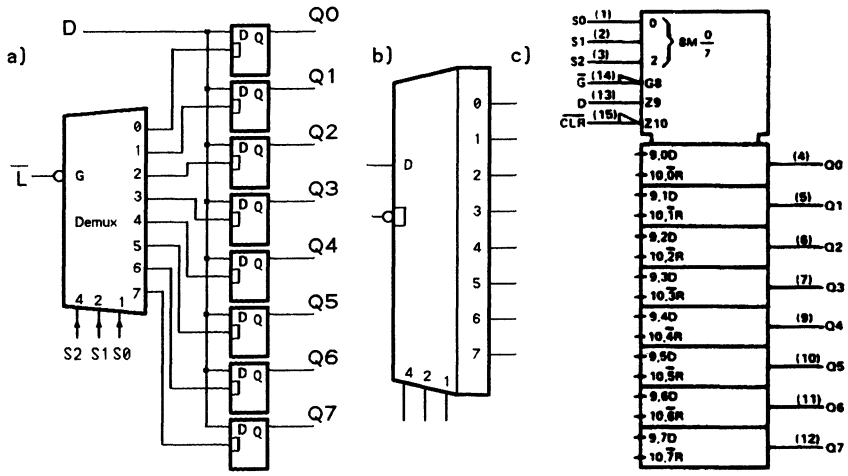


Fig. 3.9 Structure and symbol of an 8 output addressable latch

For example, in order to control lamps or motors in a microprocessor interface, addressable latches are usually more practical than registers since each flip-flop is modified without taking the others into account (section 8.5.6).

3.1.7 Designing asynchronous systems

Theoretically, any sequential system can be made up using interconnected gates. However, there is no simple methodology for *pure asynchronous systems* built with logic gates. JK and D flip-flops (figure 3.10) are the most complex circuits developed with these methods and optimized in accordance with internal propagation times. These flip-flops will be discussed in section 3.2.

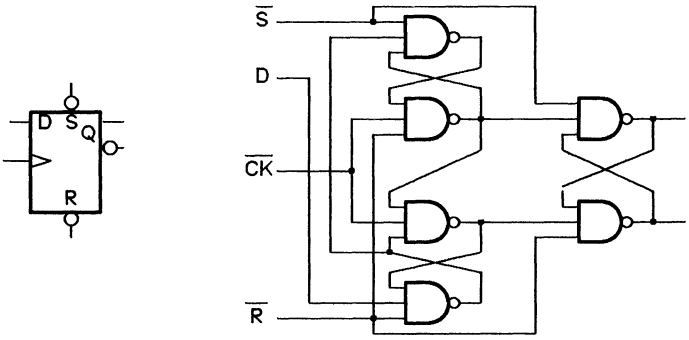


Fig. 3.10 Internal schematic for a D flip-flop (74LS74)

3.1.8 Static glitches

Dispersion of propagation times and glitches make asynchronous systems hard to design. For example, the design of a latch using direct synthesis instead of the intuitive solution presented in figure 3.8, leads to the tables and equations of figure 3.11a. When implementation is based on gates, reliability depends on the position of the inverters and on delay values. In the case of figure 3.11b, the operation will probably fail if the inverter is about twice as slow as the gates. The operation will be correct in the case of figure 3.11c, which would have been the most logical schematic if the specifications had required a $\overline{LD}$ input signal. Nevertheless, this solution is unsatisfactory because of the 4 gates delay from the LD input to the Q output.

The solution of figure 3.11d has been documented as a way to build a faster latch than those included in the "74" catalogues. It seems to work, but the operation can only be guaranteed by the manufacturer, who knows exactly what all the internal delays are.

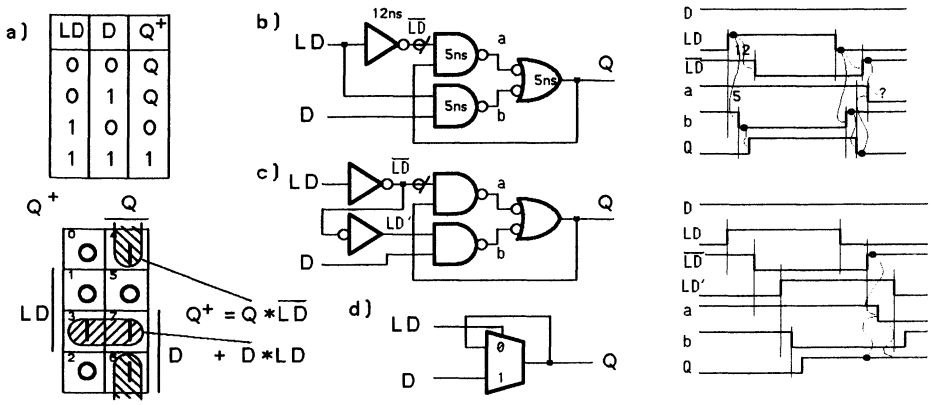


Fig. 3.11 Latch synthesis and diagram

As regards asynchronous systems, these *static* race conditions can be avoided by studying the Karnaugh map. The two simplified terms Q*$\overline{LD}$ and D*LD do not overlap. When changing from one term to the other, there may be some discontinuity. By adding a *redundant covering term* to the minimal solution provided by the usual simplification method (half toned in figure 3.12), the operation becomes reliable. That solution is preferable to the one in figure 3.11c.

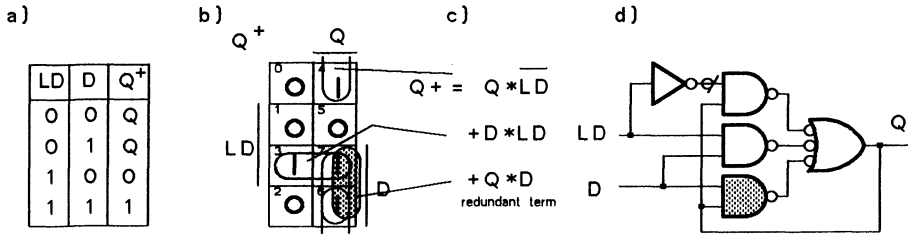


Fig. 3.12 Latch synthesis without race

3.1.9 Example

Pure asynchronous logic is often encountered as a simple retroaction. The example of figure 3.13a shows an error locking mechanism using the SR flip-flop of figure 3.2g. Memorizing the selection of an address in a processor with multiplexed data/addresses uses AND decoding conditions, the latch shown in figure 3.12 as well as an AND gate to validate the selection (figure 3.13b). This type of schematic is often used in programmable logic (chapter 5).

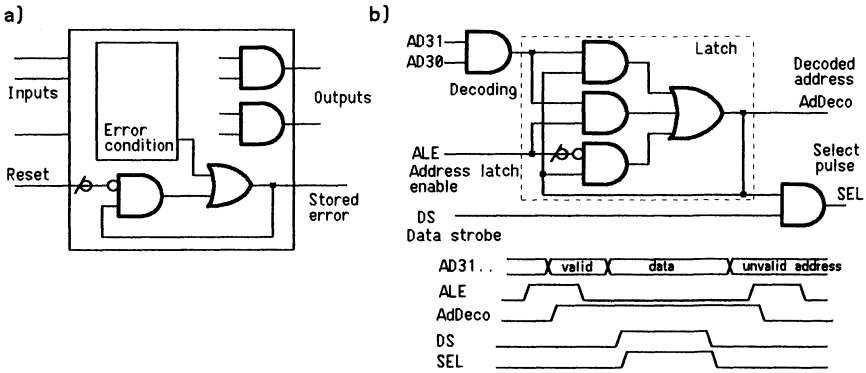


Fig. 3.13 Asynchronous sequential logic

3.2 Synchronous flip-flops

Synchronous sequential systems evolve at specific moments in time, in phase with a common signal known as the clock pulse.

3.2.1 The clock pulse

The *clock pulse* (CK or CP) is a signal whose positive (or negative) edges are active; these edges must be quite steep, i.e. much shorter than the evolution time of the signals they modify. A small triangle characterizes inputs which are edge triggered. Flip-flops with a clock edge are known as dynamic flip-flops, as opposed to latches which are static.

All flip-flops and registers are not triggered by the same pulse edge. An active falling edge is highlighted by an inverting circle before the triangle (figure 3.14a). An active rising edge carries no inverting circle (figure 3.14b).

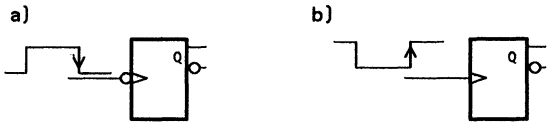


Fig. 3.14 Edge convention

3.2.2 Note

The convention used for dynamic inputs is not a very satisfactory one. It would have been more natural to consider that a clock pulse is positive and has a falling active edge (figure 3.14a) and not add an inverting circle in this case. In order to be convinced, one only needs to cascade counters (figure 3.15) or to try to replace a latch by a register in a microprocessor interface. Therefore, the direct name will be used in this book to refer to positive clock pulses, linked to a clock input preceded by a circle, as shown

in figure 3.15. For this reason, it is normal that the link between output C3 and the divide by 16 counter carries no pseudo-inverting circle.

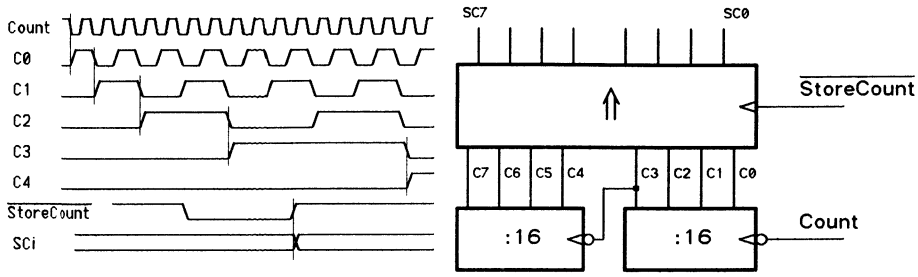


Fig. 3.15 Examples of dynamic name signals

The sequential system of figure 3.15 is known as a *quasi-synchronous* one (and sometimes wrongly as an asynchronous one) because the various clock pulses are not synchronous with each other.

3.2.3 Multi-phase clock

Depending on the semiconductor technology, it is frequently cheaper to design a synchronous system in which several overlapping clocks are included [Annaratone86]. The symbol for a flip-flop using two inverted but non-overlapping clocks is given in figure 3.16a. Out of phase signals are generated on the basis of the schematics of figure 3.16b to d, allowing for technological constraints.

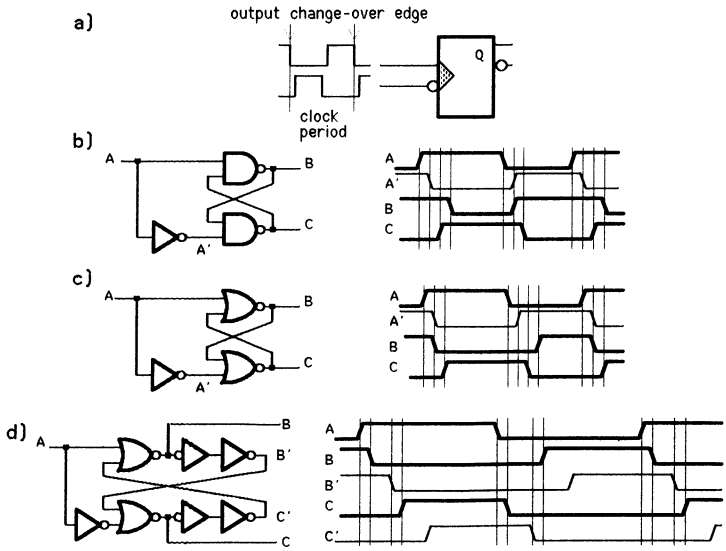


Fig. 3.16 Flip-flop with two-phase clock and out of phase signal generation

3.2.4 D flip-flops

D flip-flops take the state of input D during the clock pulse. In common flip-flops, input D is matched to the rising edge of the clock (figure 3.17).

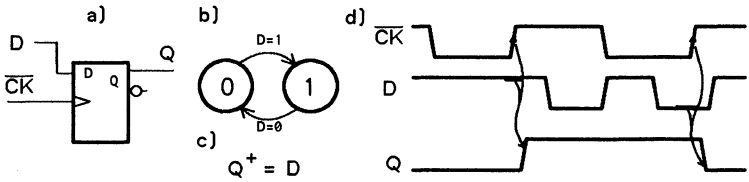


Fig. 3.17 Characteristics of D flip-flops

Static set and reset inputs have priority over clock and data inputs (figure 3.18).

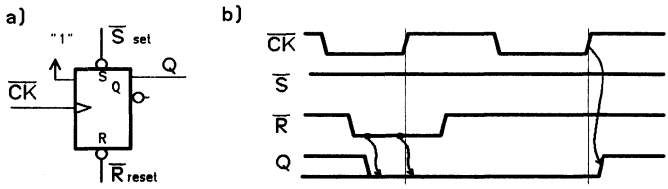


Fig. 3.18 Priority effect of static S and R signals

A D-type flip-flop can be built with two latches controlled by inverted and non-inverted clocks (figure 3.19). VLSI designers frequently use this type of schematic, based on latches made with transfer gates (figure 3.8f). The other approach is based on optimized flip-flops (figure 3.10) is

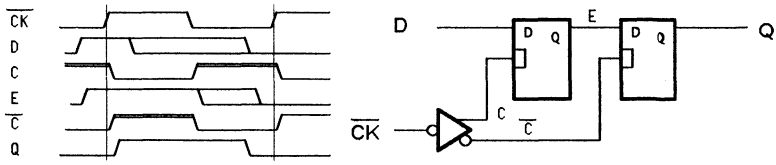


Fig. 3.19 D-flip-flop assembled with two latches

3.2.5 Output conventions

A flip-flop is often perceived as having two states Q and Q' whose values are complementary. For example, the same flip-flop can generate the "Read" and the "Write" signal. The symbol of figure 3.20a is coherent. If the flip-flop is considered as being linked with an output variable, be it a true or a false one, it is preferable to use the symbol of figure 3.20b, i.e. the one generally adopted in this book.

Strictly speaking, when the flip-flop is used to memorize the inverted variable, the symbol of figure 3.20c should be used. When studying 3.20b,c it is important to note that the reset and set pin numbers have been swapped. Therefore, this symbol is often the cause of mistakes and will not be used in this book.

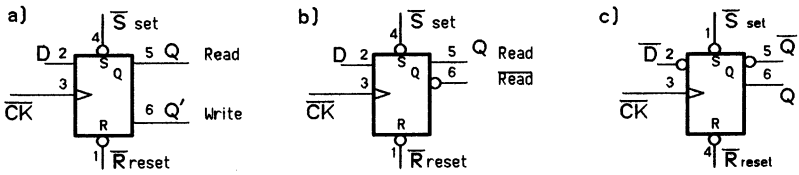


Fig. 3.20 Symbols representing the same D flip-flop (half a 74xx74 circuit)

3.2.6 Working conditions

Correct working conditions of D and JK flip-flops are only ensured if the length of the clock pulse, the *set-up time* t_s and *hold time* t_h are longer than given minimum values (figure 3.21). If these times are not respected, metastability can occur and *propagation time* t_p can no longer be guaranteed.

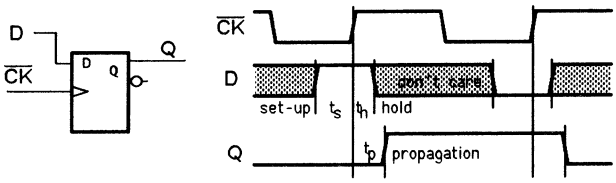


Fig. 3.21 Good working conditions for D flip-flops

The manufacturer usually guarantees that propagation time t_p is longer than hold time t_h. This is a necessary condition for cascading two flip-flops with a common clock, e.g. in a shift register.

3.2.7 Metastability probability

When set-up and hold times t_s and t_h are not respected, *metastability* can occur [Marino81] [Wakerly90]. Metastable states occur in a relatively narrow zone (figure 3.22), whose position is a function of fine technological parameters which the manufacturer cannot document.

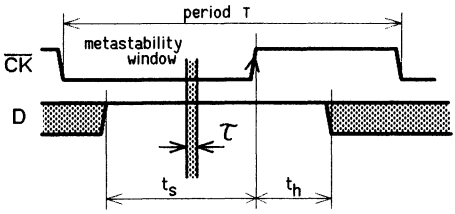


Fig. 3.22 Metastability zone

If the width of this zone is τ, the clock frequency f_c and the frequency of the sampled signal f_d, the *MTBM (Mean Time Between Metastables)* lasting longer than δt is

$$\mathrm{MTBM}(\delta t) = \frac{1}{f_c \cdot f_d \cdot C} \cdot e^{\frac{\delta t}{\tau}}$$

Parameters C and τ are determined experimentally and are only very partially documented by manufacturers of standard circuits, PLDs and gate arrays. They are vital for calculating the average mean time between failures (MTBF). It is therefore difficult to carry out a serious analysis.

As a rule, PLDs appear to be rather sensitive, much more so than fast logic families (F, AS). For the flip-flops of a fast PLD (16R8D), the value of $C = 0.27\ \mu s^{-1}$ and that of $\tau = 1.0$ns. The parallel register 74AS374 is given with $C = 6.5\mu s$ and $\tau = 0.34$ns. The graph of figure 3.23 shows that, every 10 seconds, an LS circuit which samples 500 kHz data at the rate of 1 MHz may go into a metastable state for 40 ns. However, a state lasting 70 ns would only occur every 10 years.

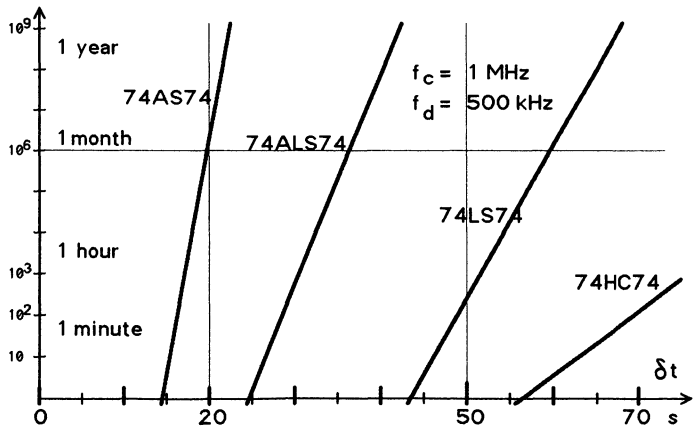


Fig. 3.23 Metastable characteristics of Texas Instruments 74xx74

3.2.8 Double synchronization

In order to reduce metastability probability, it is necessary to carry out a *double synchronization* (figure 3.24). The first X flip-flop can often fall into a metastable state but the probability that one of these states should last longer than the clock period (minus set-up and hold time) is minute and can be reduced even more by increasing the clock period. From a practical point of view, if the frequency of CK is lower than 10MHz and the synchronization flip-flops are of the 74F or 74AS types, it is possible to say that the probability of a breakdown due to a metastable state is less than that due to component defects. Recent circuits contain a double row of registers in order to simplify implementation.

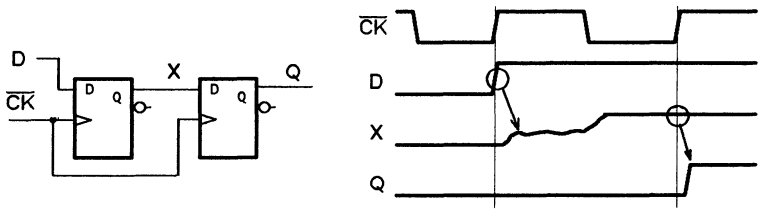


Fig. 3.24 Synchronization flip-flops

3.2.9 Standard D flip-flops

The catalogue of D flip-flops is a very poor one. The 74xx74 is the only one in the "TTL" family. In CMOS, the old 4013 circuit has non inverted S and R inputs. Catalogues of standard cells or gate arrays contain many variations on the D flip-flop, with or without static reset, output buffers, etc.. Synchronous systems generally use the D flip-flops found in parallel registers (section 3.4) and in programmable logic circuits (section 5.3).

3.2.10 JK flip-flops

The future state of a *JK flip-flop* depends on the J and K inputs (figure 3.25). The exact behaviour can be expressed through a graph (figure 3.25b) or an equation (figure 3.25c). The intuitive evolution rule is easy to remember: a JK flip-flop can only switch over to state one if J=1 and switch over to 0 if K=1. If J=K=1, it switches over at each active clock edge (divider by 2).

There are three main functionalities for JK flip-flops, which in fact apply to any type of flip-flop but are only really found in JK models.

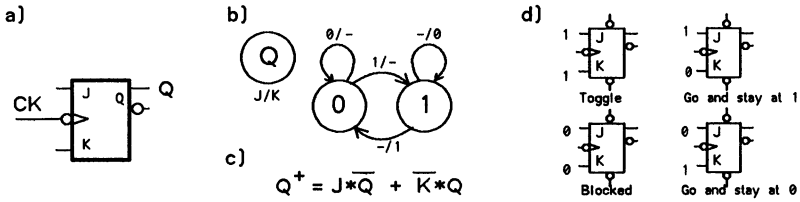


Fig. 3.25 Characteristics of JK flip-flops

Normal flip-flops are *edge triggered* by their clock input, usually on the falling edge (figure 3.26a). If propagation and hold times are respected, the next state depends on the state of JK at the active edge and the output evolves immediately, allowing for propagation time. The D flip-flop discussed in section 3.2.4 behaves in such a manner.

Another type of JK flip-flop is *pulse triggered*. While the clock is active, an internal latch memorizes the input states. The evolution on the falling clock edge depends on this memorized value (figure 3.26b). These pulse triggered flip-flops should be avoided, as should the up/down counters including them.

The third type of flip-flop, known as a *data lockout*, is made up internally of two cascaded edge triggered flip-flops respectively triggered by the rising and the falling edge. Inputs are therefore sampled on the rising edge and copied onto the output on the falling edge (figure 3.26c).

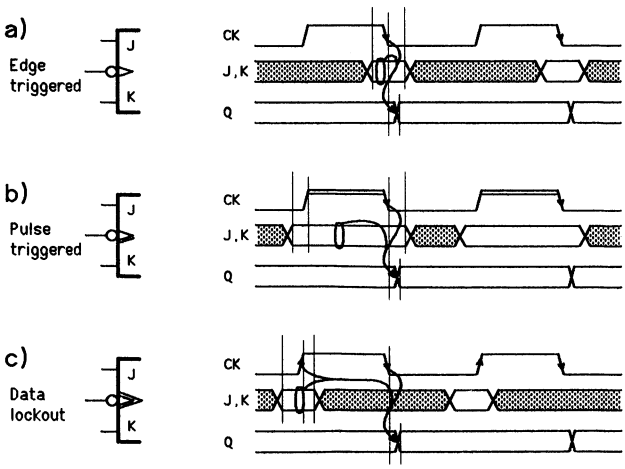


Fig. 3.26 Types of JK flip-flops

Data lockout flip-flops are useful in systems where flip-flops are far apart and, consequently, propagation times are great. Differences in propagation time between clock and data cannot be controlled. Incorrect operations may result if data is delayed less than the clock used for sampling (figure 3.27). With a data lockout flip-flop, the difference between delays or *skew* can equal half a cycle, less set-up and propagation times.

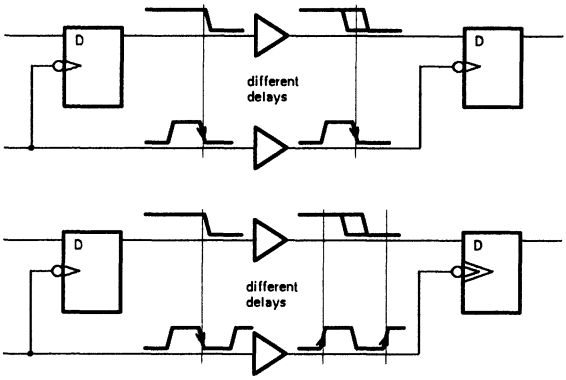


Fig. 3.27 Differences regarding propagation times

3.2.11 Standard JK flip-flops

Many JK flip-flops are available. Sometimes, the same number within different logic families corresponds to different clock functionalities. Manufacturers' symbols do not show up these functional differences explicitly.

3.2.12 Dynamic SR flip-flops

Dynamic SR flip-flops, unlike the static SR flip-flop of section 3.1.2, are frequently found in books on logic systems and are sometimes found in VLSI libraries and PLDs. Dynamic SR flip-flops are identical to JK flip-flops, although state S=R=1 is not allowed, which saves several transistors.

3.2.13 T flip-flops

T flip-flops are simple dividers by two and are sometimes equipped with a T input which enables toggling (figure 3.28). TTL catalogues do not include T flip-flops since they can easily be made up using a D or JK flip-flop (figure 3.29). On the other hand, T flip-flops are found in VLSI libraries where it is always useful to be able to save a few transistors.

When implementing binary synchronous counters, a T or G or Cin enable input is required, as well as a Cout (*carry out*) output. Cout is the logic AND between input Cin and output Q (figure 3.28b) and can be used to build synchronous divide by 2^n counters: the solution of figure 3.28c is slower than that of figure 3.28d, which computes the reports in parallel. In all cases, glitches are found on Cout reports.

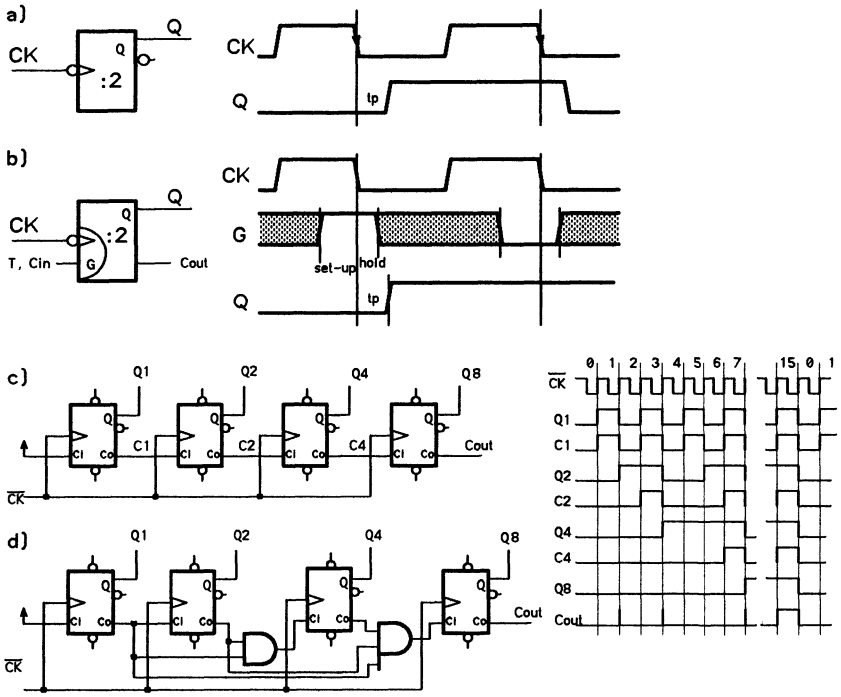


Fig. 3.28 T flip-flop and synchronous binary counter

3.2.14 Standard T flip-flops

All T flip-flops inside the "74" family are associated with binary counters. VLSI libraries include them as building blocks for designing counters of various sizes and speeds.

3.2.15 Universality of D and JK flip-flops

It is more or less easy to replace a D flip-flop by a JK one and a T flip-flop by a D or JK type. Figure 3.29 shows the corresponding diagrams. It is also possible to replace a D or JK flip-flop by a T one, but such diagrams lose all interest because of the relatively complicated logic required and the fact that T flip-flops are only rarely used.

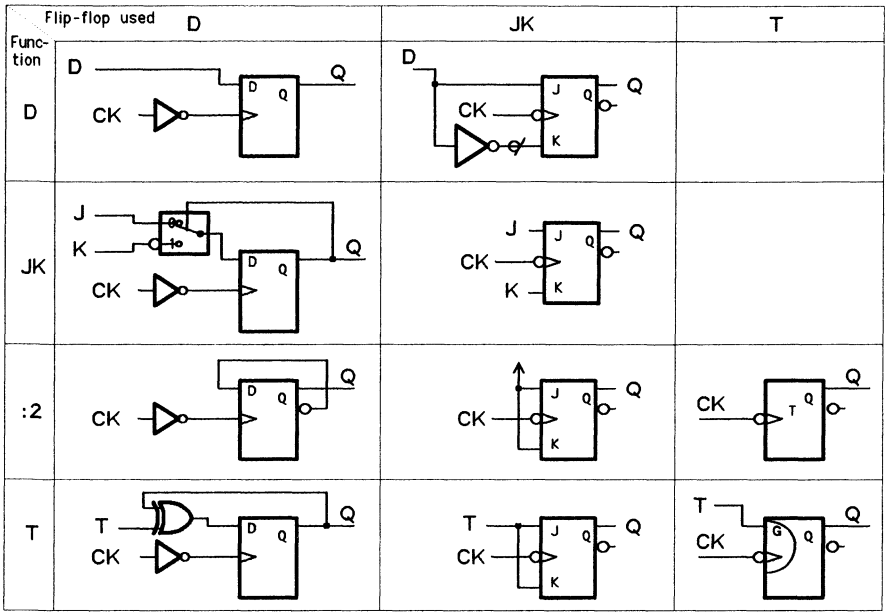


Fig. 3.29 Equivalents

3.2.16 One's catchers

As indicated by its name, a *one's catcher* circuit switches over as soon as there is a "1" state at its outputs. The action can be a static one (figure 3.30a): a brief pulse is all that is needed to switch over until the next reset. The action is a dynamic one if it is sampled by a clock edge. The solutions of figure 3.30b and c have already been discussed in the case of the comparator connected in series shown in figure 2.50.

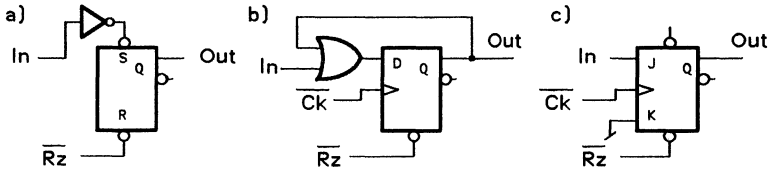


Fig. 3.30 Static and dynamic one's catchers

3.2.17 Double flip-flops

Double flip-flops have two dynamic inputs which independently set the output at one or at zero (figure 3.31a,b). This essential functional element can be made up using two simple flip-flops (figure 3.31c,d). When the two control signals are never simultaneously active, the solution given in figure 3.31e is applicable, but is only found at the VLSI level.

Double flip-flops are not found in TTL and HC-MOS catalogues but are starting to appear in some PLDs (Amd22IP6). Double flip-flops are very important functional elements, albeit little known ones. They can help to clarify documentation in the case of block diagrams (figure 8.27).

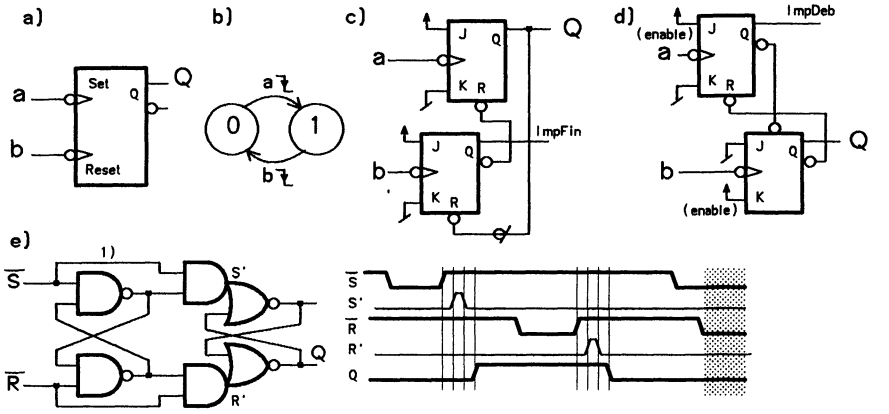


Fig. 3.31 Double flip-flop and schematic based on JK flip-flops and gates

3.2.18 Pinpointing an operation

An operation is characterized by a signal indicating its start and another indicating its end. These signals can be states (maintained ones or brief pulses) or edges (figure 3.32).

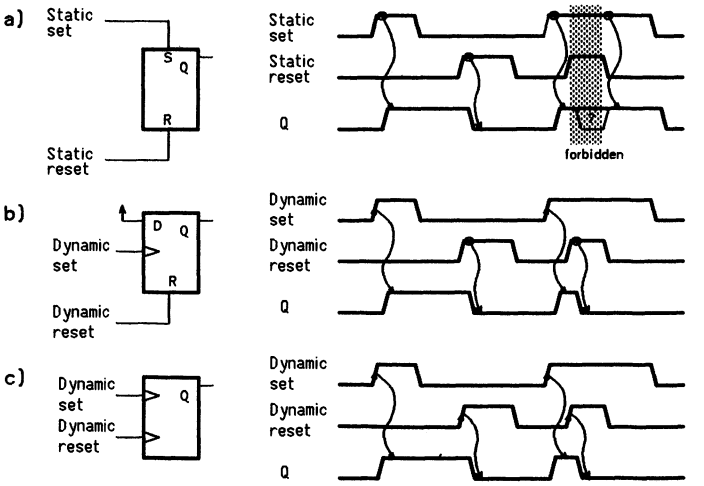


Fig. 3.32 Examples of elementary types of sequencing

Static flip-flops are suitable when the set and reset operations are static and never active simultaneously (figure 3.32a).

If one of the actions is a dynamic one and the other a static one, a D flip-flop (or JK with J=1 and K=0) is suitable (figure 3.32b). One application is to move a carriage forward, dynamically starting it with a push button and stopping it statically at the end of its run with a limit switch. Since the switch has full priority, there is no risk of restarting the motor with the push button once the limit has been reached.

When both actions are dynamic ones (figure 3.32c) a double flip-flop is necessary.

3.2.19 Cycle samplers

Cycle samplers generate an output signal equal to one cycle when faced with the edge of an input signal. Two flip-flops are required (figure 3.33a). Flip-flop X memorizes the request until the beginning of the next period (figure 3.33b). As soon that request is satisfied, by activating Out at the next clock edge, X is cleared. Signal "In" loses one active edge if the cycle is not finished. The timing diagram of figure 3.33b and the state diagram of figure 3.33c describe this behaviour.

If signal In is asynchronous when compared to CK, there is a risk that a metastable state may occur if X is activated while the clock is sampling it. A synchronization flip-flop XS is required in order to bring down to an acceptable level the probability of these metastable states (figure 3.33d).

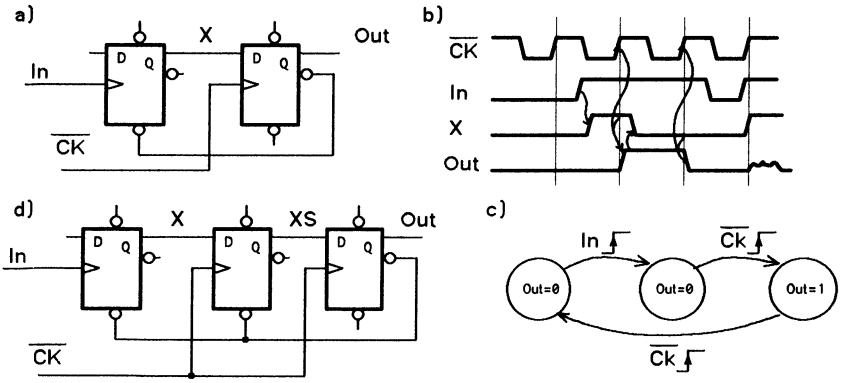


Fig. 3.33 Cycle sampler

3.2.20 Pulse sampler

Sampling a pulse (half a cycle) instead of a full one requires an additional AND gate (figure 3.34a). This does not cause a problem if the active part of the clock is taken. A glitch will usually occur if the first part of the clock is taken (figure 3.34b). The cure is to re-synchronize using a higher frequency signal, or to shift the inverter and check delays, as discussed in figure 3.1.8.

A little known solution, which is suitable for VLSI and sometimes for PLDs, uses three gates (figure 3.34c). However, this means that the signal followed by half a period must be triggered by the clock, with a delay much lower than half a period (figure 3.34d).

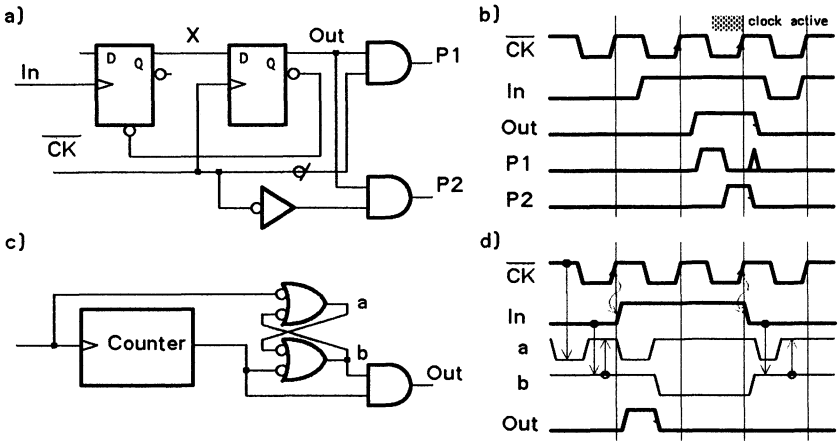


Fig. 3.34 Pulse sampler

3.3 Synthesis of sequential systems

A few comments about asynchronous systems have already appeared in section 3.1.7. The present section only deals with a few aspects of sequential systems. Systematic methods are discussed at length in books on logic systems and must be guided by the designer's experience.

3.3.1 Synthesis of a semi-synchronous system

With a little experience, it becomes easy to synthesize a sequential system, even a very complex one; this is done by interconnecting elementary blocs (counters, registers) with simple sequencers which pinpoint an elementary operation thanks to a flip-flop (figure 3.32). A number of diagrams can result from this approach where the designer's flair can come up with very efficient solutions but where a large amount of experience is required in order to avoid bad working conditions. Several examples of this commonly used method can be found in chapters 5 and 8.

3.3.2 Synthesis of a synchronous system

Synchronous systems follow a simple evolution rule. This makes it possible to opt for a systematic synthesis method, which is explained in great detail in all books on logic systems [Lewin85] [Fletcher80]. This section briefly runs through the principles of the method.

3.3.3 Principles of the method

In a synchronous system, all flip-flops which characterize the state of the system are triggered by the same clock and evolve in perfect synchronism, allowing for limited clock skew and propagation times. The next state of each flip-flop depends on the system's inputs and on the state of all the flip-flops (figure 3.35a). Flip-flops are often grouped in a parallel register; n flip-flops are required in order to encode a 2^n state system. The next state is loaded on the next active clock edge and it is important to respect set up and hold times.

External and internal outputs can be taken as outputs of the combinational system or of the register. In the first case, the input switch-over will modify the output immediately (after one propagation time). In the second case, switch over is synchronized by the clock. Input states cannot change anytime. They must be doubly synchronized so that the next state is stable at the clock edge (figure 3.35b). Double synchronization is not required for internal inputs indirectly resulting from the outputs of the combinational logic or from the register, if phase relationships are known.

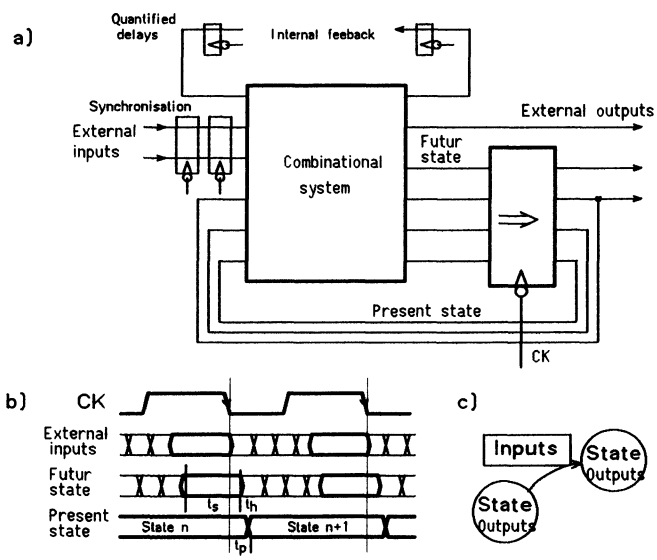


Fig. 3.35 General sequential system

The evolution of the system can be shown quite clearly in a state graph (figure 3.35c): states are circled and input functions which trigger an evolution towards one or other state are framed (the frame is sometimes omitted to increase legibility).

3.3.4 Conclusion

Comparing purely synchronous or semi synchronous methods, whether generally or based on a given example, is a futile exercise. Pure synchronous synthesis are always either limited to five or six variables or based on a repetitive structure. Complex systems are always semi synchronous ones. It is up to the designer to decompose intelligently and put the emphasis on good working conditions, which are a consequence of the delays introduced by the decomposition and the technology used.

3.4 Complex sequential circuits

The catalogue of sequential integrated circuits is as rich and confusing as that of combinational circuits. Besides various alternatives for pining as well as input and output inversion, there are subtle differences regarding clocks and reset.

3.4.1 Standard sequential functions

Registers can be parallel (figure 3.36a) or serial (figure 3.36c). Parallel dynamic registers consist of D-type flip-flops equipped with a common clock. Latches often serve the same purpose (store a state), but they load information on a logic state, and not on a clock edge.

An addressable register loads the addressed output with the value, without modifying the other outputs. Catalogues only contain addressable latches (figure 3.36b), but there is no reason to prohibit the use of synchronous addressable registers.

Serial In Serial Out (*SISO*) registers are the most simple ones (figure 3.36c). Serial In Parallel Out (*SIPO*) registers sometimes carry a parallel register on their outputs (figure 3.36d). Parallel In Serial Out (*PISO*) registers are sometimes equipped with an input register. The loading of a PISO register will be discussed in section 3.4.4. Bidirectional serial registers can shift both ways. The basic element in a complex register always consists of a D flip-flop preceded by a shifter which selects the next state.

A counter (figure 3.36e) generally sends a binary code onto its outputs but BCD codes and one out of n codes are also found. The names given to the outputs located inside the package must stress output significance, and hence the counting code. The counter can be synchronous (one common internal clock for all flip-flops) or asynchronous (the flip-flop output triggers the next flip-flop). There can be a reset, parallel loading and a command variable for counting/decounting. When dealing with an asynchronous counter, it is necessary to use the most significant output to cascade two counters.

Synchronous counters are equipped with a Ripple Carry Out (*RCO*) output which is used to authorize counting of the next level. A counter can be a programmed to divide by n by loading an initial value each time it is reset. It can be equipped with inputs which fix the division factor. Books on logic systems analyze counter synthesis in detail. The behaviour of counters found in the TTL and HC-MOS catalogues will be discussed in section 3.6. There are only few basic sequential systems but the catalogues show a great variety of implementation solutions.

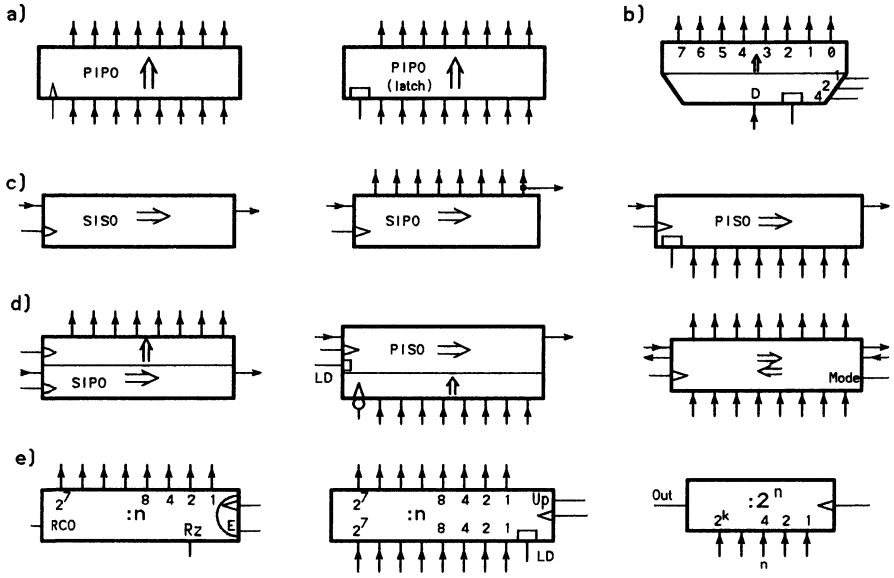


Fig. 3.36 Standard sequential systems

3.4.2 Inhibiting the clock

These last two cases do not apply to "74" circuits. The solution of figure 3.37d has been implemented in a few circuits because it corresponds to a frequent case i.e. the beginning of a sequence is a random one, but its end is synchronized by a counter. Figure 3.37 shows these clock inhibition cases for a parallel register; serial registers or counters are dealt with in a very similar manner.

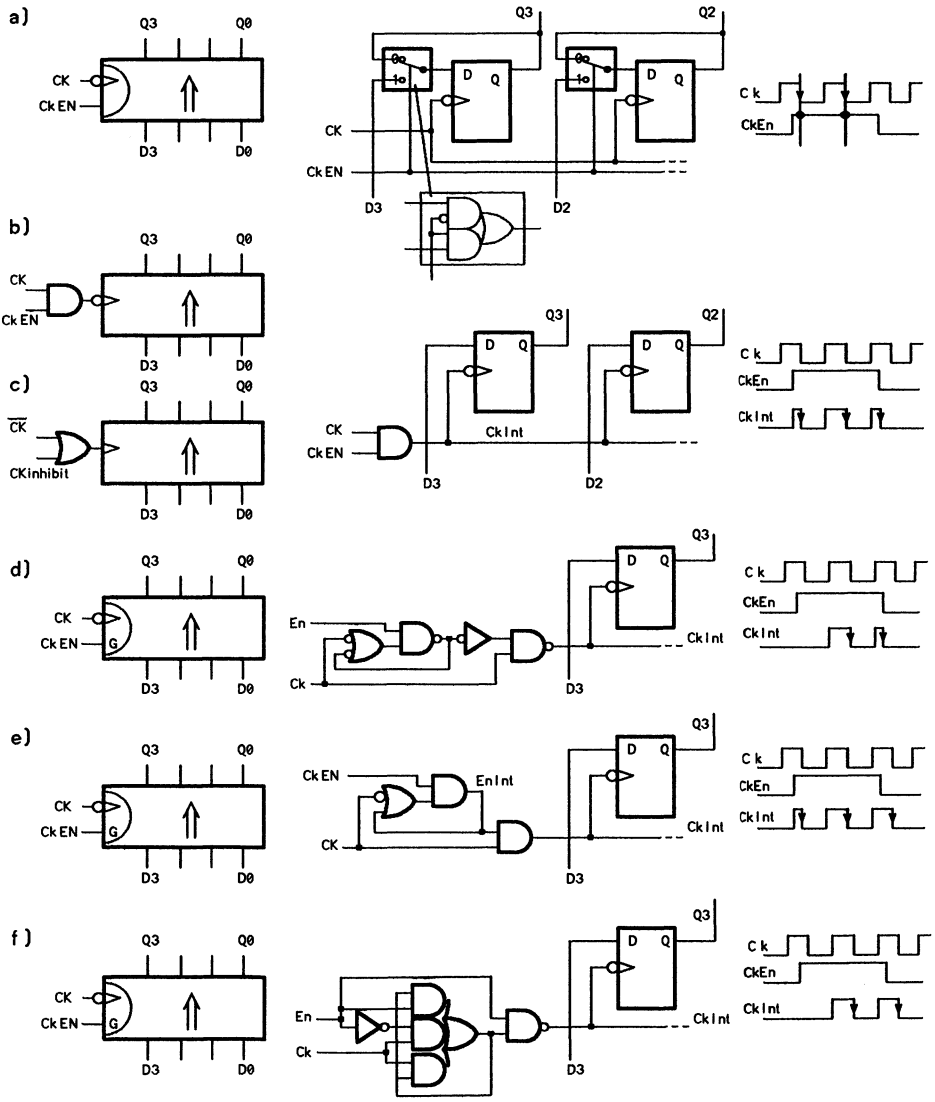


Fig. 3.37 Inhibiting the clock

3.4.3 Asynchronous and synchronous reset

A reset can act directly on the reset input which is technologically easy to incorporate in a flip-flop (figure 3.38a). So that the reset occurs in phase with the clock, all flip-flop D inputs must be equipped with an AND gate (figure 3.38b). The Reset input must be inverted on the AND gate input, since counting must be authorized when Reset is inactive. The fact that the reset input is active high or active low is a technological detail which is taken care of at the end of the design stage.

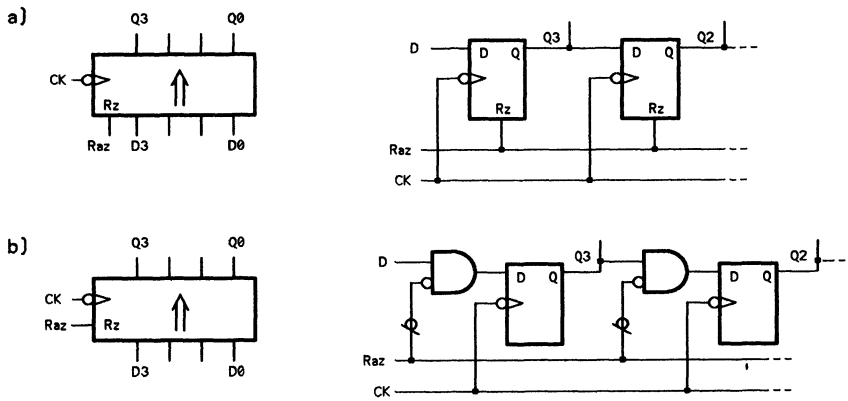


Fig. 3.38 Asynchronous and synchronous reset to zero

3.4.4 Asynchronous and synchronous loading

A flip-flop can be loaded asynchronously, acting on the flip-flop static Set and Reset inputs through AND gates (figure 3.39a). If e.g. D3 is equal to zero, the output of gate 2 is active when LD is active, and the flip-flop is reset. Two different implementations are shown in figure 3.39a: the one using NAND gates is generally chosen.

In the case of synchronous loading, a parallel channel is created with switches positioned by the load command (figure 3.39b). These two loading possibilities apply both to counters and registers. In order to stress this important difference, synchronous LD inputs are drawn on the same face as the clock throughout this book and its appendix.

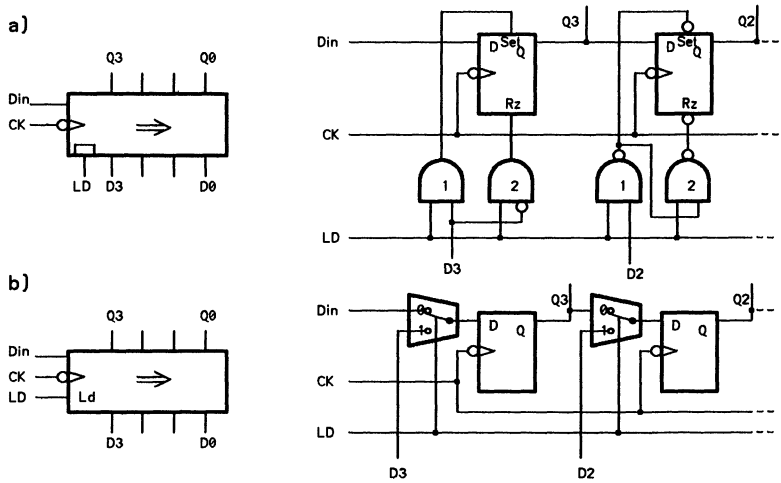


Fig. 3.39 Asynchronous and synchronous loading

3.5 Registers

The different register types have already been mentioned in section 3.4.1. They are described here in more detail.

3.5.1 Parallel registers and latches

Dynamic *parallel registers* and static *latches* have a similar functionality and similar symbols (figure 3.40) which are sometimes even identical within manufacturers' documentation. In many applications, when the transfer pulse is a brief one, either a latch or a register can be used although the designer must beware of pulse polarity.

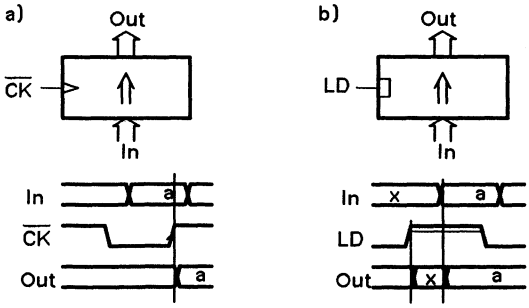


Fig. 3.40 Differences between registers and latches

The catalogue listing registers and latches is quite a large one. Many types include a clock enable and/or a reset input. Combinational elements can be preceded or followed by registers or latches. Figure 3.41 shows the case of a switch.

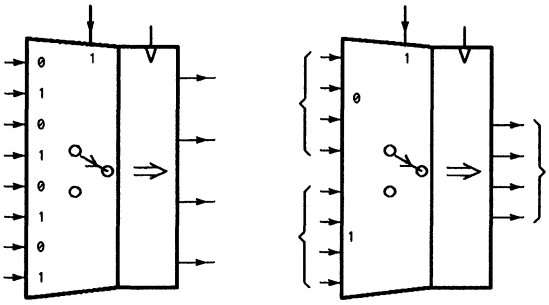


Fig. 3.41 Example of a special register (two equivalent schematics)

3.5.2 Read-back registers

In many microprocessor applications, it is useful to be able to read the state of a previously loaded register. In such cases, a driver is included in these *read-back registers* as well as the necessary commmand logic (figure 3.42). This type of register is quite a recent one; the catalogue mostly lists read-back registers of the latch type.

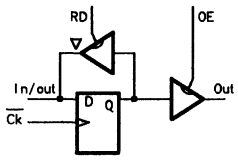


Fig. 3.42 Read-back register

3.5.3 Serial-parallel registers

Although a serial register always shifts synchronously (triggered by a clock edge), parallel loading and reset can be either asynchronous or synchronous (figure 3.43) as is explained in sections 3.4.3 and 3.4.4.

Figure 3.43 shows two well-known serial registers with our symbology and the AIEE one (section 3.8.2). Let us note that these 74xx165 and 74xx166 circuits are equipped with two clock inputs, known to the manufacturer as *Ck* and *CkInhibit*, but which are in fact quite symmetrical (section 3.4.2). The OR input gate, for clocks triggered by a rising edge, should be seen as an AND gate for clocks triggered by a falling edge. Keeping one of the clocks at state "1" inhibits the function of the other clock input. The state of the input used for inhibition purposes must not be modified when the clock on the other input is at state "0".

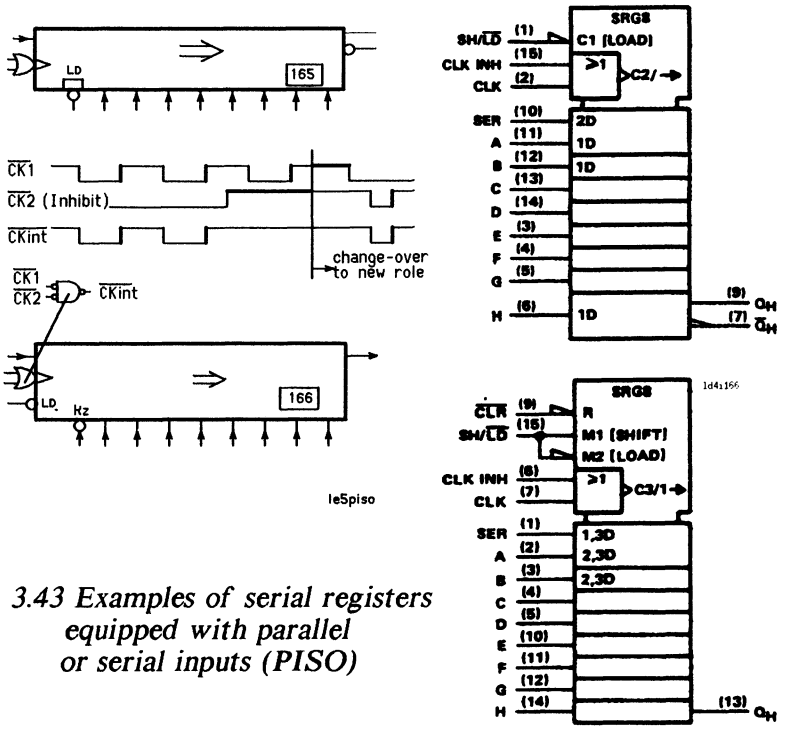


Fig. 3.43 Examples of serial registers equipped with parallel or serial inputs (PISO)

3.5.4 Universal registers

So called *universal registers* have working order inputs defining the shift direction and parallel loading (figure 3.44).

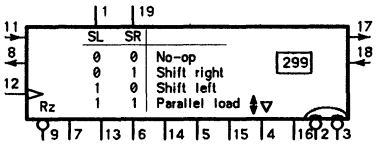


Fig. 3.44 Universal synchronous register

3.5.5 Case analysis

The catalogue of serial registers is quite rich and includes 4, 8 and 16-bit registers equipped with parallel inputs or outputs and sometimes with an input or output register. A good example is circuit 74xx589, which is functionally interesting but difficult to use (figure 3.45). This is because the transfer from the input parallel register to the shift register is an asynchronous one. The Ld input is exceptionally drawn on the clock side, but the latch symbol is used. If the transfer is to occur at high speed without interrupting the shift, an extremely precise timing analysis (figure 3.45b) must be carried out and the maximum speed, e.g. in the case of a graphic screen, falls below that indicated by the manufacturer.

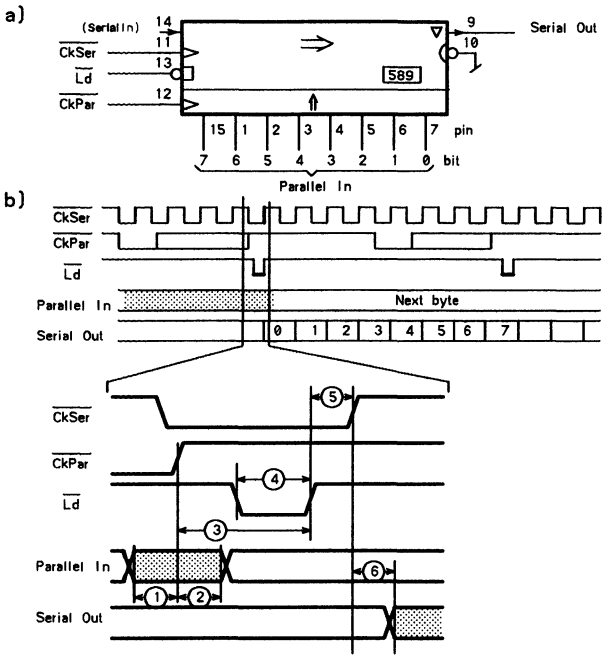


Fig. 3.45 Serial register equipped with a parallel input register

In order to analyze the working conditions for this register, it is important to check in the right order that:

1) the input register is recording the right information (respecting times 1 and 2)
2) the information has been stable for long enough in this register so that the asynchronous transfer into the shift register can be carried out correctly (time 3 equals the sum of propagation time in the input register and set-up time in the serial-parallel register).
3) the load pulse is sufficiently long (time 4)
4) the time allowed for stabilization of the information inside the serial register is sufficient for the shift clock to have a foreseeable effect (time 5)
5) the delay with which the serial information is gathered is compatible with the application (time 6)

Once the circuit technology has been chosen (74LS589 or 74HC589 in this case), it is possible to find out the various times from the manufacturer's catalogue and design a sequencer respecting all conditions. With an excessively slow circuit, the problem is solved by using two integrated circuits (a parallel and a synchronous parallel-serial register), a solution which offers a better functionality or a greater technological choice (the circuits may belong to faster families).

3.6 Counters

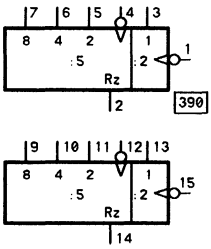
Counters can be asynchronous or synchronous. As is the case for registers, synchronous counters can be equipped with synchronous or asynchronous loading and reset inputs (sections 3.4.3 and 3.4.4).

3.6.1 Asynchronous counters

So called *asynchronous counters* (quasi synchronous would be more appropriate) are made up of cascaded flip-flops. This causes differences in propagation times and glitches when decoding which can be unfortunate. Random counting values can be obtained by acting on the asynchronous reset and set inputs, but nevertheless it is necessary to check that good working conditions are met.

Asynchronous counters are reserved for large division factors, which are often dealt with using other methods (programmable "timer" interface; program inside a microprocessor). Figure 3.46 shows an example of an asynchronous counter (divider by 10) and its related symbols.

The catalogue of asynchronous counters is a very rich but rather unorthogonal. Some asynchronous counters can handle divisions up to 2^{24} and sometimes have an oscillator connected to the input.



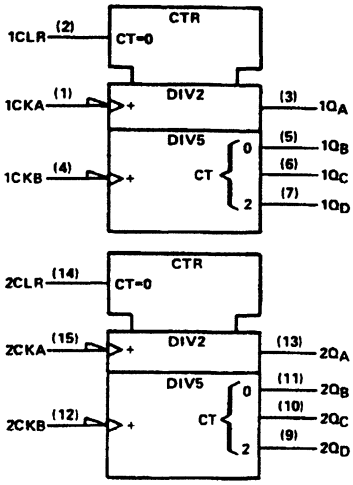


Fig. 3.46 Example of asynchronous counting circuits (LS390)

3.6.2 Divide by 2^n counter

The frequency at which 'LS292 and 294 counters divide is a power of two defined by command inputs (figure 3.47a). Functionally, these counters include 2^n-1 flip-flops and a 2^n input switch (figure 3.47b). If the value programmed at the input is modified without carrying out a reset, a variable length of time can elapse

before the desired division factor occurs. Moreover, these dividers jam when the value of n equals 0 or 1 (they do not divide by 1 and 2) due to the implementation chosen by the manufacturer.

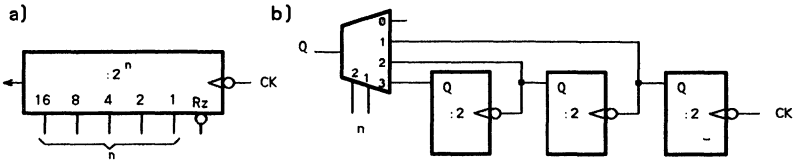


Fig. 3.47 Divide by 2^n counter

3.6.3 Synchronous counters

All flip-flops in a *synchronous counter* are fed the same pulse and therefore switch over simultaneously, allowing for slight differences which are very much smaller than the propagation times.

The clock may synchronize other functions such as the reset or the parallel load (figure 3.48). Correct symbols avoid confusion, since there are several kinds of counters. As a rule, inputs synchronized by the clock are drawn on the same face as the clock. The AND gate symbol shows which inputs inhibit counting without necessarily affecting reset or load. As is the case for an assembly language instruction, a symbol cannot express true wealth of the implemented function. The AIEE standard is slightly more explicit, but does not provide the same quick overview of the main functionalities.

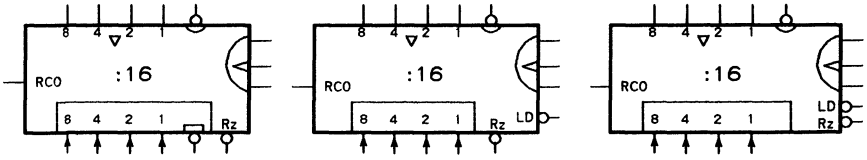


Fig. 3.48 Synchronous counting circuits

Synchronous counters are equipped with a cascading output (*RCO Ripple Count Out*) which is active in the state preceding the counter overflow. This RCO output often presents glitches during counting and should not be used as a clock (except in the case of counters carrying out an internal product with the clock and where the output is known as RCkO).

The two counting authorization inputs are generally referred to by manufacturers as P and T (figure 3.49a). The P (*Parallel*) input is used by the fastest counters but requires a cascade of AND gates (figure 3.49b) or an increasing number of inputs (figure 3.49c) in order to calculate in parallel the report for each level. Carry lookahead circuits for synchronous counters are available, e.g. the 74AS264.

T (*Trickle*) inputs can be used for simpler (figure 3.49d), albeit slower, daisy chaining: maximum frequency is determined by the propagation time required for the report to reach all the counters but the switch over remains synchronous. In this book, the symbol for the T input is always set at the same height as the RCO output to which it is linked in the simple diagram of figure 3.49d which can almost always be used in the case of common 16 or 32-bit counters.

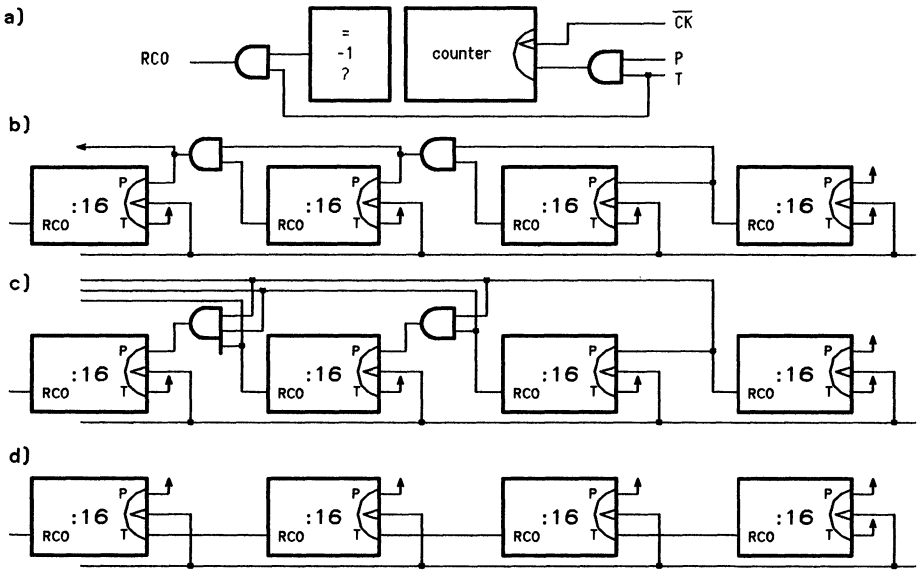


Fig. 3.49 Wiring cascaded synchronous counters

3.6.4 Programmable dividers

A counter equipped with parallel inputs can be used for counting by any value lower than the maximum one. A preset value must be loaded when RCO is activated (figure 3.50). A synchronous counter is preferable. If a down counter is used (figure 3.50a), value n-1 must be preset for a divide by n operation. If a counter is used (figure 3.50b), the 2-s complement is required, and reset must be avoided since it means that the length of the initial cycle will be a complementary one. An asynchronous action is dangerous, since the RCO signal disappears as soon as one of the counter flip-flops has been been loaded. State "0" (in the case of a down counter) only lasts a few nanoseconds, and value n must be preset. The same comments apply when dealing with several cascaded counters, but in this case, there is little hope the asynchronous solution will work witout a trick.

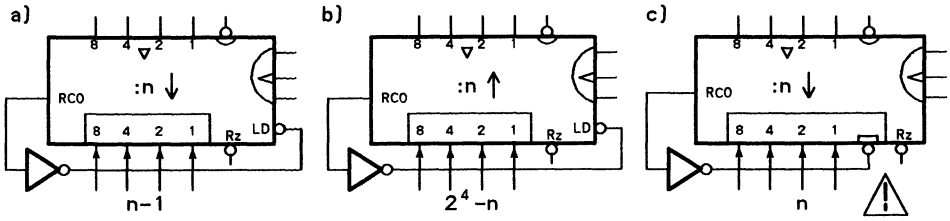


Fig. 3.50 Programmable dividers

3.6.5 Counters with registers

The family of synchronous counters is very rich. Some are equipped with input or output registers as well as multiplexers (figure 3.51). Problems related to proper operating conditions of these registers can be analyzed in a very similar manner to that described in section 3.5.5.

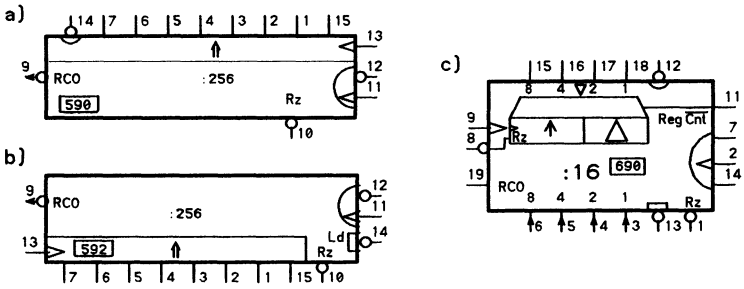


Fig. 3.51 Counter equipped with a register

3.6.6 Pseudo-random counters

Shift registers which loop onto themselves can generate long sequences which do not usually go through all states; an initial reset is required. *Ring counters* are often used for simple counting (figure 3.52), since only one of their bits is altered at each switch over and they are not prone to decoding glitches.

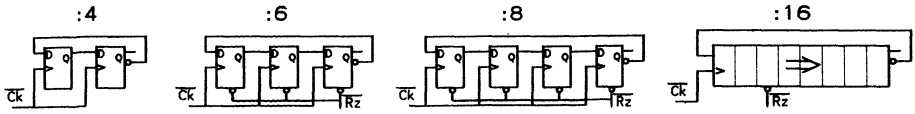
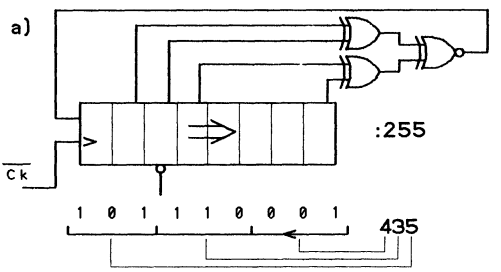


Fig. 3.52 Ring counters

Linear registers develop retroaction with exclusive ORs and are connected with an interesting theory of irreductible polynomials [Peterson72]. Figure 3.53a gives the diagram for a counter by $2^8-1=255$. One state is normally missing in the counter since state "1" on all flip-flops always loops onto itself; if the output inverter of the final exclusive OR is removed, state zero becomes the trap. If an exclusive OR input, i.e. a "1" and a "0", is associated with each wired output, the result is a binary word which, when coded in octal starting from the left, gives 435. Figure 3.53b shows the corresponding octal values for counters from $7=2^3-1$ to $2^{20}-1$. These counters are remarkable because the bit sequence they generate is a pseudo-random one which can be used in various applications to generate random values for encoding/decoding serial messages and for detecting errors during serial transmissions.



b) $:2^n-1$

n bits	formula	n bits	formula
3	13	12	16005
4	23	13	34003
5	45	14	70005
6	103	15	140001
7	203	16	264001
8	435	17	440001
9	1003	18	1004001
10	2011	19	3240001
11	4005	20	4400001

Fig. 3.53 Pseudo-random counters

3.6.7 Rate multipliers

A *rate multiplier* generates a pulse train according to a preselected k value. If the circuit includes n flip-flops, k evenly distributed pulses are generated every 2^n pulses. The output frequency is lower, but in reality, the circuit multiplies by a variable value and divides by a fixed one, hence its name. The internal structure is formed by a synchronous binary counter and logic gates which deduct 2^{n-1}, ..., 2, 1 pulses according to the preset data, and perform the logic sum (figure 3.54b). No pulse is generated when the counter reaches its maximum value; this allows a cascaded lower weight circuit to add its own pulses during this time. There are 6-bit binary and 4-bit BCD rate multipliers which are easy to cascade (figure 3.54c). Available circuits include an OR gate which is used to cascade two circuits without external components. A typical application is given in section 8.2.

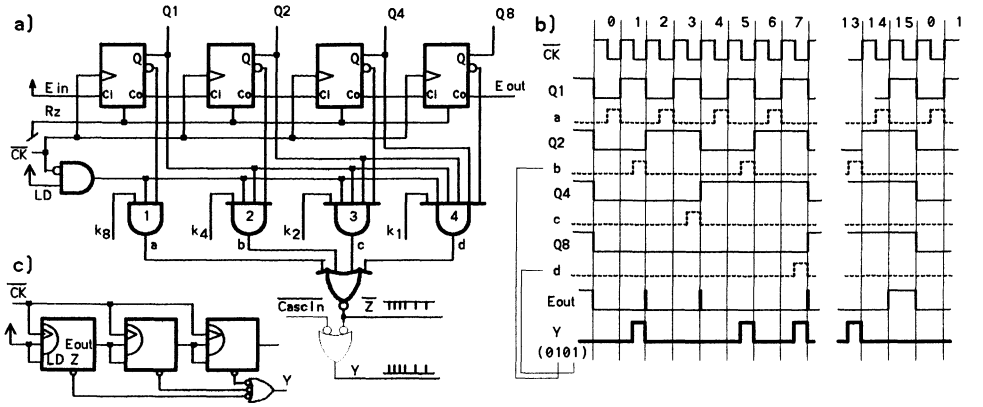


Fig. 3.54 4-bit rate multiplier

3.7 Delays and pulses

A delay may be required in order to compensate for another delay created by propagation times or it may even be absolutely essential for the application. Starting with a delay element and a logic gate, it is easy to generate delayed signals and pulses (figure 3.55).

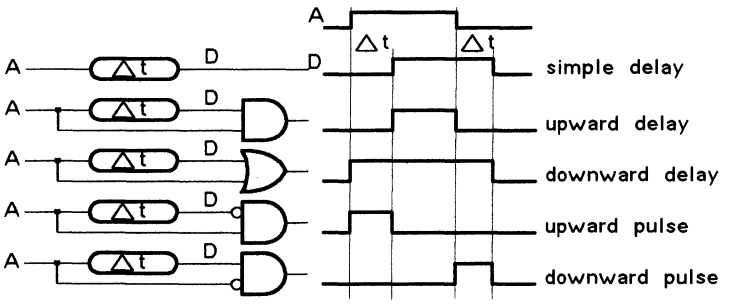


Fig. 3.55 Elementary generation of delays and pulses

Thanks to analog (continuous variations) or logic solutions (quantified by a clock), there are many ways of generating short delays (a few nanoseconds), long delays (a few milliseconds) or very long delays (a few seconds or hours).

3.7.1 Delay lines

Delay lines are used for very brief delays (1 to 50 ns). They are rather expensive products based on distributed inductances and capacitors. Some are passive (figure 3.56a) whereas others include output amplifiers (figure 3.56b) or even a shifter which can be used to program the length of the delay (figure 3.56c).

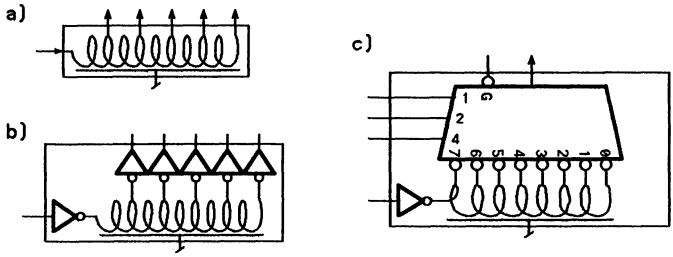


Fig. 3.56 Delay lines

3.7.2 RC networks delays

The RC resistor-capacitor network discussed in figure 1.1b is an easy way of generating a delay between 50 ns and 1 second. A Schmitt trigger inverter is required to regenerate the signal. This type of delay is quite commonly used to filter contact bounce (figure 6.5a) and more generally when the required delay is long and not very precise. Sensitivity to temperature and to voltage is significant.

Use of deriving circuits (figure 1.1c) to generate short pulses should be avoided because they are too sensitive to interferences.

3.7.3 Quantified delays

Designers try to avoid generating delays and pulses with RC (resistor-capacitor) or LC (inductance-capacitor) networks. It is more reliable to generate discrete delays using a single clock, even if this implies using a lot more logic. Synchronization introduces quantification errors and increases the risk of metastable states (section 3.2.8) which must be taken into account.

Shift registers are ideal elements for generating delays as well as any type of pulse by combining delayed outputs. Counters can be used for bigger delays using less flip-flops but care must be taken if decoding glitches exist. Figure 3.57 shows two typical solutions which can be used as a basis to imagine other ways of generating delays (rising or falling ones) and brief pulses.

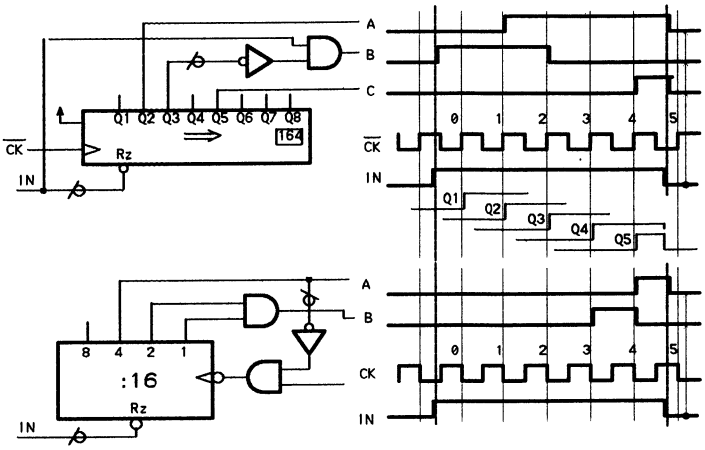


Fig. 3.57 Quantified delays and pulses

3.7.4 Analog oscillators

There are many schematics for oscillator circuits. The simplest one uses a Schmitt trigger inverting gate and an RC network (figure 3.58a). This type of oscillator is not very stable when voltage and temperature change. Maximum frequency is fixed by the propagation time of the inverter whereas minimum frequency is a function of the maximum R value which still guarantees the levels and the leakage current in the capacitor. This oscillator can be blocked, but the first period is very different from subsequent ones (figure 3.58b).

The famous 555 circuit is frequently used for a great variety of oscillators and delay circuits at low frequencies. Internally, this versatile circuit consists of two analog comparators (section 6.2.1) and an SR flip-flop which controls a transistor enabling the capacitor to discharge rapidly (figure 3.58c). When an oscillator is wired (figure 3.58d), as soon as the bottom comparator reaches the Vcc/3 level (1.7V), the flip-flop is set and the capacitor can charge through R_1 and R_2. When the level reaches 2/3Vcc (3.3V), the flip-flop is reset and the transistor is discharged through R_2. Due to the fixed 1/3 factor, charge time is equal to $0.69 \cdot (R_1+R_2) \cdot C$ and discharge time to $0.69 \cdot R_2 \cdot C$.

A *VCO (Voltage Controlled Oscillator)* is optimized so that its frequency depends on the input voltage. These components are used for disk interfaces and serial data transmissions, when decoding pulse trains and synchronizing with unstable signals.

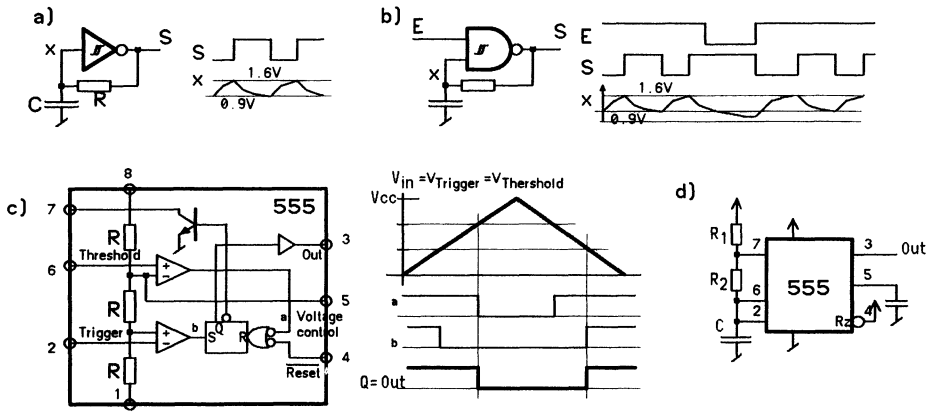


Fig. 3.58 Analog oscillators

3.7.5 Crystal oscillators

Quartz *crystal oscillators* are universally used. Except for special applications, there is no need to build the electronic part: a large number of oscillators between 10kHz and 100MHz are available. Some include a programmable frequency divider. An amplifier (74xx04 inverter) is usually recommended in order to improve transfers on clock lines which are usually loaded. The duty cycle is not always 50% and the designer should be careful in cases where the clock must be available both in a direct and an inverted form. An inverter creates a delay which may not be acceptable (figure 3.59a). In critical cases, the frequency of the double frequency quartz must be divided by two and a parallel register added in order to synchronize the direct and inverted clocks (figure 3.59b).

At high frequencies (30MHz and more), such as those found in recent microprocessors and screen interfaces, clock line impendance should be adapted. ▿Carelessness can lead to deformed signals, especially if the line has long branches (figure 3.59c). It is better to use several clock amplifiers and to insulate each branch. A series resistor at the buffer output, preferably with a terminating resistor (figure 3.59d) or an adaptation bridge (figure 3.59e), guarantees the best result. High frequency signals also cause a great amount of crosstalk; this is dealt with once the layout is finished, and the designer also checks that clock lines do not zig-zag through the circuit and never stay close to the same lines for long distances.

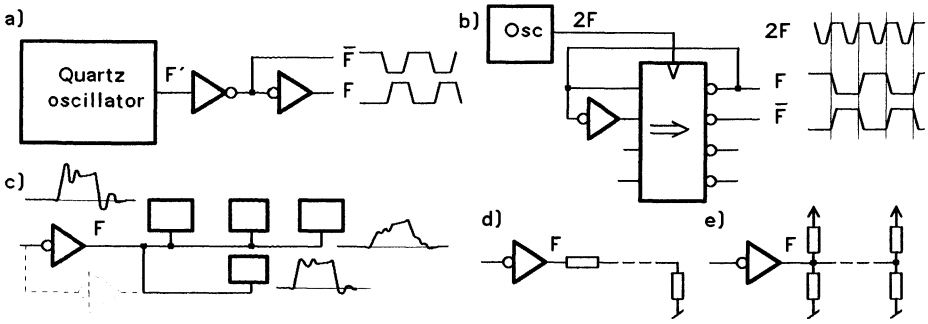


Fig. 3.59 Crystal oscillator

Several microprocessors and interfaces include an oscillator- amplifier circuit; the quartz and usually two capacitors must be added. The layout of these components must be compact; manufacturers give precise directions in each case.

Some oscillators have an input for disabling the oscillating input. When the output is a three-state one (figure 3.60a), it is possible to select among several connected oscillators. A complete reset is generally required after switch-over. As a rule, if a clock has to be disabled, an AND gate is suitable (figure 3.60b), as long as the enable input does not change when the clock is active. The solutions of figure 3.37d,e,f apply here. A JK flip-flop is frequently used to control a clock (figure 3.60c), but this requires doubling the input frequency. By acting on J (or K) only, the state of the disabled clock is known (figure 3.60d).

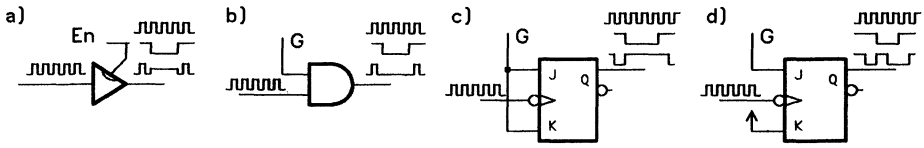


Fig. 3.60 Gating a clock

3.7.6 Monostables

A *monostable multivibrator* or *one-shot* circuit has a single stable state (state "0"). It temporarily switches over to "1" when faced with an edge (just like a dynamic flip-flop) but only stays in this state for a time defined by the time constant of an RC network or a counter.

If a signal edge occurs before the end of the pulse length, the edge is either ignored or can extend the pulse by a whole pulse length starting from the edge. In the first case, the monostable circuit is known as *non retriggerable* whereas is is *triggerable* in the second case (figure 3.61). These two types have the same symbol.

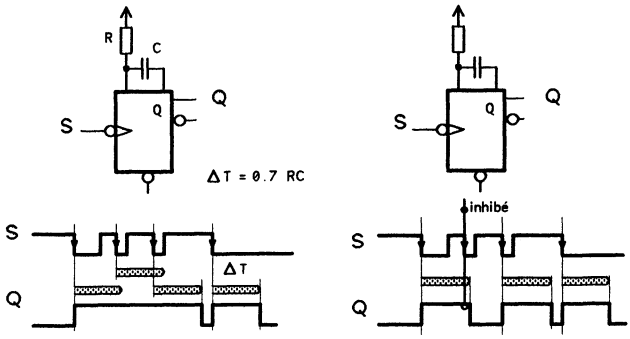


Fig. 3.61 Retriggerable and non retriggerable monostables

Manufacturers' catalogues offers a choice of simple and dual retriggerable and non retriggerable monostable circuits. Monostable circuits are replaced as often as possible by sequencers which count a certain number of clock pulses; the B signal of figure 3.57a is a typical example.

3.7.7 Power-up reset

Every sequential system, especially microprocessors, must be forced into an initial state at power up. The supply voltage rises progressively, due to the filtering and decoupling capacitors. Figure 3.62 represents a typical signal evolution. When the voltage reaches between 1 and 3V, the reset logic can work. It must be active before the system's circuits are, in order to ensure that no parasitic action occurs during power-up. Reset must stay valid for some time after the voltage has reached the minimum operating value, i.e. 5 or 10% below the nominal voltage Vcc.

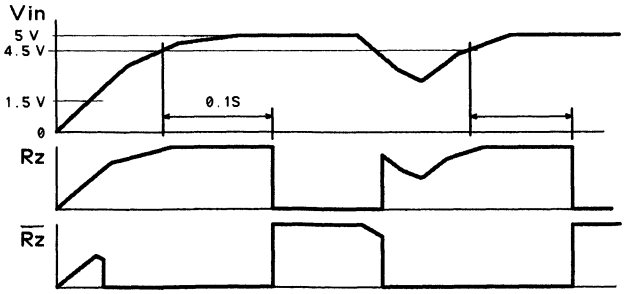


Fig. 3.62 Power-up reset signals

Within a synchronous system, reset length is expressed by a number of clock cycles, but complex VLSI circuits include some circuitry for the generation of a polarizing voltage, which may take a few fractions of a second to become settled. If there is a voltage drop, reset must occur before the circuit reaches the voltage where the operation is no longer guaranteed. Computer systems generate an interrupt as soon the main power supply notices the voltage drop, a fraction of second before reset is activated.

A simple RC network is usually used to generate the reset pulse (figure 3.63). The diode must discharge the capacitor rapidly in case a short parasitic power down occurs. Without that diode, the circuit behaves as shown by the broken line in figure 3.63b. An active circuit, as found in dedicated reset circuits, would be more efficient. Using a *voltage detector* ensures precise voltage levels (figure 3.63c). Several models are available with features corresponding to the most common applications.

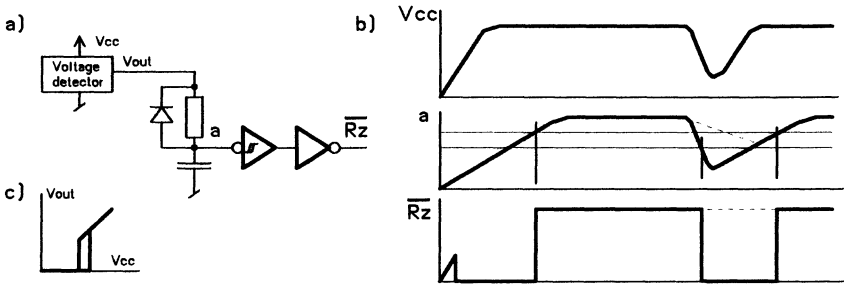


Fig. 3.63 Usual reset solution

3.8 Standardized logic symbols

A logic diagram can describe a very rich functionality. The previous sections have repeatedly stressed how important it is to use the appropriate symbology, whose historical evolution is worth mentioning.

3.8.1 Main standards

There are three main standards for logic symbols. The difference is quite obvious in the case of gates and flip-flops (figure 3.64). The German *DIN41859 standard* (figure 3.64a) is seldom used nowadays. The American *MIL-STD806b standard* (figure 3.64b) is widespread in publications and application notes related to microprocessors and interfaces. State-owned industries have adopted the *IEC-617* standard, better-known in the USA as ANSI/IEEE-91 or *AIEE-91* (figure 3.64c). Manufacturers of integrated circuits, led by Texas Instruments, have started to follow this new standard, which is discussed in the following sections.

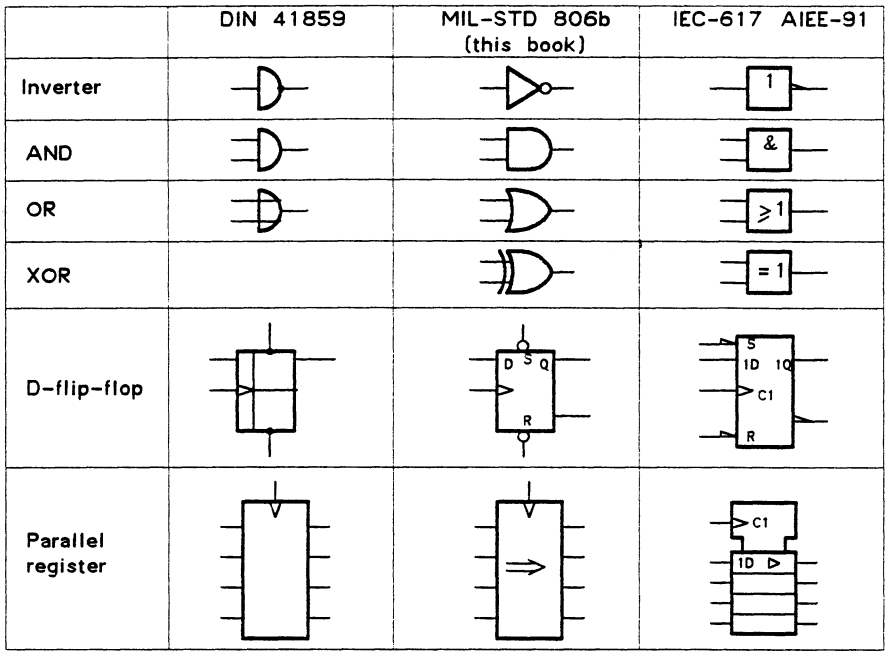


Fig. 3.64 Graphisms used for logic symbols

Differences between the three standards are greater than just the choice of a symbol for elementary functions. Over the years, better understanding of logic and the need for more accurate documentation of complex diagrams has resulted in a more precise graphic syntax which ensures greater diagram legibility and reliability. The first CAD systems had a very negative influence on this evolution: their inability to draw curves encouraged the use of rectangles and alphanumeric characters instead of easily recognizable shapes; lack of suppleness when accessing symbol libraries often meant that the correct symbol could not be used in cases where two representations of the same physical implementation were possible (figure 2.5).

The new IEC/AIEE standard has interesting aspects but, sadly enough, the distinguishing gate shape has disappeared and a rather complex alphanumeric syntax has been introduced. These notations cannot express precisely the internal richness of a circuit, just as it is impossible to express the real richness of a microprocessor instruction by using its mnemonic. The most important thing is that simple and frequently used symbols should be recognized at once and without any ambiguity, in order to leave the designer more time to become familiar with complex functions.

The symbols used in this book are derived from the 806b standard but include the best ideas of the IEC/AIEE standard. Extra symbols (figure 3.68) have been added in order to stress the traps which the inexperienced designer must watch out for.

3.8.2 IEC-617/AIEE-91 standard

Documentation about the IEC/AIEE standard is easy to get hold of since it is included, in a relatively clear and complete form, in the Texas Instruments TTL and HCMOS catalogues [Texas88b]. Only the major aspects will be discussed here.

A first important notion specific to the standard is that of *common-control blocks* and *common-output element* (figure 3.65), respectively set above (with a characteristic recess) and below the frame surrounding the operator, which is itself divided into as many parts as there are bits. The clock and its activation conditions are grouped in the common control block. A parity check, for instance, is shown in the common-output block (figure 3.65).

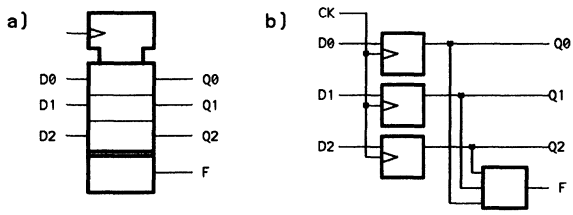


Fig. 3.65 Functional equivalents

Notations inside the symbols follow a rather precise syntax and the designer must learn to distinguish three fields inside each symbol, of which only one or two may be present:

- the first number is an indicator which characterizes the part played by one or more inputs
- the following letters characterize the function
- the last number associates the function with the inputs which influence it

For example, dependence on an AND gate is characterized by the letter G (gate) and can result in the symbols of figures 3.66a and b. Activating a three-state output is characterized by the letters EN (enable); figure 3.66c shows the symbol for the 645 bidirectional driver and its internal diagram. Address selection in the case of a decoder or a memory is referred to by the letter A (figure 3.66d).

The interest in showing dependence using an alphanumeric rather than a graphic code is arguable. This notation is difficult to use correctly and a detailed functional diagram must always be given in catalogues in order to explain the logic.

On the other hand, the IEC/AIEE standard has defined extremely precisely the qualifying symbols which characterize input and output polarity and special properties (figure 3.68). All these symbols have been included

and, sometimes completed, in this book. Although the AIEE standard takes into account the difference between positive and negative logic, there is no such distinction in this book (section 2.1.2).

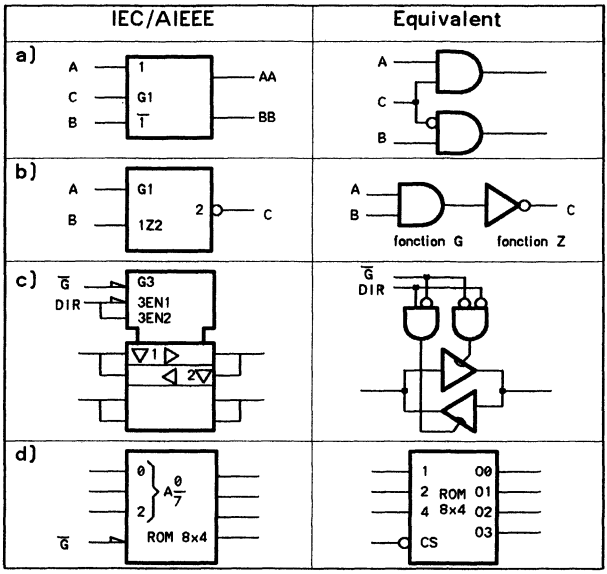


Fig. 3.66 Examples of IEC/AIEE symbols

3.8.3 Drawing conventions

Diagram legibility depends on a few basic drawing rules which must be respected and used consistently throughout the schematic. It is customary to decide that information flows from left to right and from top to bottom. Some CAD systems even place all inputs on the left and all outputs on the right, using alphanumeric references for backtracking. Bus bits should be numbered homogeneously with the least significant ones on the right or at the top (figure 3.67). Asynchronous counters have their first clock on the least significant bit side, i.e. on the right. Opposite direction conventions apply to shift registers.

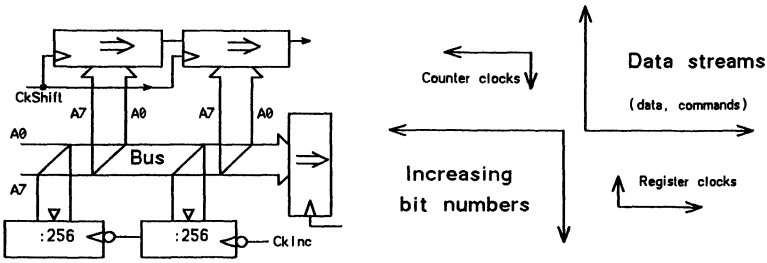


Fig. 3.67 Preferable information flow and bit order

IEC/AIEE Inputs	IEC/AIEE Outputs	This book Inputs	This book Outputs	
				direct I/O (active high)
				inverted I/O (active low)
				positive edge active (dynamic)
				negative edge active
Cm				active pulse (static)
Cm				master-slave structure
				data lockout
				Schmitt trigger input
				Analog I/O
				Non logic I/O
				Buffered output
G		G		Gating input
R		Rz		Reset input
LOAD		Ld		Parallel load input
EN				Three-state enable input
				Three-state output
				Open collector/drain output
10k		10k		same with internal pull-up
				Open emitter/source output
10k		10k		same with internal pull-down
		50		Serial internal resistor
				External pull-up
				External pull-down
		[1] 10k		Input with internal pull-up
		[0] 10k		Input with internal pull-down
				Functional pseudo-inverter
				Direction of data flow
				Bidirectional line
				Side connections
				Vcc Gnd

Fig. 3.68 Qualifying symbols

4 Memory circuits

Memories are part of a very rich field and often form the central element of a sub-system. In such cases, the combinational and sequential elements discussed in the previous chapters become essential in order to control information and guarantee correct signal sequencing.

4.1 Introduction

A memory chip uses its cells to stock binary or analog information permanently or temporarily. Although there are many technological means of storing information (magnetic, optical, etc.), only integrated memory chips will be discussed here.

Memory size (the number of cells) is expressed in *kilo* (1k=2^{10}=1024) or in *mega* (1M=2^{20}=1'048'576). Units are n-bit words, usually 8-bit *bytes*. The letter B refers to a byte and letter b to a bit. Therefore, 1kB means 1024 bytes and 1kb equals 1024 bits.

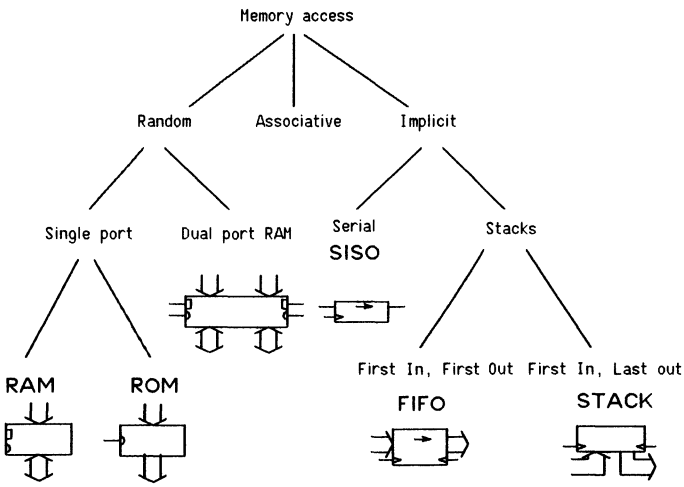


Fig. 4.1 Functional memory types

A memory cell is accessed according to various principles (figure 4.1). *Random access* locates information thanks to its address. *Implicit access* does not encode the address: the "next" piece of information follows

according to various succession rules (*serial*, *fifo*, *stack*); these terms will be defined later on. *Associative access* is made possible by giving part of the information; hence the number of possible answers to an associative access can be zero, one or many.

These types can be subdivided according to the technology used and application needs, especially as regards random memories. Simple access is generally the rule but some applications require a *dual ported memory*, which implies doubling address and memory lines. Random access memories (figure 4.2), can be divided into two types.

Read write memories are usually volatile: they can either lose or keep their information once the voltage supply is cut off. Very low consumption *volatile memories* can easily be made into *non volatile* ones by adding a battery.

Read only memories are written, either definitely or temporarily, by the manufacturer or by the user. Some read only memories can be electrically reprogrammed in a selective manner; they differ from non volatile read write memories by the fact that the write cycle is several times slower than the read cycle.

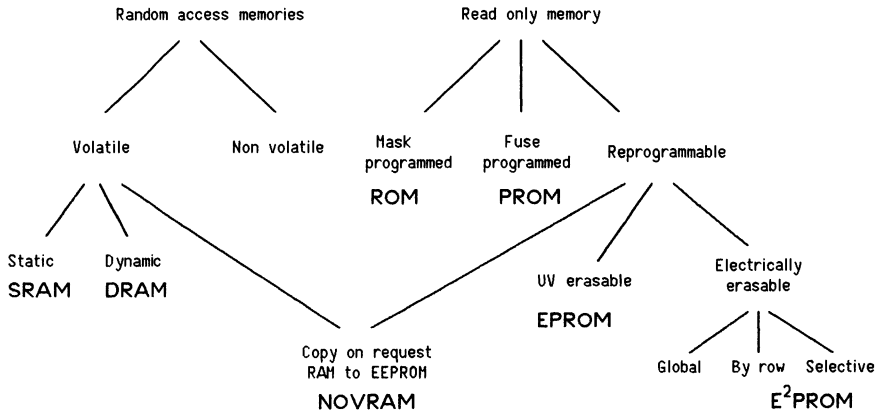


Fig. 4.2 Classification of random access memories

4.2 Read only memories

Read only memories are used for storing computer programs, data, character fonts, and sometimes for performing logic functions (section 2.2.7).

4.2.1 Principles

A *read only memory* (ROM) of 2^n words has n address lines and one or more *chip select* (CS) inputs to make decoding easier. The output is a three-state one and can sometimes be an open collector one for ROMs in TTL technology (figure 4.3).

Read only memories can hold from 256bits to several Megabits and their answer times vary from 10 to 500 ns. Bipolar ROMs are the fastest of all but can hold very little and their power consumption is high.

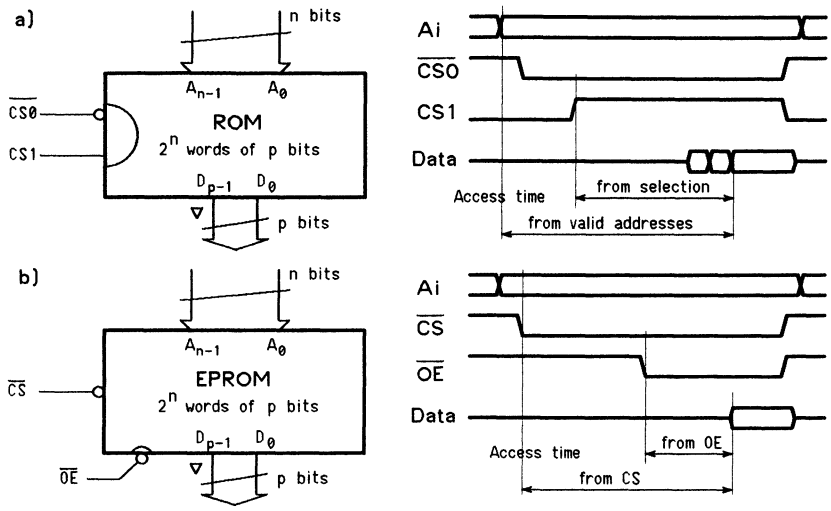


Fig. 4.3 ROM block diagram and timing diagram

The second chip select input on a ROM is often called OE (*Output Enable*) in order to stress the fact that it acts directly on the output driver with a reduced access time if the internal information is already stabilized (figure 4.3b). The manufacturer documents minimum access times from stable addresses as well as from active CS and OE. Information is only guaranteed valid if all access conditions are fulfilled.

4.2.2 Types of ROMs

ROMs are made up in a very simple manner by placing a diode or a MOS transistor at the crossroads of a selection gate. According to the information required by the client, the integrated circuit manufacturer will generate one or two specific masks. Consequently, basic ROM costs are high but fall drastically in the case of large quantities.

Programmable ROMs can be programmed once or several times according to the technology. PROMs (*Programmable ROM*) are equipped with a fuse in each memory element: information is written by blowing this fuse. According to the contents which must be found in each position, a device known as a *PROM programmer* generates the necessary current and voltage pulses on the circuit pins.

EPROMs (*Erasable PROM*) and UVPROMs (*Ultraviolet PROM*) stock information in MOS transistors (figure 1.4) equipped with a *Floating Gate MOS* (FAMOS). A high voltage pulse (12 to 24 V) charges the

capacitor formed by the gate; discharge would take more than ten years under normal conditions but in fact it takes less than 20 minutes thanks to the electronic movement created by the ultraviolet rays of a *UV EPROM eraser*.

EPROMs are equipped with two special lines for programming. The PGM line (*Program Mode*) is activated when programming pulses, of 12,5V on recent EPROMs, are sent along the Vpp line. These two signals are often multiplexed with a chip select line, with voltage encoding (12 V in the program mode) so as to avoid extra pins. PROMs equipped with fuses are programmed by acting on the data line voltage supplies.

EPROM circuits have a quartz window which allows the ultraviolet erasing rays through. One way of bringing down packaging costs is to use windowless EPROMs which come in white plastic packages and can only be programmed once. Some manufacturers call them OTPROMs (*One Time PROMs*).

The new high density EPROMs require high speed programming algorithms so as to maintain programming times within acceptable limits (less than 5 minutes).

EEPROM circuits, also known as E^2PROMs (*Electrically Erasable PROMs*) or *Flash memories* are electrically erasable, globally, by blocks or by bytes.

4.2.3 Standard PROMs

The PROMs which are now on the market are not really suitable for designers who want to implement logic functions or stock dozens of parameters in PROMs. Small PROMs are based on an outdated technology and offer a poor power/speed ratio. The PROM family is being extended by creating greater capacity fuse PROMs for micro-programming applications. No manufacturer has grasped how interesting small PROMs with an extremely low power consumption can be.

Large size PROMs can be very suitable for complex sequencers. For this reason, some PROMs are equipped with a parallel output register. However, setting up the truth table can be quite heavy-going and requires the use of adequate software.

Bipolar PROMs can be 1, 4 or 8 bits wide. Packages are not standardized but there is a certain compatibility between types, as is the case for EPROMs. Identical sized PROMs produced by different manufacturers usually have the same pinout, even if circuit names vary widely. Evolution is much slower than in the case of EPROMs. Consequently, the equipment needed to program these PROMs is simple and easy to get hold of.

4.2.4 Standard EPROMs

EPROMs were developed alongside microprocessors. The first ones to come out in 1971 held 256 8-bit words. Since then, their capacity has more or less doubled every two years. Packages rapidly switched from 24 over to 28 and then to 32 pins; they are standardized by JEDEC (figure 4.4). Corresponding ROMs sometimes carry extra programmable chip select lines in order to make end user applications easier.

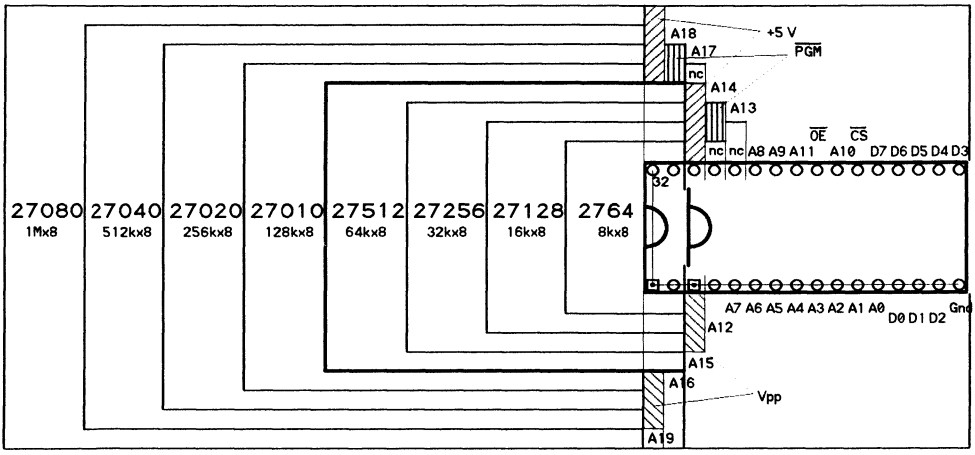


Fig. 4.4 Pinout of 8-bit EPROMs

In order to ensure compatibility within a system including various types of EPROMs, printed circuits usually include jumpers which are placed beside the Eprom sockets. Figure 4.5 shows that four jumpers are enough to ensure compatibility between 2764 (8kx8) type EPROMs delivered in 28-pin packages and the largest type (1Mx8) which come in 32 pin packages, i.e. 8 generations of EPROMs.

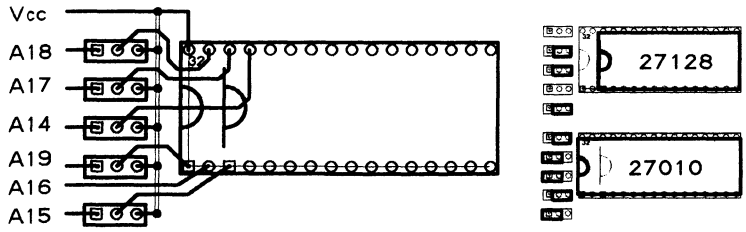


Fig. 4.5 Selection jumpers according to the type of EPROM

A new family of 16bit EPROMs with 40 pin packages has been available since 1988. They are very suitable for 16 and 32-bit systems. As is the case with 8-bit EPROMs, technical knowledge in 1991 is such that one Megabit

size models are cheap and 4 Megabit ones are available. Every 2-3 years, they go on to the next size; therefore, a new extension of the 40 pin package standard will be necessary in 1996. PLCC square packages will be generalized by then.

It is possible to use a 16kx8 page structure (*Page-addressed EPROM*) to reduce package size (figure 4.6). The idea is to write the page address (write cycle at any address) before reading the corresponding page. Write operations are only carried out when changing over to another page. This is fine for tables or program modules but can hinder the linear running of a large program stocked in ROM; however, a system ROM can be used to go from one page to the next.

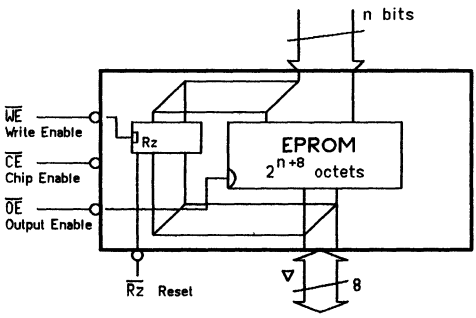


Fig. 4.6 Principle of page-accessed EPROMs

A *Write Enable* $\overline{WE}$ pulse memorizes data written in a register corresponding to the most significant addresses (figure 4.7). The initial page is guaranteed by a reset. Hence, the address limit in a 28 pin package is of 4 Mega bytes or 32 Megabits.

4.2.5 E²PROMs

EPROM floating array capacitors can also be discharged by electric pulses. Nowadays, EEPROM circuits, also known as E²PROMs (*Electrically Erasable PROMs*) or *Flash memories* do not require special voltage supplies but are too expensive for their use to be generalized. The number of write cycles may not exceed 1000, which is very low. A write cycle lasts between 0.1 and 10 ms; after writing in an input latch, a pin (READY or $\overline{BUSY}$ signal) or an internal flag must be tested before writing the next word. So as to avoid having a flag at a predefined address, it is possible to read the complement of the written value, as long as the internal cycle is not over. Every write cycle is followed by read operations until the read value is identical to the written one. E²PROMs are available in packages which are compatible with static RAMs for sizes between 512 and 32k bytes.

NOVRAMs (*NonVolatile RAM*) include both a RAM and and E^2PROM, thus avoiding the stumbling block formed by the length of the write operation and the limited number of write cycles; a series of commands allow information to be transferred between the two memories (figure 4.7). These NOVRAM memories are usually equipped with a serial interface which restricts the package size. A shift register is used to transmit in series a command and an address. The next pulses shift the contents (in or out). The contents of the RAM can be stored in the NOVRAM by using a command transmitted in series or by activating a special line; a circuit which keeps a watch on the power supply can therefore automatically save information then retrieve it. Consumer products often use this device to memorize the selected channel.

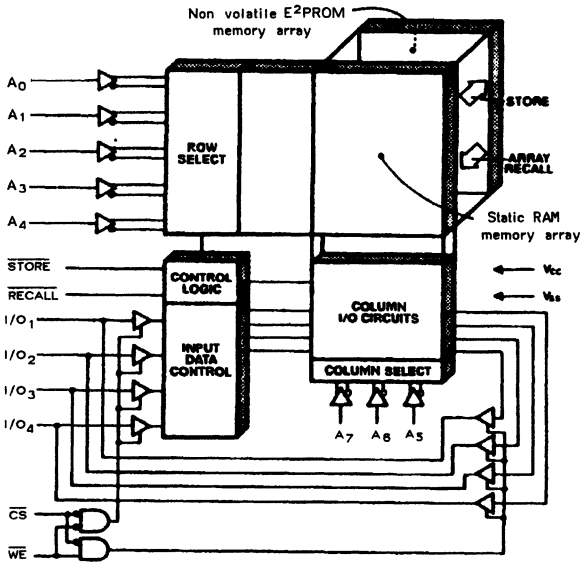


Fig. 4.7 Basic NOVRAM diagram

4.3 Read Write memories

Random access memories store information inside computer systems of any complexity. For the past 15 years, technological progress has been so rapid that, every year, memory capacity has increased by 70% and speed by 15%.

4.3.1 Static RAMs

A memory bit requires between 4 and 6 transistors to memorize and access information. Figure 4.8 shows a CMOS technology memory bit. In order to switch over the memory element, it is necessary to short-circuit the output using selection transistors which control X and $\overline{X}$ and have a lower internal resistance.

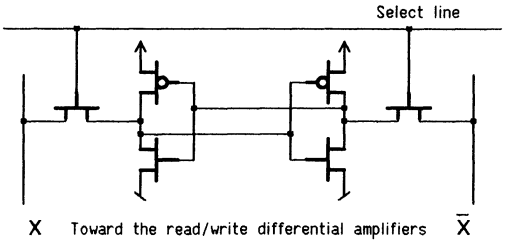


Fig. 4.8 CMOS memory cell

Two different approaches coexist as regards memory access:

- a $\overline{CS}$ select signal and a R/$\overline{W}$ transfer direction signal (more simply known as $\overline{WR}$). If R/$\overline{W}$ is at "1", a read cycle occurs, if it is at "0" a write cycle occurs (figure 4.9 a and b).
- a $\overline{CS}$ select signal, a $\overline{WRP}$ (*write pulse*) write signal and an $\overline{RDP}$ (*read pulse*) read signal. The latter is in fact an $\overline{OE}$ (*Output Enable*) which activates the three-state output drivers. A read access occurs if $\overline{CS}$ is active and $\overline{RDP}$ becomes active. A write access corresponds to $\overline{CS}$ active and an active $\overline{WRP}$ signal (figure 4.9 c and d).

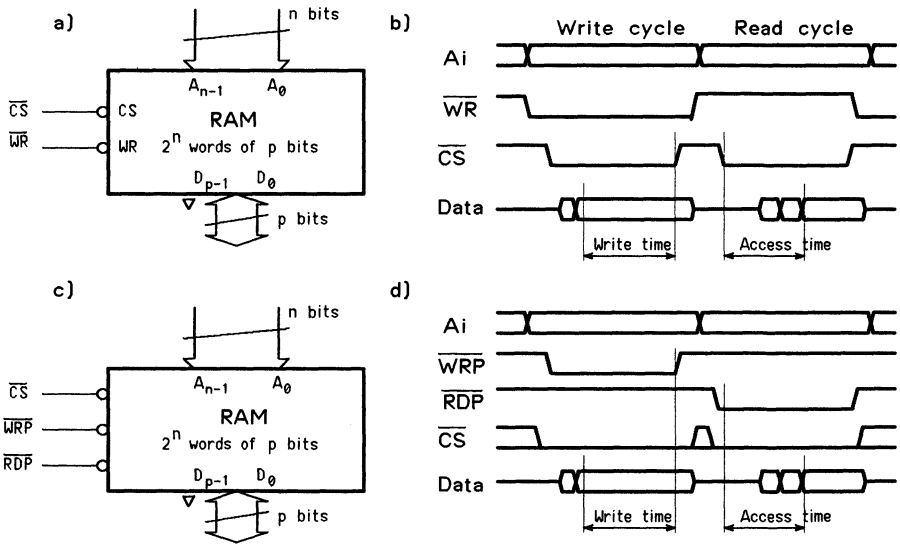


Fig. 4.9 Bloc and timing diagram for a RAM

Pseudo-static memories are dynamic memories (section 4.3.4) which are automatically refreshed. As it is impossible to refresh and access simultaneously, refresh operations are carried out between two accesses or by keeping a watch on a line known as "ready" or "busy" and extending the cycle if an internal refresh operation has begun just before the access.

4.3.2 Serial memories

Shift registers were formerly used as memories, especially for the first video screen interfaces. This technology was dropped since it is easy to simulate a shift register using a memory whose addresses are controlled by a counter.

There are a few specialized circuits available which resemble long shift registers and are especially appropriate for treating video signals. The Hitachi 63021 circuit (*16k Line Memory for Video Processing*) can be used to generate variable delays and interpolate on a screen. CCD (*Charge Coupled Device*) cells can be used to make up analog delay lines used in video as well as linear or plane sensors for image processing (solid state camera).

4.3.3 Dual ported memories

Many applications require memories which can be independently accessed by two sub-systems (*Dual ported RAM*). Theoretically, all that is required is a shifter on the data and address lines and an arbitrer to manage simultaneous accesses. The memory sends out an acknowledge signal (figure 4.10a). *Register banks* are rapid TTL circuits which are designed to be used in aritmetic units. Their input and output accesses are quite separate (figure 4.10b).

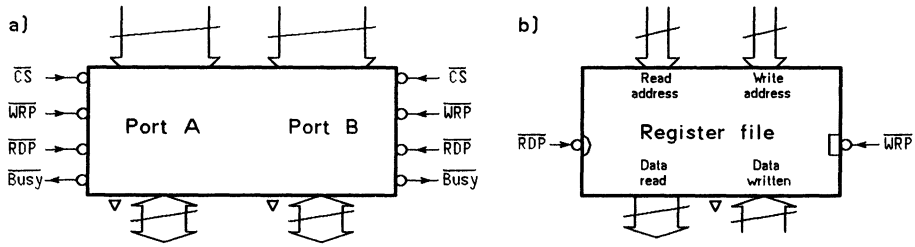


Fig. 4.10 Dual ported memory and register bank

The catalog of dual-ported memories is very poor; functionality is often linked to a given application.

4.3.4 Dynamic RAMs

Dynamic RAM memories or DRAMs use a single transistor to memorize an information bit and another one to access it. The bit is memorized in the parasite capacitor which lies between the gate and the substrate in a MOS transistor. As the capacitor discharges, it must periodically be refreshed by a read or a write access. A special *refresh* cycle occurs on average every 16 μs in most memories, thus guaranteeing that all positions are refreshed often enough.

Dynamic RAMs are organized in matrixes. Addresses are multiplexed and give access first to a *row* and then to a *column*. This is possible thanks to two select signals $\overline{RAS}$ (*Row Address Strobe* which validates the row address and is active on the falling edge) and $\overline{CAS}$ (*Column Address Strobe* which validates the column address and is active on the falling edge). An R/$\overline{W}$ signal specifies in which direction the read or write transfer occurs. When $\overline{RAS}$ is activated, an important current has to flow from the power supply: careful decoupling (100nF close from each chip) is required.

Dynamic RAMs can be 1 or 4 bits wide. One-bit wide circuits are usually equipped with a separate input and output which are interconnected in simple operating modes.

At any given moment in time, dynamic RAMs offer 4 times more storage capacity than static RAMs. In 1991, 4Mbit memories are being manufactured in series, 16 Mbit memories are available in small quantities and 64 Mbit models are being developed in laboratories. Power supply for 16 and 64 Mbit memories will fall to around 3,3V so that size can be reduced even further (electrostatic breakdown risks between conductors placed close to one another).

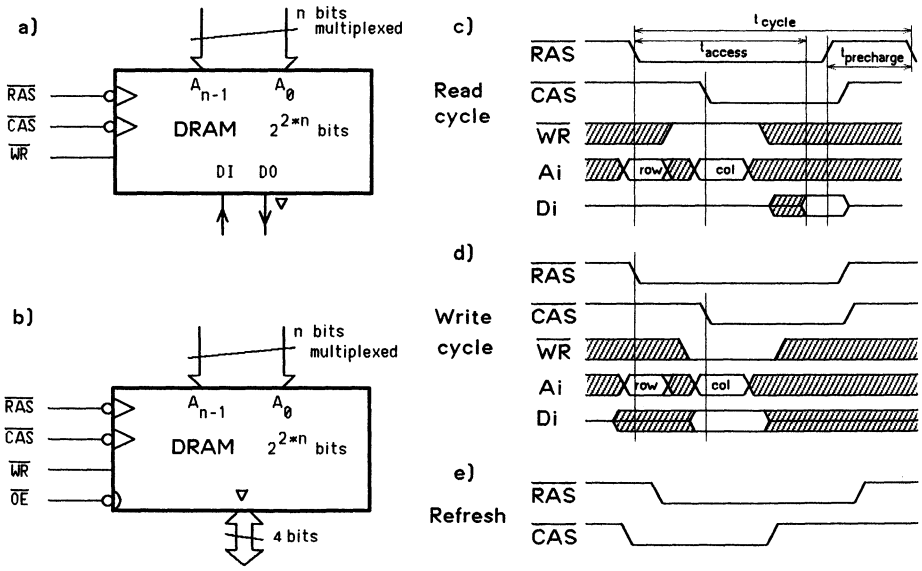


Fig. 4.11 Symbol and equivalent diagram of a dynamic RAM

Figure 4.11 presents the symbol of a dynamic RAM and its three main cycles: read, write and refresh. Moreover, dynamic RAMs offer other refresh cycles, read-modify-write cycles and faster cycles for accessing in the same row (page mode, static column).

A refresh cycle acts on all cells in a row; an internal counter gives the address of the row to be refreshed. It is interesting to note that cycle time is approximately twice as long as access time; internal amplifier capacitors must be reinitialized before each cycle, and this requires a rather long *precharge time.*

Typical DRAMs are shown in figure 4.12. The same package is used for two successive generations of memories since a single pin is enough to multiply the capacity fourfold. Package shape is also changing. The traditional DIL package is gradually being replaced by the zig-zag or flat package. Printed circuit modules equipped with 8 or 9 memory circuits offer a satisfactory solution for system evolution and reduce necessary space.

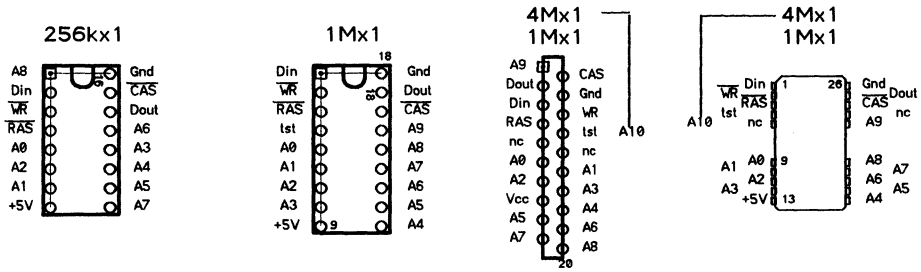


Fig. 4.12 Sample DRAM packages (scale 1)

4.3.5 Video RAMs

In a dynamic RAM, the falling edge of the $\overline{RAS}$ signal triggers a read and refresh cycle for all of the selected row. Hence, a 64k memory carries 256 amplifiers working in parallel; the requested data is chosen according to the second part of the address, valid on the falling edge of $\overline{CAS}$. Associating each output amplifier with a shift register cell and allowing this serial information to be shifted through extra lines has proved very satisfactory. This is a particularly interesting solution when building video screen interfaces, hence the name videoram.

If the RAS, CAS and DT lines are appropriately sequenced, normal accesses can be distinguished from transfer accesses towards the serial registers called (*SAM Serial Access Memory*) [Nicoud88]. Videorams come in a 64kx4 size, which is very suitable for screens. The 256kx4 type has been available since 1989 in a 28 pin package.

4.3.6 FIFO memories

FIFO (*First In First Out*) memories are dual ported memories offering two accesses, one of which is only a write operation and the other a read one. There is no addressing; incoming information is stocked in the fifo and the first input becomes the first output. In order to function properly, fifo memories require a write pulse, a read pulse and orders to stop reading if the fifo is empty or to stop writing if it is full (figure 4.13)

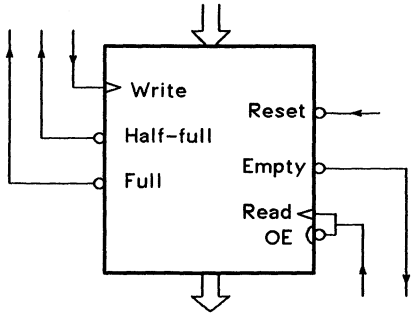


Fig. 4.13 FIFO memory circuit

According to the technique used to build the fifo, it is necessary to distinguish access time from *fall through* time. Available fifo circuits vary in size from 64x5 to 4096x9. Internally, they consist either of latches with an asynchronous transfer logic (figure 8.31) or of an addressable memory which is selected by two counters.

4.3.7 LIFO memories

LIFO memories (*Last In First Out*) are seldom used as individual components since they are easy to build using a RAM and an up/down counter. Microcontroller manufacturers offer a few low capacity circuits in connection with their applications.

4.3.8 Associative memories

It is difficult to cascade associative memories and the algorithm used for multiple comparisons is very dependent on the application. A few such circuits have been developed recently for special applications.

5 Programmable logic

Programmable logic can be used to reduce the number of components performing the same function and make corrections easier during the adjustment stage. Theoretically, the designer "only" needs to write the equations or draw a diagram based on a library of elementary functions before letting the appropriate software generate the circuit. In fact, he must choose the most suitable programmable components, share out the overly complex functions between various circuits and spend a long time learning to master the possibilities and limitations of programmable circuits and of the associated development tools [Bolton90] [Wakerly90].

5.1 Regular logic

The basics of programmable logic have already been shown in figures 2.21 and 2.22. This chapter goes over the main principles and describes in detail how to use PALs®, which have become widespread in the industrial world since 1980. They resemble PLDs, GALs, LCAs and other programmable devices described later and marketed under a series of names, which really refer to very similar concepts presented in quite different material and software packages.

In this book, we shall use the generic term *PLD* (*programmable logic device*) since *PAL®* is a registered trade mark which can only be associated with MMI/AMD products.

5.1.1 A simple example

Let us consider an elementary adder equipped with three inputs A, B and C as well as with two outputs S and R (figure 5.1). An analysis of the binary addition operation shows that output S (sum) is active if only one input is active or if the three are active. Output R (report) is active if at least two of the outputs are active. Both Karnaugh tables and direct and simplified logic equations can therefore be drawn up using + for a logic OR, implicit * for logic AND and overlining for inversion:

$$S = A\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + \overline{A}\,\overline{B}\,C + A\,B\,C$$
$$R = A\,B\,\overline{C} + A\,\overline{B}\,C + \overline{A}\,B\,C + A\,B\,C$$

$$S = A\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + \overline{A}\,\overline{B}\,C + A\,B\,C$$
$$R = A\,B + A\,C + B\,C$$

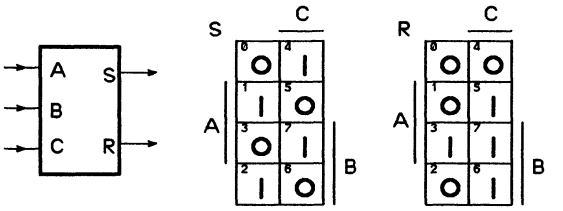


Fig. 5.1 Binary adder

5.1.2 Solution based on a decoder

A decoder or demultiplexor generates all the minterms of the input variables. Any simple logic function can therefore be performed easily (figure 5.2a). The simplified notation of figure 5.2b is very clear and will often be used hereafter. The large dots pinpoint fixed links and the crosses show the links which depend on the application.

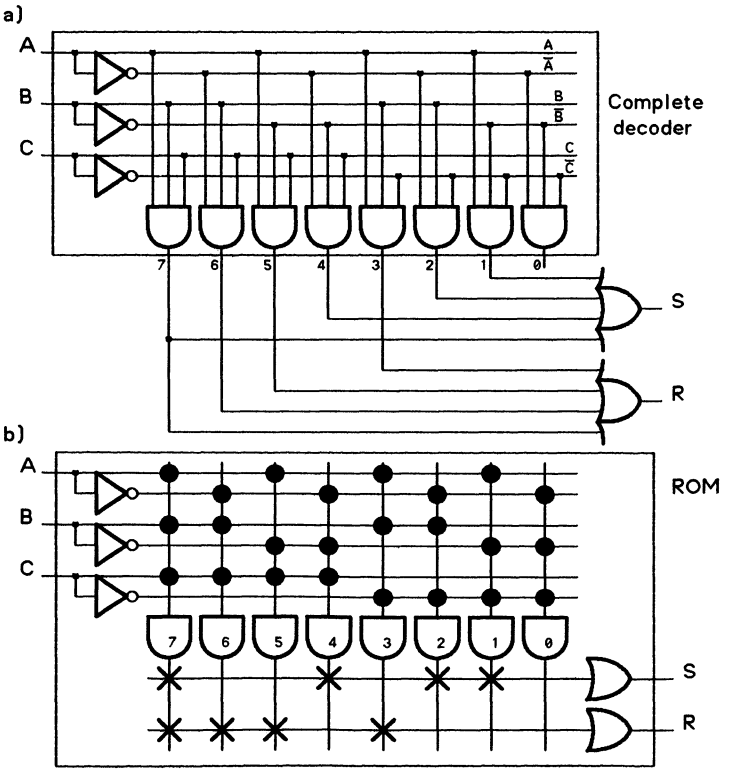


Fig. 5.2 Adder made up using a decoder

A ROM is in fact a decoder followed by a large OR gate for every output bit. It is therefore easy to perform any logic function. Figure 5.2b shows how to program an 8x2 (8 2-bit words) PROM to build the adder. The crosses correspond to transistors or fuses which are maintained or created by the programming process.

When dealing with more complex logic functions, the size of the decoder increases exponentially with the number of inputs and therefore rapidly makes solutions including multiplexors or ROMs quite unsuitable.

5.1.3 Completely programmable solution

A ROM usually includes several unused lines. *PLAs* (*Programmable Logic Array*) can be used to program only the significant lines of the decoder. Such devices enable the designer to start with a simplified function in order to reduce the number of terms and the possible sharing of common terms (figure 5.3).

Some applications do not of course use all the available terms of a given PLA circuit, whereas others cannot be implemented because of the limited number of terms.

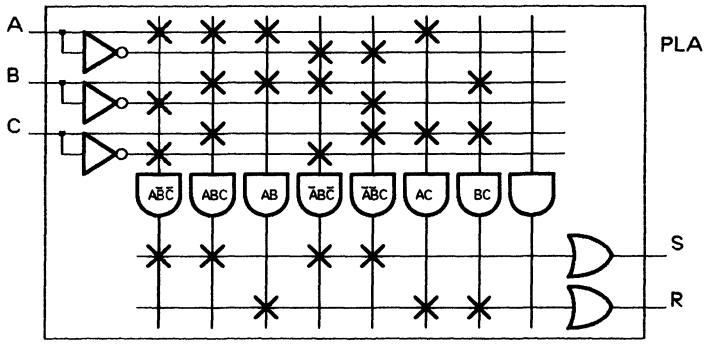


Fig. 5.3 Adder based on a PLA

Size is characterized by the number of inputs, the number of minterms and the number of outputs. In the example shown in figure 5.3, the minimum size is 3 x 7 x 2.

5.1.4 Partially programmable solution

Known as a PAL®, MMI's invention is a PLA which only includes programmable minterms. The sum is predefined and for the same complexity (total number of programming fuses) different circuits can be built. This is why they are grouped by families in PLD catalogues. The circuit of figure 5.4 uses a PLD whose OR gates are equipped with four inputs. Since the AND output gates are all wired to the OR gates, the useless

AND gates must be neutralized meanwhile guaranteeing that the output remains at zero; for example, it is possible to create an $A \times \overline{A}$ minterm and leave all the fuses intact.

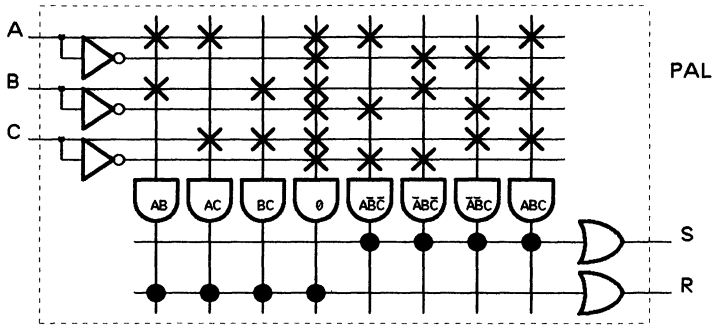


Fig. 5.4 Adder built using a PLD

The size of a simple PLD is characterized by its number of inputs, its number of minterms for each output and its number of outputs. The PLD of figure 5.4 is of the 3x4x2 type. More complex PLDs are available and will be discussed in section 5.3.

5.1.5 PROM logic

Unnecessarily called PLEs (Programmable Logic Element) by one manufacturer, PROMs and EPROMs (section 4.2) can be used to make up any combinational system and are essential when dealing with complex functions. However, the number of memory elements increases exponentially with the number of address inputs and solutions resorting to large ROMs are usually quite costly and rather slow. Moreover, they dissipate a lot of power.

PROM logic allows the designer great latitude. Input signals can be either direct or inverted ones but the permutations which must be made in the truth table require a lot of attention. In order to increase outputs, several memories must be connected in parallel or in series, as is the case in microprocessor memories.

Most PROMs are prone to large hazards when inputs vary; the output must always be synchronized by a parallel register.

The contents of a PROM are usually prepared with an assembler program using a truth table. Filling a large PROM can prove quite tedious. Repetitive zones are easily generated with a good editor or a macroassembler. Specialized tools (PLE-ASM, LogIC) generate an EPROM from logic equations.

5.1.6 PLA logic

PLAs or *FPLAs* (*Field Programmable Logic Array*) or *IFLs* (*Integrated Fuse Logic*) are characterized be their number of inputs, their number of minterms or products and their number of outputs. The functionality as well as the number of alternatives at the input-output level has a lot in common with PLDs, which will be discussed in the following section.

Standard PLAs did not catch on because of manufacturing and implementation difficulties. Inside integrated circuits (microprocessors, programmable interfaces), almost all combinational and sequential logic is performed using PLAs strewn around the circuit.

For many years, Philips/Signetics and Texas have been trying unsuccessfully to manufacture and market their PLAs. Philips has called its family IFL (Integrated Fuse Logic) and offers variants known as FPGAs (Fuse Programmable Gate Array), FPLAs (Fuse Programmable Logic Array) and FPLS (Fuse Programmable Logic Sequencer). Only few FPLA/FPGAs are available; they are rather expensive, they dissipate a lot of power and associated programmers are hard to find.

PLAs are not yet widespread and corresponding development tools are quite expensive. Information about available development tools and prices can be obtained from PLA and programmer manufacturers.

5.2 PLD logic

PLDs have been very successful because they offer a good technological compromise, quite a complete catalogue and are relatively easy to use. *GALs* (*Generic Array Logic*) and *EPLDs* (*Erasable PLD*) are new PLDs whose greater universality allows the designer to simulate most first generation PLDs (section 5.3.7). They have the advantage of being electrically or UV erasable.

5.2.1 Simple PLDs

A *PLD* is equipped with a certain number of inputs and outputs. Each output is an AND/OR function of a predefined number of minterms. Minterms can be programmed by blowing fuses. The presence of all these fuses is sometimes indicated by a simple cross in the AND gate (figure 5.5). If all the fuses are present, the AND gate output is at zero, which is a neutral condition. If all the fuses are blown, the AND gate output is still active, which also jams the OR gate output. The two representations of figure 5.5 are used by manufacturers.

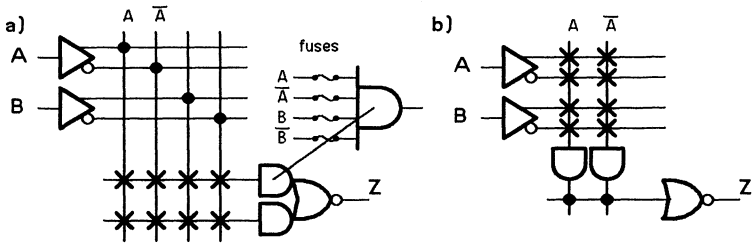


Fig. 5.5 Diagram and symbolism for PLD circuits (all fuses are in place)

Historically, two types of PLDs were developed and are still widely used. The H series has direct outputs as shown in figure 5.6. The L series has inverted outputs. Since any logic function can be obtained in a complementary form, simple functions can be performed quite easily using one or other type of PLD. More complex functions may be possible with only one of the types of PLDs, or with neither when the function cannot be simplified sufficiently.

Figure 5.6 shows how to implement elementary functions with both types of PLDs. For PLDs with an inverted output, it is important to distinguish clearly between the name of the output variable, which is associated with the integrated circuit pin number, and the name of the variable before the output has been inverted, which corresponds to the generated equation.

One kind of PLD is characterized by the letter X and includes the XOR function in order to simplify adders and parity generators. This type is seldom found in very optimized schematics and is being dropped by manufacturers.

In order to program a PLD, one need to provide in a file the signal names with their true polarity, the related pin numbers and then the equations. Examples can be found in figures 8.20 et 8.36.

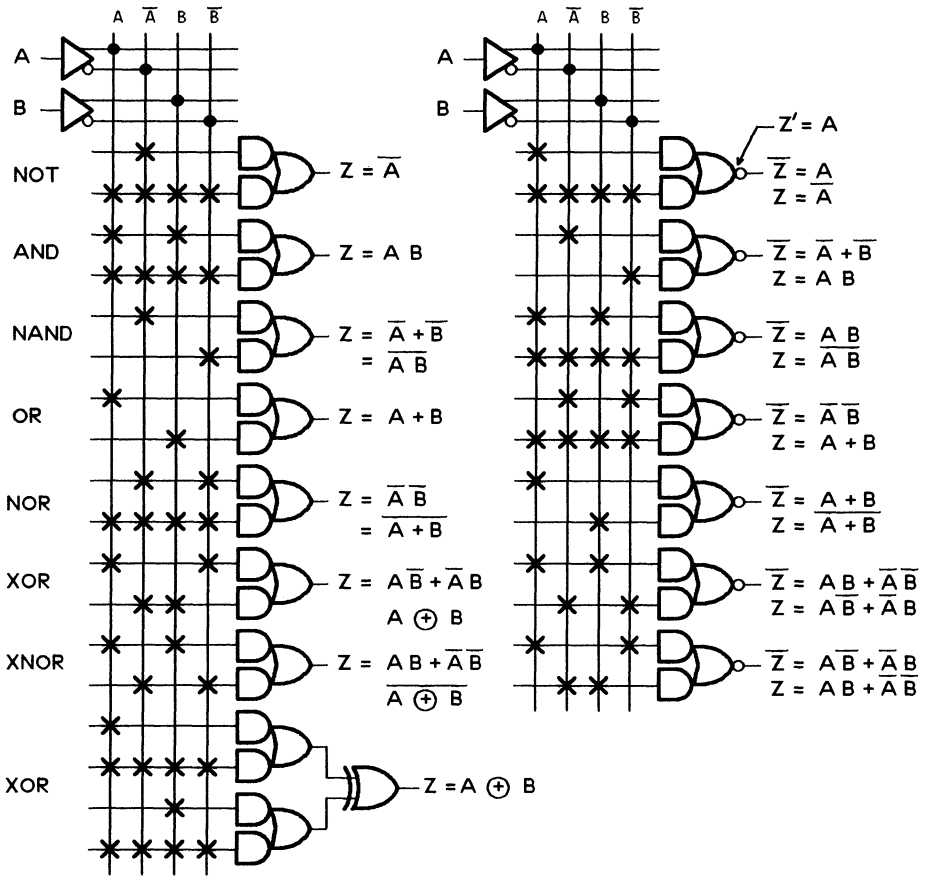


Fig. 5.6 Elementary functions with H and L series PLDs

Representing the function through a fuse plan, a practice manufacturers are rather too fond of when writing application notes, helps to understand the internal structure of PLDs. It is useless if the designer has access to a PLD assembler (Palasm, Abel, Cupl, LogiC, etc.), an absolute necessity in order to get good results. Reasoning based on equations is quite simple and efficient.

5.2.2 Simplifying logic equations for PLDs

In some cases, the natural specification of a PLD output is an equation. For example, output S must be active if A and B are active, **except if** C **or** (D **and** E) are active. The equation can be written as

$$S = A*B*(\overline{C + D*E})$$

To be suitable for a PLD, any equation must be reduced to a sum of products. In the above case, we have

$$S = A*B*(\overline{C} * \overline{D*E}) = A*B*\overline{C}*(\overline{D}+\overline{E}) = A*B*\overline{C}*\overline{D}+A*B*\overline{C}*\overline{E} \qquad (1)$$

If this function is to be performed by a PLD equipped with an inverter (type L), we can therefore write

$$\overline{S} = \overline{A*B*(\overline{C+D*E})} = \overline{A*B} + (C+D*E) = \overline{A}+\overline{B}+C+D*E \qquad (2)$$

In the latter case, it is obvious that a PLD with at least 4 minterms must be used. If the above S function is to be included in a 12H6 type PLD, equation (1) must be used in the form required by the assembler (figure 5.7). If it is a 12L6 type PLD, equation (2) should be used and care should be taken to use one of the two acceptable outputs (only outputs on pins 13 or 18 in figure 5.7b can stand sums of 4 minterms).

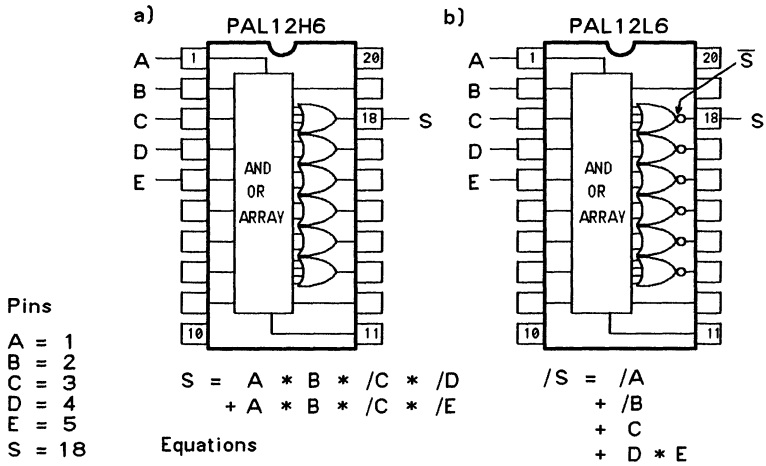


Fig. 5.7 Implementing the S function in two real PALs

It is interesting to see that the PLD assembler does not deprive a variable of its inverted value, be it on an input or on an output. The equation must define the function before the output inverter.

Choosing between direct or inverted functions is easier with new PLDs, whose polarity can be programmed: an "exclusive OR" gate equipped with a fuse on one input can be used to program the output inverter or not. Such PLDs often carry the letter P in their number (e.g. 16P8)

5.2.3 Correcting existing PLDs

Unlike EPLDs and GALs, which can be erased using ultraviolets or electrically, fuse PLDs containing mistakes must be thrown out and redone. In some cases it is possible to "over" program in order to correct certain functions. A factor can be taken away from a minterm by blowing the corresponding fuse. If there is a free row of fuses, an extra minterm can be added (figure 5.8).

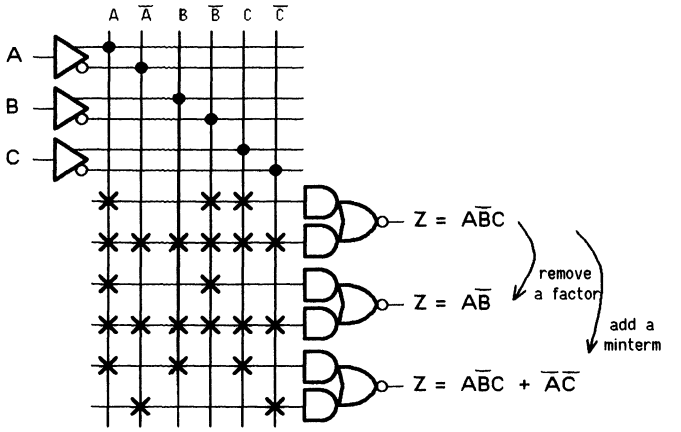


Fig. 5.8 Correcting a previously programmed PLD

5.2.4 Application example: 68008 decoder

A memory card for a 68008 must include 8 K octets of ROM, 8 K of RAM and a few input/outputs. During start-up, the ROM must take the place of the RAM, but only for read purposes. Selecting the normal ROM address reestablishes the final memory plan. A BOOT flip-flop is required to differentiate between these two working modes. The most significant addresses are decoded in order to create an 8 K input/output space. Figure 5.9 shows the memory map and the block diagram of the required interface.

Memory space is not completely decoded since the 2 high address lines A19 and A18 are ignored. 8k address groups are defined since lines A12 to A0 do not go into the decoder. Peripherals are decoded by a standard decoder, which is cheaper and consumes less power than a second PLD or a PLD equipped with more inputs and outputs.

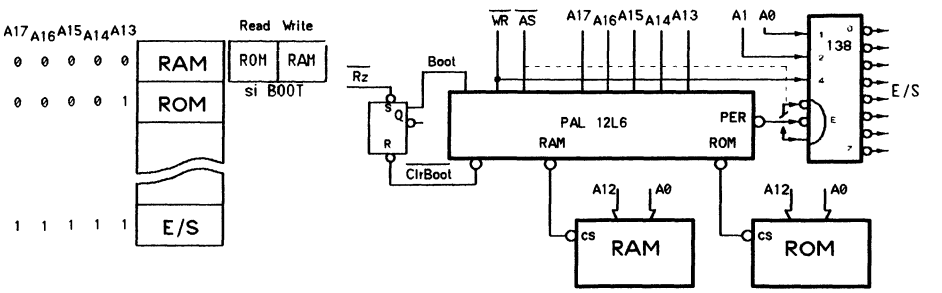


Fig. 5.9 Memory interface for the 68 008

The selected PLD is a 12L6 in a 20-pin package. It includes 12 inputs and 6 inverted outputs. Input signals are A13, A14, A15, A16, A17, /WR, /AS and BOOT. Output signals are /ROM, /RAM, /PER and /CirBoot.

The equations can be written directly, keeping in mind the function to be performed: the ROM is selected at its normal address plus that of the RAM if BOOT is active in a read cycle. The RAM is selected if BOOT is not active or when dealing with a write cycle. Peripherals are selected within their address space. The BOOT flip-flop is cleared when the ROM is selected at its normal address.

```
ROM= /A17 * /A16 * /A15 * /A14 * /A13 * AS * BOOT * /WR
   + /A17 * /A16 * /A15 * /A14 *  A13 * AS
RAM= /A17 * /A16 * /A15 * /A14 * /A13 * AS * /BOOT
   = /A17 * /A16 * /A15 * /A14 * /A13 * AS * WR
PER=  A17 *  A16 *  A15 *  A14 *  A13 * AS
ClrBoot/A17 * /A16 * /A15 * /A14 *  A13 * AS
```

Once this stage has been reached, the design is complete. Pins must be attributed before the PLD can be simulated and programmed if it meets requirements. Since some of the PLD lines are not used, other functions can be included. The BOOT flip-flop, which is nothing but a simple SR flip-flop, can easily be programmed in the PLD, as will be shown later.

A careful study of figure 5.9 and the equations shows that the peripherals are selected with a delay related to $\overline{AS}$ equal to the sum of the propagation times inside the PLD and inside the decoder. PLD time can be saved by wiring $\overline{AS}$ onto the decoder (see dotted lines), and removing AS from the PER equation.

5.2.5 Asynchronous sequential logic

Simple static SR flip-flops (figure 3.2) and latches (figure 3.8) can easily be made up using a single PLD output as shown in figure 5.10. It is important to remember that, in order to guarantee that such an asynchronous system functions well, it must include unnecessary minterms according to the simplification of the logic functions (section 3.18).

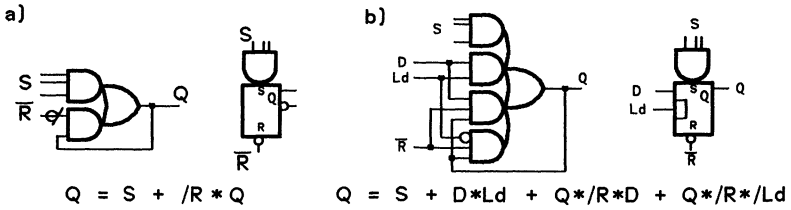


Fig. 5.10 Static flip-flop and latch made up using a PLD

5.2.6 Example: boot flip-flop for 68008

Let us examine once again the example of the 68008 decoder (section 5.2.4). The BOOT flip-flop can be included in the PLD instead of the ClrBoot output which controls it. BOOT must be activated as

soon as Rz is active and must retain its value as long as the CirBoot function is not true. The equation to be performed is:

$$
\begin{aligned}
\overline{BOOT} &= RZ + BOOT * \overline{/ClrBoot} \\
&= RZ + BOOT * \overline{(/A17 * /A16 * /A15 * /A14 * A13 * AS)} \\
&= RZ + BOOT * A17 + BOOT * A16 + BOOT * A15 \\
&\quad + BOOT * A14 + BOOT * /A13 + BOOT * /AS
\end{aligned}
$$

This equation is problematic for it includes too many minterms for most PLDs. It is necessary to synthesize the inverted function which comes to mind naturally when thinking about normal working conditions for the NoBoot=$\overline{BOOT}$ variable: it becomes active when the ROM is selected and remains so as long as the reset is inactive:

$$
\begin{aligned}
\overline{BOOT} = NoBoot &= ClrBoot + NoBoot * \overline{RZ} \\
NoBoot &= /A17 * /A16 * /A15 * /A14 * A13 * AS \\
&\quad + NoBoot * /RZ
\end{aligned}
$$

This equation only includes two minterms and can easily be implemented. In order to modify the other equations, BOOT must be replaced by /NoBoot and /BOOT by NoBoot.

5.2.7 Three-state output PLDs

Some combinational PLDs are equipped with three-state outputs which can be controlled by a single minterm (figure 5.11a). Moreover, these three-state outputs are also inputs, which allows for great flexibility.

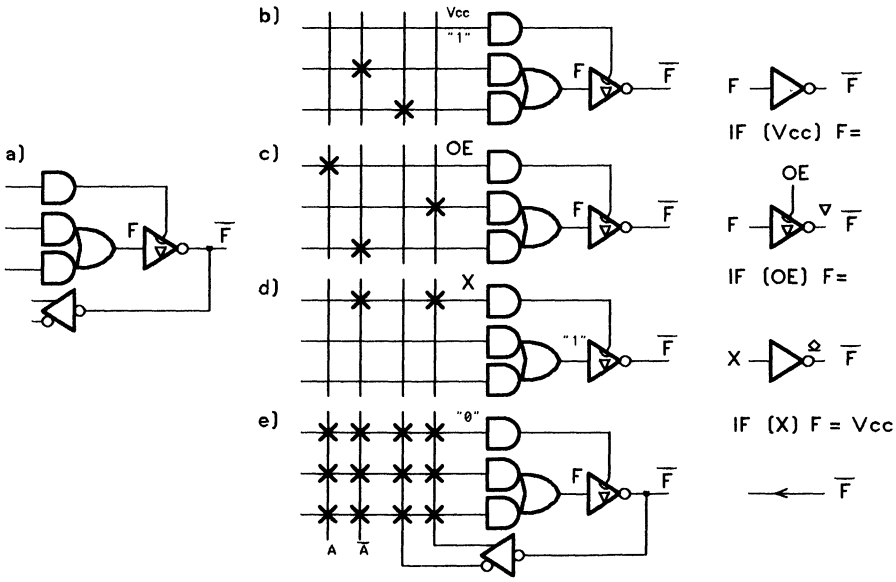


Fig. 5.11 Use of a three-state output (for example 16L8 type PLD)

The designer can simulate a normal output by blowing all the fuses of the product which controls the three-state output (figure 5.11b), make up a three-state output (figure 5.11c), simulate an open collector output by placing the output logic function on the three-state control (figure 5.11d) or even simulate a simple input by leaving all the fuses in place (figure 5.11e). Building an open emitter is a suggested exercise. The syntax of the PALASM program is a little special, for it includes an IF to say which product controls the three-state output. Another notation writes F.OE for the term that controls the OE related to F output.

5.2.8 Application example: resetting the 68000

The 68000 and 68008 have a rather special encoding on the $\overline{\text{Reset}}$ and $\overline{\text{Halt}}$ lines. These lines sometimes function as inputs, sometimes as outputs, according to the following truth table:

$\overline{\text{Reset}}$	$\overline{\text{Halt}}$	
1(in)	1(in)	Normal execution
1(in)	0(in)	The processor stops working
0(in)	0(in)	Processor reset
0(out)	1(out)	The RESET instruction is performed
1(out)	0(out)	Double bus error, the processor refuses to continue

The end-user must therefore use two lines to control the reset and suspend activity; moreover, he must be sent a logic error signal if the processor stops without being asked to. Figure 5.12 shows the gate diagram and the bloc diagram using a 16L8 type PLD or equivalent.

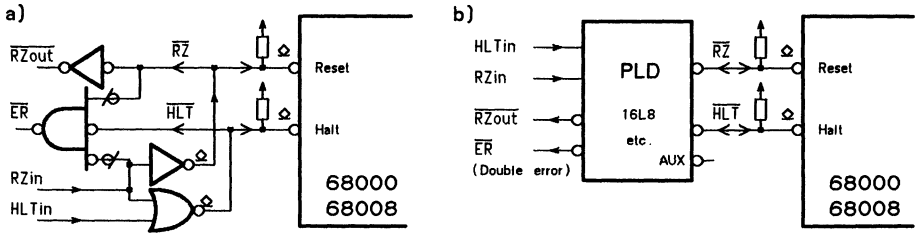


Fig. 5.12 Reset and Halt circuit for 68 000 and 68 008

The PLD inputs are the RZin and HLTin signals. The outputs (some of which are also inputs) are $\overline{\text{ER}}$, $\overline{\text{RZout}}$, $\overline{\text{RZ}}$ and $\overline{\text{HLT}}$.

The equations are given hereafter. They can be derived from a truth table which shows all possible cases. It is easy to find them directly by using common sense to analyze the necessary operations, as shown in the diagram of figure 5.12a. Let us note that these equations are written for signals coming from outputs which are active at "1", since an inverted PLD has been chosen.

IF (RZin)	RZ	= 1	(1)
IF (RZin+HLTin)	HLT	= 1	(2')
IF (VCC)	RZout	= RZ	(3)
IF (VCC)	ER	= HLT * /RZ * /HLTin	(4)

For the selected PLD, and more generally all PLDs equipped with programmable three-state outputs, it is impossible to activate the HLT output (RZin+HLTin), since the function is the sum of a series of products and not a simple product. This function must therefore be generated in the same way as an auxiliary variable. Such a solution uses an input/output pin and adds 20–40 ns propagation time, which does not cause any inconvenience in the case of this particular application.

The second equation becomes:

IF (AUX)	HLT	= 1	(2)

The auxiliary output equation is:

IF (VCC)	AUX	= RZin+HLTin	(5)

If the PLD outputs are equipped with inverters, which is the case of the 16L8 shown in figure 5.12b, all the designer has to do is give equations (1) to (5) to the PLD assembler. If there is no inverter, the equations must be inverted, as seen in section 5.2.2. Whenever it is handy to go into the PLD with the real inverted RZin signal instead of the direct one, the designer only needs to change the list of signals and their corresponding pins. The corresponding equivalent diagram is given in figure 5.13 in order to materialize the transformation but, as is the case with more detailed diagrams including fuse plans, this type of diagram does not give any more information than that already offered by the equations as regards correct comprehension of functionality; moreover, it is of no help when looking for design mistakes.

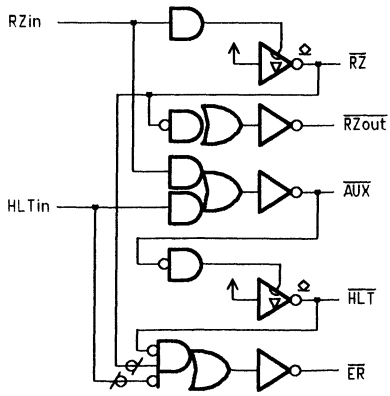


Fig. 5.13 Diagram equivalent to equations

5.3 Registered PLDs

Combinational PLDs can be used to design simple asynchronous systems, and be associated with flip-flops in order to build semi-synchronous or synchronous systems. Registered PLDs usually make sequential systems less complex.

5.3.1 Basic structure

Registered PLDs include several flip-flops which share a common clock. The outputs are three-state ones but they are controlled by a common external pin. The flip-flop state (and not the output state) can be used internally (figure 5.14).

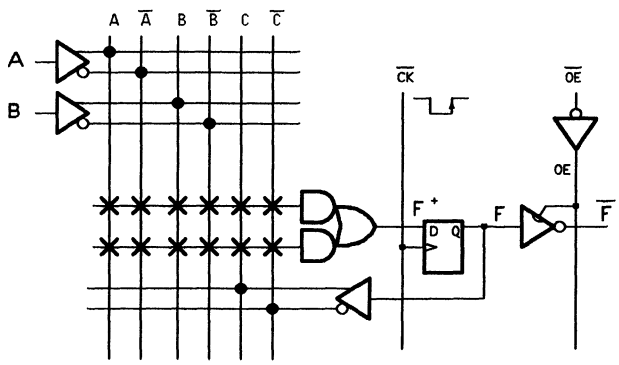


Fig. 5.14 Part of a registered PLD

Interestingly enough, the manufacturer's diagrams indicate that the $\overline{Q}$ flip-flop output is used as an internal variable, as is the case in figure 5.15. This can be used to locate the fuses exactly but is of no use to the designer using a PLD assembler. On the other hand, one can say that the outputs are inverted compared to the outputs of the corresponding flip-flops, i.e. compared to the next state indicated by the equation.

The equations concern the D input and define the next state. The := sign included in PLD equations replaces the + sign which usually characterizes this next state. Within the equations, the name of the next input variable of a flip-flop is identified using the flip-flop output name. Figure 5.15 shows how to build some simple flip-flops. The D-flip-flop has as an equation $Q^+=D$. The equation required in the case of figure 5.15a refers to the internal flip-flop input. Because of the output inverter, it should be written /Q:=/D. The same reasoning applies to figures 5.15b and c, which correspond to a JK flip-flop and a D flip-flop whose clock is dynamically inhibited.

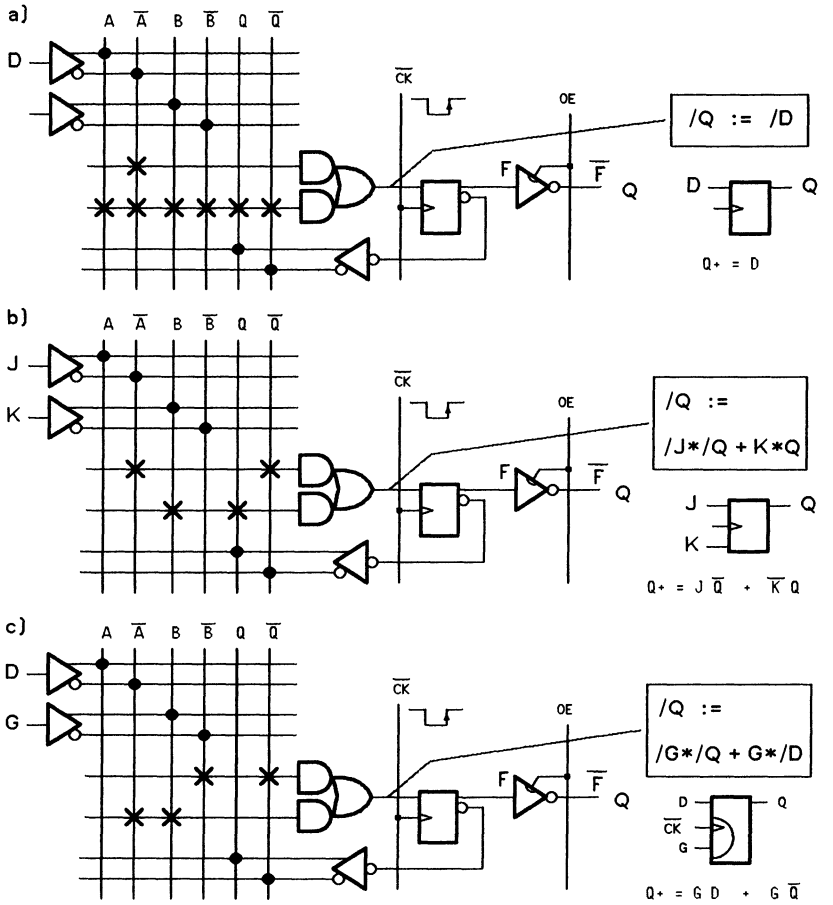


Fig. 5.15 Building common flip-flops using a registered PLD

5.3.2 Binary counter

A binary counter made up using D flip-flops is based on the diagram of figure 5.16. The first level changes state on every clock pulse, the second one changes whenever the first one is at state "1", the third one whenever the two previous ones are at "1", etc.

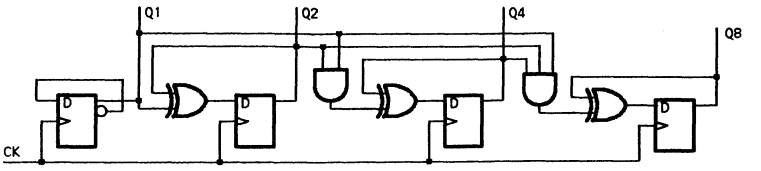


Fig. 5.16 Diagram of a binary counter (:16)

The corresponding equations are

Q1 := /Q1
Q2 := Q1 ⊕ Q2
Q4 := (Q1 Q2) ⊕ Q4
Q8 := (Q1 Q2 Q4) ⊕ Q8

These equations can be developed or simplified thanks to Karnaugh tables and lead to the following equations:

Q1 := /Q1
Q2 := Q1*/Q2 + /Q1*Q2
Q4 := Q1* Q2*/Q4 + /Q1*Q4 + /Q2*Q4
Q8 := Q1* Q2* Q4*/Q8 + /Q1*Q8 + /Q2*Q8 + /Q4*Q8

At this stage, the work is over. The equations are fed into a development tool. Fuse programming is preferably preceded by simulation.

5.3.3 Gray and special counters

Binary counter equations soon become very complex. PLDs adapted to this type of equation (16X8, 20X8) are required for 8-bit counters. A binary counter is seldom required. Gray counters lead to simpler equations and, as an extra advantage, enable glitch-free decoding. As another simplification, two counters can be cascaded inside the same PLD, as shown in figure 5.17. The first divide by 4 counter is a ring counter. The second is identical but includes a synchronous count enable (figure 3.37a); it can only count if the first counter is at state 11 (or any other state). This conditional evolution principle can be used in many cases. According to the application, the designer must find the counter structure which ensures the simplest design.

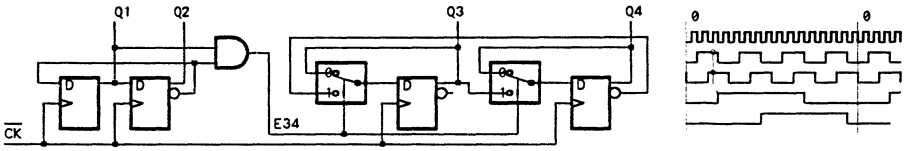


Fig. 5.17 Divide by 16 counter

The equations of figure 5.17 counter are copied directly from the schematic:

Q1 := /Q2
Q2 := Q1
Q3 := Q1*Q2 * /Q4 + $\overline{Q1*Q2}$ * Q3
Q4 := Q1*Q2 * Q3 + $\overline{Q1*Q2}$ * Q4

Developing the last two equations leads to the final equations:

Q3 := Q1*Q2*/Q4 + /Q1*Q3 + /Q2*Q3
Q4 := Q1*Q2* Q3 + /Q1*Q4 + /Q2*Q4

5.3.4 Independent clocks

A major defect of registered PLDs in many applications is due to clock sharing, which seems to prevent transitions being triggered by the edge of any signal other than a clock. In some PLDs, known as asynchronous ones, the clock on each flip-flop can be controlled by a product (figure 5.20). Such models are often rejected because of availability, price and speed.

This problem can be solved by synthesizing a D flip-flop asynchronously. This means using two daisy-chained latches equipped with an inverted clock, but it is necessary to check the effect produced by internal delays. The recommended solution (figure 5.18) sacrifices a flip-flop in order to generate a brief control pulse at the desired clock time and requires control pulses to be longer than the main clock period. If the CK signal is asynchronous compared to the CP clock, a metastable suppression flip-flop, shown by dotted lines in figure 5.18a, is required. Figure 5.18b gives as an example the analysis and equations of a T flip-flop, while figure 5.18c corresponds to a D flip-flop.

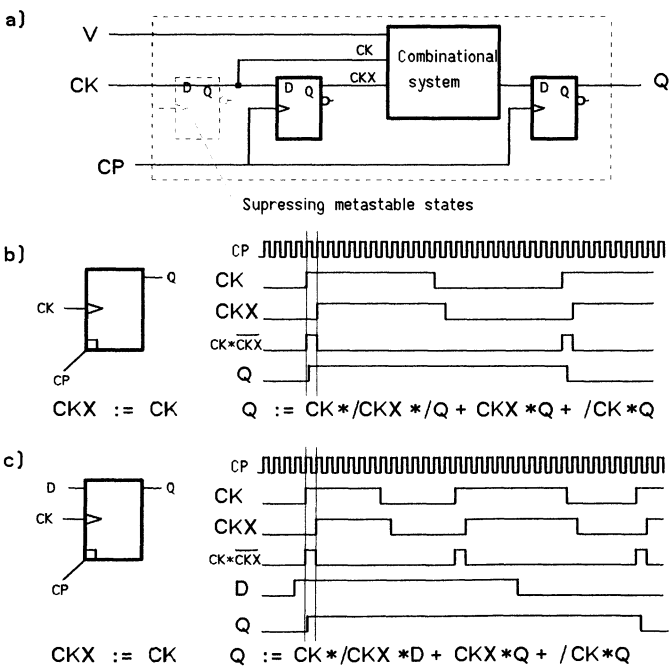


Fig. 5.18 Flip-flops equipped with independent clocks

For example, in order to build the two first levels of the counter appearing in figure 5.17 using an asynchronous clock which is slower than the PLD clock, the designer will have to add an auxiliary CKX flip-flop which is the copy of CK at the next CP clock time. Counter transition can only occur when CK * /CKX is active, if switch-over occurs on the rising edge of CK (in fact at the next CP clock time). The /CK * CKX product characterizes the falling edge. The schematic diagram of this counter can be found in figure 5.19; its undeveloped equations are as follows:

CKX := CK
Q1 := (CKX * /CK) ⊕ Q1
Q2 := (CKX * /CK * Q1) ⊕ Q2

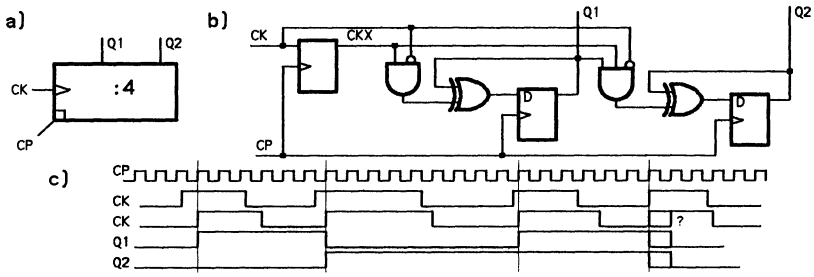


Fig. 5.19 Counter with independent clock

5.3.5 Tricks

PLDs cannot be used in non trivial applications without their limits becoming a constant hindrance: lack of inputs, minterms, outputs, common clock, activation of common outputs, no reset during start-up, high power consumption, relatively long propagation times, price, availability, need for a special programmer, 2nd source, etc. The PLD family is growing in order to overcome these inconveniences but the recently announced products are expensive and their availability seldom comes up to dealer promises.

If there are not enough inputs, it is necessary to use either a PLD whose outputs can be used as inputs or to encode input functions. When faced with a lack of outputs, the designer can use parallel or serial connections or even a crafty combination of both. As a general rule, two PLDs must always exchange encoded information. Decoding inside the PLD is never a problem.

When faced with a lack of minterms, the designer can invert the output (and add an outside inverter), use a different PLD, change the encoding or

even use two levels and let the first part of the function be performed outside or inside the PLD.

In order to improve output characteristics in the case of a PLD simulating an open collector output (figure 5.11d), it is wise to disable the three-state command after the main variable (one internal propagation time or one clock cycle). Tension is thus forced to "1" before becoming an open collector and rising time is significantly reduced.

5.3.6 Buried registers

A complex sequencer often uses many flip-flops whose output value is not used directly. Therefore, it is unnecessary to have an output amplifier and a pin for these *buried* flip-flops. Some PLDs have 8 flip-flops equipped with normal outputs and 6 buried flip-flops; unused buried flip-flops cost internal transistors and power but do not take up any room on the printed circuit.

5.3.7 Macrocells, PLDs and GALs

End-users are soon faced with the limits imposed by simple PLDs and would often like to be able to use a mixture of outputs (direct, inverted, registered, etc.), which are not available in catalogues. One solution is to plan all usual functionalities for each output and program the desired working mode. Figure 5.20 shows the architecture of a *macrocell* whose output can be a direct one, an inverted one, a combinational one or a register, according to the position of programmable switches. Retroaction can be initiated at three levels, which means that synchronous and asynchronous logic can be used and that the output can become an input and function as a buried register. The clock and the three-state output can be controlled from outside or by a product, allowing for certain limitations.

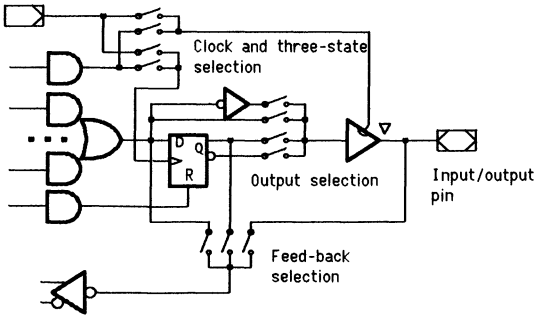


Fig. 5.20 Macrocell examples

The more universal the macrocell, the more expensive and power consuming it becomes, even in a CMOS version. Only "zero-power" versions offer really low power consumption. Each manufacturer is striving to find the best compromise. *EPLDs* (*Erasable Programmable Logic Device*) are PLDs whose fuses are floating gates of MOS transistors and can therefore be erased using ultraviolets. *GALs* (*Generic Array Logic*) can be erased electrically and their macrocell structure is designed to generate all simple PLDs. These abbreviations are linked to a group of manufacturers and the available products are still changing.

5.3.8 Existing PLDs

There is a large choice for PLDs. Most manufacturers name their part number with a similar syntax which helps when selecting parts:

- the first number corresponds to the maximum number of inputs
- the middle letter depends on the internal logic (figure 5.21)
- the second number corresponds to the number of outputs.

Hence, a 16R8 has up to 16 inputs and 8 outputs; letter R specifies registered outputs (figure 5.15). Speed is shown by appending as a suffix -A, -B or -D (for 30ns, 15ns, 10ns access time), or -30, -15 and -10, which is clearer. Old PLDs of the L and R type are the most widely used models, since they are available through many manufacturers with a wide choice of speed and power consumption. Moreover, they are cheap and easy to program.

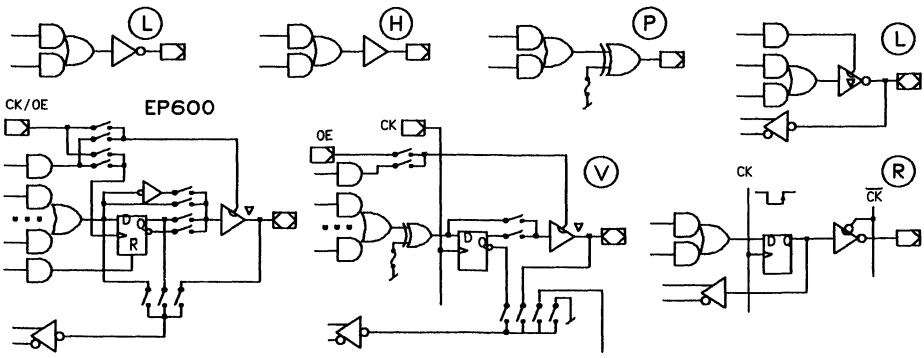


Fig. 5.21 Summary of PLD cells

5.3.9 Future PLDs

Due to its success, the PLD family is still developing. The aim is to increase the number of fuses as well as the flexibility and universality of internal functions. Adding more pins avoids resorting to several PLDs. New complex PLDs include specialized logic to be used when developing interfaces or sequencers.

Another interesting step forward has meant placing several programmable cells on a circuit, and programming the connections between cells. *LCAs* (*Logic Cell Array*) and *FPGA* (*Field Programmable Gate Arrays*) include a large network of relatively simple cells whose interconnections can be programmed (figure 5.22b,c).

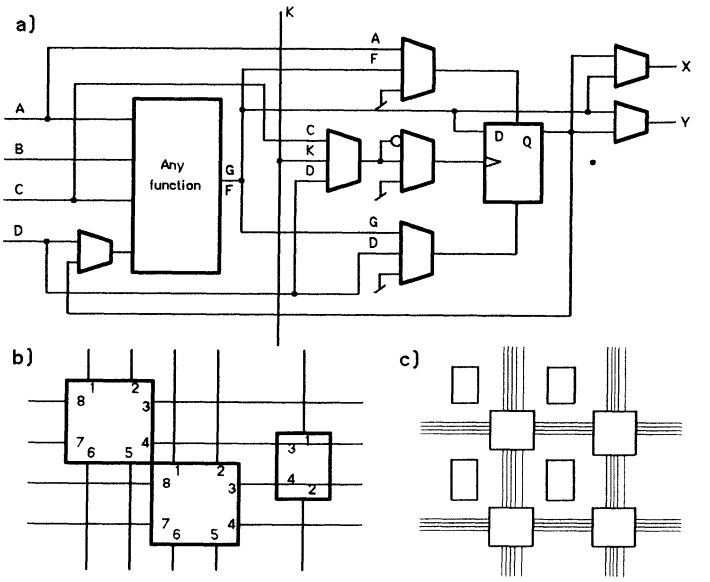


Fig. 5.22 Macrocell and links inside LCAs

Software tools which resort to the TTL library can be used to develop complex programmable circuits. The problem with LCAs in many applications is that they are relatively slow. Other manufacturers offer solutions closely resembling gate arrays but in which only the interconnections are programmable. This development could interest the industrial world for it can speed-up production of prototypes and preseries and, when the series become larger, help to convert programmable circuits into gate arrays or standard cells without changing the whole printed circuit.

6 Input-output interfaces

The interfaces linking the logic part, made up of TTL, HC-MOS or VLSI, to the outside analog and electro-mechanic world are both rich and varied. The few books which cover this field [Tompkins88] are somehow incomplete and the designer always has to search through the manufacturer's documentation. This chapter only describes a few principles and the most frequently used techniques.

6.1 Logic inputs

A logic or Boolean input has two separate states for voltage, current, frequency, etc. Books on sensors and electronics provide a solution to one problem, i.e. how to convert this state into a voltage between 0 and 3V, with an impedance lower than few hundred Ω; this means using the logic components discussed in the previous chapters.

6.1.1 Filtering and galvanic insulation

The link leading to an outside logic signal can suffer from parasites. These signals often reaches industrial voltage levels, e.g. 24 volts. In addition, *noise filtering* and some protection against *overvoltages* are absolutely necessary in order to guarantee good operating conditions. An RC network, diodes to divert overvoltages and a Schmitt trigger can be used and will prove sufficient in most cases (figure 6.1a). Using an HCMOS gate means that the value of R can become quite great if necessary (high input voltage or desire for low consumption). *Galvanic insulation* using an *optical coupler* (figure 6.1b) is recommended so that pulses travelling along a common ground wire do not disturb the logic. Input resistor R can therefore stand several hundred Ohms in order to produce the required 5 to 10 mA current. A transformer is used for coupling in the case of alternate signals with no continuous component (figure 6.1c).

The optical coupler catalogue is a very large one. Contrary to TTL integrated circuits, compatible circuits produced by different manufacturers do not carry the same numbers.

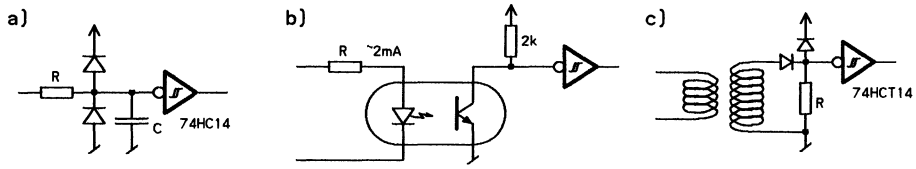


Fig. 6.1 Filtering and galvanic insulation

6.1.2 Digital filtering

Digital filtering is preferable to the RC network shown in figure 6.1a, especially if a VLSI integrated circuit is being developed for the interface. There must be some protection against overvoltages (input resistor and diodes); it is often included at the integrated circuit input. The simplest filter consists of two flip-flops synchronized by a clock whose cycle is longer than the irregularities at the beginning and the end of the signal, e.g. bouncing of mechanical switches (figure 6.2). Since the signal is sampled by the same correctly phased clock, a single flip-flop can be used when the metastable states do not affect the following stages.

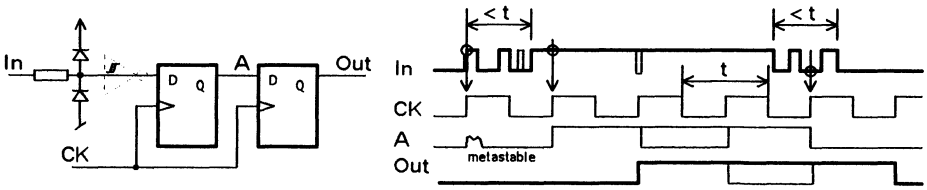


Fig. 6.2 Filter using two flip-flops

Figure 6.3 shows two solutions used in available circuits. In the first case, the HCTL-2000 quadrature encoder (\ssn 6.1.7) which checks rotation direction (figure 6.3a) also filters bounces and only accepts pulses which exceed two clock pulses. The first flip-flop synchronizes the input; its possible metastable state must be shorter than a clock period. The two following ones, used jointly with the AND gates, make up a low-pass filter.

Another crafty solution is found in the CMOS 4490 circuit (Motorola MC14490). As it uses a serial register to delay the signal, a parallel load occurs when input and output are identical, which destroys any parasites propagating through the register at that particular moment (figure 6.3b). The reader can carry out the time analysis. This circuit does not seem to include a flip-flop for suppressing metastable states; the manufacturer probably neglected this problem because of the intended application (suppressing mechanical contact bounce).

In all applications which include a microprocessor, designers prefer a software filter which uses as a guide the algorithms wired into the previous figures.

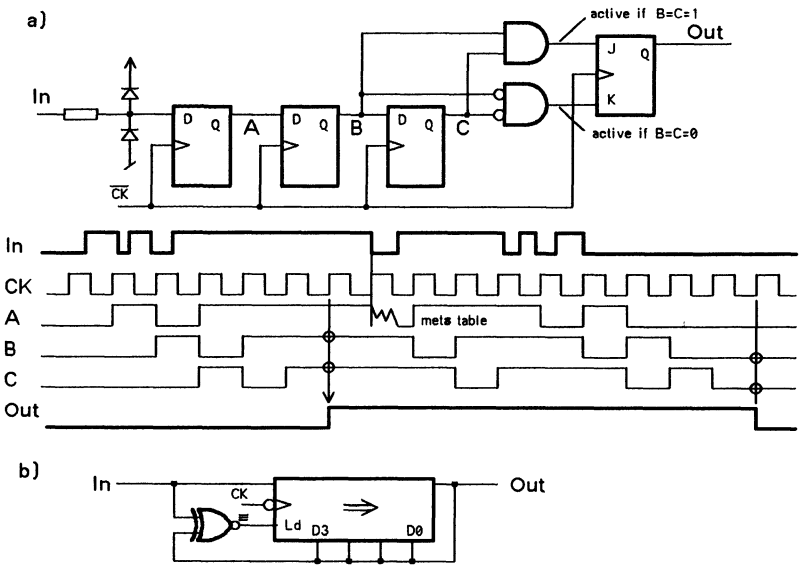


Fig. 6.3 Digital filter

6.1.3 Reading an analog sensor

Many types of *sensors* include a resistance which varies according to a physical effect. A voltage divider and a Schmitt trigger can be used as an interface when the resistance varies sufficiently for the voltage to go way over the hysteresis thresholds.

The resistance in a *photodiode* reaches about 100kΩ in darkness but drops to less than 1kΩ when lit up. A 2,2kΩ resistor, even an adjustable one, can be used (figure 6.4a). If the amplitude is not great enough, the Schmitt trigger can be replaced by an analog comparator (figure 6.4b). A *damp contact* is more delicate because the effect is not repetitive; it requires a logarithmic 1MΩ potentiometer (figure 6.4c). The principle of most *photo-, thermo- or piezo-resistors* is the same, if their resistance varies sufficiently (figure 6.4d).

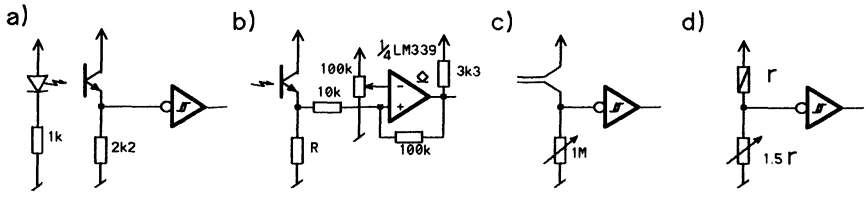


Fig. 6.4 Interface for common sensors

If the resistance included in the sensor varies around r, the potentiometer resistor must be adjusted to the same value (a 1,5 r switch is therefore ideal). In order to reach the switch-over thresholds, resistance r must change by more than 40%, since a circuit like the HC14 has a hysteresis of 1V. An analog comparator is used for smaller resistance variations.

6.1.4 Suppressing contact bounce

Contact bounce for a simple switch can last from 1 to 10ms according to the size of the mechanism. Filters for simple contacts use the aforementioned solutions. The solution shown in figure 6.5a is the most common one, but it is important to note that, when there is no resistor R' between the capacitor and the switch, the important voltage discharge caused by the capacitor sets a limit to the switch lifespan.

Toggle switches offer cleaner solutions with an SR type flip-flop (figure 6.5 b and c). Good operating conditions naturally imply that there is no alternate contact on either side during switch-over. The solution of figure 6.5d is satisfactory with C-MOS circuits; the short-circuit current only lasts 50ns and is limited to a few milliamperes, contrary to the case shown in figure 6.5a which does not include a resistor connected in series and where the current can instantly reach several amperes.

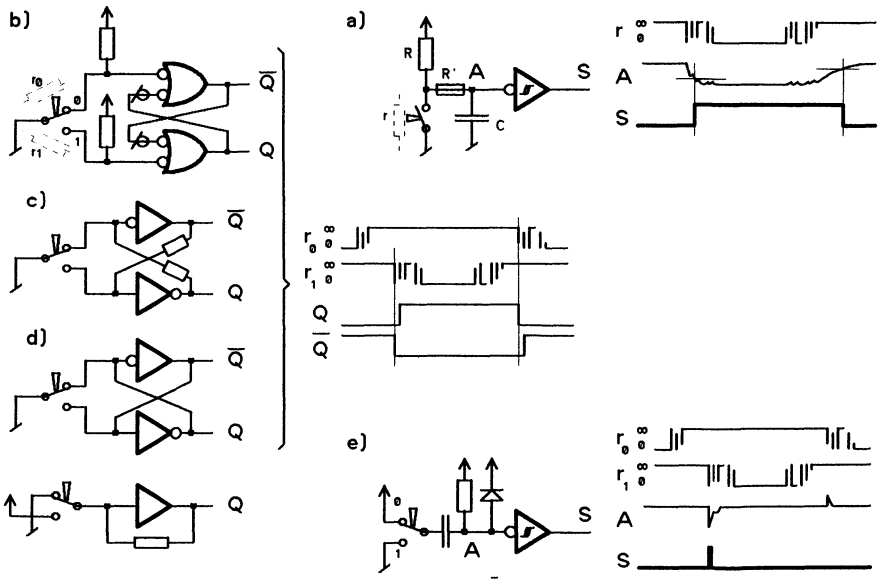


Fig. 6.5 Solutions for suppressing mechanical bounce

It may be convenient to be able to generate a brief pulse by pressing a key. The solution of figure 6.5e is sometimes used but is based on the theory that the first contact bounce of the switch lasts long enough to charge the capacitor almost completely. It is better to use a monostable or a pulse on the rising edge to generate the brief pulse after filtering.

6.1.5 Grouped inputs

In many cases, the number of switches is high (keyboard) and they can be grouped using a matrix layout before being scanned, thus simultaneously solving any problems related to bounce suppression.

As a matter of fact, this is how *keyboards* are built (figure 6.6). A priority encoder can be used instead of the three-state driver. Mechanical contacts can be replaced by suitably polarized phototransistors or transistors, which act as zero or infinite resistors.

In figure 6.6 only one of the S_0...S_3 outputs is active at a given instant of time (row at zero). Inputs E_a...E_c are read globally or successively. One of these zero inputs means that there is a closed "contact" in the selected row. Rows S_i are scanned by software (or hardware), which interprets the states read in E_i. Only four write/read cycles are needed to know the state of all these keys.

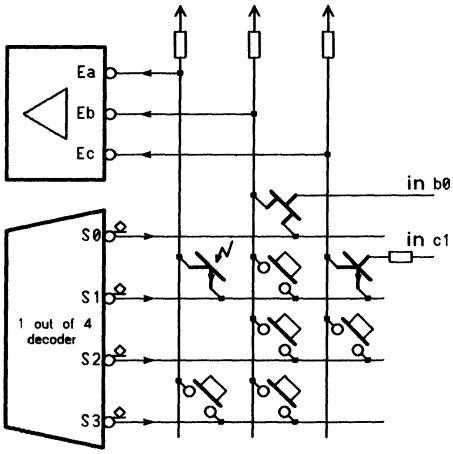


Fig. 6.6 Scanned keyboard interface

A *phantom key* problem may arise if three keys are pressed simultaneously, as in figure 6.7a. Currents can form a loop through unselected closed keys thus making a fourth open key appear. Diodes avoid these parasite currents and are required when phantom keys are unacceptable or cannot be eliminated using software (figure 6.7b).

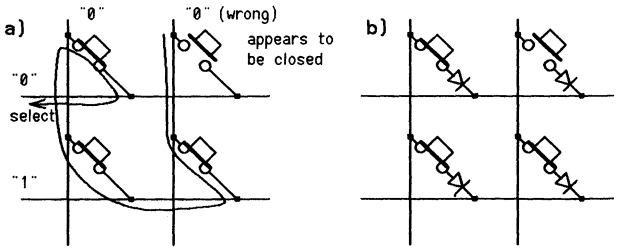


Fig. 6.7 Phantom key and suppression diodes

There are many keyboard interface circuits; the only ones which are still interesting offer a single 5V power supply and low power consumption. If a microprocessor is included, software scanning often works out cheaper although it takes up computing time.

6.1.6 Angular encoder

An absolute *angular encoder* can be used to pinpoint the position of an axe. A linear encoder is based on the same principle (disc with an infinite radius). Gray's code is used to avoid uncertainties between two successive values, since a perfect line-up is impossible (figure 6.8).

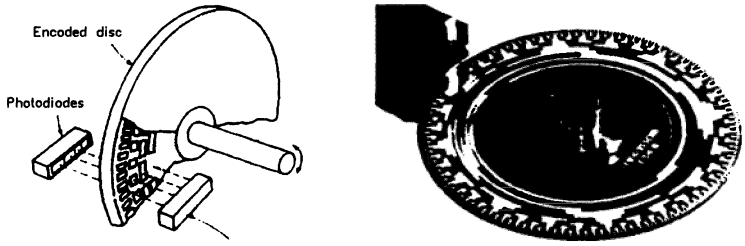


Fig. 6.8 Angular encoder

Converting Gray into binary is easy when dealing with a 2^n moment code (figure 6.9). It can be done by software using a derived algorithm. Encoding in the other direction is very similar. If the information is available in a serial form, converting Gray to binary (figure 6.9c) and binary to Gray (figure 6.9d) only requires very little hardware.

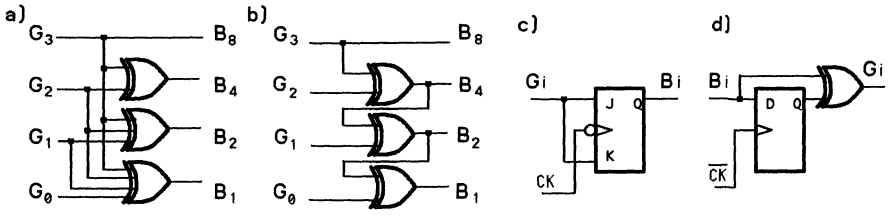


Fig. 6.9 Converting Gray to binary

6.1.7 Quadrature encoder

The angular encoder of figure 6.8 is expensive. A cheaper version can be made up using a single slot disc and two sensors which are out of phase by n+1/2 half-periods (figure 6.10). Decoding and counting up and down the increments is carried out thanks to a sequential system, which is wired, microprogrammed into a ROM or a registered PAL, or programmed into a microprocessor. If X and Y are the two channels, the algorithm copies X every time Y changes in order to generate signal X-. Direction then becomes a matter of the exclusive OR between X- and Y. Sequencing must be analyzed closely in order to control an up/down counter correctly; of course, a change of direction should be avoided during the active edge of the computing pulse. The digital filter and phaser of figure 6.3a is the first stage of a modern quadrature encoder.

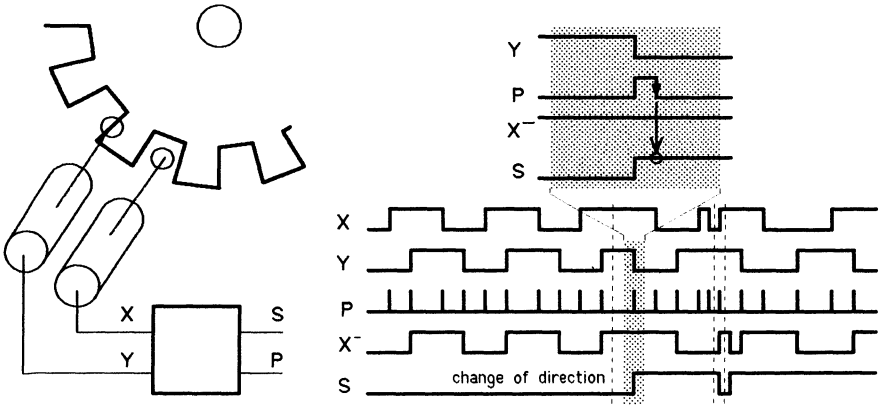


Fig. 6.10 Quadrature encoder

If the resolution can be reduced by a quarter, the schematic (or the software) becomes a lot simpler. The solution of figure 6.11 is easy to implement in software with X linked to an interrupt input triggered by an edge and Y linked to an input port.

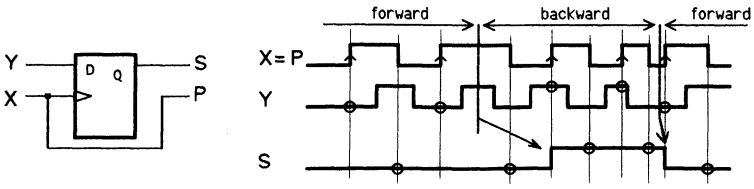


Fig. 6.11 Simplified encoder

It is interesting to note that with some initial position indicator, it is easy to simulate an absolute angular encoder with a quadrature encoder and an up/down counter controlled by the encoder shown in figure 6.11. This is much more economical and absolute encoders are only used when an initial reset is impossible.

There are a few circuits which make it easier to build an angular encoder. Double photodiodes include the logic for rotation direction. Other circuits are designed for use with external diodes but they incorporate an up/down counter; they have an advantage over a microprocessor program since they can keep up with higher frequency signals.

6.1.8 Analog angular encoder

A simple or multiple turn potentiometer gives an absolute angle indication, with a limit to the rotation amplitude. In order to overcome this limitation, potentiometers with two endless paths must be used (figure 6.12). A simple comparison indicates which of the values is inside the validity field.

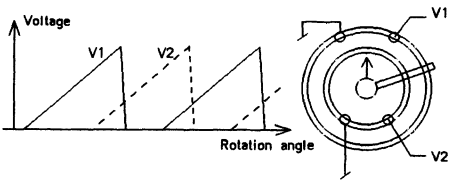


Fig. 6.12 Potentiometric absolute encoder

6.2 Analog inputs

The field of analog electronics is a form of art which is bound by rich technical rules [Jung81] [NS86]. The goal of these few pages is to enable digital circuits specialists to recognize major components and be able to compute basic schematics.

6.2.1 Operational amplifiers

An *operational amplifier* or *op-amp* has two inputs marked by + and - signs. The relative position of these two signs is random and depends on the schematic legibility. Two voltage supplies, typically +12V and -12V, are required for the power supply (figure 6.13a). In some cases, the system can operate between 0 and 5V, with a reference point at 2.5V.

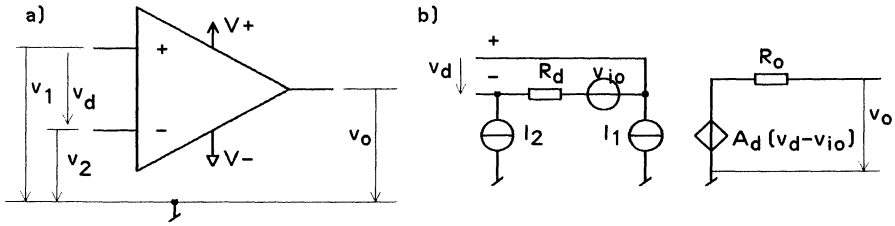


Fig. 6.13 Operational amplifier

The output voltage is given by the formula

$$v_o = A_d \cdot v_d + A_c \cdot v_c = A_d \cdot v_d (1 + \frac{1}{\rho} \cdot \frac{v_c}{v_d})$$

where

$v_d = v_1 - v_2$	differential mode voltage
$v_c = \frac{v_1 + v_2}{2}$	common mode voltage
A_d	differential mode gain
A_c	common mode gain
$\rho = \frac{A_d}{A_c}$	common mode rejection ratio CMRR
R_d	input resistance
T_o	output resistance
v_{io}	offset voltage

The differential mode gain is around 10^5, that is 0.1 V/μV. Saturation obviously occurs before the maximum output voltage is reached. The CMRR (*Common Mode Rejection Ratio*) is an important parameter of op-amps and characterizes their ability to operate in a differential mode over a wide range; its value is typically 60 to 110 dB. Cut-off frequency is more or less one MHz. In response to a voltage change, output voltage changes with a *slew rate* which is typically 0,01 to 100 V/μs.

The equivalent low frequency schematic is given in figure 6.13b. An *offset current* I_1-I_2 is required for a zero output voltage and a *bias current* $(I_1+I_2)/2$ of a few nA is necessary so that the input amplifier functions correctly.

The variation of the output voltage when the input voltage is fixed is known as *drift*. Several factors influence this drift: input current, voltage change, temperature change and ageing. *Noise* is defined as an unplanned output signal. It can be random or repetitive, internal or external, and can occur at high or low frequency. Mastering noise problems within high gain amplifiers requires great experience.

6.2.2 Ideal model

In the case of an ideal op-amp, $R_d = \infty$, $R_o = 0$, $A = \infty$, $I_1 = I_2 = 0$ et $v_{io} = 0$. Bandwidth is infinite, noise and drift are inexistant. The schematics of the following sections refer to this ideal case which is a realistic one for good quality op-amps working far from their limit conditions.

6.2.3 Basic schematics

The inverter amplifier requires two resistors (figure 6.14a) whose ratio defines the gain G. One have $V_o = G \cdot v_i$. The non-inverting amplifier of figure 6.14b necessarily has a gain greater than 1. The unity-gain amplifier of figure 6.14c has a high input impedance and can be found in input stages (e.g. for maintaining a capacitive input), as well as in output stages. Many other schematics are documented in specialized books [NS86] and manufacturers' application notes. When combined with analog switches (section 2.7.9), they offer numerous possibilities for controlling analog parameters digitally.

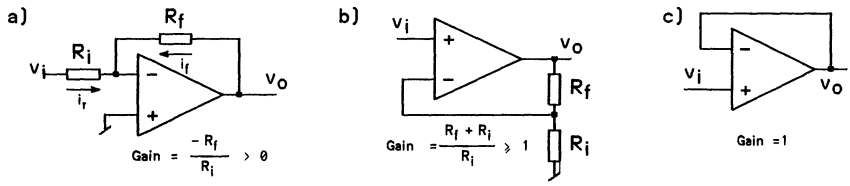


Fig. 6.14 Op-amp basic schematics

Adding capacitors to the schematic of the inverting amplifier creates *active filters* for low (figure 6.15a) and high (figure 6.15b) frequencies. Cut-off frequency depends on component value. Higher order filters, switched capacitor filters and many special techniques ensure a better frequency response.

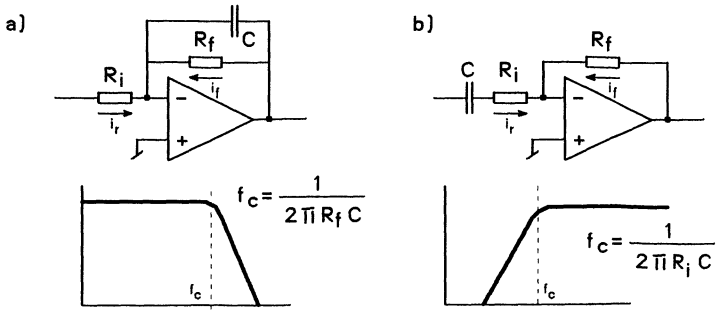


Fig. 6.15 Active filters

6.2.4 Comparators

An *analog comparator* is a high gain op-amp, whose output is always saturated. When v_1 is higher than v_2 (used as a reference voltage), the output drops to the lowest voltage (figure 6.16a). Using two resistors provides some hysteresis (figure 6.16b,c), which is required in most applications for filtering the noise generated by the switch-over. Computing these two resistors is easy thanks to Ohm's law. The output is frequently of the open-collector type, and a pull-up resistor, not shown in figure 6.16, must be added.

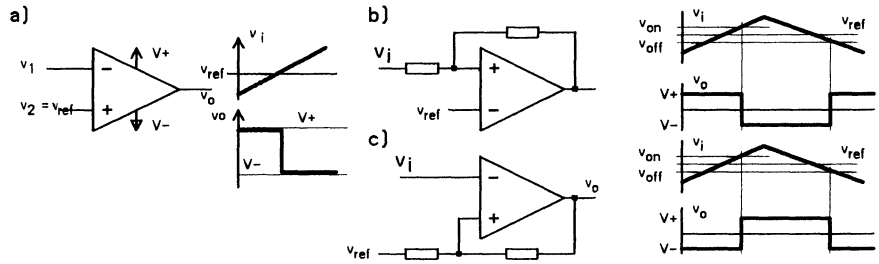


Fig. 6.16 Comparator and typical applications

6.2.5 Instrumentation amplifiers

Instrumentation amplifiers are used to amplify very weak signals which include a large amount of noise when in common mode. E.g. a resistive bridge makes a good noise antenna (figure 6.21). The main features of these amplifiers are:

- High input impedance
- Low output impedance
- Wide linearity
- Stable gain
- Very good CMRR

An instrumentation amplifier can be assembled using a few op-amps, as long the circuits are *matched*. Moreover, resistor symmetry must be respected in order to provide an important CMRR (figure 6.17a). However, ampli-ops are gradually being replaced by more efficient and flexible integrated circuits, since the latter are getting cheaper and their characteristics come close to those of an ideal amplifier. These circuits have gain program pins (1, 10, 100, 1000) which make them easy to adapt. Figure 6.17b shows a typical example based on Burr-Brown's INA102 circuit. Two op-amps complete the schematic in order to adjust the output level within the field of the selected converter. The first op-amp has a high gain and adds a voltage which allows the instrumentation amplifier to work in its linear zone (1 to 6.12V). The second op-amp shifts the measured range by 1V.

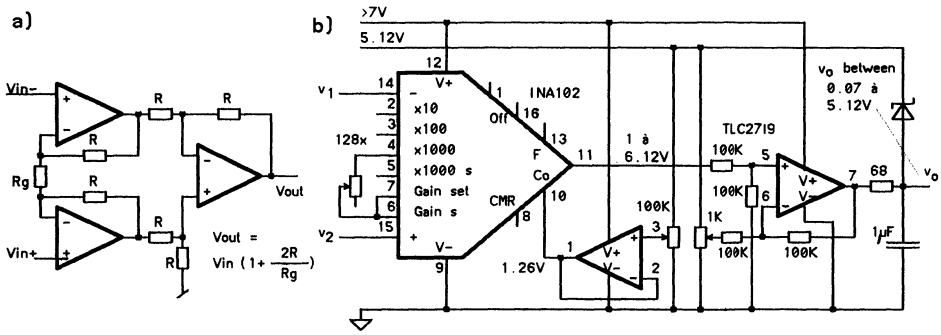


Fig. 6.17 Instrumentation amplifier

6.2.6 A/D converters

Analog-digital converters (ADC) give the encoded value of an analog input value to be found typically between 0 and 2.56V. The conversion process is a parallel or sequential one and the read operation is validated by a suitable signal. A preamplifier is required with most sensors (figure 6.18). It is often used to hold the analog value during conversion (*sample and hold*) in order to avoid errors due to signal variations.

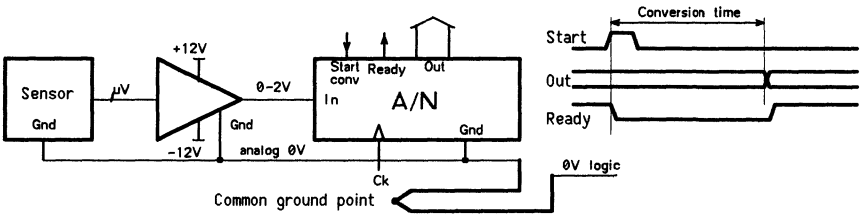


Fig. 6.18 A/D converter with preamplifier

One big problem to be solved when measuring sensors' low voltages is how to get rid of the disturbances due both to the environment and to the microprocessor itself, which includes many signals oscillating at different frequencies. The ground wires must be carefully drawn and linked to the power supply at one single point. The converter must not share the same power supply as the processor and the interface circuits. Some cases require total galvanic insulation, which means adding extra power supplies and decoupling signals optically.

Distant sensors offering very low levels can be read by transferring the voltage to be measured into a capacitor which has been galvanically insulated first from the converter, then from the sensor (figure 6.19). Sealed mechanical *reed type relays* are used to guarantee perfect insulation. This solution is gradually being replaced by operational amplifiers whose galvanic insulation can stand up voltages to 2400V.

Another seldom used solution consists in placing the converter and the processor close to the sensor and transmitting at a distance digital information, which is thus protected from small parasites.

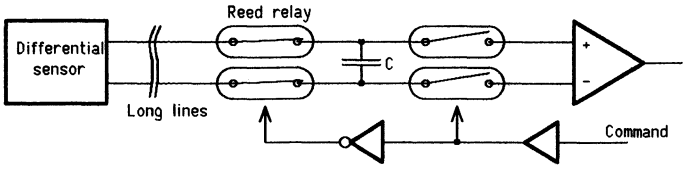


Fig. 6.19 Analog measurement by charge transfer

6.2.7 Serial interface

The converter interface shown in figure 6.18 is a parallel one. By using different command signals, it can be plugged directly into the 8 or 16 bit bus of a microprocessor [Tompkins88]. In order to simplify wiring, several converters are equipped with a 3 wire serial interface which is suitable for slow data acquisition by a microcontroller. In reply to a select operation and a series of pulses, the converter sends back the value of the serial measurement (figure 6.20). This principle will be discussed in more detail in section 8.5.4.

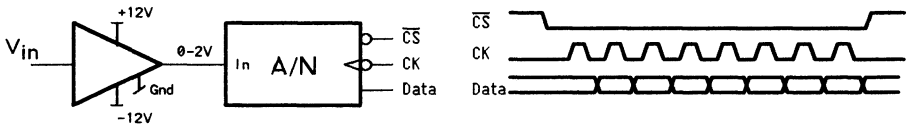


Fig. 6.20 A/D converter equipped with a serial interface

6.2.8 Analog sensors

There are various types of sensors for measuring pressure, temperature or distortion. In the case of a resistive principle, it is possible to enter an A/D converter equipped with direct or differential inputs by using a voltage divider or a *Wheatstone bridge* (figure 6.21). When the input amplitude is too low, an instrumentation preamplifier is required (section 6.2.5).

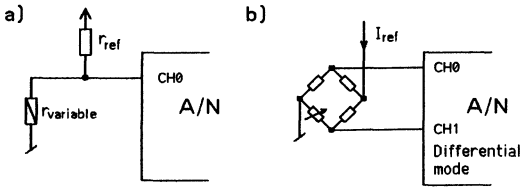


Fig. 6.21 Interfaces for analog sensors

6.2.9 Voltage-frequency conversion

Capacitive sensors, as well as resistive and inductive ones, are easy to include in an oscillating circuit. In such cases, oscillation frequency is linked to the phenomenon to be measured and it is easy to create intervals in order to increase sensitivity.

6.3 Output interfaces

The TTL and HC-MOS circuits used in microprocessor interfaces are designed to control a few milliamperes, enough for luminous diodes and very low power relays. Special circuits, the main types of which are discussed in this section, make it easier to control common cases implying voltages reaching 100 volts and 3 amperes. When going above these values, it is necessary to resort to very special electronic techniques.

6.3.1 Microcontroller outputs

Microcontrollers and programmable interface outputs are usually TTL-compatible, but the output current may vary a lot depending on the output amplifiers. These outputs are mostly bidirectionnal, and are based on four basic schematics. The first solution (figure 6.22a) includes a direction flip-flop which controls the output amplifier. The processor reads the I/O pin and can thus carry out some tests: if the output is not short-circuited to the ground or Vcc, the processor must read again what has been written. This is not possible with the schematic of figure 6.22b, in which the processor reads what has been written into the output flip-flop before the output amplifiers.

The solution shown in figure 6.22c does not require a direction flip-flop, but the output flip-flop must be disabled in the output mode. The input signal must have a low enough impedance compared to the pull-up resistor. Such an output is not adequate for generating output current and loading parasitic capacitors rapidly, but its main advantage is that a short-circuit toward the ground is not damaging and can be detected by software.

The last solution (figure 6.22d) adds a transistor which is activated only during the few hundredths of a ns during which the processor writes a "1" on the output. This loads the output capacitance rapidly, and is not dangerous if a short-circuit occurs. Some programmable interface circuits include additional flip-flops for programming a three-state or open-collector mode, and for controlling I/O active polarity thanks to an exclusive-OR gate.

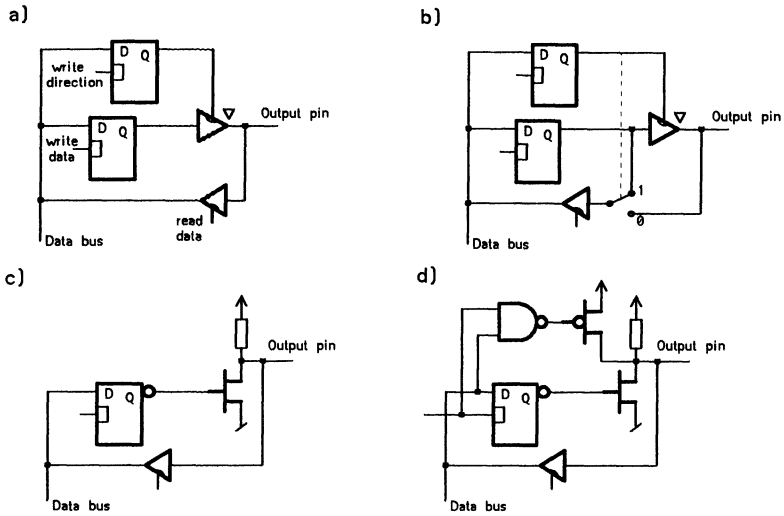


Fig. 6.22 Bidirectionnal I/O lines on a microcontroller or programmable interface

6.3.2 D/A converters

Converting digital to analog can be done quite easily by referring to a resistor scale. Integrated circuits which convert digital to analog are available and cover a large spectrum ranging from rapid (4 to 6 bits, 100 MHz) ones meant for screens up to 12 to 16 bit audio converters.

6.3.3 Display controllers

Alphanumeric or graphic displays are available in high voltage technologies (plasma, electro-luminescent). Manufacturers of these displays also produce special circuits.

LCDs (liquid crystal displays) seem to have greater potential, due to their low voltage consumption, and are controlled very simply by a few volts. However, in order to avoid reducing the contrast by polarization, this voltage must be inverted with a frequency of a few kHz. The solution which resorts to a positive and a negative power supply (figure 6.23a) is never used. One of the electrodes is sent an alternate signal. If the other electrode is out of phase, which can easily be controlled by an exclusive OR, the LCD is activated.

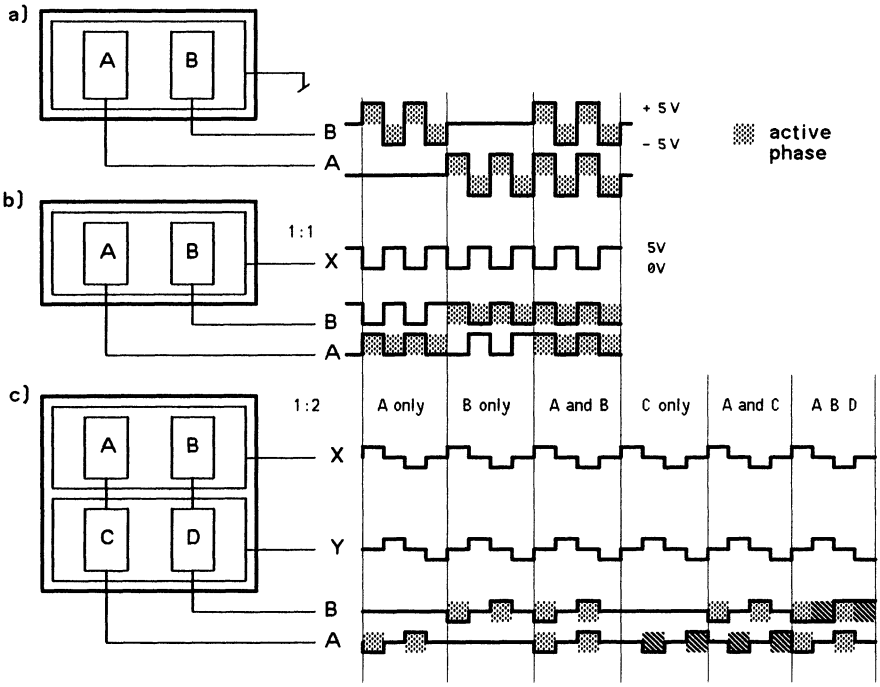


Fig. 6.23 Controlling an LCD display

Multiplexing can be carried out with a slightly more complex wave which includes half-voltages. When the voltage at the LCD electrode terminals equals half the voltage, the LCD remains inactive. Normally, this multiplexing operation slightly reduces contrast. The multiplexing principle can be extended to 1:3 and more.

Control circuits for LCDs are equipped with a serial or serial-parallel interface. Several 4-bit microprocessors include an interface for a 1:2 and 1:3 multiplexed LCD. A wide choice of LCD displays including the interface does exist.

6.3.4 Power amplifiers

Power amplifiers require large radiators when they do not operate in an on/off mode. High current *Darlington transistors* (figure 6.24a) are available but the new *V-MOS or MOS FET* (figure 6.24b) transistors are preferable, for their voltage drop is a manufacturing parameter proportionate to the current. Protection diodes or more efficient schemes for recuperating energy are required in the case of inductive charges. It is important to remember that the floating gate capacity is high and implies using good input amplifiers in order to reach maximum switch-over speeds. Some transistors have an additional gate for measuring the current.

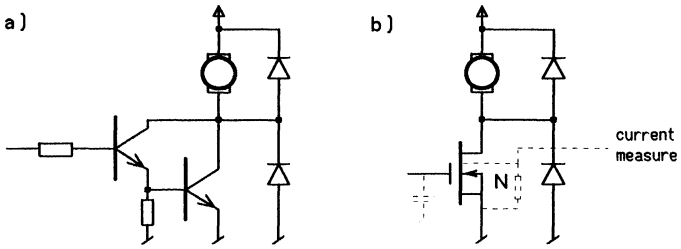


Fig. 6.24 Analog or digital control

6.3.5 Relays

Electro-mechanical relays are very slow amplifiers but they are very easy to implement. They provide a perfect galvanic insulation between the input and the output and are available in the same size as an integrated circuit or larger when high currents are required (figure 6.25a). With an inductive load, sparks are created when the contact opens, significantly reducing reliability and disturbing the surrounding logic. An RC network is used to damp the sparks and is empirically selected by adding a capacitor whose value in microFarads is equal to the switched current in Amps, as well as a resistor equal to the load resistor value.

A *static relay* is a power amplifier equipped with a galvanic insulation. When it is connected, the designer should beware of signal polarity and select among models based on transistors, thyristors or triac, depending on whether the commuted current is continuous or alternative.

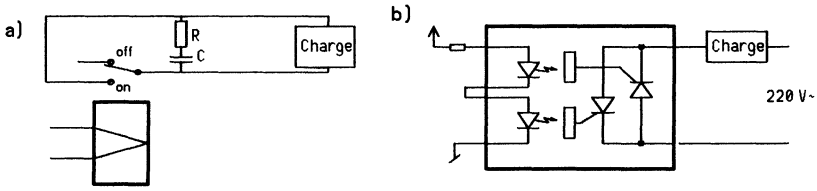


Fig. 6.25 Electromechanic and static relay

6.3.6 Driving motors

In order to drive a motor in either direction, there are three basic possibilities. There is a simple solution based on two transistors and dual coil motors which are rather difficult to find (figure 6.26a). Using two transistors and two power supplies (figure 6.26b) is expensive. It is preferable to use four transistors and one power supply (figure 6.26b and c). P-type transistors are more expensive and offer lower performances than the N-type transistors they have to work with. Protection diodes are required because of the voltage surges due to the inductive motor load.

Usually, motor power supply is higher than the logic 5V supply. Level conversion is carried out thanks to interface circuits which have two voltage supplies, the 5V one preferably being generated internally (figure 6.26b,c). Open collector gates are also used if their output breakdown voltage is high enough (figure 6.26d). Optical couplers (figure 6.26e) offer one advantage, i.e. an efficient galvanic insulation.

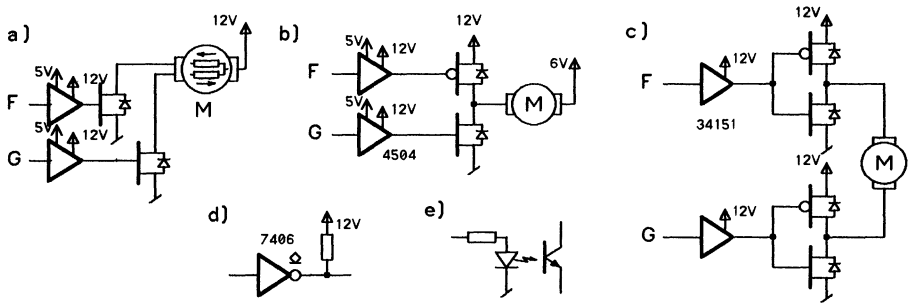
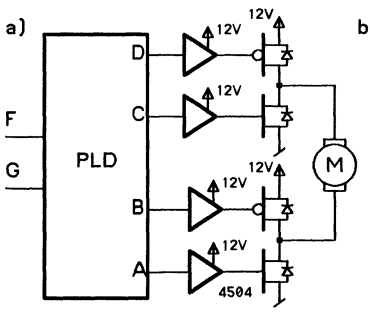


Fig. 6.26 Driving a motor

In the motor control interface of figure 6.26c, the two F and G signals control the forward and backward direction. When they are both at zero or both active, the motor is short-circuited by the low impedance of the transistors and diodes through which the current can pass. A more general interface which implies using some additional easy logic (figure 6.27a) specifies a mode in which the motor is open and continues to run freely when power is switched off. Coding these states can require two or three wires, according to the preferred truth table (figure 6.27b). Controlling three phased motors (figure 6.30) follows the same principle. Implementating power amplifiers is made easier when hybrid power transistor sets including 1, 2 or 3 transistor pairs and diodes are used.



b)

G	F	DCBA	
0	0	1111	blocked
0	1	0011	forwards
1	0	1100	backwards
1	1	1010	free wheel

L	E	S	DCBA	
0	0	-	1111	blocked
-	1	0	0011	forwards
-	1	1	1100	backwards
1	0	-	1010	free wheel

Fig. 6.27 Complete motor control

Under certain circumstances, it is possible to drive motors set out in a matrix. This means that up to 5 bidirectional motors can be connected to 6 amplifiers according to the diagram of figure 6.28 Several motors can be controlled simultaneously, but they must be driven in the same direction and the total voltage must remain at a tolerable level.

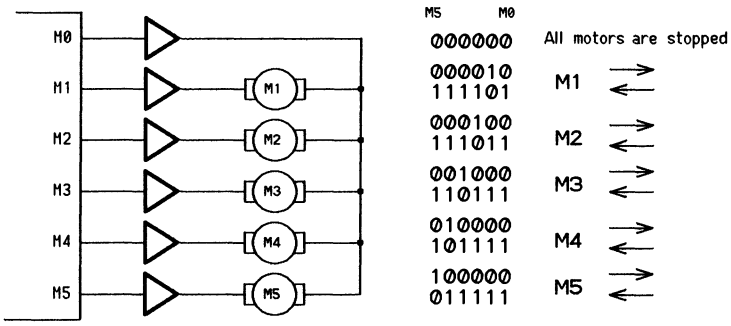


Fig. 6.28 Interface for 5 bidirectional motors

6.3.7 Proportional control

A motor can be controlled by variable voltage using a D/A converter followed by a power amplifier (figure 6.29a). Half of the dissipation thus occurs in the amplifier and large coolers are required.

A motor can be controlled proportionally by *Pulse Width Modulation* (figure 6.29b), which only requires e.g. a counter and a comparator. Mechanical inertia smoothes the movement and if the switching frequency is higher than 20 KHz, there is no disagreeable noise. As regards bidirectional motors, switching can be handled by an extra transistor or included in the logic of figure 6.27.

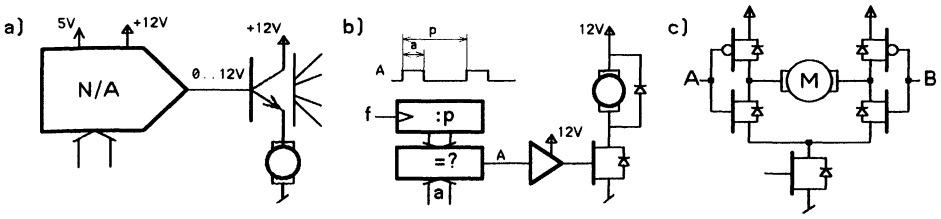


Fig. 6.29 Proportional control

6.3.8 Collectorless motor

A collectorless motor is a synchronous motor equipped with permanent magnets switched on/off according to its position by a transistor bridge. Three Hall effect sensors or one incremental sensor are required and the control signals can be pulsed in order to control power (figure 6.30). This control part will be included in the motor in the near future due to the low power dissipation of digitally switched transistors.

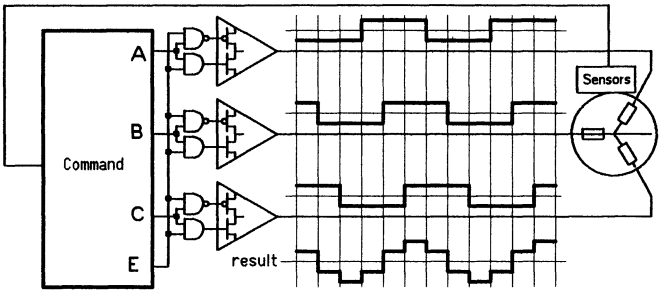


Fig. 6.30 Collectorless triphase motor

6.3.9 Step by step motor

A step by step motor is controlled by two pairs of coils whose excitation currents are out of phase (figure 6.31). In the simpler solution, a complete cycle corresponds to four steps and rotation direction is a function of how out of phase the excitation currents are (figure 6.31b). In the half-step mode (figure 6.31c), an extra state corresponding to a deactivated coil (both bits at "1" or at "0") generates a half-step. Steps are generated in the correct sequence and at the right frequency thanks to an interface circuit.

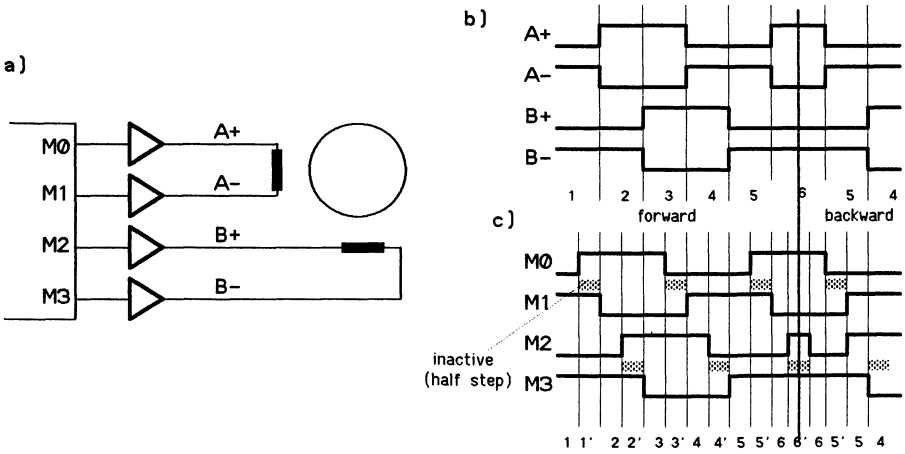


Fig. 6.31 Interface for step by step motor

Normally, a step by step motor is controlled by forward pulses and a direction indication. An interesting problem to solve is how to use these two I and S signals to generate signals A and B which control coil excitation. There are a few low powered integrated circuits for step by step motors. Some of them are really microprocessors and generate acceleration and braking ramps.

6.3.10 Regulated power supplies

An auxiliary voltage, which is sometimes controlled by software, is often required on a board. Depending on how much power is needed, this continuous current conversion (DC-DC) can be carried out using various types of integrated circuits:

- linear DC converters. They lower the tension by at least 2V since they render transistor conductivity dependant on the chosen output voltage.
- switching regulators. A capacitor is used to transform energy pulses measuring between 40kHz and 1MHz which are then filtered by an output capacitor. This type of power supply is not suitable if weak analog signals are to be amplified on the same board. Step-up converters increase voltage whereas step-down converters reduce it.
- charged pumps are based on the same principle as voltage doublers
- AC/DC converters. They take their own power supply straight from the 220V line and provide an output current of a few mA at 5-10V.

Circuits which are easy to use provide current and tension for applications based on ampli-ops and A/D converters.

Checking a battery and switching between the external and the battery supply is an important function in portable systems. Special circuits called battery monitors or managers can do the job.

7 Testing circuits

Testing integrated circuits and sub-systems during manufacturing is an extremely expensive process. The designer must therefore be very aware of testing difficulties and design a product which is easy to test, even if implementing test facilities means creating a more complex product.

7.1 Types of tests

Similar problems appear when testing VLSI components, when testing boards comprising between 10 and 100 circuits or when testing complete systems; they can be solved in many different ways.

7.1.1 Debugging

For a long time, *debugging* a circuit meant using a voltmeter, a function generator, an oscilloscope, a logic analyzer, etc. to check the behaviour of the circuit, and subsequently modifying the circuit until it behaves correctly. Nowadays, this procedure has become quite impossible with integrated circuits and is becoming more and more difficult with printed circuits.

It has become essential to simulate the circuit completely before building it; however, this complete simulation can often only be carried out on limited parts of the circuit: sequencers, counters, decoders. In most cases, it can prove either impossible or very expensive to simulate circuits which include processors or programmable interfaces, since the files which describe such circuits are highly complex. Some debugging techniques will be described in section 7.5.

7.1.2 Manufacturing

Once a piece of equipment has been debugged, it is mass procuced and every item must be tested before being distributed. The cost of this test largely depends on the design, the main point of such a test being to reduce the number of subsequent repairs. The further on in the designing-manufacturing-retailing chain a defect comes to light, the more it costs to put right.

7.1.3 Maintenance

When a piece of equipment breaks down after being used for a normal length of time, it is repaired by replacing the defective circuits and modules. Being able to pinpoint them easily and quickly depends both on the design and on the equipments and programs used to locate the errors.

7.2 Test aids

Equipment used to observe the signals corresponding to the test sequences can vary widely in price.

7.2.1 Logic indicators

A logic indicator or *logic probe* indicates whether a line is in one of the logic states 0 or 1 or in a floating state. This device which is both simple and cheap is very useful for debugging and repairing.

7.2.2 Oscilloscopes

An *oscilloscope* visualizes signals and shows up precisely all incorrect wave shapes, pulse lengths and delays between signals (figure 7.11). An oscilloscope must be properly synchronized, which implies the presence of relatively high frequency repetitive signals. The interesting part of the signal must always be close to the synchronization signal. The advantage of a digital oscilloscope is that it can memorize non repetitive events based on a few traces but, for the same price, such devices offer a lower bandwidth. In order to ensure that the observed signals are representative of the real signals, the probes must be well adjusted and connected to the ground very near the observed signal.

7.2.3 Logic analyzers

A *logic analyzer* samples signal amplitudes and encodes them in binary. Glitches which occur during sampling are recognized in some acquisition modes. Thanks to information compression due to this sub-sampling, it is possible to stock between 8 and 100 channels in parallel and to a depth of several K (Kilo samples). Debugging a microprocessor system necessarily means using a logic analyzer, since the consequences of a glitch, e.g. when managing an interrupt, may only come to light long after the cause. An oscilloscope is an absolute must in order to pinpoint analog problems (reflections on lines, incorrect access times).

Some logic analyser models include a digital oscilloscope and are hence more flexible.

7.2.4 Function generator

Analog or digital *function generators* produce repetitive sequences which mean that a logic analyzer or an oscilloscope can be used to observe the reaction of the excited circuits. They are rarely used for debugging digital interfaces: the processor can be programmed to generate test sequences whose behaviour is observed with a logic analyzer.

7.2.5 Signature analyzers

For the last ten years, Hewlett Packard has been offering an easy way to check that a circuit is working correctly [HP80] [Mizco86]. By definition, linear registers (section 3.6.5) carry out a polynomial division of a chain of bits, which compresses information. The tested system must generate a clock and a signal to validate the test (figure 7.1). At the end of the sequence, each observed point is matched by a signature (4 digit hexadecimal number), which must correspond to the calculated reference value. With 16 bits, the probability of not detecting an error is less than $2^{-16} \simeq 10^{-5}$.

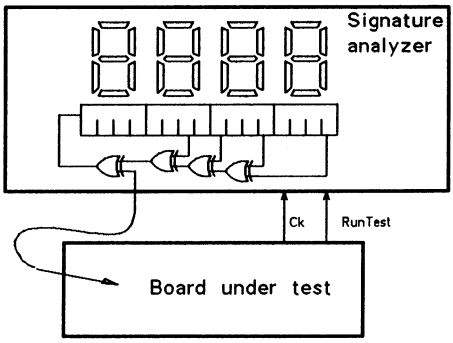


Fig. 7.1 Principle of signature analysis

7.2.6 Industrial tester

Integrated and printed *circuit testers* are extremely expensive pieces of equipment capable of generating or observing hundreds of signals within a very precise time limit. The circuits which condition the signals must be placed very near the connection points of the tested circuits, therefore the test head is often round. Preparing the test sequences takes a long time and requires the use of CAD tools.

Printed circuits are placed on a *bed of nails* which links the tester to the external and internal board signals. VLSI integrated circuits wafers are tested with a *probe card*. Circuits are again tested after encapsulation.

7.3 Test theory

Mastering test problems implies a high level of specialization. This chapter only gives the main elements needed by an engineer intent on designing for test purposes, while applying traditional methods intelligently. Many specialized books [Mizco86] [Fujiwara85] [Stevens86] describe the knowledge required when setting up tests which are particularly important from the economic and strategic point of view.

7.3.1 Break-downs and reliability

A *break-down* is the sign of a physical defect (bad manufacturing process) or of a defect in the design (mistake, omission). A defect appears sooner or later as a *glitch*, perceptible from outside.

A break-down is known as *permanent* if the error which caused it can be reproduced and therefore detected. A *transitory* break-down is fleeting and cannot be detected accurately by the test. Once the break-down has been located, the defective component can be replaced or the design modified.

A *test vector* defines a given configuration of the expected inputs and outputs. A test sequence is nothing other than a series of test vectors which enable an error to be detected. The length of the sequence is the number of vectors which comprise it.

Reliability is the probability that the system will function without a break-down during a given length of time. If a system increases in complexity, its reliability rapidly decreases. The reliability of a component or of a system follows the typical curve shown in figure 7.2. *Infant mortality* is high: *pre-ageing* drastically reduces after sale returns. Reliability decreases after a long plateau and the *average life expectation* characterizes the lifespan of a circuit [Shooman68] [Stevens86].

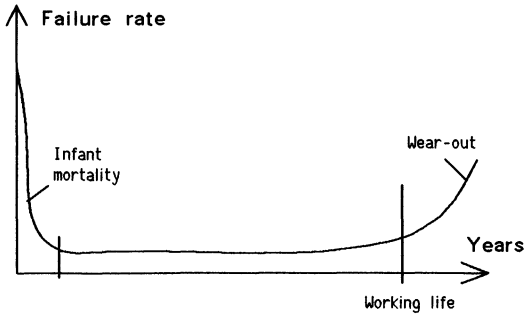


Fig. 7.2 Reliability curve

Mean Time Between Failures (*MTBF*) and *Mean Time To Repair* (*MTTR*) are the parameters used to calculate the *availability* of a system, that is D = MTBF/(MTBF+MTTR). In the case of a system which cannot be repared (integrated circuits), the *MTTF* (*Mean Time To Failure*) is the most important parameter.

7.3.2 Testability

There are two types of tests used for circuits and digital systems: *parametric tests* can be static or dynamic and check that the circuits play their part with the intended tolerance margins (regarding tension and speed); *logic tests* check behaviour under normal conditions. They are quicker and can be repeated at regular intervals during preventive maintenance. There are also functional tests which check that the function is performed correctly, statistical tests used when comparing large numbers of circuits and compact tests which use a counter or a linear register to compress the information issued by a functional test (section 7.2.5).

The environment in which the test is carried out is important. It can be carried out *on line* (while an end-user application is running) and it can be continuous or intermittent. Continuous tests are often limited to a small number of operations, e.g. checking parity in a binary word, or comparing redundant parts. An *off-line test* can be carried out in situ (in the real physical environment) or ex situ (e.g. on a tester).

Test costs depend on the cost of developing test programs and on *socket time* cost. The final price is also a function of the manufacturing yield; in the case of integrated circuits, it must exceed a given percentage in order to guarantee basic profitability.

7.3.3 Test coverage

A test can be more or less exhaustive. Its *coverage rate* is the ratio between the number of recognizable defects and the total number of defects. Obviously, test coverage should be as good as possible but being exhaustive may turn out to be too expensive.

7.3.4 Test methods

Simple pieces of equipment manufactured in series are tested and maintained by technicians who are very familiar with the diagram. When an item does not function correctly, they go up from the outputs or go down from the inputs in order to pinpoint faulty behaviour.

Preparing exhaustive test vectors means reviewing all possible breakdowns and determining their cause exactly. Test vectors can be generated using a

deterministic method, which implies good knowledge of the application. A *gold circuit* is necessary when a random method is used.

Self-testing systems including continuous tests (e.g. error detection codes) or discontinuous ones (internal generation of test sequences during the periods when it is not in use) can be designed for some applications.

7.4 Design for testability

Circuit testability is improved by using various techniques discussed in this section. The aim is to reduce test time while gaining more knowledge of the state in case of error. Simulation should guarantee that the circuit which is being designed functions correctly.

7.4.1 Simulation

When decomposing a system, there are various levels in the simulation description (figure 7.3). Roughly, simulating starts at the behaviour level and means using either a general or a specialized computer language both to describe this behaviour as seen from outside and to check the algorithms used. This simulation often already takes into account the architectural decomposition into functional blocs. *Hardware Design Languages* (*HDL*) or *Register Transfer Languages* (*RTL*) can be used to describe the functional blocs and their interconnections [IEEE87] [Yoeli90]. At the implementation stage, these are decomposed into logic elements which can be simulated independently. The basic elements are gates, whose characteristics are partly idealized: signals are at 0 or at 1 and have predefined delays. Simulating the complete system requires great computing capacity and simulation time is kept to a minimum by resorting to specialized machines. It is also possible to use real physical models as well as a simulator: the circuit is excited using an interface and the answer is sent to the simulator. Since the interface only gives a logic answer, answer times come from a table.

In order to guarantee that tension and delays work correctly, it is necessary to simulate on an electric model; *SPICE* is a well-known software used in such cases. Simulation at a higher analog level is possible thanks to macro models. In the case of a digital system, they characterize library elements and define the models used in the upper layers. SPICE uses parameters linked to transistor size and to diffusion properties; they are obtained experimentally or by simulation at a geometric level, while taking into account semiconductor physics. Obviously, the lower the simulation level, the more information there is to deal with. Each simulation must be chosen at a level which can give sufficiently precise information, before time runs out, so that it is possible to guarantee that the part under examination interacts correctly with the other parts.

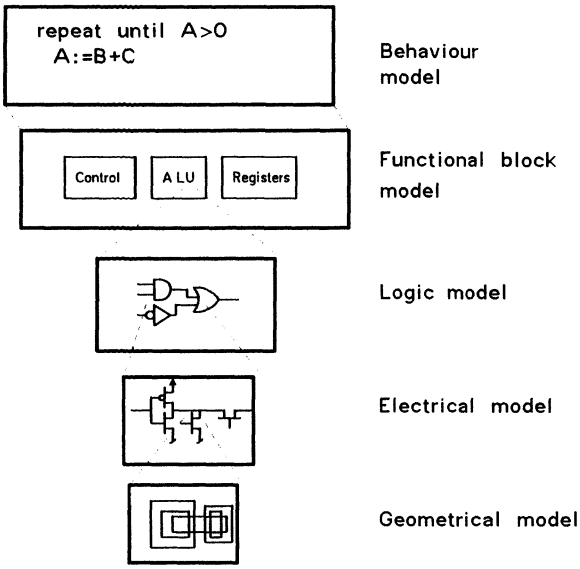


Fig. 7.3 Simulation models

7.4.2 Observability

Observability characterizes the ease with which all internal variables are accessed. Thirty years ago, discrete transistor logic meant total observability, but it must now be artificially increased for modern VLSI circuits.

7.4.3 Controllability

Controllability measures the ease with which a given internal state can be generated by applying outside signals. Good controllability significantly reduces the number of test vectors. For example, testing a 2^{16} counter requires 2^{16} pulses on the input (figure 7.4a). After having split this same counter in two in a test mode, while simultaneously increasing both controllability (counting pulse on the 2nd level) and observability (first counter output), test time falls to 2^8 pulses, i.e. 250 times less.

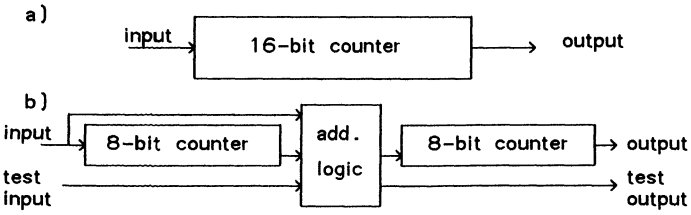


Fig. 7.4 Testing a binary counter by 2^{16}

7.4.4 Initial state

A reset input is the first requirement for such a test. Test vectors can thus be applied from a known state. It is usually important to act on all flip-flops, registers and counters in the system, which increases the cost. In a microprocessor, the reset does not act on the stack and the registers, so that it is easier to test end-user software; when working in a test mode, these parts should be initialized.

7.4.5 Test points

Adding extra *test points* obviously increases both controllability and observability. In the case of VLSI circuits, test points located in the middle of a circuit can no longer be used for recent fine geometry circuits due to the size of the test pads (40 μ x 40 μ) and the necessity not to render these areas passive (i.e. isolated). Using test points on the edge of VLSI circuit chips, without any link with the encapsulating package pins, is a good compromise in order to carry out the initial chip tests and reduce package costs. Thus, the test carried out after encapsulation becomes a superficial one which checks that links are correct.

7.4.6 Multiplexing

The number of links can always be reduced by multiplexing. For example, activating the TEST input of a microprocessor can place on a processor address lines flip-flop states coming from the sequencer used to decode instructions, so that they can be observed. A very similar variant replaces the multiplexor by a test bus capable of addressing different groups of states. In order to avoid a special TEST line, it is possible to control mode change by increasing the tension of one of the outputs to 9V.

7.4.7 Scan paths

A shift register passing through all important signals can be used to control and check a circuit (figure 7.5). This is known as the *scan path* technique and undergoes a few variations according to the cell structure: Level Sensitive Scan Design LSSD, Scan test, Boundary Scan, etc. A synchronous design makes it easier to use a scan path: a scan path flip-flop then becomes a mere flip-flop preceded by a switch.

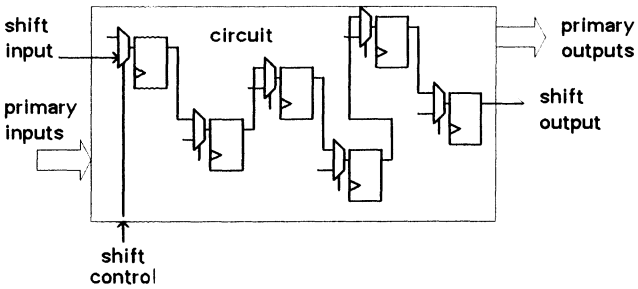


Fig. 7.5 Basic scan path principle

For example, if the counter in figure 7.4 includes a scan path which goes through all the flip-flops, the test in fact means shifting 16 different motives and observing that the counter behaves satisfactorily in these cases.

Libraries of VLSI circuits offer a choice of flip-flops which can be used to make up a scan path. Each CAD-VLSI software retailer can suggest various testing philosophies associated with semi-automatic test vector programs. The surface of a circuit including a scan path increases by 20-30%.

7.4.8 JTAG standard

The *JTAG* (*Joint Test Action Group*) P1149.1 standard was established at the end of the 1980s in order to standardize test equipment [Mounder90]. Any VLSI circuit or subsystem which includes a number of inputs-outputs is also equipped with four standard lines:

- TMS (*Test Mode Select*) acts on a sequencer which decides whether the circuit is in a test mode (input active at 1)
- TCK (*Test ClocK*) synchronizes transfers (input, active on the rising edge)
- TDI (*Test Data In*), data input and serial commands
- TDO (*Test Data Out*) data output and serial commands

The logic associated with these four lines (figure 7.6a) includes *scan path* registers known as *boundary scan* registers since the serial register observes and assigns signals at the circuit boundary, just in front of the input-output connection pins. Several circuits can be daisy-chained just like shift registers (figure 7.6b). Parallel links (separate data inputs and outputs and/or separate clocks) can also be used to increase the flow.

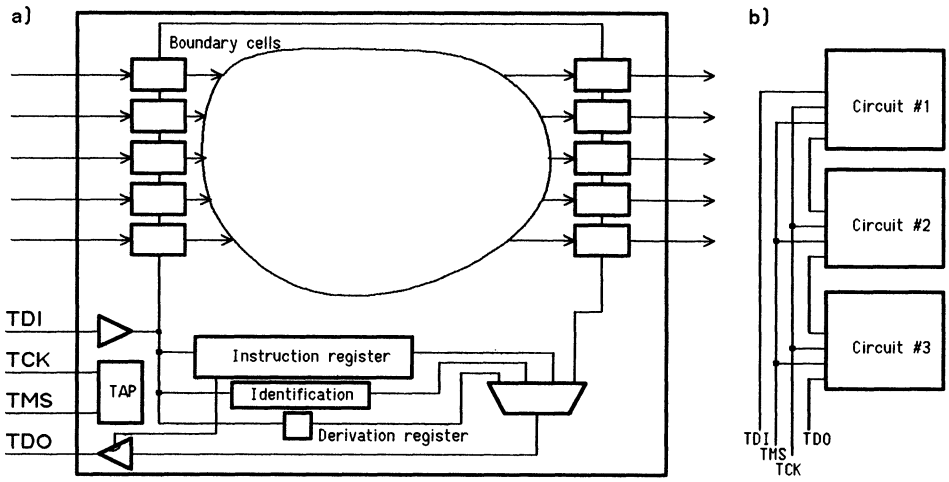


Fig. 7.6 JTAG interface

Data is shifted within the internal registers according to sequencing on the TMS line. The instruction register defines the subsequent working mode. The optional identification register gives both the manufacturer's code and that of the product. The derivation register means that it is possible to go right through the circuit with a minimum delay of one clock pulse without taking into account its internal structure. The boundary or data register assigns and retrieves useful information. It can be tripled in the most complex cases.

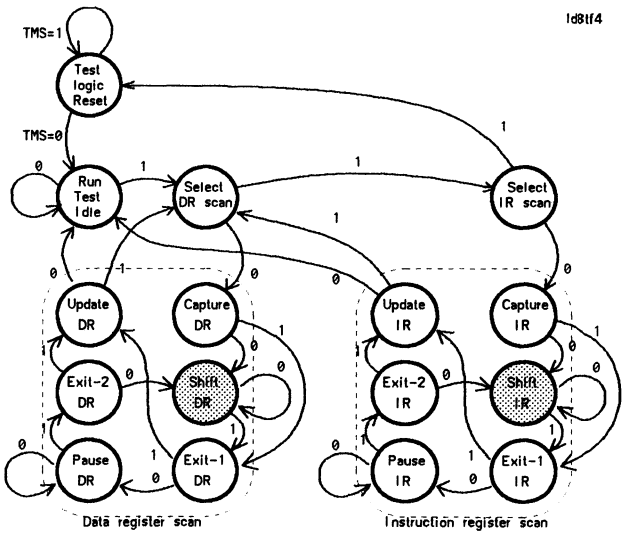


Fig. 7.7 Graph showing JTAG modes

The curve of figure 7.7 shows the states of the system, which can be controlled by a simple interface card in a compatible PC; special circuits are available to make it easier to use this interface. The TMS signal is sampled at each clock pulse in order to define the state of the system. In the *test logic reset* mode, which can be reached in all cases in 5 clock pulses with TMS=1, test logic is initialized. The *run-test/idle* rest state generally follows; it is reached and maintained by keeping TMS at 0. In this state, test logic has no effect whatsoever on the application logic.

During the first stage, the instruction register is selected and loaded (mode *Sel IR Scan*). During the second stage, the selected register (*Sel DR Scan*), which is generally the boundary register, is loaded and transferred. The sequential *TAP* (*Test Access Port*) machine generates internal signals according to each state (figure 7.8). For example, ResetZ, which initializes all the registers, is activated during the *test logic reset* and *IrUpdate* states, which loads the instruction register, is activated in the *Capture IR* state.

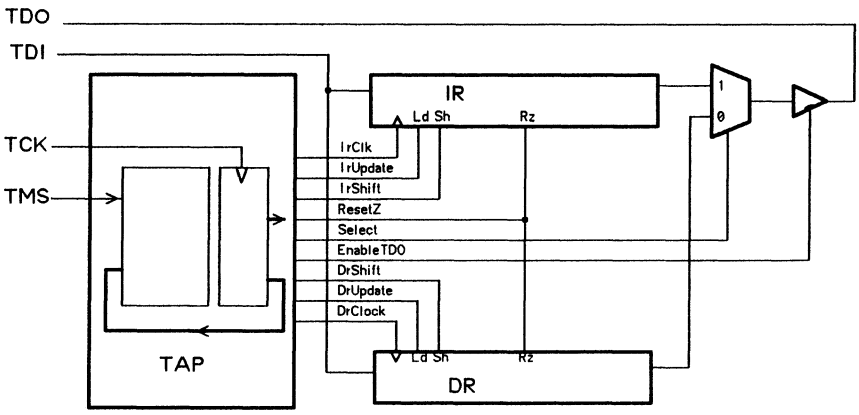


Fig. 7.8 Sequential TAP machine

The timing diagram of figure 7.9 shows a few of these signals used to load and shift registers when the instruction register has 4 bits and the boundary register 8 bits. The documentation issued by the main manufacturer, Texas Instruments, explains very clearly the various transfer modes which correspond to the standard. The graph of figure 7.7 seems quite complex but it enables elementary operations to be sequenced in a very distinct manner. For example, the register is shifted after a clock pulse so that the scan register can be loaded in parallel, and is followed by another clock pulse so that the new contents can be loaded in to the latch which sends the signals to the application logic.

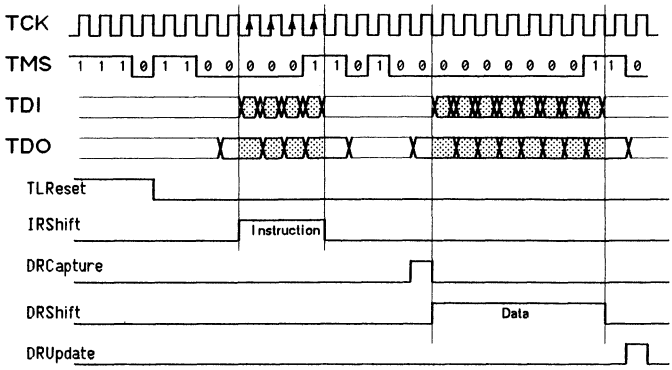


Fig. 7.9 Typical sequencing for loading an instruction and carrying out a transfer

Some register circuits and drivers of the TTL family are available with 4 extra pins and JTAG standard compatible logic (Appendix).

7.4.9 Signature analysis

The basic principle of signature analysis has already been discussed in section 7.2.5. There are several variants known as *LFSR (Linear Feedback Shift Register)* and *BILBO (Built-In Logic Block Observer)* [McCluskey86]. According to the JTAG standard, there is a mode in which the internal register can be wired so that it acts as a linear one. This enables *Pseudo-Random Pattern Generation (PRPG)* and *Pattern Signature Analysis (PSA).*

7.5 Debugging prototypes

Initial debugging of an integrated circuit or of a printed circuit starts in the same way. Physical defects are not due to identical causes and it is rarely possible to repair an integrated circuit; the designer must repeat all simulations and redo the circuit completely, which takes several months.

7.5.1 Testing integrated circuits

The designer should ideally have a tester which is supported by the CAD system used for designing and simulating. Test vectors are automatically generated using the simulation programs as a basis. However these are lengthy operations and require a good knowledge of the CAD system. The time it takes to design an integrated circuit is usually less than one tenth of the total time; the time needed for simulation and generating test vectors is always underestimated in projects.

7.5.2 Testing printed circuit boards

The complexity of a printed circuit board, which includes a microprocessor, memory and programmable interfaces, is such that a complete simulation is impossible. Moreover, it is unnecessary since a bus structure decomposes the problem into very distinct simple situations: decoding and selection timing. Random logic is found in PLDs whose logic simulation can be carried out using a large number of tools. Propagation time ranges are most often calculated manually.

When debugging and carrying out production tests, the following problems may arise:

- incorrectly inserted component (bent pin, misalignment)
- defective component
- microcut on the printed circuit track
- short-circuit on adjacent tracks
- forgotten solderings

Careful visual control marks the first stage of debugging. Microcuts and forgotten solderings are noticeable for they leave some inputs unconnected, a thing which should never occur in a normal circuit. A logic probe can pinpoint these unconnected inputs and follow the conductors along to the defect. If a circuit is influenced by a touch of the hand, it usually means that there is an unconnected input.

Short-circuited links show up on the oscilloscope because of easily recognizable half-levels; they are difficult to pinpoint with a logic analyzer. The exact short-circuit location is sometimes hard to find. One solution is to seek the lowest voltage drop (figure 7.10).

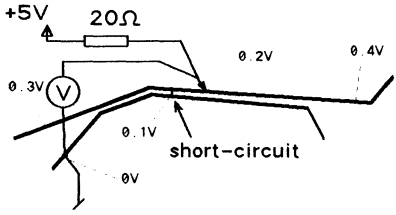


Fig. 7.10 Looking for a short-circuit

Problems are often due to poor-quality ground wires or to faulty circuit uncoupling. Circuits made up of four or more layers considerably reduce problems caused by uncoupling and crosstalk, but critical working conditions remain difficult to correct. As a rule, a well designed system should work from 4V to 6V, although guaranteed voltage is 5V±5%. Extending the voltage range compensates for the fact that the temperature range is seldom explored completely.

7.5.3 Special techniques

It is possible to debug a microprocessor system very efficiently by repeating the processor start up at a few dozen kiloHertz (section 8.3). If the clock and the reset pulse are correct, the processor will execute at least one instruction. All that is required is a program in EPROM which selects peripherals and memory. Figure 7.11 shows a 68030 with a 20MHz clock in its starting phase. The first two $\overline{AS}$ (address strobe) cycles exists even if memory is not available. The first instruction reads a peripheral named Fast. Another slow peripheral is read two instructions later.

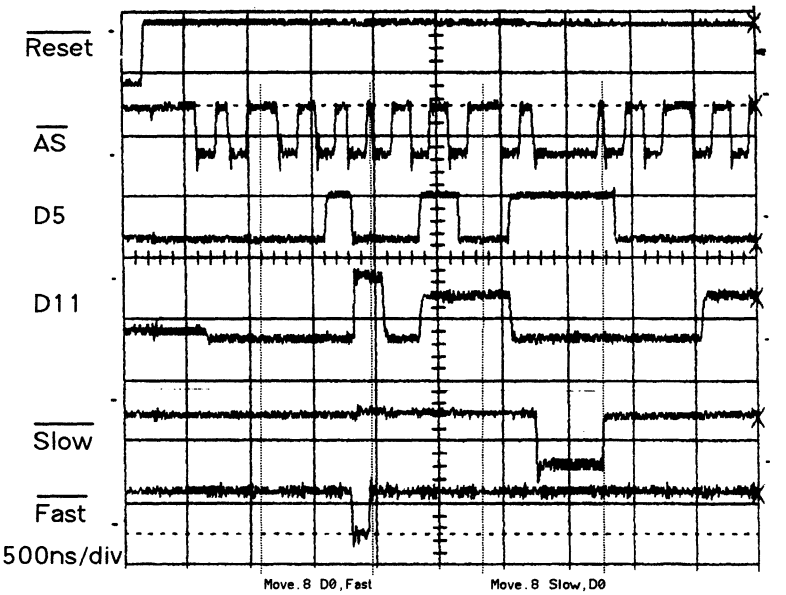


Fig. 7.11 Start up signals for a processor

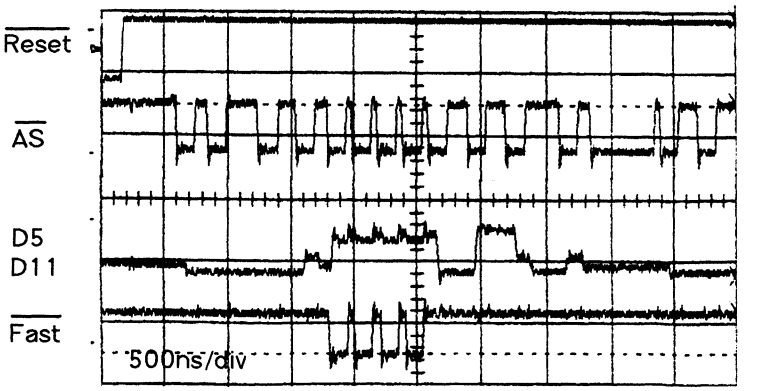


Fig. 7.12 Short-circuited processor signals

Caches and prefetched instructions complicate timing analysis, but I/O selections can act as a guideline when dealing with long instruction sequences.

Any faulty selection, short-circuit between address or data lines, signal suffering from a large overshoot, poor ground line, etc. is easily viewed on an oscilloscope. For instance in figure 7.12, the two data lines D5 and D11 have been short-circuited. The processor can no longer operate correctly; it asks for a 32-bit word for the 8-bit peripheral named Fast and executes four 8-bit transfers. The oscilloscope shows signals with levels at about 1.5 V, indicating short circuits. Once such a signal has been viewed on a line, locating other identically shaped signals enables the designer to pinpoint other short-circuited lines. In most cases, the problem can be seen at a glance. The test of figure 7.11 is then required if the tracks are routed under packages or inside the internal layers.

As soon as random signals such a interrupts and DMA are in service, a logic analyzer becomes necessary because of the random aspect of these events. So that arbitration between two signals can be tested correctly, it must always be synchronized with a variable phase; an oscilloscope can be used to scrutinize the metastability zones no simulator is capable of analyzing correctly.

Debugging is often complicated by physical access to signals. In the case of modern high speed buses, using extenders can create problems since reflections on the lines are influenced by a change of size. Designing processor and sophisticated interface boards often raises the question of how to coordinate hardware and software teams: hardware tests are carried out using simple software in simplified hardware and software environments (few boards, few interrupts and DMA accesses). Software teams often have difficulty in separating errors due to their programs from those due to hardware defects which did not show up during the test programs.

7.5.4 Memory tests

In all modern systems, the memory plays a vital part and must be perfectly reliable. The main problems which can appear are:

- defective memory cells
- cut or short-circuited addresses
- glitches due to reflections on the lines
- faulty access times, e.g. following an arbitrator's metastable states
- insufficient recovery times
- faulty refresh

Memories are mostly very reliable components and remain easy to implement, although technology is changing extremely rapidly. Memories are based on a regular structure and their behaviour is easy to describe; therefore, they should be trivial to test. In fact, the large memory sizes which are in use nowadays require sophisticated testing algorithms, which take into account the internal design of the memory chips. When a cell changes state, it influences neighbouring cells and amplifiers on the circuit, but not necessarily within the addressing zone.

There are a large number of memory testing algorithms [Mizco86], [VanDeGoor91]. The *galloping pattern (GALPAT)* test is easy to program but its execution time is proportional to the square of the memory size. After reset, a pattern is written at the first memory address and all other addresses are checked to ensure that they have not been influenced. The same operation is then repeated for the following addresses. It is preferable to check only the area close to the affected cell or to test diagonally in order to simultaneously check two decoders in x and in y.

A special tester is required to check the parameters precisely and carry out the test as quickly as possible. When a processor tests its own memory, it is necessary to use instructions which create a read/write sequence (shift, memory increment) in order to make sure that dynamic memory recovery times are respected. If a read operation immediately follows a write one, missing memory does not show up since the bus line remains a floating one and memorizes the previous state. Repeating the test with a higher clock frequency, a 4.5 and 5.5V tension, a high temperature then a low one gives the equivalent of a safety coefficient.

Read only memories, and especially EPROMS, can lose their information after a time. The test can only be carried out if a checksum is placed at the end of the memory during the manufacturing process. Calculating this checksum at various speeds and tensions guarantees correct working order for a time. The checksum may be just a sum whose report is placed on the least significant bits. If there is a risk of several bits being degraded at the same time, it is preferable to use a cyclic sum (CRC) [DelCorso86] based on the same principle as pattern signature analysis.

7.5.5 Interferences and disturbances

A system's reliability depends enormously on the quality of its power supplies and on signal noise. Disturbances due to modern high speed circuits, and which must be reduced as much as possible, affect neighbouring circuits as well as distant pieces of equipment such as television sets.

Using *ground planes* avoids crosstalk and characterizes line impedance more precisely. A shield linked to the ground avoids outward radiation. Some applications require internal functioning not to be disturbed by abnormal electromagnetic disturbances.

There are various techniques for detecting and correcting errors. The simplest one is the parity bit. These techniques are mostly used for memories and buses [DelCorso86]; they require extra bits as well as a design which increases the price by about 30%.

It is often difficult to pinpoint the cases of critical behaviour in a system. An oscilloscope disturbs the signals it is observing; therefore, there must be a reference grounding wire very near the end of the probe. When faced with a problem, the designer should try to answer the following questions:

- are answer times correct?
- are all clocks and sample signals clean?
- are propagation times acceptable throughout the board?
- are the components sufficiently uncoupled?
- are there any metastable states?

7.5.6 Repairing

If a system which has been running satisfactorily breaks down although there have been no abnormal mechanical or electric (overload) demands, the breakdown is usually due to a single component. Drivers, PLDs and registers often blow following overloads on the inputs-outputs. PLDs also sometimes mysteriously lose bits. Mass production has shown that most breakdowns are due to a small number of components. At the design stage, it is possible to define a repair procedure. It is similar to the debugging test procedure but includes faster and more superficial stages, which are specially designed to locate a defective component. The whole battery of tests is run once the repair has been made.

7.5.7 Conclusion

Designing interfaces and digital systems has evolved enormously in the last ten years. However, most small companies have not yet adjusted to this change which implies using up to date CAD tools and a wide range of components (processors, PLDs, programmable arrays) supported by specific development tools. Large-scale production is influenced more by VLSI circuits and complex programmable components. This also means greater investments as well as teams including large numbers of specialists for each component and software tool.

It will still be possible for some time to produce small series (up to a few thousand pieces) based on quite simple components and use the techniques described in this book. The small teams of engineers who design such products must keep abreast of a wide range of products and their associated software tools.

8 Design examples

In order to design an interface or a specialized digital system, the designer must possess a sound knowledge and a good imagination. A small number of examples cannot cover enough situations to enable the reader faced with a new problem to get a feel for the various possible approaches and know where to find the required information. The aim of this chapter is to treat each problem in depth, to stress the various solutions and to justify completely the solution which is finally chosen.

8.1 Simple sequencer

It seems quite easy to build an interface using the TTL or CMOS circuits which are on the market. There are a large number of solutions and they are easy to find. The aim of this first artificial example is to show the different guises under which a simple problem based mainly on flip-flops can appear.

8.1.1 Pulse generators

It is necessary to generate two signals which are out of phase with a period of 6 clock pulses (figure 8.1).

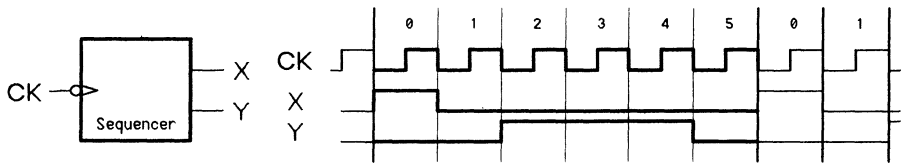


Fig. 8.1 Description of the problem

The solution is a trivial one; a counter by 6 controls a combinational system which is easy to synthesize (figure 8.2a). The X and Y equations are a simple consequence of the Karnaugh map (figure 8.2b). It is interesting to note that the decision to activate X for state 0 rather than for a next state is rather an arbitrary one. In some cases, the equations can be simplified by shifting the initial state.

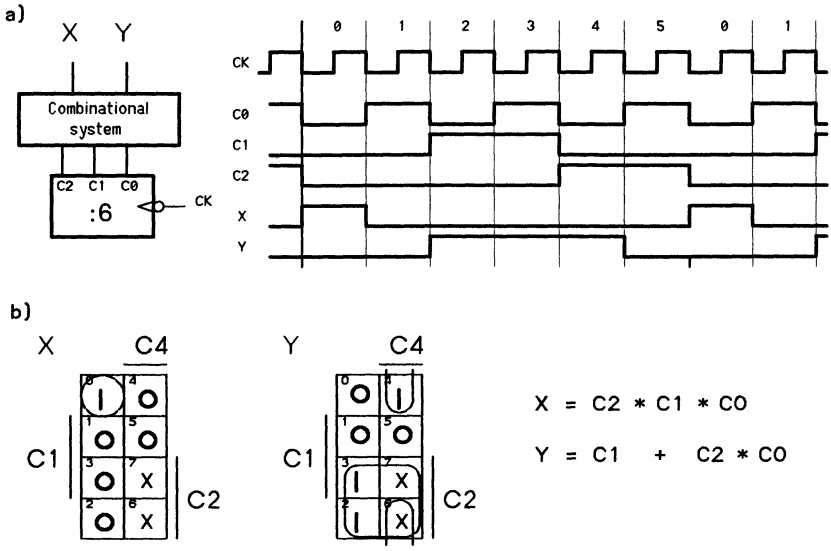


Fig. 8.2 Solution using a binary counter by 6

8.1.2 Solution using decoders

As a reminder, let us mention that logic functions can sometimes be performed satisfactorily using complex modules like decoders, although this is not the case here. When this type of solution is adopted, problems related to glitches during decoding usually remain as bad if not worse than before.

8.1.3 Getting rid of glitches

In reality, any implementation using gates or decoders can become problematic because of propagation times and *glitches* due to decoding. The problem is very obvious in the case of an asynchronous counter but can also appear with a synchronous counter.

Figure 8.3 analyzes the solution based on an asynchronous counter by 16 which has been made into a counter by 6, while maintaining an identical propagation time for all the logic elements. X and Y implementation is based on the equations of figure 8.2. Let us note that X is shortened by the propagation times, especially when the asynchronous reset occurs, and that, like Y, it suffers from glitches.

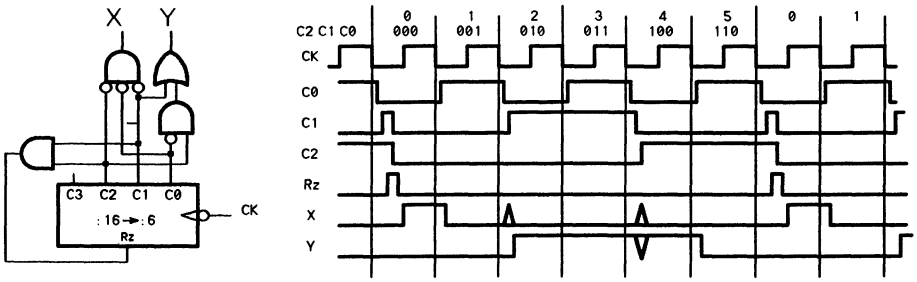


Fig. 8.3 Timing diagram showing propagation delays

Glitches due to decoding, as shown in figure 8.3, are not usually acceptable. Using a synchronous binary counter can help, but is not absolutely safe. One general way of solving the problem is to synchronize the outputs with a parallel register made up of D flip-flops (figure 8.4). A latch controlled by the CK signal, which is therefore closed while the signals change, is a valid solution, albeit a slower one (maximum frequency is reduced by half).

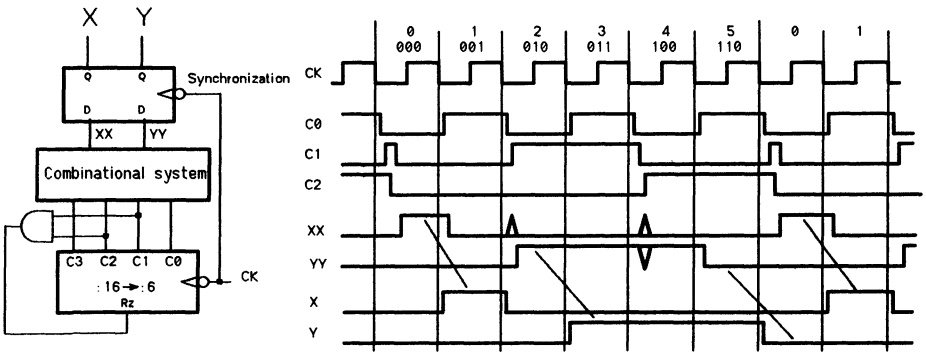


Fig. 8.4 Synchronizing the outputs

In order to suppress the synchronization level caused by glitches due to decoding, it is necessary to use a Gray code counter whose only variable changes from one state to the next. In the case of a counter by 6, a *ring counter* is the most efficient solution (figure 8.5).

The equations of the **X** and **Y** functions are easy to set up. **X** must be active at 0, and **Y** active for the states which are now known as 6, 7 and 3, i.e. when **B** is active.

$$X = A * B * C$$
$$Y = B$$

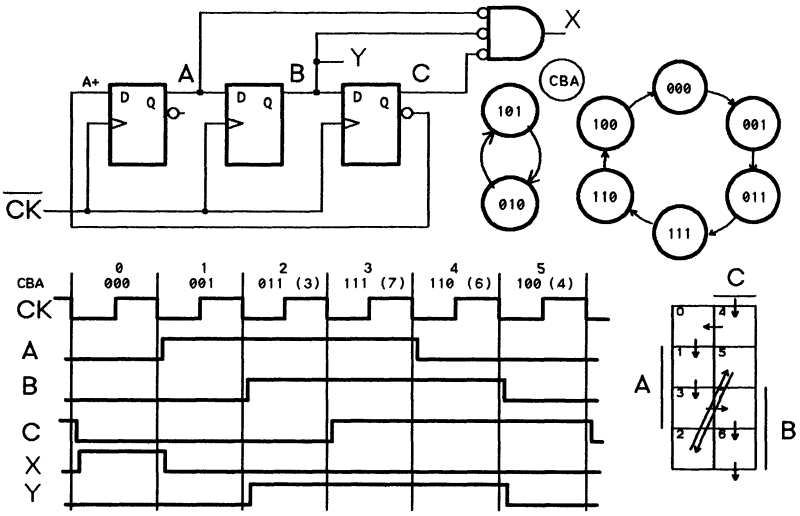


Fig. 8.5 Ring counter and graph

The previous counter may operate in the two extra states (2,5). This can be avoided by introducing an initial reset or forcing the change-over from state 5 to state 4 by adding two logic gates (figure 8.6).

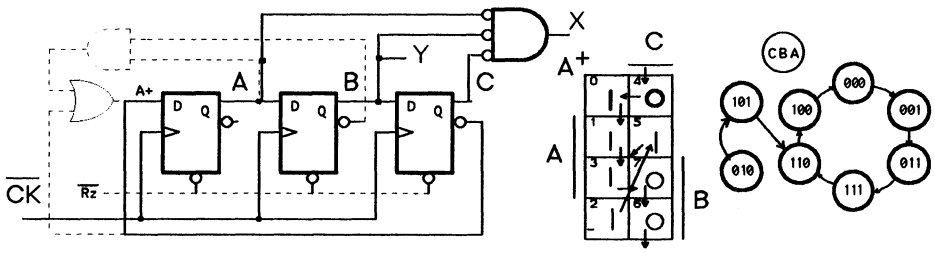


Fig. 8.6 Suppressing unwanted states

8.1.4 Optimized solutions

A simple logic function can often be performed using a flip-flop. The above X variable is activated e.g. on C's falling (or on $\overline{C}$'s rising) edge and deactivated when A is active. A D flip-flop with $\overline{C}$ on the clock input and $\overline{A}$ on the reset can therefore replace the gate equipped with three inputs and shown in figure 8.5. The two gates which force the counting mode into a counter by 6 can be replaced with an extra flip-flop which changes over to 1 at state 000, and thus confirms that the B flip-flop must be at zero in this phase (figure 8.7). In states 3, 4 and 5, the X output is at zero due to the reset forced by B. This diagram is the most elegant one of all, since the X signal is identical to that of the extra flip-flop, which plays a double part. Moreover, let us note that the system is a synchronous one and therefore avoid the delay of X compared to Y.

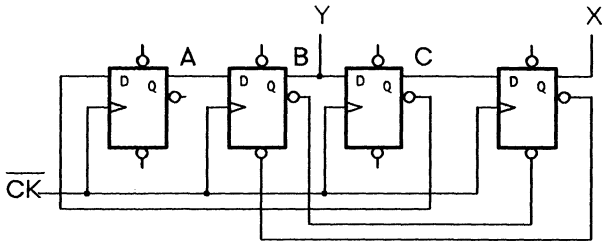


Fig. 8.7 Solution without decoding gates

8.1.5 General solution

The general solution for sequential systems is shown in figure 8.8. Four flip-flops are used with an encoding which generates signals **X** and **Y** directly in order to avoid glitches. This encoding choice is very important; if direct encoding is not possible, it is preferable to generate the output using a flip-flop rather than gates, so that the outputs remain properly synchronized with each other. The latter solution is the best suitable in the case of a PLD, e.g. a 16R4 or 16R8.

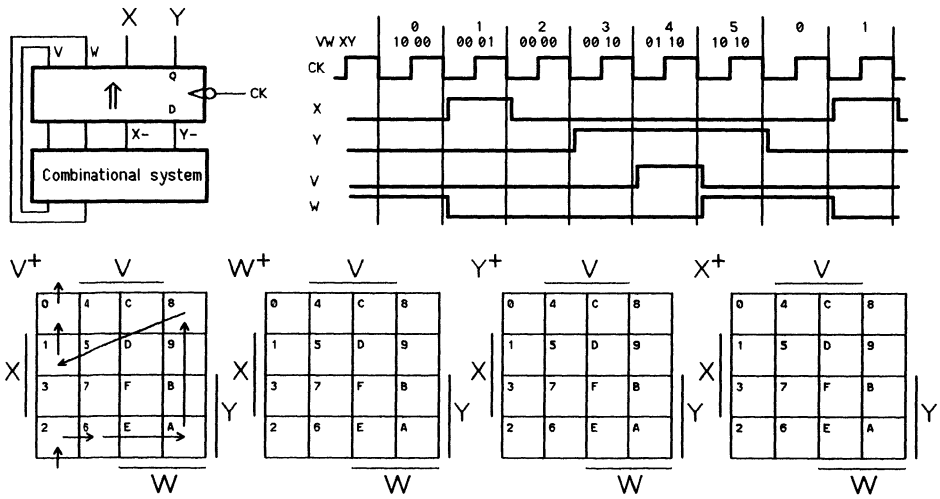


Fig. 8.8 Sequential machine method

If encoded differently, a more elegant solution can be found. The disadvantage of general solutions is that they are seldom elegant. On the other hand, they systematically analyze illegal states. The design is frozen as soon as the first general solution has been found when development time must be minimized and when the cost of a regular decoding structure is negligible, as is the case with PLDs.

8.2 Frequency divider

It is easy to divide a frequency by an integer: it just has to be divided by the number. A fractional division is also possible but requires a little more logic and produces non equidistant output pulses. Clock circuit manufacturers are faced with this problem when they want to generate 100Hz using a 32kHz quartz, i.e. 2^{15} = 32'768Hz. These quartz crystals are mass produced and cheap and it is preferable to add a hundred transistors rather than to have to use a 32'000 or a 100'000Hz quartz.

8.2.1 Rate multipliers

The *rate multiplier* shown in section 3.6.7 solves the problem completely (figure 8.9a). A 15-bit multiplier must be used with an input value of 100, which is of course encoded in binary i.e. 2'1100100. Thus, at each counter cycle, 100 pulses are generated on the output with a placing error of half a pulse. This is not a realistic solution since building a 15-bit multiplier means using gates with too many inputs; it is therefore necessary to work out how to decompose it using a library of standard elements.

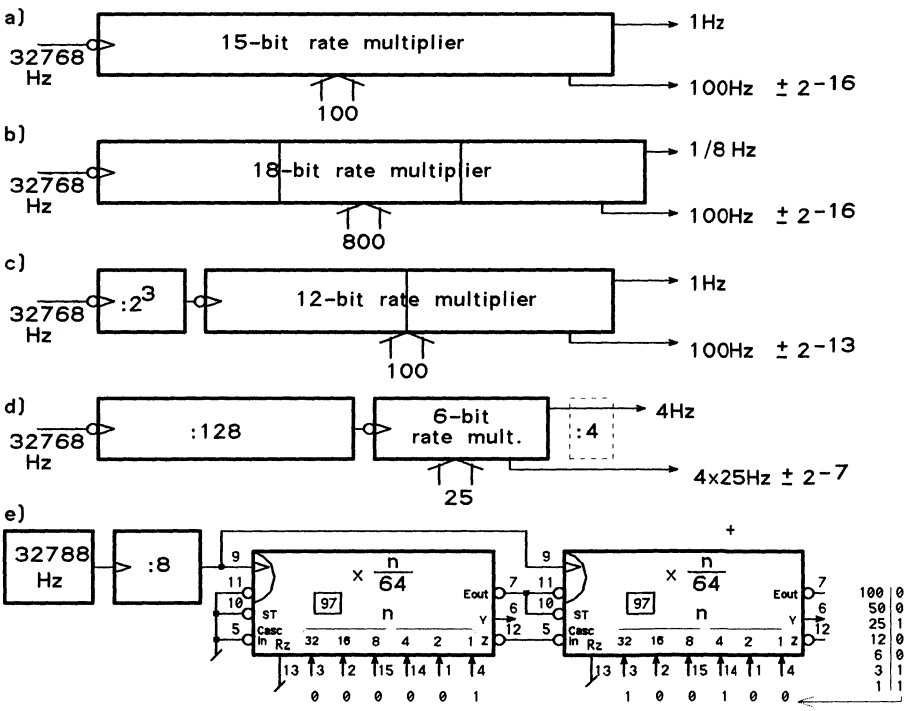


Fig. 8.9 Solution based on rate multipliers

The solution of figure 8.9b is not very satisfactory. It is preferable to predivide using a binary 3-bit counter (figure 8.9c), which brings precision down to 2^{-13}. This solution uses two TTL 7497 or CMOS 4089 type circuits (figure 8.10e). A cheaper albeit much less precise (1%) variation only requires one 6-bit multiplier (figure 8.9d).

Programmable multipliers are suitable when generating variable frequency pulses but the fact that both these circuits have not been developed for LS and HC technologies shows that they are seldom needed or that applications which call for multipliers are not solved in such a manner industrially.

8.2.2 Other solutions

When seeking a solution which means reinventing programmable multipliers, it is easy to fall back on the idea of using a counter which divides by an approximate value and to compensate by subtracting or adding pulses. It is thus possible to start with 256Hz in order to end up with 100Hz. With 250Hz, a division by 2.5(alternatively by 2 then 3) would be suitable. It is only necessary to suppress 6 pulses out of every 256, i.e. one pulse out of every 42. This leads to the block diagram shown in figure 8.10a, with a counter by 5 generating two pulses for every cycle, a counter by 42 which inhibits a counting pulse at each cycle and a counter by 256 which performs a complete reset every second. The timing diagram (figure 8.10b) can be used to analyze sequencing carefully and a fine error analysis can lead to a variation on the diagram which may gain a few per-cent. In such cases, an extremely complete simulation or a test is absolutely essential in order to avoid mistakes which the human brain has difficulty locating among such long quasi repetitive sequences.

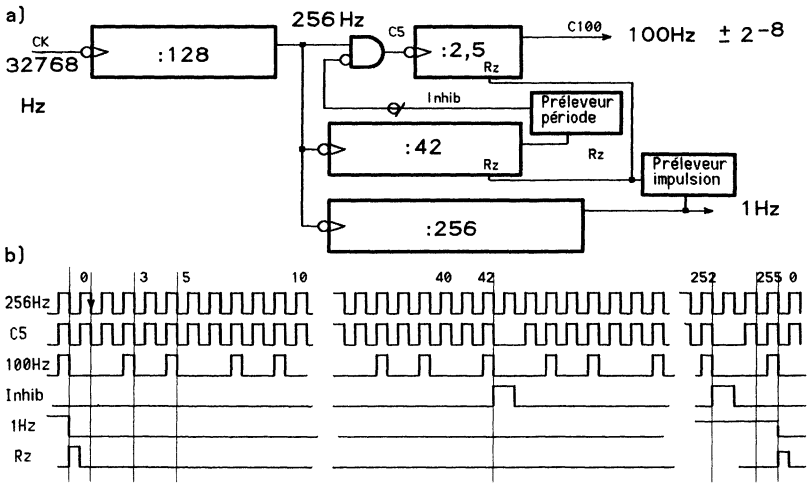


Fig. 8.10 Not very precise solution using three counters

EM (EM Microelectronic-Marin SA, Switzerland) has used a more elegant solution in its VLSI *real time clock* circuits. It is based on the fact that 100Hz·320=32'000Hz. The 768=256·3 pulses which must be subtracted every second can be detected using a counter by 128, which generates the 256Hz required by the circuit. The block diagram therefore becomes very simple (figure 8.11). Since 128/3=42,66, states 0, 43 and 87 of the counter by 128 are decoded in order to inhibit the activity of the counter by 320. One advantage of this solution is that it guarantees maximum precision with a simple hardware.

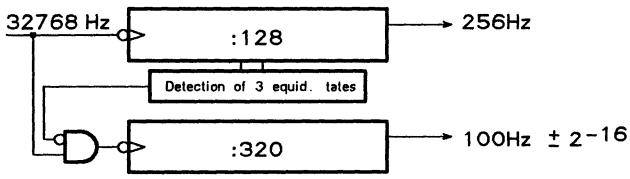


Fig. 8.11 Solution based on two counters

8.2.3 VLSI implementation

An ideal VLSI clock circuit uses as few transistors and as little power as possible. Therefore, counters are quasi synchronous since propagation times are relatively unimportant at such low frequencies. Seven flip-flops divide by 128. The decoding of the three states which require the clock to be inhibited is not done for the first three aftermentioned values i.e. 0, 43 and 87. Any three equidistant values will do; they should be chosen so as to make implementation easy. Decoding state 0 means using a NOR (AND with inverted inputs) gate with 7 inputs, which is not suitable in CMOS. EM has chosen states 11, 53 and 96 so that each value is decoded by a NOR gate and a NAND gate cascaded with less than 5 inputs (figure 8.12).

The counter by 320 is built like a counter by 64 followed by a counter by 5. The counter by 5 uses 4 flip-flops rather than 3 flip-flops and a gate. It really is a counter by 4+1. State 4 is copied onto the D flip-flop. Reset time equals the sum of the two last flip-flop propagation times, which seems sufficient to guarantee that the first two flip-flops have enough time to reset although this must be simulated with care. For an unknown reason, the 5Hz clock is generated by a half cycle sampler (figure 3.34c). This VLSI circuit was built using precharacterized VTI VLSI library elements completed and improved by EM.

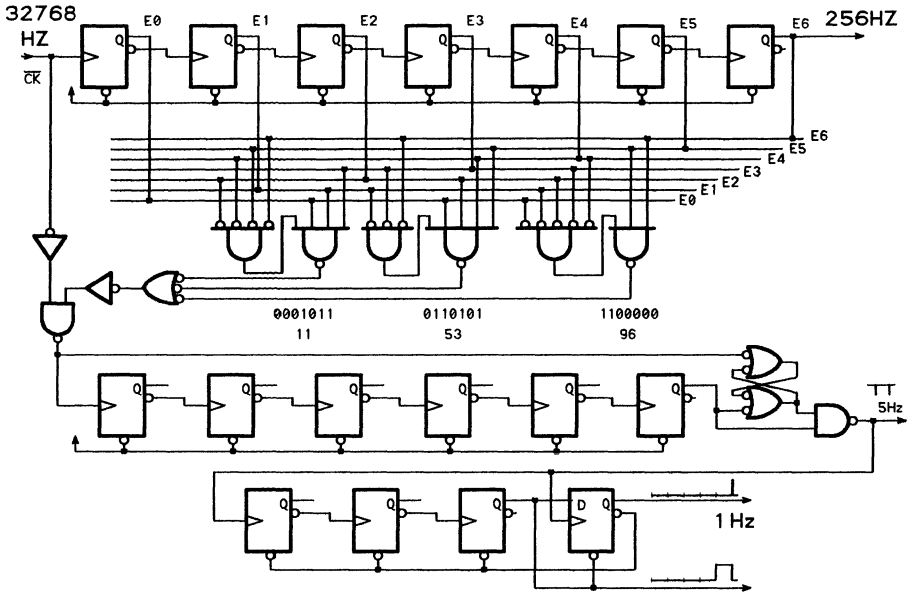


Fig. 8.12 VLSI diagram used in the EM circuits.

This example shows the variety of solutions which can be used for a relatively simple but non trivial example. The solution which has been chosen is strongly influenced by the application as well as by the standard modules which must be used for the application.

8.3 Sequencer for repetitive resets

The repetitive reset technique (section 7.5.3) is very efficient for debugging microprocessor systems and may be used in other applications. The only thing needed is a counter which generates **RzOut** pulses (figure 8.13), although extra flexibility requires that pulse length and width be modifiable. Moreover, a shifted **InterOut** pulse is useful when generating a second signal, i.e. an interrupt.

8.3.1 Problem specifications

As the device must be easy to use and optimized for a fast processor like the 68030, the specifications are as follows:

- input frequency of 50MHz Max
- length of reset pulse at 50MHz: 1μs, 10μs
- pulse period: 10μs, 100μs, 1ms, ∞
- interrupt delay: 10μs, 100μs

Since the aim here is to understand the synthesis method, the reader should repeat this exercise based on his own problem specifications. They will be modified at a later stage in order to make implementation easier. This is quite common: the service which the client requires can be performed with more or less comfort and it is sometimes difficult to come up with the best compromise between the quality of this service and the price to pay.

The circuit block diagram (figure 8.13) shows that an output has been planned in order to synchronize the oscilloscope. This signal is identical to **RzOut** but one of the modes deactivates the repetitive reset. In this case, a 1st input is used for a manual reset and a 2nd input synchronizes the oscilloscope.

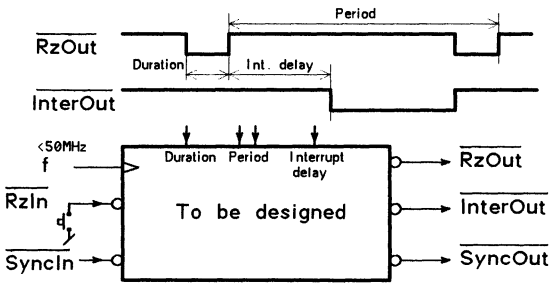


Fig. 8.13 Block diagram

8.3.2 Possible solutions

From 50 MHz to 1 KHz, a frequency division of $5 \cdot 10^4 \simeq 2^{16}$ is necessary. The required precision is not very great and it is preferable to use a predivider, which is also necessary because of the high input frequency. Let us choose a predivider by 4, which can be by-passed when the input frequency falls below 12 MHz. This gives the decomposition shown in figure 8.14a and means that the maximum input frequency of the programmable part is 12,5 MHz. The extra **RzIn** and **SyncIn** inputs only have an effect in the "NoRepeat" mode; a provisional decomposition with an output switch means that these two signals can be put aside until the design is almost ready so that the designer can focus his attention on sequencing.

A binary counter (figure 8.14b) can be used to divide the frequency as desired. Depending on the mode, the state of the counter is monitored by logic which generates a reset pulse on the counter in order to produce the corresponding period. The asynchronous 74HC4020 counter by 2^{14} has a maximum frequency of 20 MHz and seems to be very suitable. However, the 38ns propagation time between the clock and the last flip-flop can become very awkward when dealing with partially synchronous logic. Moreover, this solution is inconvenient since the logic block requires a large number of inputs, which is not very compatible with PAL use. Therefore we will not pursue its study any further.

A much preferable solution uses the counter by 2^n studied in section 3.6.2. Its greatest advantage is that it offers large division factors (2^{31} with the 74LS292 circuit) and leads to a logic block which has 5 inputs and 8 outputs (figure 8.14c).

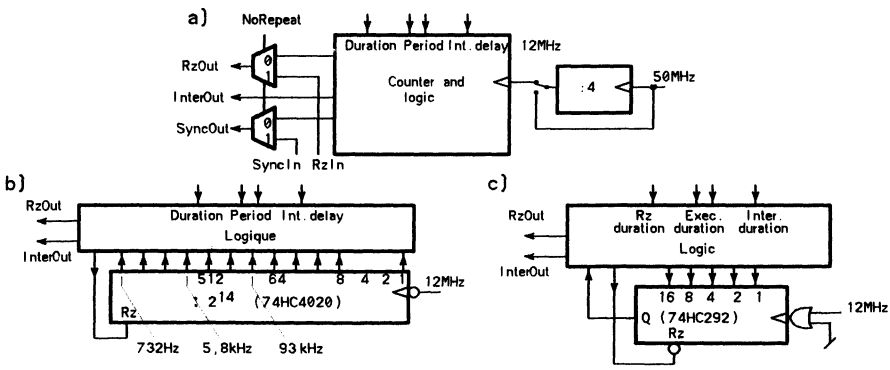


Fig. 8.14 Initial solutions

8.3.3 Simplified solutions

The basic idea with the *counter by* 2^n is to modify the counting value during a cycle in order to set reset length and running time. This means a first change in the problem specifications; it is unimportant since the reset is only a short one.

The divider by 2^n must be reset each time the value of n changes. If the desired **RzOut** pulse length equals n_1=4 clock cycles and the active period n_2=256 cycles, the 74LS292 must be preceded by a switch in order to alternately count by n_1 and by n_2 and thus conform to the timing diagram shown in figure 8.15. The period then becomes n_1+n_2, which is not important. The counter input must successively be $2=00010_B$ ($\log_2 4$ and $8=01000_B$ ($\log_2 256$) The sequencer synthesis is obvious: at each cycle of the programmable divider, it is necessary to change the state of **RzOut** and generate a brief **RzCnt** pulse (figure 8.15a). The **RzCnt** flip-flop is active for half a period; good operating conditions are guaranteed since the programmable divider has a long propagation time (>38ns), and the reset must be removed from the flip-flop only a few ns before the clock edge. Maximum operating frequency is limited by this reset sequence.

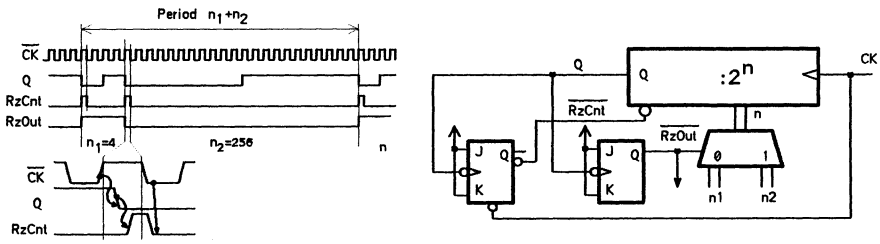


Fig. 8.15 Quasi synchronous solution

A synchronous solution is preferable but means that the reset lasts a whole pulse length and that consequently the counter will remain in state zero for two successive pulses (figure 8.16a). This does not matter in the case of the present application. Combining the Q signal with the same signal delayed by a QQ pulse generates the reset pulse and the switch-over condition of the **RzOut** flip-flop. If the divider by 2^n was capable of generating a one period Cout at the end of its counting, the way synchronous counters do, this flip-flop would be unnecessary. The **RzCnt** reset should not go through a flip-flop if this pulse is not to be delayed and a third zero state added, which could maybe be acceptable in this application. The initial equations are given in figure 8.16a Their developed version, which can be used by the PLD assembler, is given on the following page.

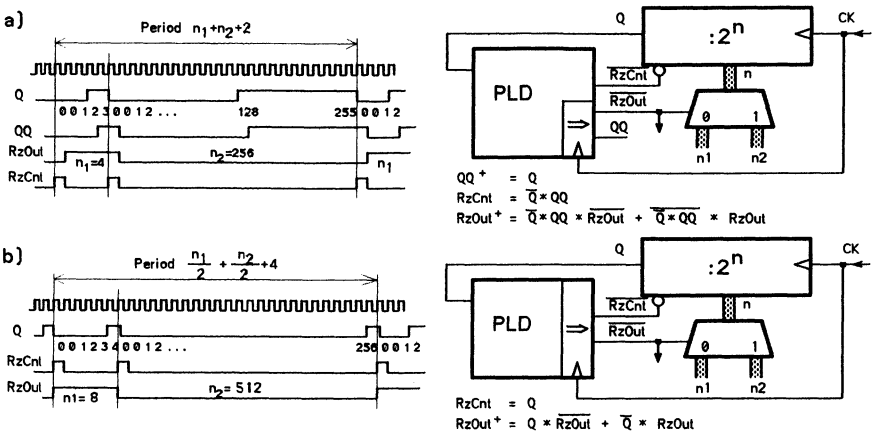


Fig. 8.16 Synchronous solutions

QQ := Q
RzCnt = /Q * QQ
RzOut := /Q * QQ * /RzOut + Q * RzOut + /QQ * RzOut

The output amplifier inverts these terms, which is practical in the case of **RzCnt** and **RzOut** and is of no importance as regards **QQ**. If an inversion is necessary, the equations can easily be inverted algebraically or using the Karnaugh method (section 2.3.1).

One variation (figure 8.16b) saves the QQ flip-flop and indirectly replaces it by one of the flip-flops of the counter by 2n (i.e. and sets a limit on the maximum counting value). The double value is preselected. When the counter's Q output switches over to one, once half of the counting value has been reached, the reset is activated during a whole period (which maintains state 0 for two successive periods) or during the first half-period (which is slightly less compatible with a synchronous PLD but may sometimes prove a good solution). A purely synchronous solution, like that discussed for the cycle sampler of figure 3.33, is extremely dangerous since the divider by 2^n is equipped with 32 cascaded flip-flops; specifications require the reset pulse to last 130ns.

8.3.4 Complete solution

Because of the problem specifications, several values of n_1 and n_2, as well as n_3 for the interrupt, must be programmed according to the mode. Five 3-input switches in front of the counter is not a realistic option. It would be easy to implement a ROM but the appropriate circuits are not available. The best solution uses a PLD like in figure 8.16. The block diagram including all necessary signals is shown in figure 8.17.

A PLD equipped with 7 inputs and 10 outputs is used. This type of PLD is available but according to the wishes expressed in the problem specifications, the module must be small and cheap. Moreover, more elegant solutions can only be sought after the first one has been found.

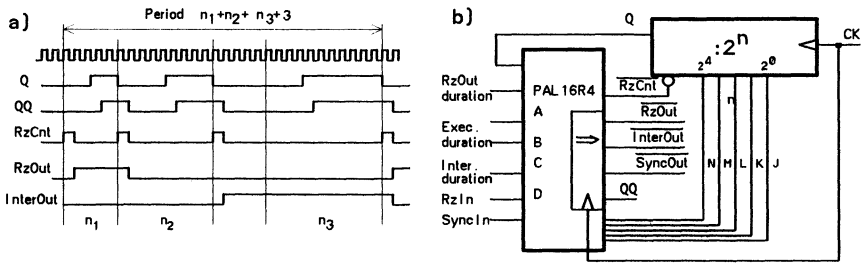


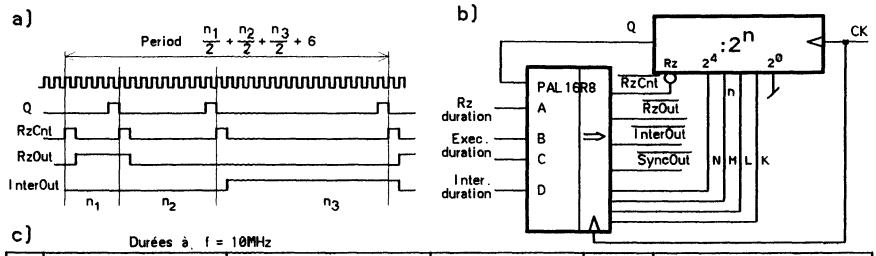
Fig. 8.17 Solution compatible with a 10 output PLD

8.3.5 Optimizing

The solution can be made simpler by suppressing the QQ output as shown in figure 8.16, i.e. by limiting maximum delays to 2^{30} clock periods. It is also possible to use only four inputs of the programmable counter. Delays which can occur are multiples of 4 if the least significant input is ignored; since the problem specifications are based on delays which are multiples of 10, multiples of 16 are acceptable. Ignoring the most significant line limits the lowest frequency to $12MHz/2^{15}$, i.e. around 400Hz which would not be acceptable and will not be pursued. It is now possible to use a simple 16R4 or 16R8 type PLD (figure 8.18b) since the inputs into the divider by 2^n can be direct or synchronous. The values of n can be modified either at the beginning or at the end of the divider reset.

The designer will now need to muster all his concentration in order to carry out a very tedious operation. The problem specifications must be defined exactly taking into account the constraints imposed by the chosen implementation. This leads to the table in figure 8.18c, in which the values of n_1, n_2 and n_3 have been chosen so as to offer a range of usable lengths of time. A acts on n_1 which is valid when RzOut is active. B and C act on n_2 and n_3. In the manual mode, B=C=1; the counter must be jammed in a neutral state. RzCnt can be forced into this mode so as to guarantee that Q=0.

The equations for **RzCnt**, **RzOut** and **InterOut** take into account the graph of figure 8.18a. There is a change of state each time Q is activated.



c) Durées à f = 10MHz

	Interrupt duration				Execute duration				Reset duration				Period	n for IntOut RzOut =			
	D	n3	f *	µs	C B	n2	f *	µs	A	n_1	f *	µs	µs	11	10 n_3	01 n_1	00 n_2
0	0				0 0	8	2^7+2	13	0	4	2^3+2	1	14			00100	01000
1	0				0 0	8	2^7+2		1	8	2^7+2	13	26			01000	01000
2	0				0 1	12	$2^{11}+2$	204	0	4	2^3+2	1	205			00100	01100
3	0				0 1	12	$2^{11}+2$		1	8	2^7+2	13	217			01000	01100
4	0				1 0	16	$2^{15}+2$	32000	0	4	2^3+2	1	32000			00100	10000
5	0				1 0	16	$2^{15}+2$		1	8	2^7+2	13				01000	10000
6	0				1 1	0		infinite	0		don't care				NMLK		00000
7	0				1 1	0			1								00000
8	1	8	2^7+2	13	0 0	8	2^7+2	13	0	4	2^3+2	1	27		01000	00100	01000
9	1	8	2^7+2		0 0	8	2^7+2		1	8	2^7+2	13	39		01000	01000	01000
A	1	10	2^9+2	51	0 1	12	$2^{11}+2$	204	0	4	2^3+2	1	256		01010	00100	01100
B	1	10	2^9+2		0 1	12	$2^{11}+2$		1	8	2^7+2	13	268		01010	01000	01100
C	1	12	$2^{11}+2$	204	1 0	16	$2^{15}+2$	32000	0	4	2^3+2	1	32200		01100	00100	10000
D	1	12	$2^{11}+2$		1 0	16	$2^{15}+2$		1	8	2^7+2	13			01100	01000	10000
E	1				1 1	0		infinite	0		don't care						00000
F	1				1 1	0			1								00000

Fig. 8.18 Chosen solution and specifications

Because there is an immediate reset, Q only lasts one clock pulse, which simplifies synthesis (if not it would be necessary to include a QQ flip-flop). When D is inactive, InterOut is never activated (figure 8.19b). When D is active, the system goes through the three states (figure 8.19c). Representing the graph as a Karnaugh table is a good preparation for simplifying the next state functions. Figure 8.19d gives the table corresponding to **InterOut+**. The table for **RzOut+** is a suggested exercise. The designer must also take into account the switch of figure 8.14a. When C=B=1 (NoRepeat mode), the mode is manual and signals **RzIn** and **SyncIn** must also be taken into account. All of this leads to the four last equations of figure 8.20.

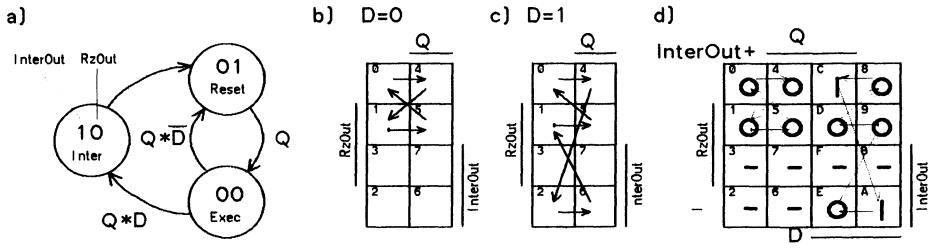


Fig. 8.19 Graph and simplification example

It is possible to write the PLD equations for the **K**, **L**, **M**, **N** functions using the table of figure 8.18c. These equations are not intuitive ones and must be simplified using the Karnaugh method. Figure 8.21a gives function **L** as an example. Since the PLD inverts the outputs, the inverted function must be simplified, which is not really a problem and leads to the equations of figure 8.20.

Let us note that a well-spaced page-layout makes reading easier and therefore reduces the risk of making mistakes. Although using explicit signal names may seem tiresome to the beginner, it is also an effective way of avoiding confusions.

```
.TITLE    RzRep.PAL   PAL for Repetitive reset
.NAME   PAL16R8
.PIN
GND       =   10                      N          =   19
VCC       =   20                      M          =   18
CK        =   1                       /RzOut     =   17
OE        =   11                      /SyncOut   =   16
Q         =   2                       /InterOut  =   15
/SyncIn   =   3                       /RzCnt     =   14
/RzIn     =   4                       L          =   13
A         =   5   ; RzOut duration    K          =   12
B         =   6   ; Exec duration
C         =   7   ; idem
D         =   8   ; Interrupt duration
.EQUATION
/K        :=    /InterOut
            +                                  /B
/L        :=    /RzOut * /InterOut          * /B
            +   /RzOut * /InterOut             * C
            +    RzOut                  *  A
            +              InterOut           * /C
/M        :=    /RzOut * /InterOut             * C
            +    RzOut                  * /A
/N        :=     RzOut
            +    InterOut
            +    B
            +   /C
RzCnt     :=     Q
            +   C * B
RzOut     :=    /C * /D *  Q * /RzOut * /InterOut
            +   /B * /D *  Q * /RzOut * /InterOut
            +   /C       * /Q *  RzOut
            +   /B       * /Q *  RzOut
            +   /C *  D *  Q          *  InterOut
            +   /B *  D *  Q          *  InterOut
            +   C * B * RzIn
InterOut  :=    /C * D *  Q * /RzOut * /InterOut
            +   /B * D *  Q * /RzOut * /InterOut
            +   /C * D * /Q          *  InterOut
            +   /B * D * /Q          *  InterOut
SyncOut   :=    /C * /D *  Q * /RzOut * /InterOut
            +   /B * /D *  Q * /RzOut * /InterOut
            +   /C       * /Q *  RzOut
            +   /B       * /Q *  RzOut
            +   /C *  D *  Q          *  InterOut
            +   /B *  D *  Q          *  InterOut
            +    C * B * SyncIn
.END
```

Fig. 8.20 PLD equation

Simulating these equations shows that behaviour is acceptable. The module can then be built according to the final diagram shown in figure 8.21b

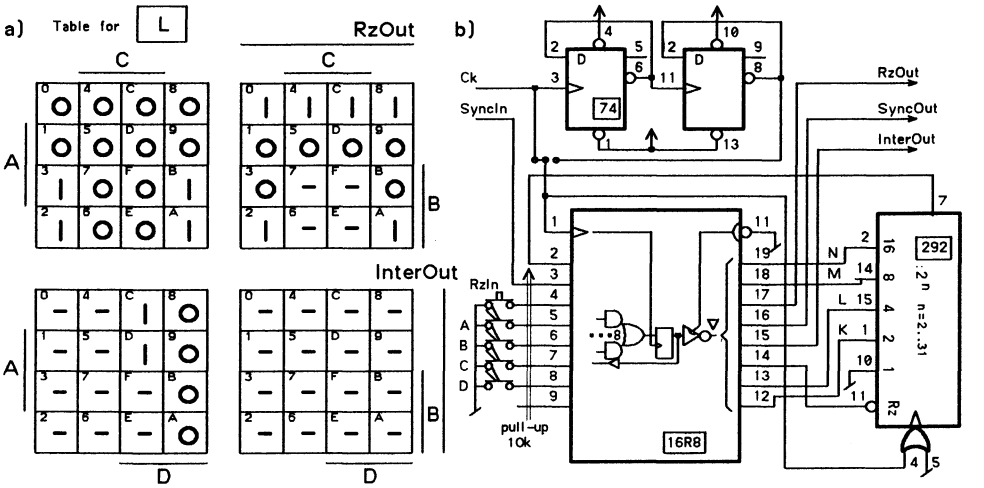


Fig. 8.21 Simplification example and diagram

This quite complex example shows how flexible and powerful PLDs are. The reader will have no trouble imagining improved variations since there is a wealth of possible solutions, especially if the problem specifications can be altered so as to benefit more from elegant technological solutions.

8.4 Double buffers

This example stresses the problems encountered when setting up specifications and describes asynchronous and synchronous design techniques.

8.4.1 Data and options

A simple buffer is a parallel register which stores available information at a given moment in time in order to enable the receiver to freely take this information at his own rhythm. Exchanges are synchronized by a flag (figure 8.22).

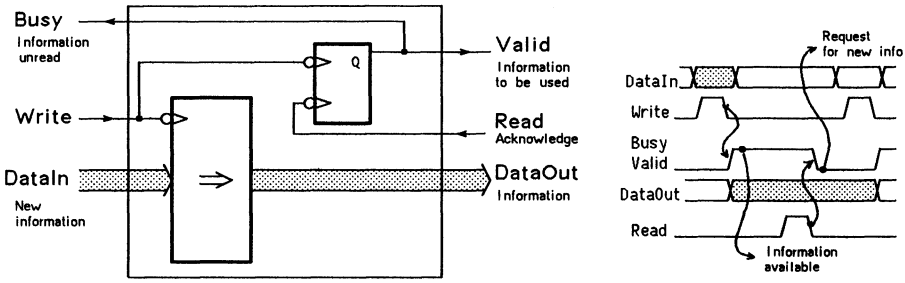


Fig. 8.22 Simple buffer

Double buffers generalize this notion through two registers. Silo registers generalize for n registers; the bigger n, the more irregular speed may be during transfer. Small buffers, such as the one studied here, are often found in serial and parallel programmable interfaces, as input/output transmission buffer.

The block diagram is shown in figure 8.23. The input Write signal is the clock which loads the input register (or the pulse which loads an input latch). This information is transferred into the output register and a signal tells the receiver located further along the line that there is some information to read.

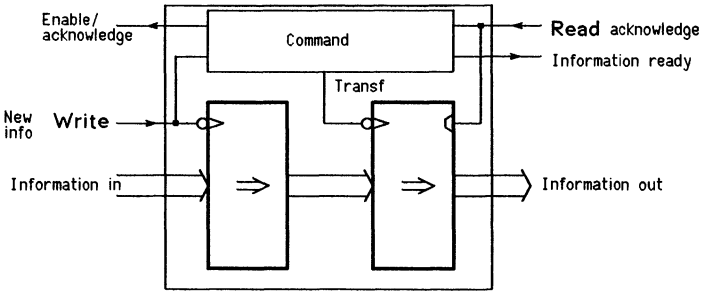


Fig. 8.23 Double buffer block diagram

If a new piece of information arrives at this moment, it can be loaded into the input register but it is necessary either to send an indication along the line saying that the buffer is full or to refrain from acknowledging.

This input interface must be well-defined, which means choosing between three philosophies used to synchronize input exchanges. The first (figure 8.24a) returns a Busy signal indicating that the interface is active and dealing with the information stored in the register. The Busy signal is activated as an answer to a load pulse and is deactivated once the content has been transferred into the output register. The second philosophy (figure 8.24b) seems very similar but the acknowledge pulse, known as AK, is only generated when the input register is free.

The third philosophy (figure 8.24c) does not send out a positive acknowledge but uses a BFull (bufferfull) signal to indicate that the buffer can no longer be filled.

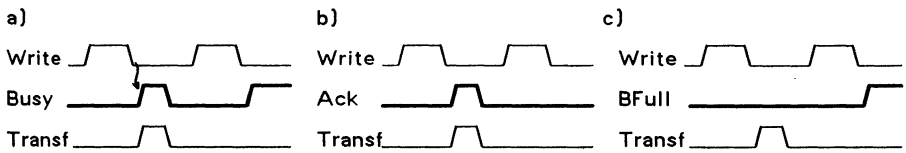


Fig. 8.24 Synchronizing input exchanges

There are two output solutions which are described in figure 8.25 through timing diagrams; they presuppose that two write operations have been performed before reading. In the first case (figure 8.25a), the Valid signal is activated as soon as a piece of information is loaded. Valid is deactivated by the Read signal, which can activate the three-state output of the output register. In the second case (figure 8.25b) an Empty signal is activated when the output register cannot be read.

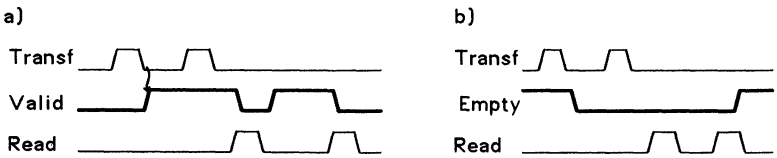


Fig. 8.25 Synchronizing output exchanges

8.4.2 Problem specifications

The solution which has been chosen uses Busy and Valid signals (figure 8.26a). In order to analyze it, the output values and switch-over conditions must be clearly described in a graph (figure 8.26b), which must take into account all possible states and change-overs. When information is

read as soon as it is received, the system goes through the left loop. When the receiver located further along the line is one transfer late, the system goes through the right loop. The graph shows all possible evolutions.

Timing diagrams such as those shown in the previous figures are not comprehensive. They are used to feel what is happening and to imagine all possible cases; once synthesis is over, they can be used to document typical cases. Figure 8.26c corresponds to the timing diagram given in figure 8.24a and figure 8.26d to that of figure 8.25a.

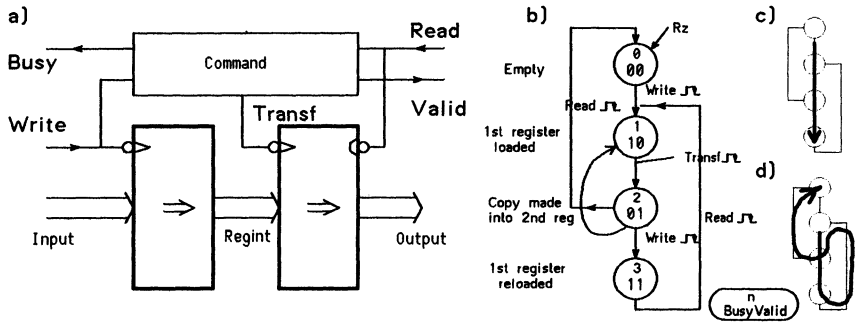


Fig. 8.26 Block diagram and graph for first solution

It is difficult to decide when the graph is complete. Simultaneity problems are particularly delicate. If Write and Read are simultaneously deactivated in state 2, (i.e. the internal logic cannot decide if the next state is state 3 or state 0), how can the designer be sure that the next state is state 1 and that a pulse or a switch-over has not been added during this short-cut? This means carrying out a comprehensive graph study, similar to that which is called for when dealing with metastable states. Once found, the solution will require detailed analysis.

8.4.3 First synchronous solution

Synchronous synthesis is both easier and more efficient in simple cases which only include two or three flip-flops, as is the case here due to the very simple graph. One difficulty is how to generate the internal transfer pulse called Transf. In an asynchronous solution, it is necessary to use a monostable or to generate an internal delay.

The functional diagram, which is a natural consequence of figure 8.26, uses two *double flip-flops* and a monostable (figure 8.27a). The Busy flip-flop is activated by Write and deactivated as soon as the information has been transferred into the second register. The Valid flip-flop is activated by this transfer and reset when the information is read. The Transf monostable gives a pulse if Busy is active while Valid is inactive.

Since double flip-flops are not available in TTL, it is better to generate a brief pulse when the monostable switches back over to zero (figure 8.27b). In this case, this IFin pulse lasts two flip-flop propagation times (Valid and Ifin); the system can only operate correctly if the brief pulse is sufficiently long to act on the Valid flip-flop.

One variation avoids including a monostable and uses a delay generated by an RC network. The diode discharges the capacitor rapidly and is only required if the Write signal can be reactivated very soon after Busy has been deactivated. This solution is not very elegant in TTL but may be interesting in VLSI since it is easy to adjust delays by changing transistor sizes or adding pairs of inverters.

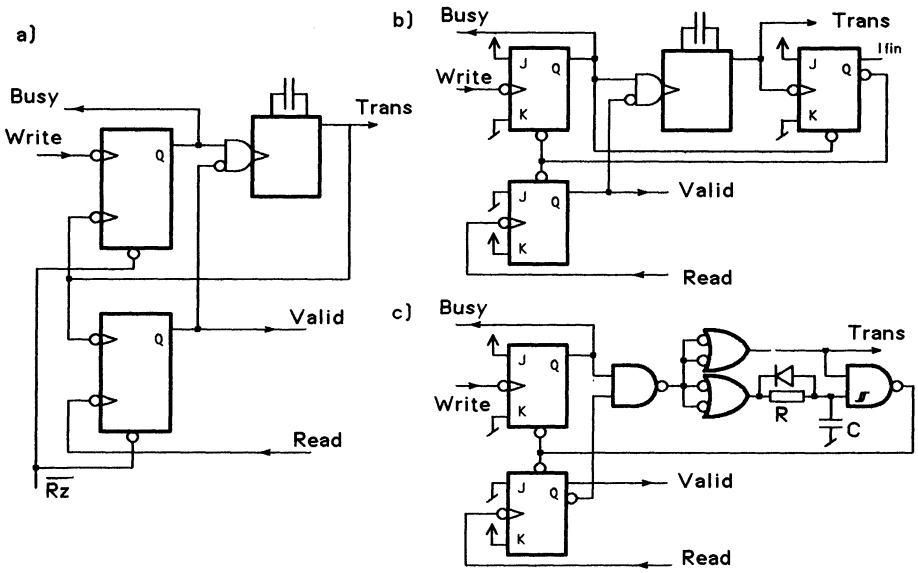


Fig. 8.27 Diagrams for the first asynchronous solution

We must now examine one specific case, i.e. the Write and Read simultaneity while Valid is active. The only tricky point is to be found at the monostable input: however, a time analysis comparing cases where the Read signal occurs slightly before or slightly after the Write signal shows that there is no real dangerous zone (figure 8.28).

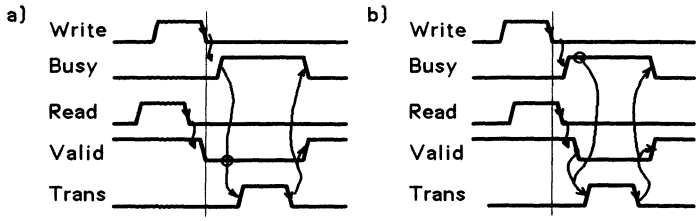


Fig. 8.28 Simultaneity analysis between Read and Write

8.4.4 Second asynchronous solution

The application graph must be drawn slightly differently when taking into account the number of valid words stored in the silo: 0, 1 or 2. A change of state occurs when there is a write or a read operation (figure 8.29a). In reality, a special operation must be performed when switching over to state 1 since the intermediate register must also be transferred.

This operation was delayed in the previous figure. In this case, it can be generated while the read or write pulse is active, which leads to the graph of figure 8.29b. This operation could be performed asynchronously using a brief pulse generated by the internal change of state but this brings us back to the aforementioned solution.

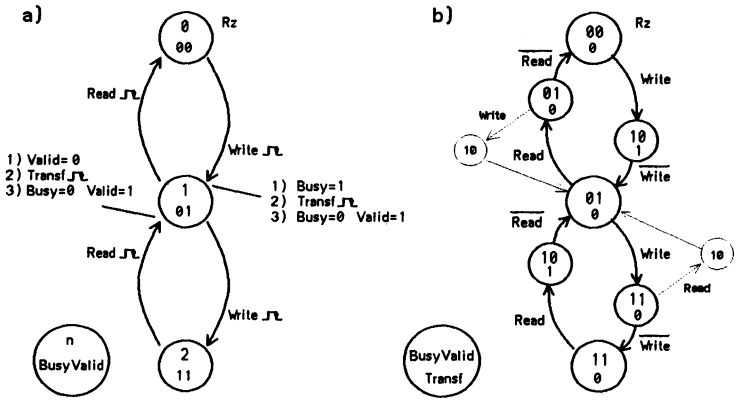


Fig. 8.29 Graph of the second solution

A timing diagram (figure 8.30a) is a direct consequence of this graph and naturally leads to a synthesis using double flip-flops, as long as the designer knows how to choose the right intermediate transfer sequence. The solution shown here was found by a trial and error method which simultaneously led to the diagram and to the timing diagram including correctly positioned arrows and delays. The corresponding diagram is shown in figure 8.30b and can easily be built using two TTL or HC-MOS circuits (figure 8.30c).

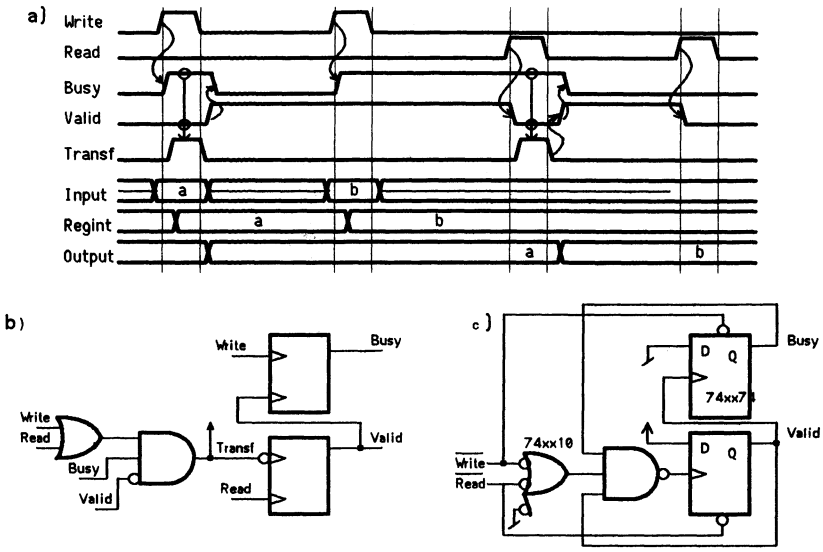


Fig. 8.30 Diagram of the second solution

One question still needs answering: what happens when a read pulse occurs at the same time as a write one in state 1? This case is shown using a fine line in figure 8.29b. However elegant this diagram may be, before using it the designer must carry out both a careful analysis of borderline cases while taking into account metastable states and an experimental check while minutely adjusting simultaneous signal phases.

8.4.5 Purely asynchronous solution

It is quite hard to fathom which synthesis method was used for the schematic of the *FIFO* 74HC40105 circuit included in the Philips/Signetics catalogue. Figure 8.31 gives its diagram after it has been drawn again more clearly and limited to two levels. With a little patience, it is easy to grasp how it works. For example, as soon as the WR pulse is activated, the flip-flops are activated and deactivated in a cascade, except for the penultimate one. P1 and P2 are successively activated, which copies the information right through the FIFO. At each read operation, as long as Valid is active, pulses P2 and P1 are generated in order to shift the content of the FIFO latches to the right. These two phase latches have been seen in figure 3.8f. The switch-over and the reciprocal inhibition sequence is very subtle and compatible with a totally asynchronous action between the input and the output. The delay occurs thanks to an even number of inverters and according to the pulse length required by the latches; it is absolutely necessary to carefully simulate this type of diagram which is very efficient because of the number of transistors used but intrinsically sensitive to element delays.

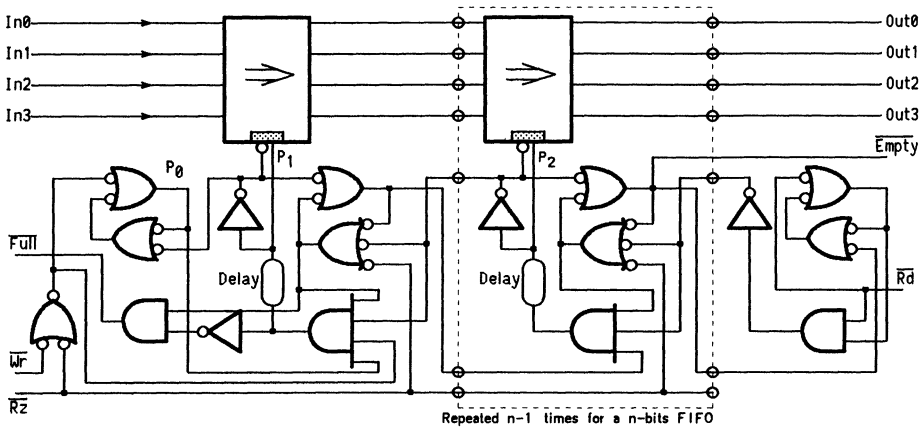


Fig. 8.31 Partial diagram of the 74HC40105 circuit

8.4.6 Synchronous synthesis

The evolution of a synchronous system is wholly determined by a single clock. This means that a monostable is not necessary since there is only one clock period to be counted. On the other hand, it is important to synchronize the inputs twice in order to avoid creating metastable states.

The diagram is fundamentally similar to that of figure 8.26 but some states must be doubled in order to detect switch-overs (figure 8.32a). Let us note that speed can be optimized by planning simultaneous signal evolutions like that which means cycling through state 8.

The corresponding block diagram is given in figure 8.32b. In order to simplify the problem, it is possible to set aside the passage through state 8. This means that 3 flip-flops are enough to encode the sequencer's next states. This requires a PAL with 10 outputs, at least 7 of which are flip-flops. It is possible to use 20G10 (Cypress) or 22V10 (Amd) types. A recommended variation is to encode Busy and Valid on state flip-flop outputs. This solution may enable the designer to use smaller and cheaper PALs of the 16R8, 16V8 or EP320 type.

The main disadvantage of a synchronous solution is the quantified delay of the resulting action. Busy and Valid are activated 3 clock pulses after the Write and Read signals which generate them. Moreover, the length of these two latter signals must be greater than one clock sample period in order to be sure that the signal is perceived.

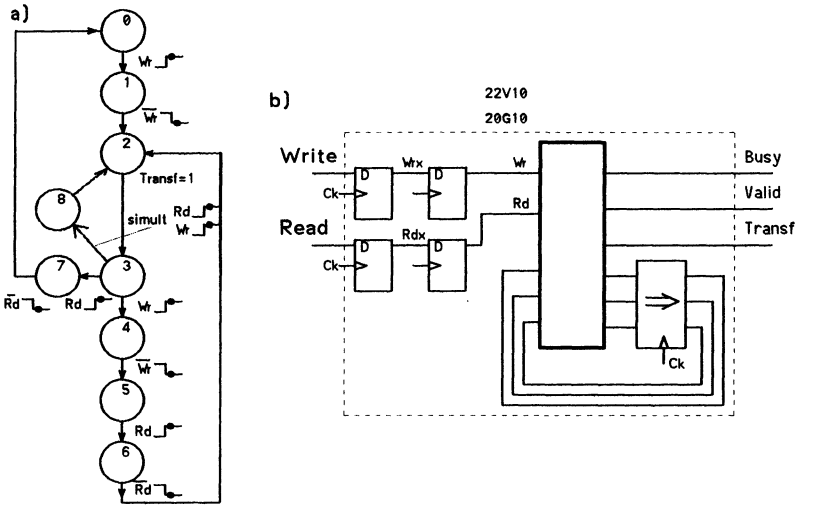


Fig. 8.32 Graph and block diagram of the first synchronous solution

A more elegant solution is based on the availability of the input signal as well as its value after a one pulse delay. An AND condition ($Wrx * \overline{Wr}$) is used to locate the switch-over. The graph is identical to that of figure 8.26 and the diagram can be seen in figure 8.33.

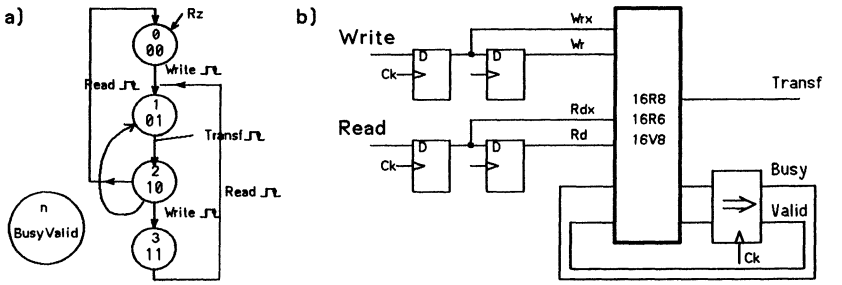
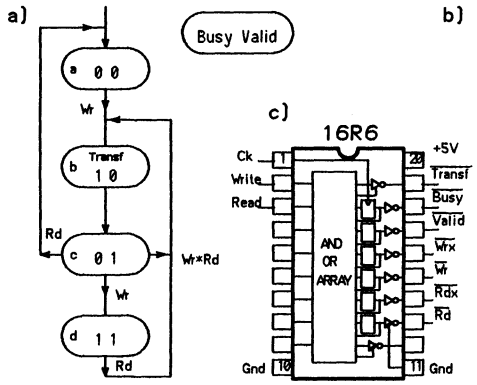


Fig. 8.33 Second synchronous solution

Synthesis is easy since there are only three output variables. Because evolution only depends on the $Wrx * \overline{Wr}$ and $Rdx * \overline{Rd}$ products, these are the terms used.

8.4.7 Detailed synthesis

According to the usual technique (section 3.3), synthesis is performed after having assigned a combination of states to each variable (figure 8.34a) and drawn up a table of present and next states (figure 8.34b).



	Present state R W	B V	Future state B+V+ T	
a	0 0	0 0	0 0 0	
c	0 0	0 1	0 1 0	
b	0 0	1 0	0 1 1	
d	0 0	1 1	1 1 0	
	0 1	0 0	0 0 0	error
c′	0 1	0 1	0 0 0	
	0 1	1 0	1 0 0	error
d′	0 1	1 1	1 0 0	
a′	1 0	0 0	1 0 0	
c″	1 0	0 1	1 1 0	
	1 0	1 0	0 1 1	error
	1 0	1 1	1 1 0	error
	1 1	0 0	0 0 0	error
c‴	1 1	0 1	1 0 0	
	1 1	1 0	1 0 0	error
	1 1	1 1	1 1 0	error

Fig. 8.34 Assignation and next state table

Some PAL synthesis programs can be fed this type of truth table after it has been rearranged in accordance with the program syntax.

```
.NAME   PAL16R6

.PIN                    .EQUATION

VCC     = 20            Wrx     :=      Write
GND     = 10            Wr      :=      Wrx
CK      =  1
/OE     = 11            Rdx     :=      Read
                        Rd      :=      Rdx

Write   =  2            Busy    :=      Busy*Valid + Busy*Rdx*/Rd + Valid*Wrx*/Wr + /Rdx*Wrx*/Wr + Rd*Wrx*/Wr
Read    =  3
                        Valid   :=      Valid*/Rdx + Valid*Rd + Busy*/Valid
/Transf = 19                          + Busy*/Wrx*/Rdx + Busy*/Wrx*Rd + Busy*Wr*/Rdx + Busy*Wr*Rd
/Busy   = 18
/Valid  = 17            Transf  =       /Valid*Busy*/Wrx*/Rdx + /Valid*Busy*/Wrx*Rd + /Valid*Busy*Wr*/Rdx
/Wrx    = 16                          + /Valid*Busy*Wr*Rd
/Wr     = 15
/Rdx    = 14            .END
/Rd     = 13
```

Fig. 8.35 PAL equations

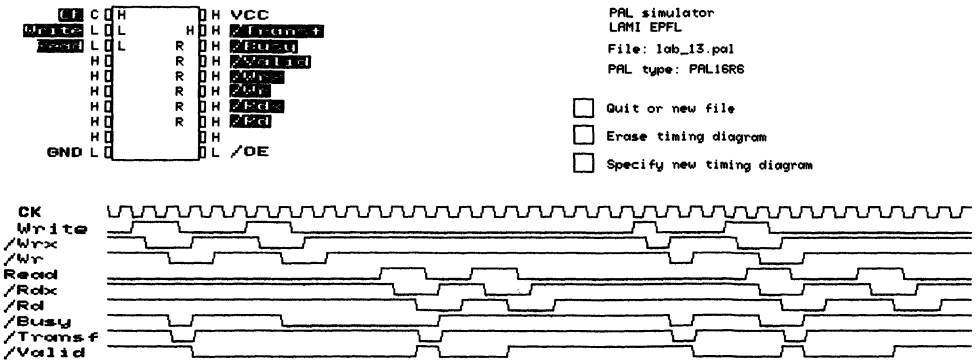


Fig. 8.36 Operating simulation (screen copy)

The designer must now simplify patiently and draw up the Karnaugh tables. The listing of figure 8.35 is obtained by substituting W=Wrx*/Wr and R=Rdx*/Rd. A simulation such as that of figure 8.36 means that correct behaviour can be checked before the fuses are blown.

Inverted inputs must be declared in the pin assignation. It is more difficult to invert the outputs since the 16R8 requires synthesizing the inverted equations. Inverted outputs can be programmed when using a 16V8.

8.4.8 Quasi synchronous solution

Let us mention one last solution which does not entail the aforementioned constraint about the Write and Read pulses, i.e. that they must be greater than one clock period in the case of a synchronous system. The basic principle of the diagram of figure 8.37 is that it memorizes the action which is occurring in a flip-flop, that it synchronizes this request, which is then suppressed by an asynchronous reset as soon as it has been recognized by the sequencer. This solution is sometimes more difficult with a PLD, for only few PLDs enable the flip-flop clock to be controlled individually.

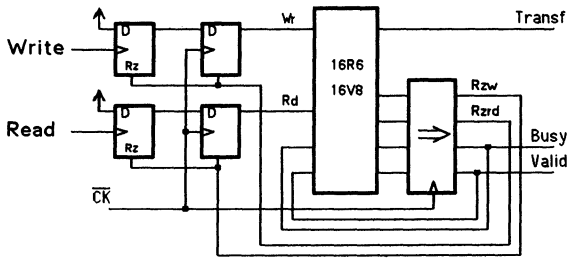


Fig. 8.37 Semi-synchronous solution

These few synthesis examples show that there are a large number of methods and solutions to be mastered by an experienced designer in order to come up with an optimum solution. Design generally proceeds faster and more easily using synchronous solutions but the most important thing is to analyze the situation carefully and to take all special cases into account.

8.5 Multiplexing in a microprocessor interface

There is often a lack of inputs or outputs on a microcontroller. One solution is to find a processor which comes in a larger package; multiplexing inputs and demultiplexing outputs usually works out cheaper because TTL and HC-MOS circuits are cheap and available in miniature packages. This section describes various solutions and examines in a little more detail the solution of a real interface in which 32 diodes, a motor, an optic sensor and a 64 key keyboard are controlled by a set of 17 bits taken on the input-output ports of the 68705F25 microcontroller.

8.5.1 Registers and latches

The distinction between parallel and serial registers, latches and addressable latches is an important one (section 3.5.1). Parallel registers are often referred to as PIPOs (parallel in parallel out). *Dynamic parallel registers* (figure 8.38a) are the most common: information arriving at the inputs is stored on the rising clock edge.

Such registers are not the same as *latches* or *transparent registers* (figure 8.38b) which copy information when the load input is active and jam the last piece of information on the inputs when the load input is deactivated. A latch can possibly be replaced by a register and vice-versa but the clock must be equipped with an inverter since the active edges are different.

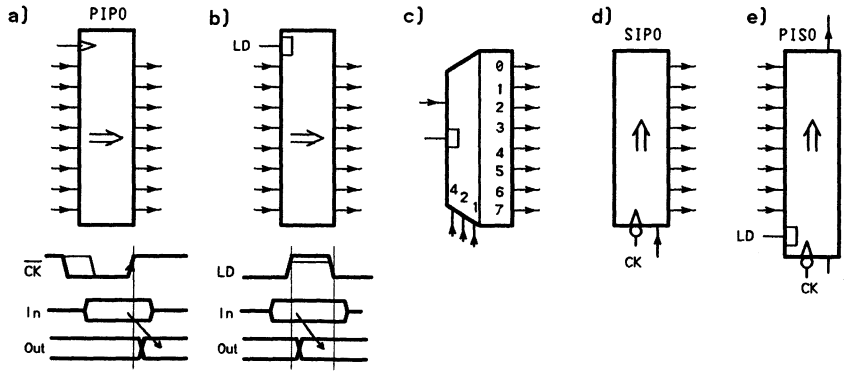


Fig. 8.38 Parallel register and latch, addressable latch and serial register

Addressable latches include both a demultiplexer and a register whose latches are fed independent load pulses controlled by the demultiplexer (figure 8.38c).

They share the D inputs and information is stored in the flip-flop which is addressed by the demultiplexer/decoder. Addressable latches are an advantage when only one output line needs modifying at a time.

The first category of serial registers is equipped with parallel inputs (PISO = parallel in, serial out), with a static load input and a clock input for shifting the content of the register (figure 8.38d). The second category is equipped with parallel outputs (SIPO = serial in, parallel out) with a serial input and a clock input (figure 8.38e). Other variations are of less interest here.

8.5.2 Multiplexers and demultiplexers

Let us go back over a few elements of section 2.8. A *multiplexer* transmits one of 2^n inputs according to the combination of the n selection inputs (figure 8.39a). A *demultiplexer* copies its input onto one of the 2^n outputs which are selected by n inputs (figure 8.39b). As a rule, a *decoder* always has one active output; practically, decoders can control output activation ("Enable"), which makes the difference between demultiplexers and decoders imperceptible.

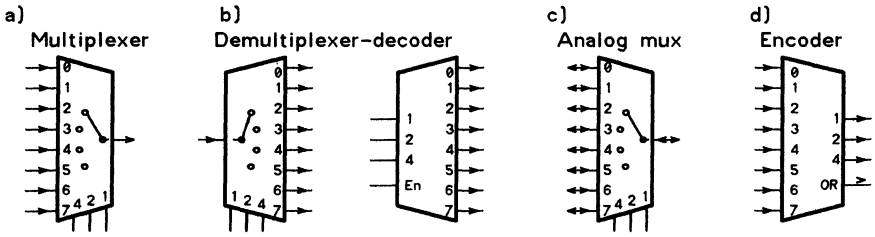


Fig. 8.39 Multiplexer, demultiplexer and decoder

Some analog multiplexers are available (figure 8.39c); they are also demultiplexers since they are perfectly bidirectional. Their only disadvantage is that they do not regenerate the signal but they can be used to create applications which would otherwise be impossible. *Encoders* give the number (encoded in binary) of the active input and, when faced with a simultaneity, give priority to the highest number (figure 8.39d).

As is the case for all integrated circuits, inputs and outputs of the aforementioned circuits are sometimes inverted; the corresponding variable is thus active at zero. The outputs are often three-state ones and include a control variable to activate them; when this variable is not active, the outputs it controls are floating ones (figure 8.40a). *Three-state outputs* are also available as outputs for parallel or serial registers (figure 8.40b).

Open collector outputs are only active at state zero. A pull-up resistor is required on each line to ensure state 1 and limit charge speed (8.40c). An activation input is sometimes associated with open collector outputs; when this input is inactive, none of the outputs it controls can be in state zero.

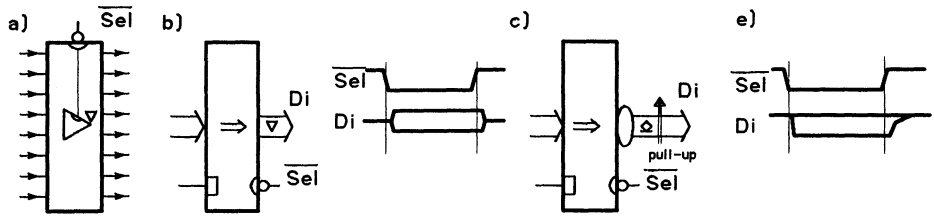


Fig. 8.40 Three-state and open collector outputs

8.5.3 Demultiplexing outputs

There are many ways of increasing the number of outputs. In order to choose the right interface, it is important to know whether the combinations on the outputs are unimportant or whether only one output is active at a time. The designer also needs to know whether the outputs can stand glitches (brief pulses) while the state is switching over, what degree of simultaneity is required when several outputs are modified at the same time (millisecond, microsecond or nanosecond). The maximum switch-over frequency must be defined. According to these criteria, it is easy to find the cheapest solution once the basic solutions are known.

8.5.4 Demultiplexing using a shift register

A shift register is equipped with a data input whose value is stored and shifted at each clock edge. It is possible to control as many signals as desired thanks to only two microcontroller outputs (figure 8.41a).

During the register shift, the outputs go into unwanted transitory states which are not problematic when dealing e.g. with a light emitting display. They can be suppressed using a parallel register which is loaded once the serial register is completely ready (figure 8.41b). This costs an extra control line which can even be suppressed by using a retriggerable monostable since the pulse window end will operate the shift, at the expense of an additional component.

The only disadvantage of this serial solution is that it is slow. Transfers can be stepped up by filling several registers in parallel (figure 8.41c).

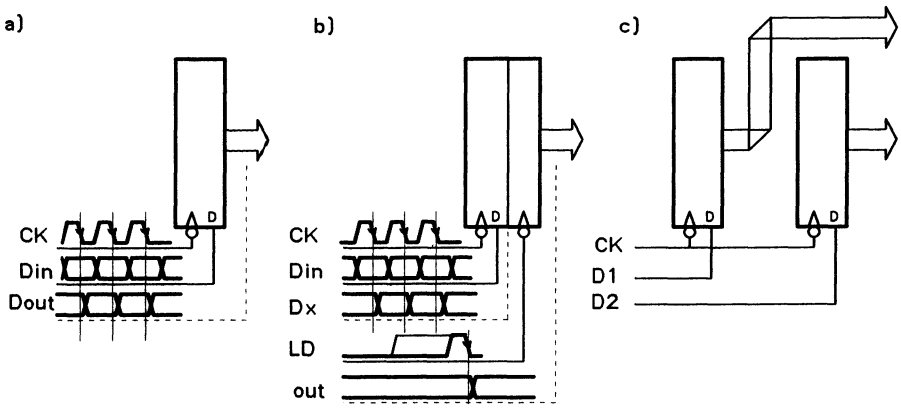


Fig. 8.41 Serializing information

8.5.5 Demultiplexing using a decoder or a PAL

If only one output is active at a time, a decoder offers a simple solution (figure 8.42a). Using a decoder equipped with exclusive-OR outputs (figure 8.42b) is a way of choosing polarity; moreover, when dealing with a display control, this means that all the lamps can be tested by deactivating (E=0) and inverting (INV=1) if the lamps are lit by a high state on the decoder outputs.

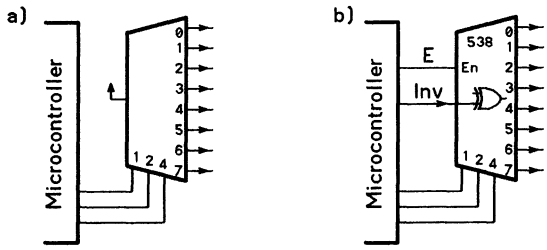


Fig. 8.42 Decoding

The decoder catalogue is quite rich. Some decoders are equipped with an input register on the selection lines, which can save even more lines.

When there are only few possible sets of values for the outputs, they can be encoded using a PAL. This is what happens when demultiplexing a 7 segment display for which special circuits are available. The disadvantage of PALs in this case is that they are equipped with less outputs than inputs and that their power consumption is high.

8.5.6 Demultiplexing using an addressable latch

A variation on the previous solution latches the written value. It is possible to control 8 outputs, as long as there is only one switch-over at a time, by using 3 selection inputs, one data input (value to memorize) and one latch pulse, that is only 5 signals (figure 8.43a).

In order to control two *addressable latches*, it is possible to share the selection addresses either with the data or with the latch pulses (figure 8.43b); this second solution modifies two outputs at a time.

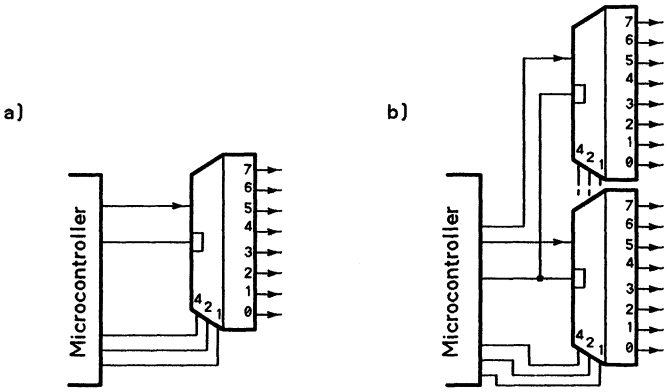


Fig. 8.43 Demultiplexing using an addressable latch

8.5.7 Demultiplexing using a parallel register

The most frequently used solution in microprocessor interfaces is to load registers or latches which share common inputs (bus). Selecting a register activates the corresponding outputs. With n 8-bit registers it is possible to control n×8 bits using n+8 lines (figure 8.44a). If all the outputs are to switch-over at the same time, an extra register layer is needed and the transfer must end by loading the upper register (figure 8.44b).

Since only one register is selected at a time, a decoder can demultiplex the selections in order to save lines; in such a case, the designer must beware of switch-over glitches and use an activation line in addition to the addressing lines (figure 8.44c). A decoder equipped with a register on the address inputs can reduce the number of lines to the bare minimum.

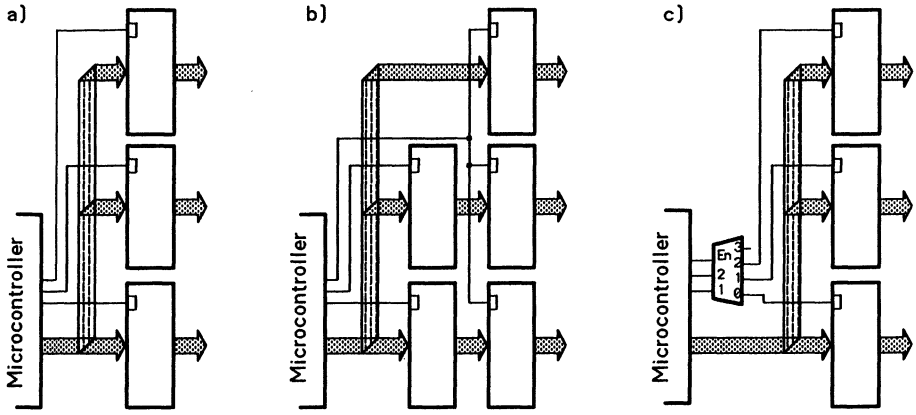


Fig. 8.44 Demultiplexing using a register

There is a rather astute solution which is very suitable for controlling a network of light emitting diodes and only uses 9 lines to control up to 64 diodes. The basic idea is to use a selection latch instead of the decoder shown in figure 8.44c. Moreover, the load input of the S selection latch must be connected to the inverted activation input of the three-state outputs (figure 8.45a).

Pull-down resistors guarantee state 0 when SEL=1, i.e. when the three-state outputs are floating. These outputs control latches 0_0 to 0_7, which are all connected to the same data bus (figure 8.45a). The outputs of these latches are also three-state ones and are activated by the FLASH variable in order to light the diodes.

The standard timing diagram can be seen in figure 8.45b. The SEL signal is activated (1) in order to store in latch S the address of latch 0_i which has not yet been loaded (2). The binary word on the bus only includes a single 1. During this transfer, the S_i outputs are deactivated. When SEL is deactivated (3), the output shows the value that has been loaded and which is temporarily loaded into the selected O_i register.

The microprocessor places the new value on the bus (4) so that the value can be copied into the still selected O_i latch. Reactivating SEL (1) deactivates the three-state S output so that S_i is deactivated and O_i latches the value onto the bus. At the same time, the bus value changes in S, although this has no effect since these outputs have already been deactivated.

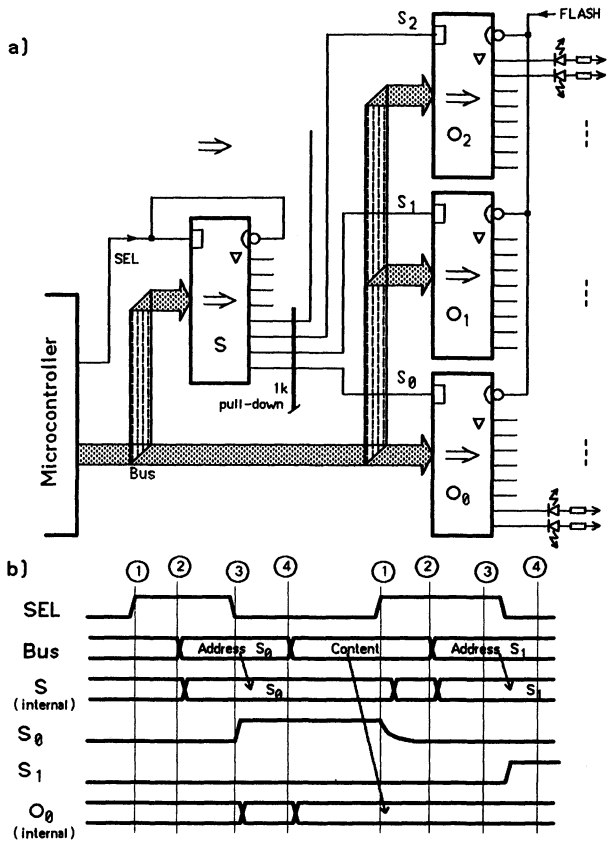


Fig. 8.45 Demultiplexing in the case of 24 light emitting diodes

The address of the next O_i register is ready on the bus, SEL is deactivated, etc. until all the O_i registers are loaded. A FLASH pulse can then be sent in order to light all the light emitting diodes at once. In the case of our application, the load loop requires about 20 instructions which means that the 6805 needs approximately 30μs to load the registers.

8.5.8 Multiplexing the inputs

In order to multiplex inputs, the microcontroller must have free outputs. The designer needs to know whether the inputs are going to be scanned and be acquainted with the temporal and electronic characteristics of the signals he wants to read. The last two aspects will only be discussed briefly here. Temporal signal hesitations (contact bounce) are filtered either analogically or using software. The input signal levels must be compatible with the TTL or CMOS logic used.

8.5.9 Multiplexing using switches

A multiplexer circuit controlled by n outputs can be used to read one of 2^n inputs on a single line (figure 8.46a). When using two smaller multiplexer circuits, it is possible to swap an output for an input so that two inputs can be read at the same time (figure 8.46b). In cases where it is important to read the value of all the inputs at exactly the same moment, an extra register is used to "photograph" the inputs and keep them constant during scanning (figure 8.46c). A register or a latch can be used indifferently if the control signal wave is satisfactory.

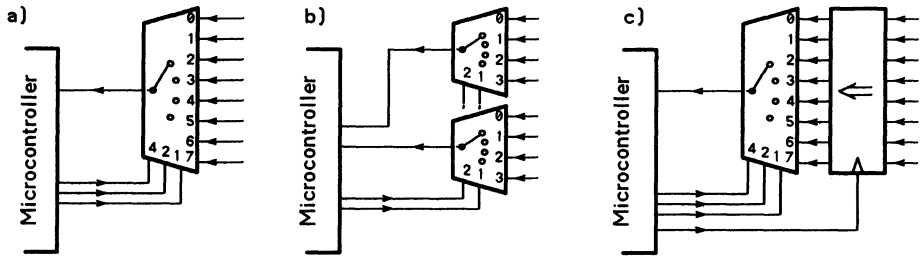


Fig. 8.46 Multiplexing 8 input signals

8.5.10 Multiplexing using registers

A PISO register is often used to read inputs which change slowly and to transfer them in series towards the microcontroller (figure 8.47a). Two microcontroller outputs and one input mean that it is possible to read as many bits as desired. If the register is loaded dynamically, a clock pulse is required while the "Load" input is active (figure 8.47b). Exceptionally it may be useful to use a register equipped with an input register; this means that a new value can be loaded while the previous information is still being shifted (figure 8.47c).

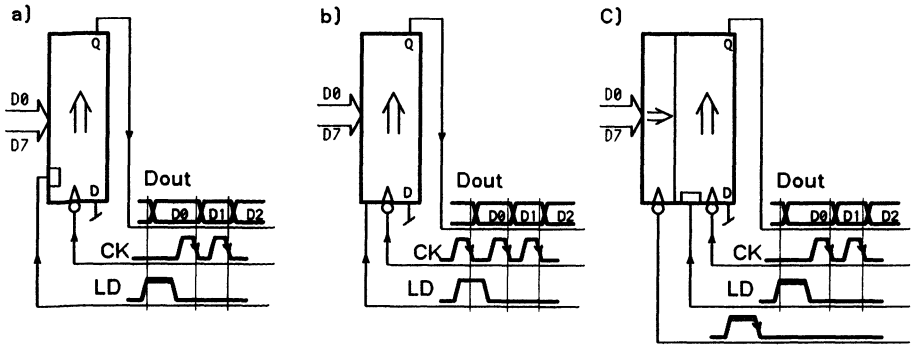


Fig. 8.47 Multiplexing using shift registers

8.5.11 Multiplexing using a three-state driver

Several three-state drivers can be linked by a bus to the system's inputs (figure 8.48a). Only one driver can be activated at a given moment. As is the case in microprocessor buses, a decoder is usually used as soon as there are more than two drivers (figure 8.48b). If the bus is only an input one, this decoder can remain permanently active as long as the microcontroller port is always an input.

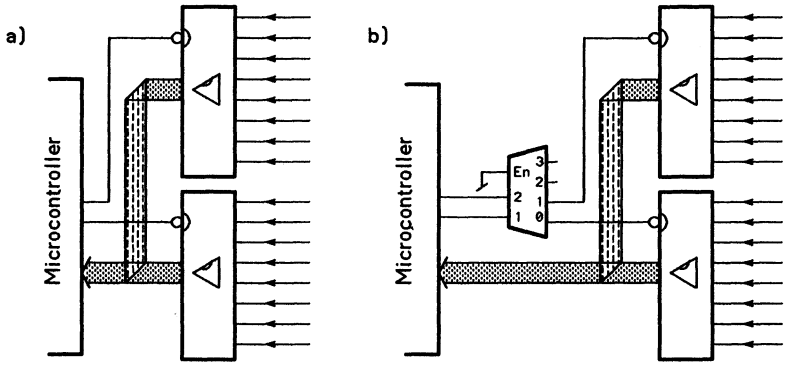


Fig. 8.48 Bus multiplexing

8.5.12 Multiplexing using a matrix

When the input signal is a mechanical or analog switch, a matrix structure can be used. Therefore, when dealing with 16 keys, there must be 4 outputs to select columns and 4 inputs for the rows (figure 8.49a). Unless otherwise specified, pull-down resistors set the inputs at zero. If a switch is closed while the column it is in is activated, the corresponding input is activated. Four selection cycles which go through the columns one after the other can give the state of all the keyboard keys. The outputs must be open emitter or source (open collector or drain if the outputs are inverted) in order to avoid creating a short circuit between two decoder outputs if two switches are closed in the same line.

Since only one column can be activated at a time, a decoder offers certain advantages. If this decoder is equipped with inverted outputs (figure 8.49b), there must be pull-up resistors and a zero will appear at the input when the corresponding switch is closed. If the decoder does not have any open collector outputs, the outputs must be equipped with a serial diode in order to avoid a short-circuit between the two corresponding decoder outputs if two keys are depressed in the same column.

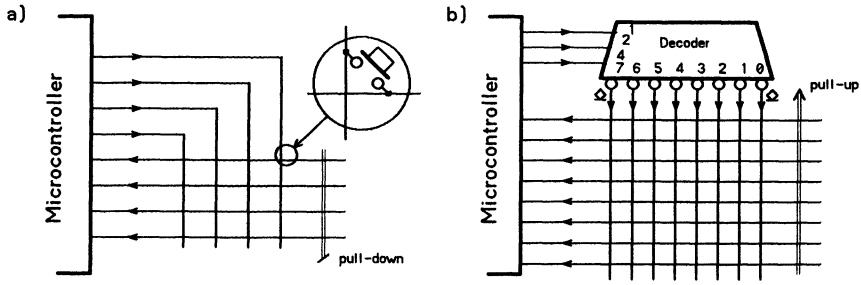


Fig. 8.49 Scanned keyboard

Inserting contacts in a matrix layout can create *phantom keys*, as seen in section 6.1.5 (figure 6.7). The only solution is to add a serial diode to each contact, which gives normal outputs but is more expensive and often incompatible with keyboard technology.

8.5.13 Multiplexing input-outputs

Multiplexing several inputs and outputs to save lines does not call for new techniques. Depending on the availability of the microprocessor, whose free lines are usually a combination of simple inputs, of simple outputs and of bidirectional programmable input-outputs, the designer must use various combinations of solutions the basic principles of which have already been discussed. This section will only give a typical example.

8.5.14 Scanned keyboards

The bare minimum required to efficiently scan a 64 key keyboard is one output line and 8 bidirectional lines. The decoder of figure 8.48b is replaced by a register. This solution (figure 8.50a) means that output registers can be added if necessary as seen in figures 8.44 and 8.45.

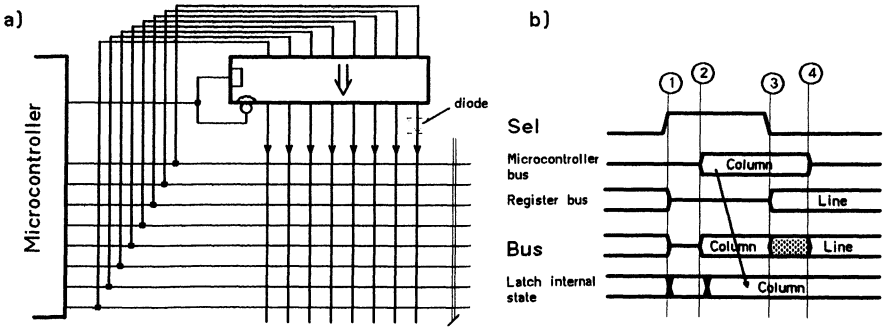


Fig. 8.50 Minimum solution for scanning a keyboard

Instead of a register equipped with an open collector output, which would be ideal, it is possible to use one of the many three-state latches available in all technologies and add diodes if there is a risk of depressing two keys simultaneously in the same line.

Detailed functioning is described in figure 8.50b. Activating SEL (1) deactivates the three-state output and it becomes possible to place on the port, which is programmed to write, the column selection code (2) which is copied into the register. During deselection (3), the latch output is active and a short-circuit occurs because the bus outputs are still active. Within a few microseconds, the microprocessor can set the port in an input mode (4) and the short-circuit will cease to be destructive or dangerous for LS or HC circuits. Resistors can be added in order to limit the short-circuit current.

Obviously this solution is not appropriate if power consumption has to be very low. The diagram is perfect if the key matrix is no larger than 7 by 7 (49 keys). The port's eighth line is used to control SEL with a pull-down resistor. Internally, SEL remains active and selection/deselection is carried out by making the output port into an input; the latch switches-over simultaneously and a short-circuit is possible only during the fractions of a nanosecond switch-over lasts.

8.5.15 Conclusion

The wealth of the integrated circuit catalogue and the possibilities offered by three-state and open collector outputs open up a large number of avenues for designers. When it is necessary to save a line on a microprocessor interface, there is always an elegant solution requiring patience and imagination. In the case of a real project, a microcontroller with a 28 pin package was used thanks to a combination of the solutions shown in figures 8.45 and 8.50. The diagram is simpler and faster than a solution based on shift registers; moreover, it only uses one type of circuit (HC373).

References

[Annaratone86] M. Annaratone, *Digital CMOS Circuit Design*, 1986, Kluwer Academic Publishers

[Blood82] W.R. Blood et al., *MECL System Design Handbook*, 1982, Motorola Semiconductor

[Bolton90] M. Bolton, *Digital Systems Design with Programmable Logic*, 1990, Addison-Wesley

[Buchanan90] J. Buchanan, *CMOS/TTL Digital Systems Design*, 1990, McGraw-Hill

[DelCorso86] D. DelCorso et al, *Microcomputer Buses and Links*, 1986, Academic Press

[Fletcher80] W.I. Fletcher, *An Engineering Approach to Digital Design*, 1980, Prentice-Hall

[Fujiwara85] H. Fujuwara, *Logic Testing and Design for Testability*, 1985, MIT Press

[HP80] *A Designer's Guide to Signature Analysis*, Application note 222, 1980, Hewlett Packard

[Hunsperger85] R.G. Hunsperger, *Integrated Optics: Theory and Technology*, 1985, Springer-Verlag

[IcMaster91] *IC Master*, Yearly, Hearst Business Communications, Garden City NY

[IEEE87] *IEEE Standard VHDL Language Reference Manual*, IEEE Std 1076-1987

[Jung81] G.J. Jung, *IC Op-amp Cookbook*, 1981, Howard Sams

[Lewin85] D. Lewin, *Design of Logic Systems*, 1985, Van Nostrand Reinhold

[Marino81] L.R. Marino, *General Theory of Metastable Operations*, IEEE Transactions on Computers, C30-2, February 1981, pp.107-115

[McCluskey86] E.J. McCluskey, *Logic Design Principles: with Emphasis on Testable Semicustom Circuits*, 1986, Prentice-Hall

[Miczo86] A. Miczo, *Digital Logic Testing and Simulation*, 1986, Harper and Row

[Mounder90] C.M. Mounder, R.E. Tulloss, *The Test Access Port and Boundary-Scan Architecture*, 1990, IEEE CS press No 2070

[Muller89] J.M. Muller, *Arithmétique des ordinateurs*, 1989, Masson

[Nicoud88] J.D. Nicoud, *Video-Rams: structure and applications*, IEEE Micro, February 1988, pp 8..27

[Nicoud91] J.D. Nicoud, *Dedicated tools for Microprocessor Education*, IEEE Micro, February 1991.

[NS86] *Linear Application Handbook*, 1986, National Semiconductor

[Peterson72] W. Peterson, E.J. Weldon, *Error Correcting Codes*, 1972, MIT Press

[Piguet90] C. Piguet et al., *Conception des circuits ASIC numériques CMOS*, 1ère partie, 1990, Dunod

[Schwartzlander90] E.E. Schwartzlander, *Computer Arithmetic I, II*, 1990, IEEE CS press Nos 1931, 1945

[Shooman68] M.L. Schooman, *Probabilistic Reliability: an Engineering Approach*, 1968, McGraw Hill

[Stevens86] A.K. Stevens, *Introduction to Component Testing*, 1986, Addison-Wesley

[Texas88a] *Bus Interface Circuits, Application and Data Book*, 1988, Texas Instruments

[Texas88b] *High Speed CMOS, Data Book*, 1988, Texas Instruments

[Tompkins88] W.J.Tompkins, *Interfacing sensors to the IBM®PC*, 1988, Prentice Hall

[VanDeGoor91] A.J. Van de Goor, *Testing Semiconductor Memories, Theory and Practice*, Wiley, 1991

[Wakerly90] J.F. Wakerly, *Digital Design, Principle and Practice*, Prentice-Hall, 1990

[Weste85] N. Weste, K. Eshraghian, *Principles of CMOS VLSI Design*, 1985, Addison-Wesley

[Yoeli90] M. Yoeli, *Formal Verification of Hardware Designs*, 1990, IEEE CS press No 2017

Appendix

Integrated circuits' catalogue

This integrated circuit catalogue provides a unique view of the standard digital circuits used for developing microprocessor interfaces. It exhaustively covers bipolar and complementary MOS circuits parts of "74" and "400" families, as well as the commonly used PLDs. It also mention also several other circuits, but does neither try to cover all the special circuits used for sensors, actuators and displays, nor for designing architectures based on special memories or fast arithmetic circuits, nor for transferring data over long distances. A catalogue three times as thick would be required and the principles described in this book are not sufficient for the implementation of many of these more specialized circuits.

In order to help the reader willing to design his projects quickly and successfully, the table of page 222 allows each category of product to be cross-referenced with the corresponding manufacturers. Addresses of the preferred sales representatives will be found by consulting the exhibitors list of a national exhibition devoted to electronic components. Detailed specifications and application notes will be provided by these vendors on request. Three thick books [IcMaster91] are published each year and include many cross-reference lists. They are useful for finding the manufacturer(s) of a circuit where the number is known.

The value of this catalogue is to present selected circuits with clear symbols, using a logical order which covers all technologies and manufacturers. Searching inside manufacturers' documentation is still required for getting the detailed specifications which are necessary for a reliable design. But it is difficult to find a circuit with a given functionality when looking at catalogues where circuits are always classified by numeric order. When an apparently adequate circuit is found, one additionnally needs to decode a specific symbolism and to decode complex truth tables in order to understand the detailed operations of the circuit.

This catalogue provides at a glance the package size and the multiplicity of versions. When deMorgan's theorem allows for two equivalent symbols, they are generally shown. Another important aspect is the component's vitality. When developing a system one must not use components which are

too old, or are slow, power-hungry and about to be abandoned by manufacturers. One should also be careful with attractive-sounding but recently announced circuits, available at a single manufacturer and still expensive. Special signs mark each circuit in order to express this important choice criterion:

† Old circuit with no more guaranteed availability
⌀ Old circuit still easily available, but preferably to be avoided
○ Getting-old circuit
● Old circuit full of vitality
■ Young circuit easily available with an apparently guarantied future
□ Recent circuit, apparently available, but risky due to a single source and/or a new technology
ф Recently announced circuit with the likelihood that it may soon disappear.

For each circuit, the catalogue specifies in which family it is available. Availability of SMD packages is given without specifying the family: this field is quickly expanding and one should carefully check with the selected manufacturer not only the existence of the circuit, but also the precise package size.

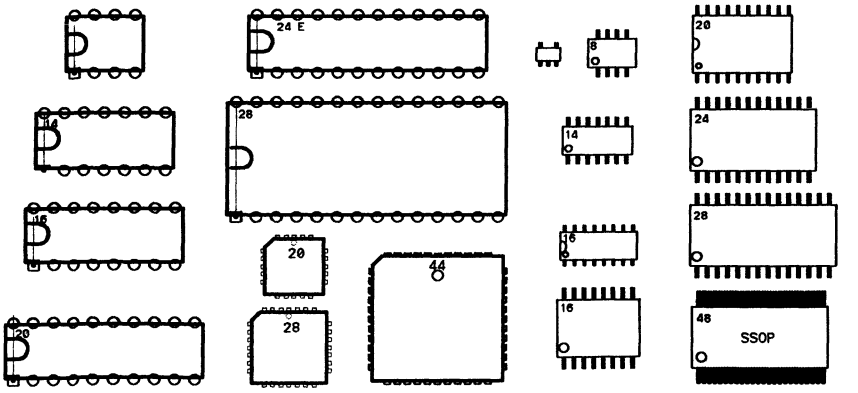
An aspect which is not part of the catalogue and which will confuse the beginner is the exact circuit number, which sometimes slightly differs from one manufacturer to another. 74' circuits are usually easy to recognize, but Texas Instruments has adopted the 29C AMD's family, referring e.g. to the 29x821 as a 74x29821 or 74x821, according to the family (x refers to ALS, AS, AC, etc.). In order to reduce the confusion in this case, the two numbers are referred to here. Each manufacturer frequently prefixes the number by own set of letters. For instance, the "4000" numbers of the CMOS family are preceded by a 14 (Motorola), a CD (RCA), a TC (Toshiba) or a 74HC (Texas Instruments). Suffix letters specify the package type and the temperature range. On IC packages, a second number is sometimes confusing: it is the manufacturing date (year and week number) or a serial number.

Circuits mentioned are almost all found in the catalogues one can get from Texas Instruments, AMD, NS/Fairchild, Philips/Signetics or Toshiba. When a manufacturer's name is mentioned for a circuit, it is usually the only one to offer this circuit. A given user usually buys 95% of his circuits from the same supplier and quickly gets familiar with its numbering system. For other circuits, speaking to the representative helps to locate the exact part number, and get the full set of letters for the selected circuit, together with its price and availability.

Integrated circuits of the 74AC11xxx family, having power lines at package centre, are not documented here, since their generalization within industry is uncertain. Circuits functionnality is the same for the same reference number; only the pinout is changed.

Appendix content

Integrated circuit manufacturers 222
Survey of
  gates 224
  three-state drivers 225
  comparators 226
  registers and latches 226
  read only memories 227
  static memories 228
  FIFO memories 228
  dynamic memories 229
Drawing conventions 230
Inverters 231
Non-inverters 232
Miscellaneous 234
NAND gates 235
AND gates 237
NOR gates 238
OR gates 239
XOR gates 240
XNOR gates 240
AND-OR gates 241
Simples three-state drivers 242
Open-collector drivers 244
Parity drivers 245
JTAG drivers and registers 245
Bidir. three-state drivers 246
Bidir. open-collector drivers 247
Bidir. drivers with parity 248
GPIB/IEEE488 drivers 249
Tridirectionals drivers 249
Multiplexers 250
Parity generators 251
Analog multiplexers 252
Priority encoders 252
Decoders 253
7-segment decoders 255
Comparators 256
Address comparators 257
Arithmetic units 257
Carry look-ahead 258
RS flip-flops 259
D-flip-flops 259
JK-flip-flops 259
Parallel registers 261
Double registers 264
Read-back registers 264
Bidir. drivers and registers 264
Universal serial-par. registers 266
Latches 265
Read-back latches 266
Bidirectional latches 267
Registers with interrupt logic 267
Registers/latches with multiplexer 267
Addressable latches 268
Serial registers 269
Serial-only registers 271
Special registers 271
Asynchronous conters 272
Rate multipliers 273
Synchronous counters 274
One-shots 277
Delay elements 277
PLDs 279
Index 285



Exact size of several IC packages

Integrated circuits' manufacturers

Manufacturers-product chart

● Major manufacturer
○ Several interesting products

	74LS,S	74ALS,AS	74F	4000 (C)	74HC,HCT	74AC,ACT	74AC11	74BCT Bus interf	ECL 10k,100k	GaAs logic	PLD,PAL,GAL	LCA,PGA,ERA	PROM	EPROM	E²PROM, Flash	Novram	SRAM	DRAM	VRAM	FIFO	Dual port	Cache, associative memory	Mem.driv.,ECC	Video int.,DAC	Real time clock	Progr.interface	Fast arith., signal processing	Comm.circuits	Line driver	Display drivers	CCD reg, camera	Opto electr	Transistors, diodes	Motion contr.	Regul. DC/DC	References	Op-amp,comp.	Filters	A/D, D/A	Delay lines
Actel											●																													
Altera											●																													
Amd/MMI											●	○	●	●	○		●			●	●	●	●	●		●	●	●	●											
AMS																												○												
Analog Devices																								●			●		●					○	○	●	●	●	●		
Atmel											○	○		○	●																							○		
AT&T												○					○										○	○	○											
Bel Fuse Inc																																								●
Brooktree																								●														●		
Burr Brown																																		●	○	●	○	●		
Chips and Technol																											○													
Cypress											●		○	○			○			○	●	●					●													
Dallas																●				●					●															●
Datel																																		●			●	●		
EM Microel.Marin																									●					○										
Elantec																								○												○				
Exar																												○	○											
Exel																									○															
Fujitsu													●	●	●		●	●	●		○						○	○												
Gazelle											○						○																							
General Instr Power																																○								
GE/RCA				●	●	●											○			○				○	●	○	○	○	○	●		○	○		○	○	○		○	
Gigabit Logic										●																														
GTE																										○		○												
GoldStar	○			○	○												○	○	○									○		○	○		○		○	○	○	○	○	
Harris/Intersil																										○		○												
Hewlett Packard																																●		●						
Hitachi														●	●		●	●	●		●	●		●	●							●		●			●			
Honeywell																																								
Hughes Micro															○	○												○		●										
Hyundai															○		○	○																						
IDT																	●			●	●						●													
Inmos: see RCA																																								
Intel											○		●	●	●	●	○	○			○	○	○		○	●	○	○												
Int.Rectifier																																	●	●						
Lattice											●																													
Level One																												○												
LoralFairchild																															○									
LSI Logic																											○													
LTC Linear Technol																																		○	○	○		○		
Maxim																								●				●	●	●		○		●	●	●	●	●		
Micro Chip														○	○											○	○			○										
Micron																	○	●	○		○																			
Micronas																												○										○	○	
MicroPowerSystem																												○											○	
Mitel																												○												
Mitsubishi														○			○	○												○										
Motorola	●		●	●	●	●		●	●								●	●			○	○		○	●	○	●	●	●	○			●	●	●	●	●	●	●	
Mosel																	○			○	○																			
National Semic.	●	●	●	●	●	●			●		●		○	○	○		○			○				●	●		●	●	●	●		●	●			●	●	●	●	
NCR																										○														
NEC										●				●	●		●	●	●	○				●	●	●	○	●	●	○	●								●	
Newport																																								●
New JRC																																		○			○	○		

Manufacturers-product chart

● Major manufacturer
○ Several interesting products

Manufacturers	74LS, S	74ALS,AS	74F	4000 (C)	74HC,HCT	74AC,ACT	74AC11	74BCT Bus inter	ECL 10k,100k	GaAs logic	PLD,PAL, GAL	LCA,PGA,ERA	PROM	EPROM	E²PROM, Flash	Novram	SRAM	DRAM	VRAM	FIFO	Dual port	Cache, Associative memory	Mem. driv., ECC	Video int., DAC	Real time clock	Progr. interface	Fast arith. Signal processing	Comm. circuits	Line driver	Display drivers	CCD reg, camera	Opto electr	Transistors, diodes	Motion contr.	Regul. DC/DC	References	Op-amp, comp.	Filters	A/D, D/A	Delay lines
OKI Semi																										○														
Panasonic																	○																							
PerformanceSemi.						○											○																							
Plessey												●								●				●			●									●	○		○	
PLX								○																																
PMI																																	○			●	●		●	
Precision Monolith.																																				○	○			
Raytheon																																			○	○	○		○	
Raytheon																																			○	○	○		○	
Rockwell																												○		○										
Samsung						●	●				○				●		●	●	○	○						○							○						●	
Saratoga																	○				○																			
Seagate																																		○	○					
Seeq											○			●	●																									
Seiko/Epson													○	○			○								●				○	○					○					
Seiko Instr.															○	○	○										○								●	●				
SGS/Thomson	●			●	●						○			○	○	○	○	○		●		●		●	○	○	○	○	○	○			●	●	●	○	●	●	●	
Sharp /Inmos													●	●			●	●		●	●								●	●	●	●								
Signetics/Philips	●	●	●	●	●		●	●	●		●		●	●	●		○			○			○	○	●	○		○	●	●			●				●		○	
Siemens	○																○	○								○		○				○								
Sierra															○									●														○		
Silicon Gen.																																		●	●	●	●	●		
Siliconix																								●					○	●			●	●	●		●			
Siplex																													○							○	○		○	
Solitron																										○		○												
Sony									●	●							●									○					●								●	
SPT																											●											○	○	
Sprague/Allegro																																	○	●						
Supertex																										○		○	○	○										
Tamarak																										○		○	○											
Teledyne																													○	○			●		●	●	●		●	
Telefunken																																●	○			○				
Texas Array																										○														
Texas Instr.	●	●	●		●		●	●			●	○	●	●	○			●	●	○		○	○		○	○	●	○	●	○	○	○		○			●	●	●	
Three Five III/V																																								
Toshiba				●	●	●		○						●			●	●	●	○		○				○			○	●	●	●	●		○					
TRW																								●			○											○	○	○
UMC																										○														
Unitrode																																	○							
USE United Semi																				○																				
Vitelic																	○	○		○	○																			
Vitesse										●																														
VTC						○																					○		○											
VLSI Technol																										○	○	○												
Waferscale											○	○	○	○																										
Weitek																											●			●										
Western Digital																										●		○												
Xicor															●	●																								
Xilinx												○																												
Zilog																					○					●		○												

Survey

Gate survey

	TTL (LS,ALS,AS,F)						CMOS (C,HC,HCT,AC,ACT,ACL)					
	1-2	3	4	5-8	>8	comb.	1-2	3	4	5-8	>8	comb.
6x	04 1004						04 4069 901 903					7006 7074
	34 1034 63						4010 4050 4504					
	265											
2x 4x 6x	8003 00 37 804	10	40 20 140	30	133 134	265 31	40107 00 4011 804	10 4023	20 4012	30 4068		7006 4501 4572
4x 6x	28 128 08 808	11	21				08 4081 808	11 4073	21 4082			
4x 6x	02 805	27	23	260			02 4001 7002 805	27 4025	4002	4078		4000
4x 6x	32 832						32 7032 4071 832	4075	4072	4078		
XOR 4x	86 386					135	86 386 4070					
XNOR 4x	810						4077 7266 810					
1x 2x	54 51	54	55 64				4085 4086	58				4506

Schmitt trigger	TTL (LS,ALS,AS,F)			CMOS (HC,HCT,AC,ACT)		
	1	2	4	1	2	4
6x 9x	14			14 4584 40106 9014 9114[1]		
9x				9015 9115[1]		
		132 24	13 18		132 7003[1] 4093	
					7001	
					7002	
					7032	

Note 1: Open-collector output

le4rscp

Open-collector	TTL (LS,ALS,AS,F)			CMOS (HC,HCT,AC,ACT)		
	1-2	3	4	1-2	3	4
	05 06[1] 16[1]			05		
	35 07[1] 17[1]			906		
	03 01 39 38	12	22	03 7003[2]		
	09	15		09		
	33					
XOR	136					
XNOR	266 811			266		

Note 1: Increased output voltage (15V ou 30V)
Note 2: Schmitt trigger input

Mono-, bi- and tri-directional drivers

	Drivers per package									
	4	4+2	6	4+4	8	9	10	16	18	20
a	126 426			241						
b	125 425	367	365	244 231 467 734 2966 230				16244		
c		368	366	240 468 730 2965 231				16240		
d										
e			365	827	747 465 541 7541 2541		827 2827	16541	16825	16827
f			366	828	746 466 540 7540 2540		828 2828	16540	16826	16828
g					456 656					
h	2910 2912 26S10				455 655					
i	2911 26S11									

	4	8	9	10	16	18	20
a	243 1243	623 1623			16623	16863	16861
b	242 1242 2242	620 1620			16620	16864	16862
c		(2949)	863	861			
d		(2948)	864	862			
e	449	245 1245 645 454 639 [1] (2947) 643			16245		
f	446	640 1640 638 [1] (2946)			16640		
g direct		657 833 659 853 665					
g inverse		658 834 664 854					

Note 1: Open collector on one side

Note 2: Open collector on both sides

	4 bits	8 bits	16 bits
4	442		
	443		
8	444		
		646 647 [2]	16646
		648 649 [2]	16648
		652 654 [1] 615 [2]	16652
		651 653 [1] 614 [2]	16651

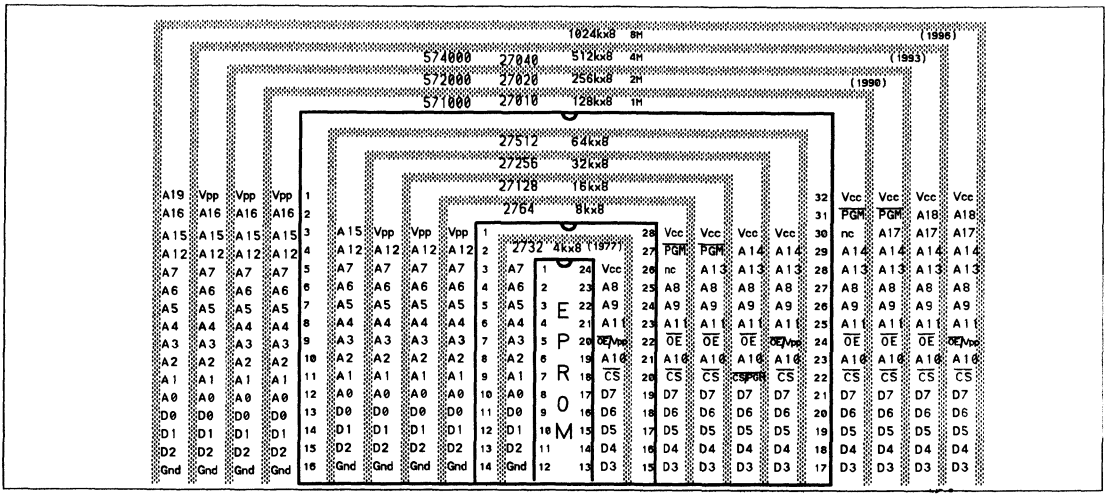
Comparators

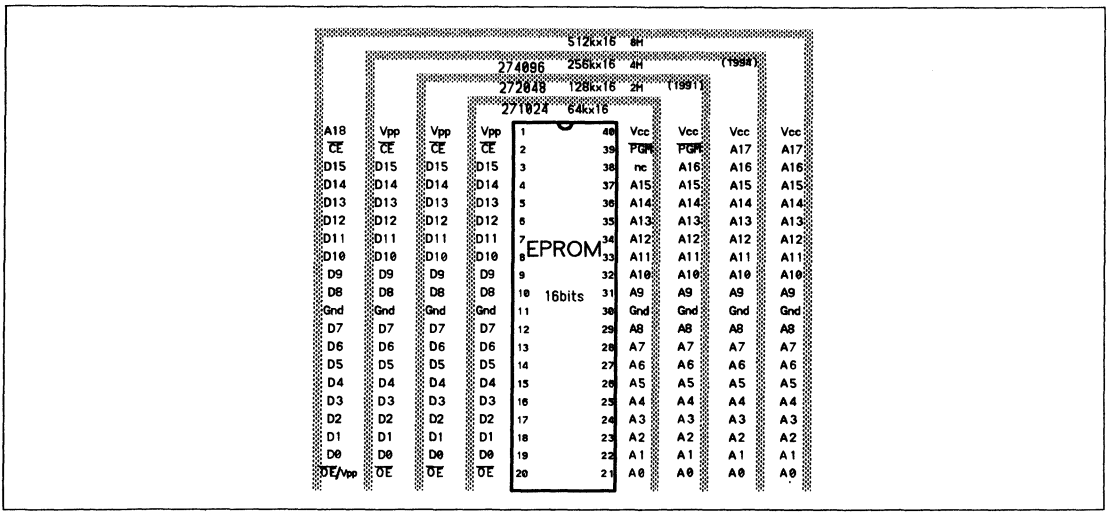
	4 bits	8bits +pull ups	8bits⌂ +pull ups	8bits with reg	with decoder
	85 4585			886 524	
				885	
		685 682 686	684 683 687		
			519 518		
		521 688 520 809	522 689		806

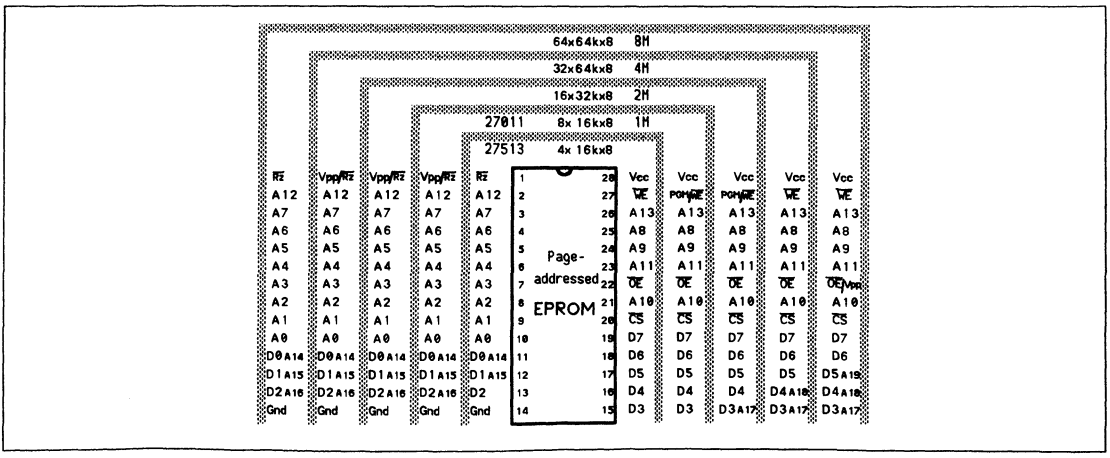
Registers and latches

Registers	number	direct	inverted	Latches	number	direct	inverted
a	4 6 8	173 [171 175] 174 378 273			2x2 2x4 8	77 [75 375 4042] 100 116 412	
b Reset synchrone	4 4+4 6 8	878	879		8 number	direct	inverted
c	8 10	374 574 821	364 564 534		4+4 8 10	373 573 841	533 563 580 842
d	4+4 8	874 878 575	876 879 577		4+4 8 9	4508 873 845 843	880 846 844
e	4 6 8	379 378 377 825	576				
f	8 9 10	25LS2520 821	822				
g	8 9 10	825 [996] 823	826 824				
h	4	298 398 399			8	604 606 ▽ 605 607 ⌂	

Read-only memories

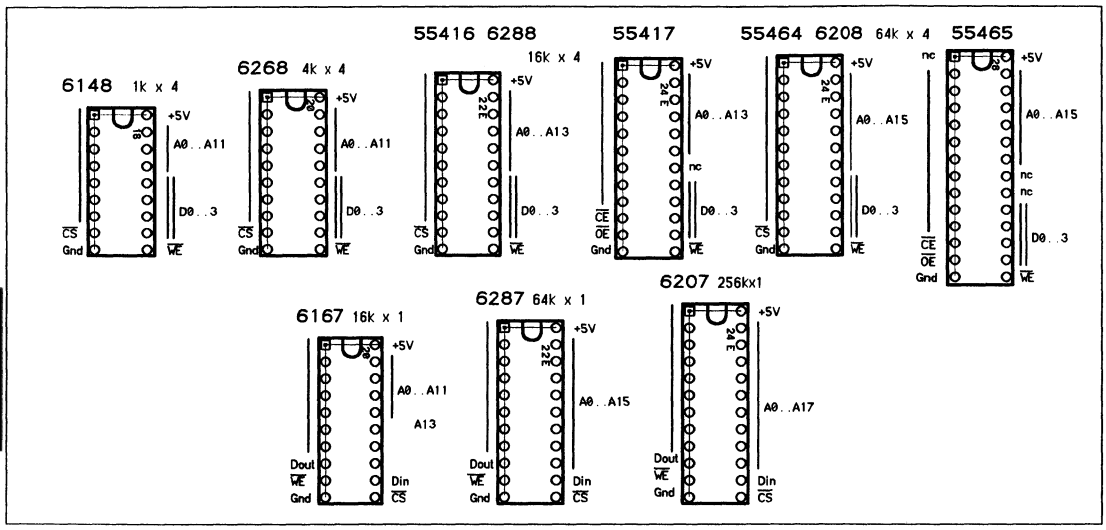


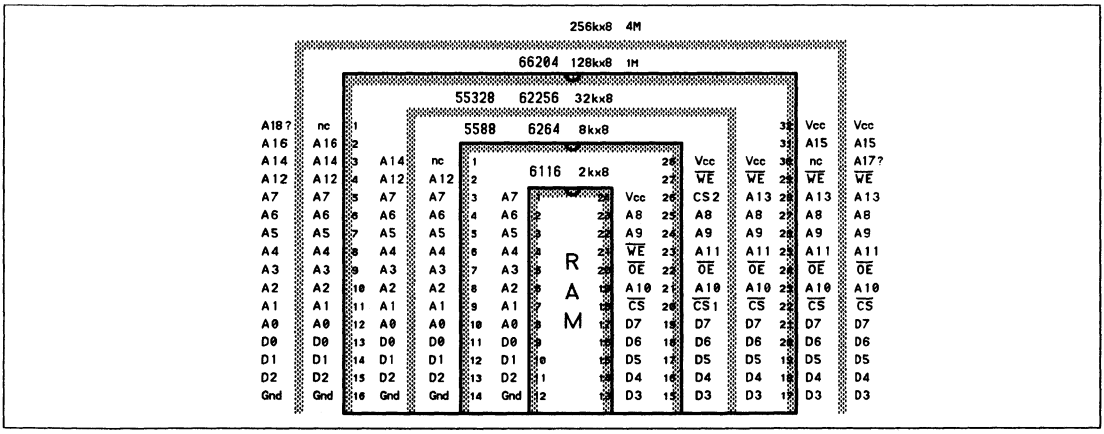




Memories

Static memories

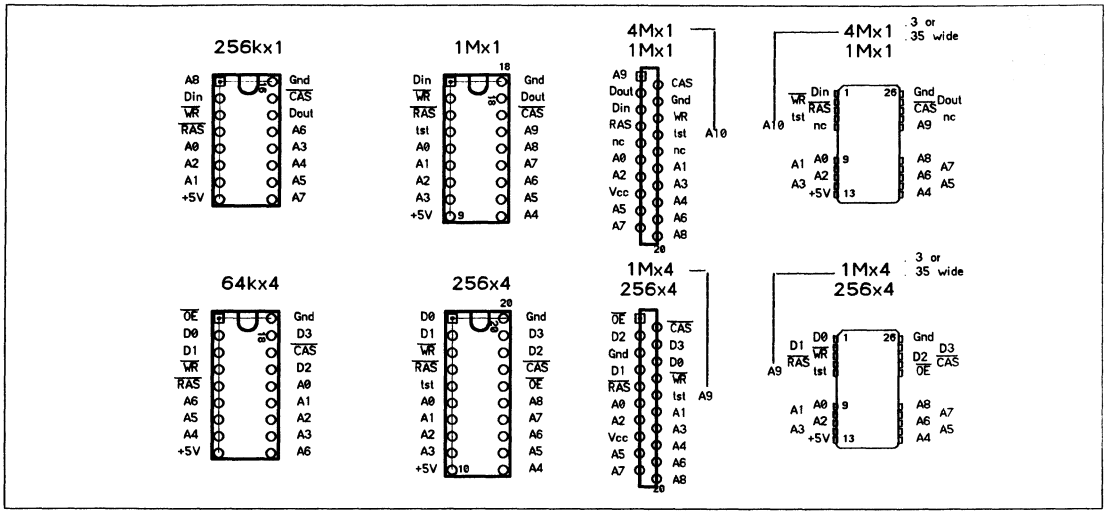




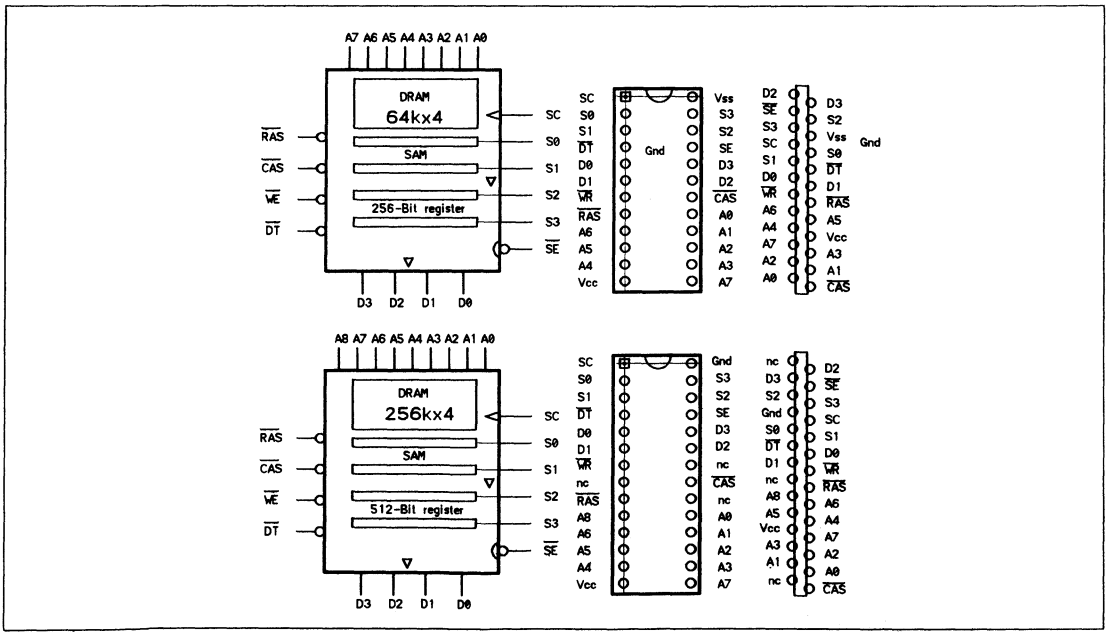
FIFO memories

Size	Type	Manufacturer	Package
16 x 4	HC40105	Philips	16
16 x 4	74ALS232	Texas Inst, NS	16
16 x 5	74ALS233	Texas Inst, NS	20
64 x 4	74ALS234	Texas Inst, NS	20
64 x 4	74F433	Fairchild	24 s
64 x 4 serial out	74F413	Fairchild	16
64 x 9	HC7030	Philips	28
64 x 9	74AC708	Texas Inst, Fairchild	28
512 x 9	7201,4501,..	SGS, MMI, IDT, Cypress, Texas Inst	28
1024 x 9	7202,4502,..	SGS, MMI, IDT, Cypress, Vitelic, Dallas	28
2048 x 9	7203,4503,..	SGS, IDT, Cypress, VLSI, etc.	28
2048 x 9 serial out	72103	IDT	28
4096 x 9	7204,..	SGS, IDT, Cypress	28
4096 x 9 serial out	72104	IDT	28

Dynamic memories (DRAM)



Video memories (VRAM)

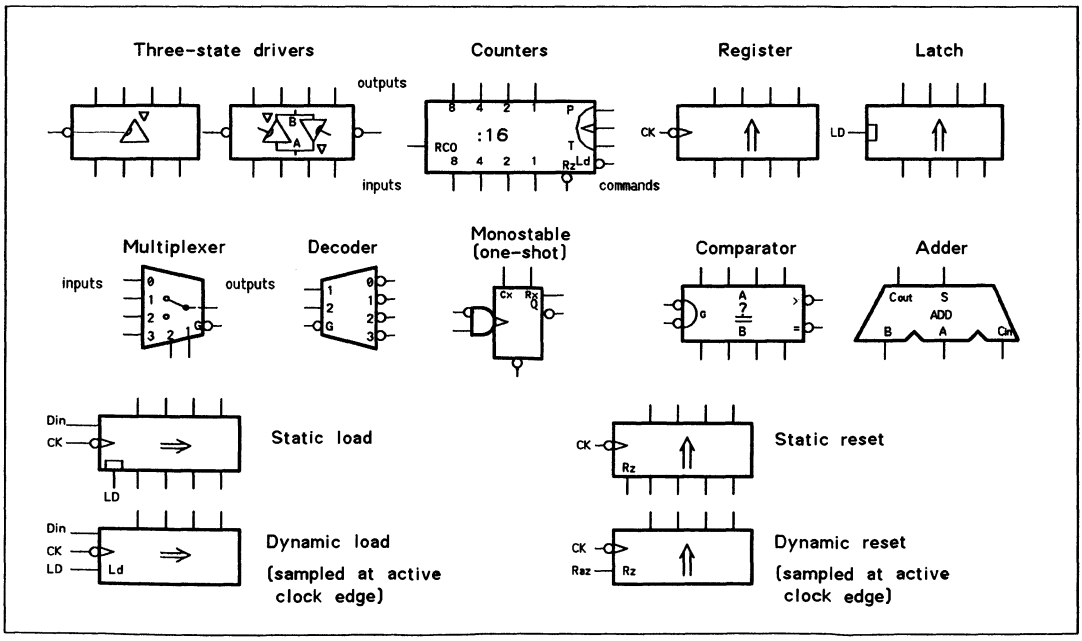


Memories

Conventions

Drawing conventions (symbols et schematics)

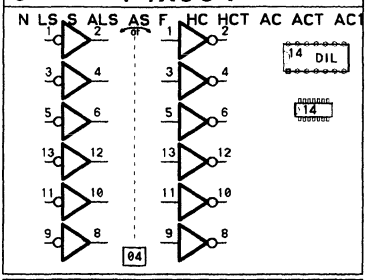
Input Output		Input Output	
	direct I/O (active high)		Three-state output
	inverted I/O (active low)		Open collector/drain output
	positive edge active (dynamic)	10k	same with internal pull-up
	negative edge active		Open emitter/source output
	active pulse (static)	10k	same with internal pull-down
	master-slave structure	50	Serial internal resistor
	data lockout		External pull-up
	Schmitt trigger input		External pull-down
	Analog I/O	[1] 10k	Input with internal pull-up
	Non logic I/O	[0] 10k	Input with internal pull-down
	Buffered output		Functional pseudo-inverter
G	Gating input		Direction of data flow
Rz	Reset input		Bidirectional line
Ld	Parallel load input		Side connections
	Three-state enable input		Vcc Gnd



Inverters

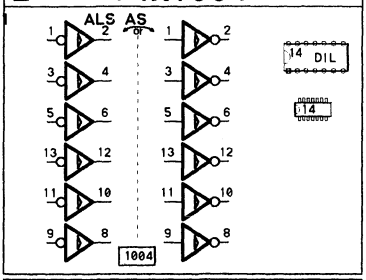
● 74x004

N LS S ALS AS F HC HCT AC ACT AC11



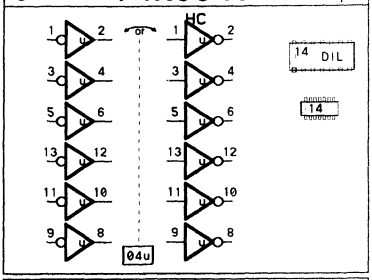
■ 74x1004

ALS AS



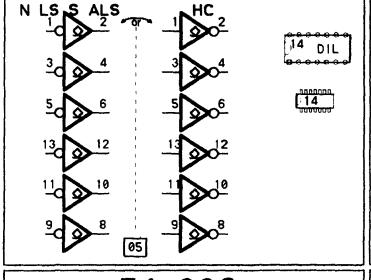
● 74x004u Unbuffered outputs

HC



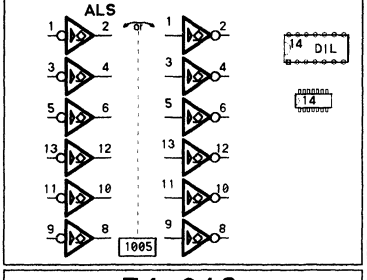
● 74x005

N LS S ALS HC



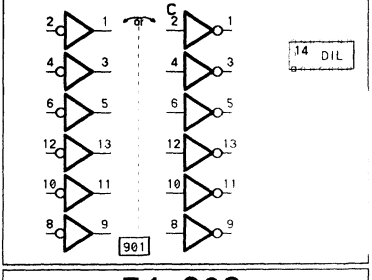
■ 74x1005

ALS



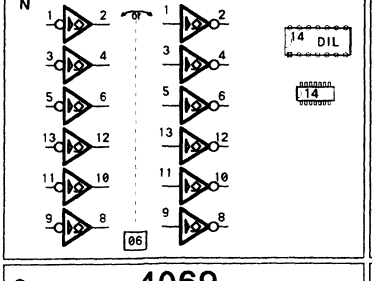
† 74x901 TTL buffer NS

C



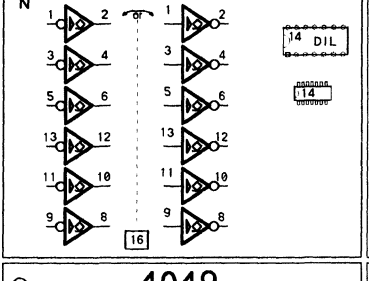
● 74x006 30V

N



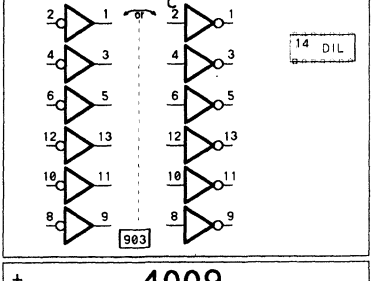
○ 74x016 15V

N



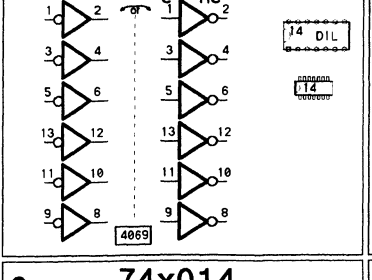
† 74x903 CMOS buffer NS

C



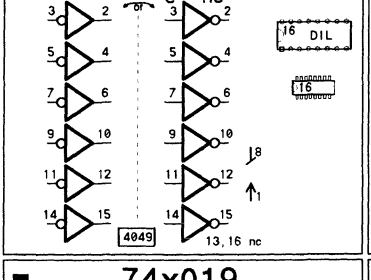
○ 4069

C HC



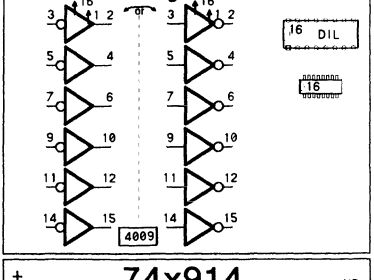
○ 4049

C HC



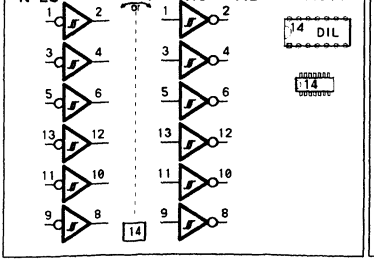
† 4009

C



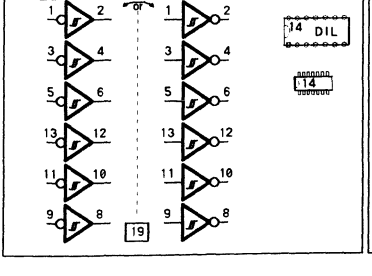
● 74x014

N LS F HC AC AC11



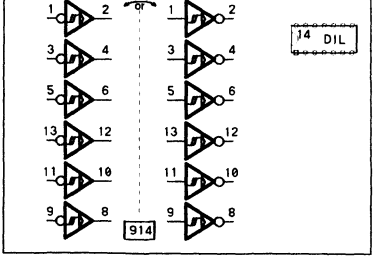
■ 74x019

LS

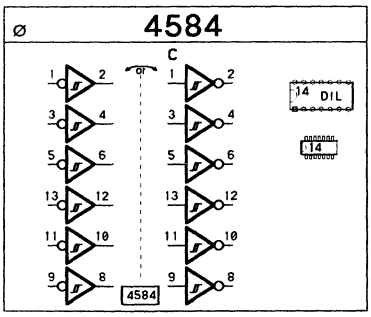


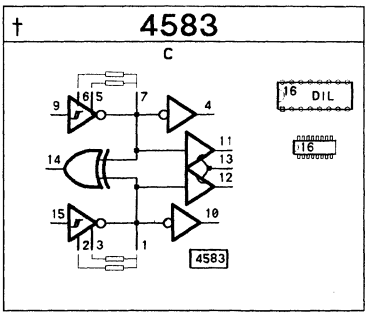
† 74x914 NS

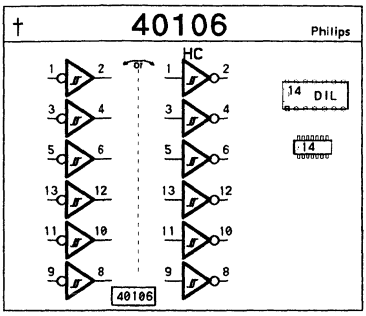
C

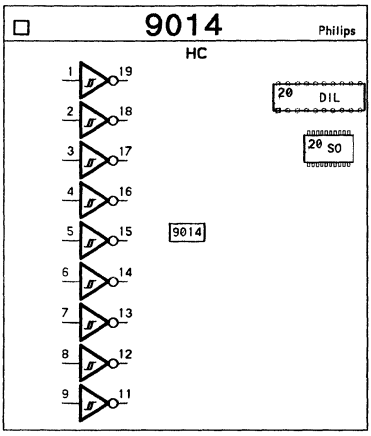


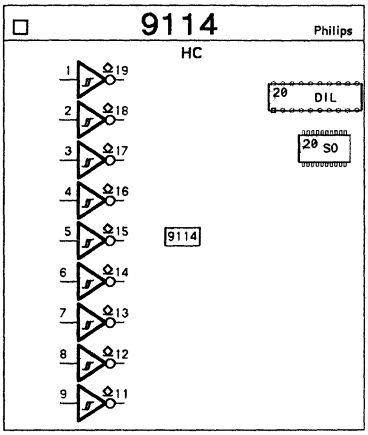
Inverters



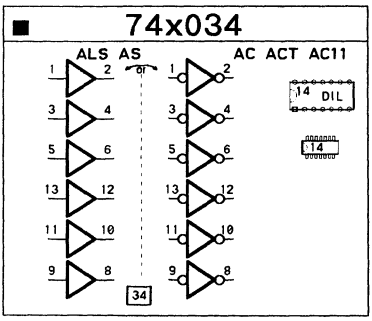


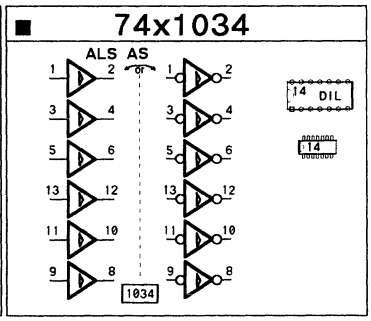


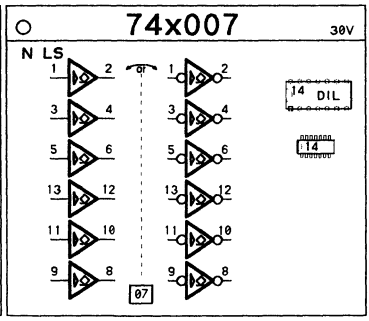


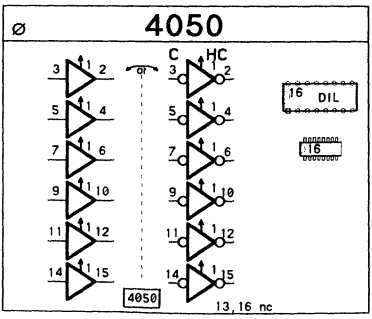


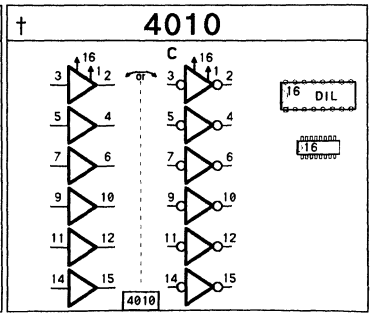
Buffers (non-inverters)

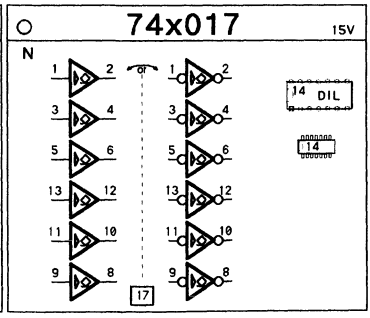






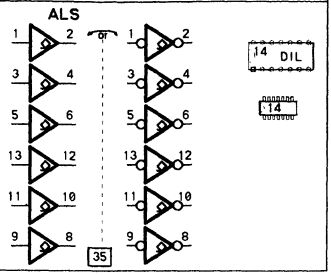






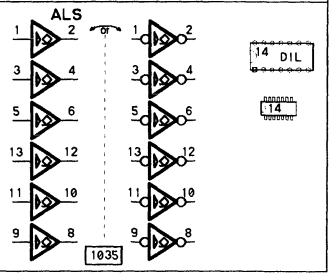
■ 74x035

ALS



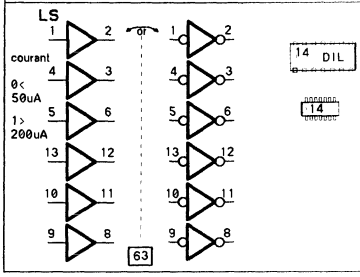
■ 74x1035

ALS



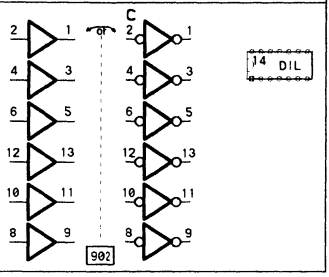
† 74x063

LS



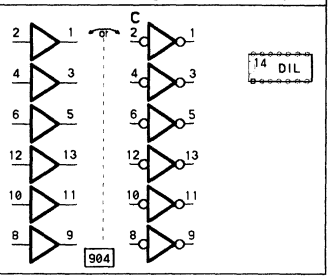
† 74x902 TTL buffer NS

C



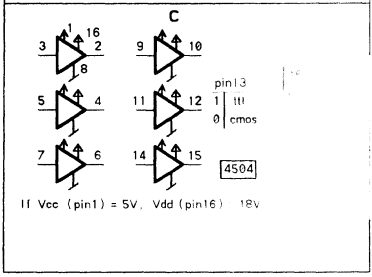
† 74x904 CMOS buffer NS

C



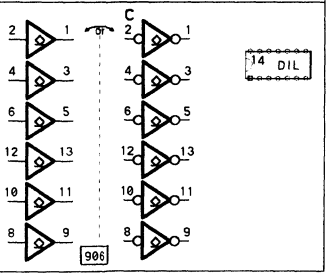
Ø 4504 M

C



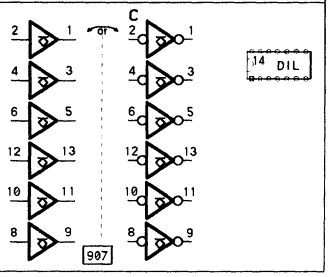
† 74x906 open drain N NS

C



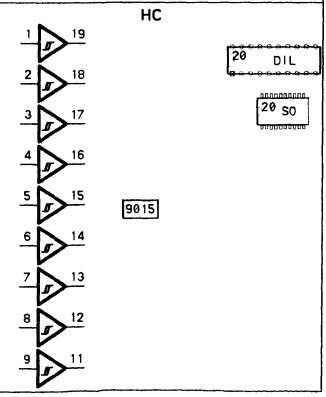
† 74x907 open drain P NS

C



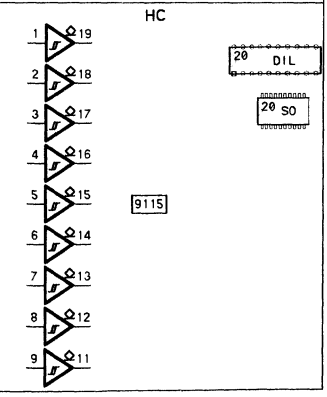
□ 9015 Philips

HC

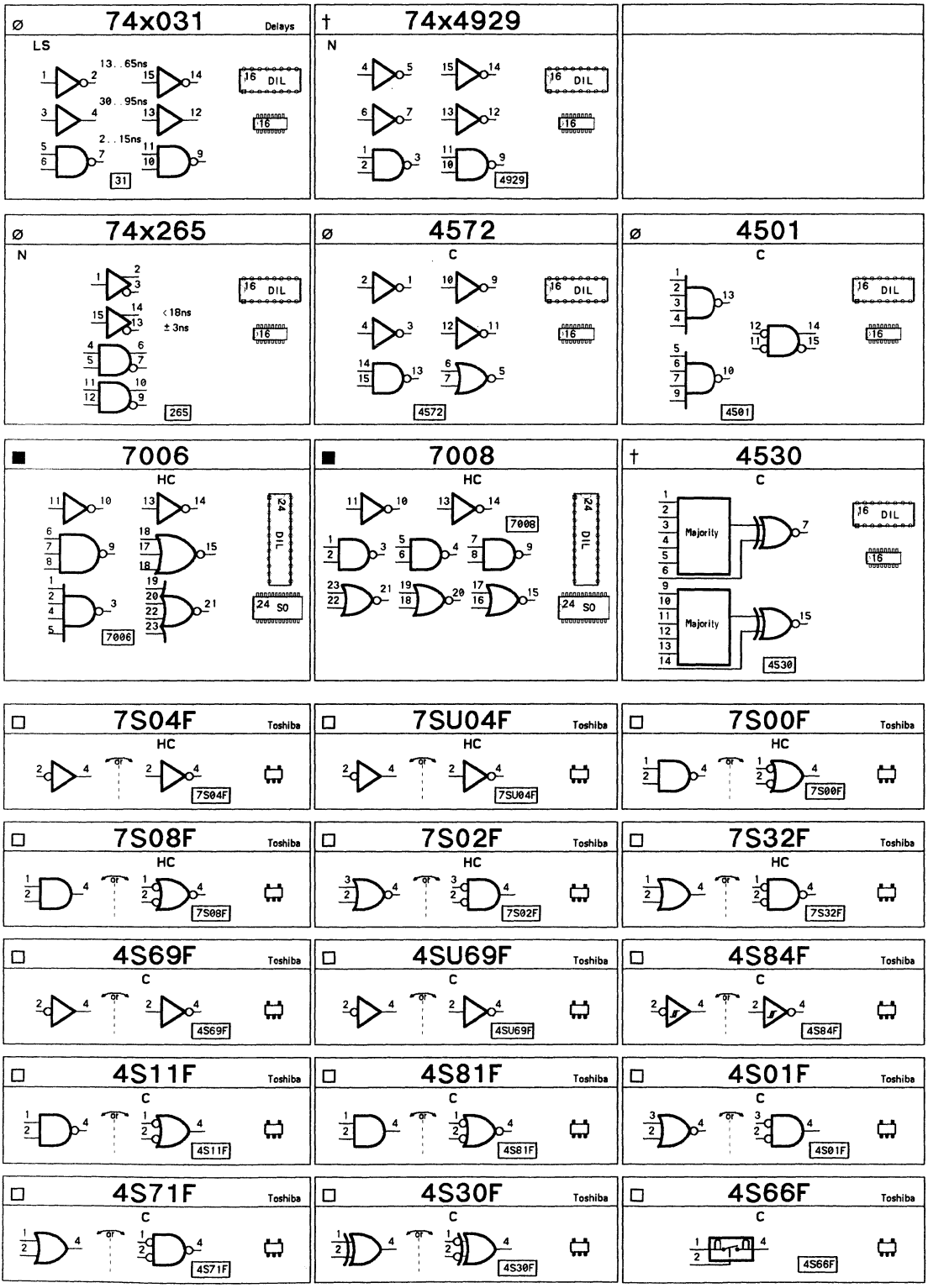


□ 9115 Philips

HC



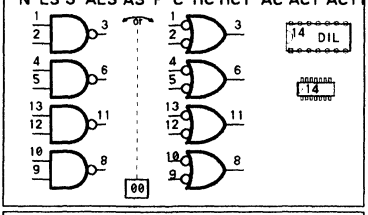
Miscellaneous



NAND gates

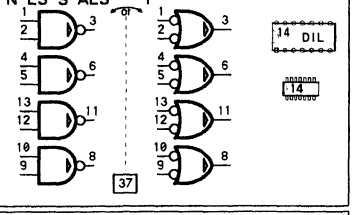
● 74x000

N LS S ALS AS F C HC HCT AC ACT AC11



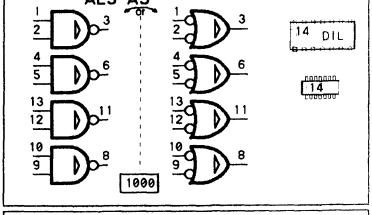
○ 74x037

N LS S ALS F



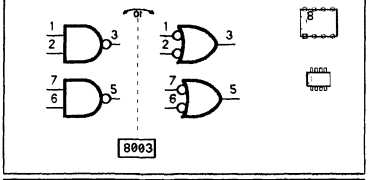
■ 74x1000

ALS AS



■ 74x8003

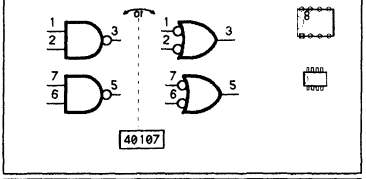
ALS



⌀ 40107

Toshiba, RCA

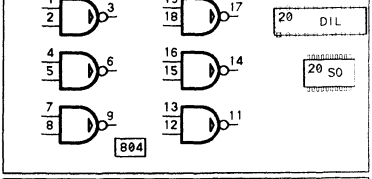
C



□ 74x804

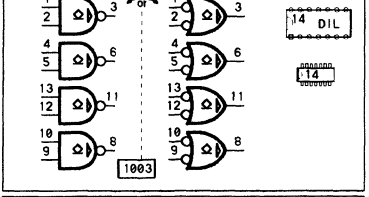
50/75Ω

ALS AS HC



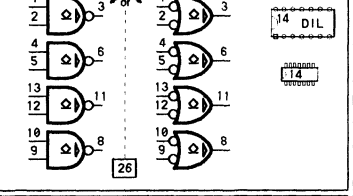
■ 74x1003

ALS



⌀ 74x026

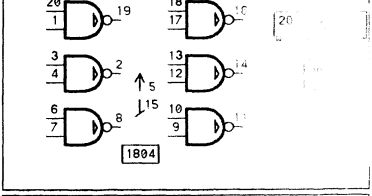
N LS



□ 74x1804

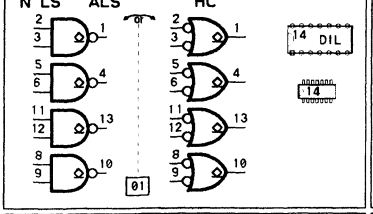
50/75Ω

ALS AS



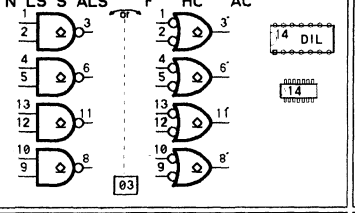
⌀ 74x001

N LS ALS HC



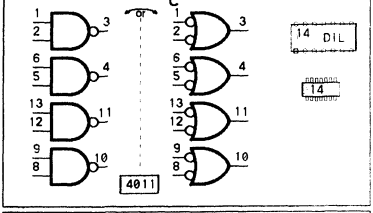
● 74x003

N LS S ALS F HC AC



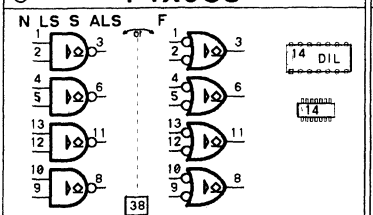
⌀ 4011

C



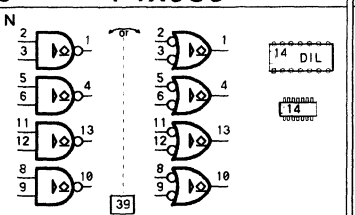
○ 74x038

N LS S ALS F



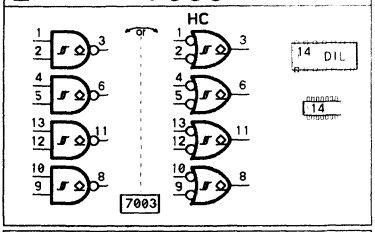
⌀ 74x039

N



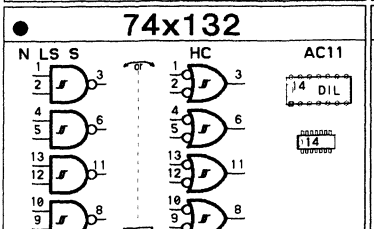
■ 7003

HC



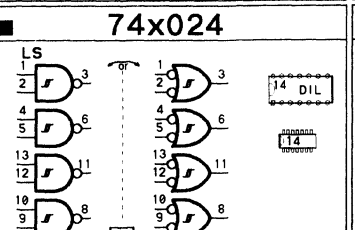
● 74x132

N LS S HC AC11



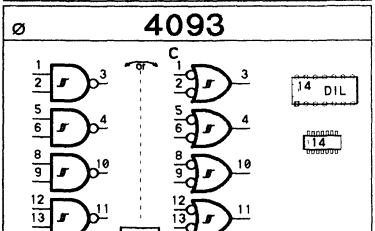
■ 74x024

LS



⌀ 4093

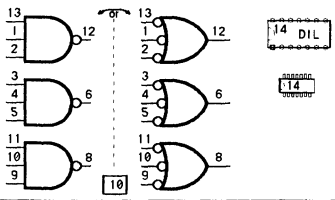
C



Gates

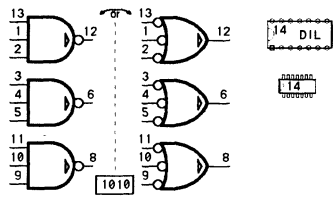
● **74x010**

N LS S ALS AS F HC AC ACT AC11



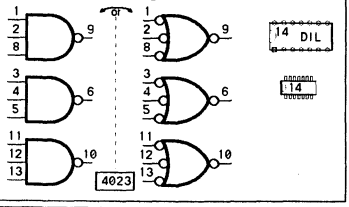
■ **74x1010**

ALS



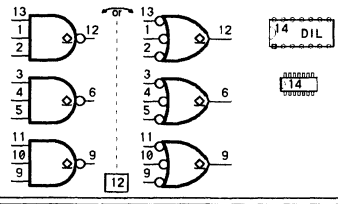
Ø **4023**

C



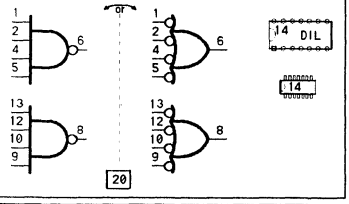
● **74x012**

N LS ALS



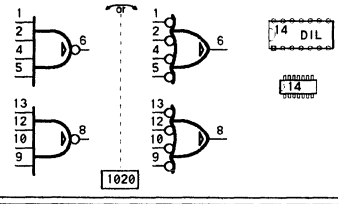
● **74x020**

N LS S ALS AS F HC AC ACT AC11



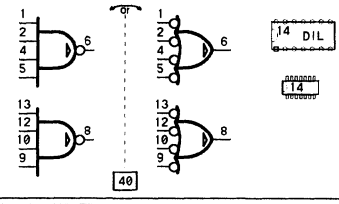
■ **74x1020**

ALS



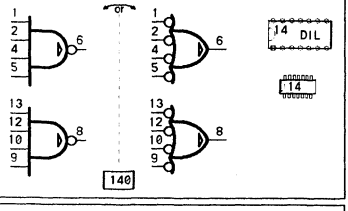
○ **74x040**

N LS S ALS F



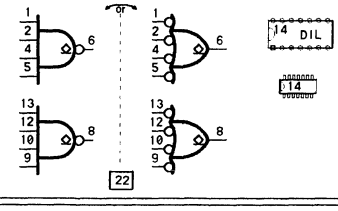
Ø **74x140** 50Ω

S



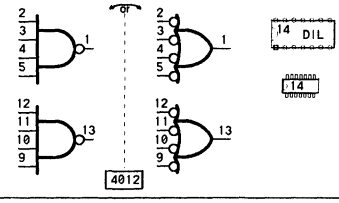
○ **74x022**

N LS S ALS



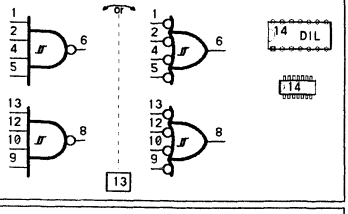
Ø **4012**

C



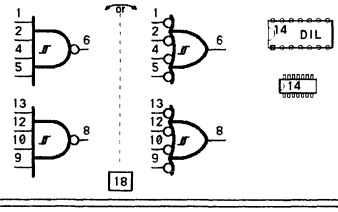
● **74x013**

N LS F AC11



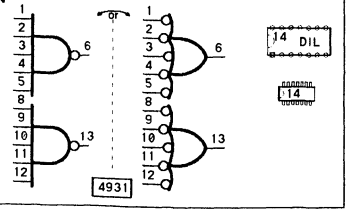
■ **74x018**

LS



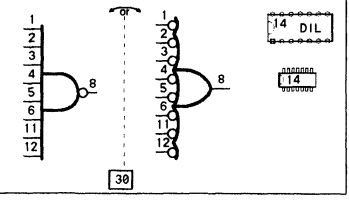
† **74x4931**

N



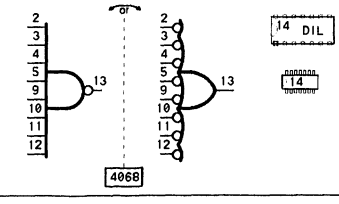
● **74x030**

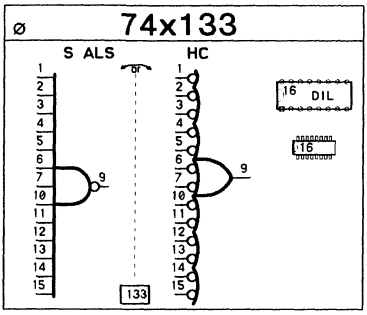
N LS S ALS AS F HC AC ACT AC11

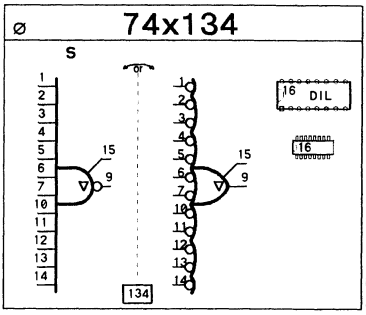


Ø **4068**

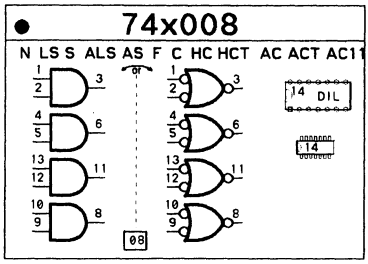
C

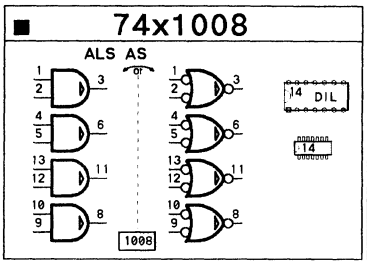


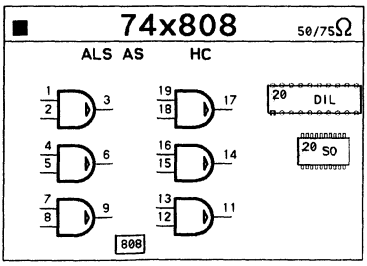


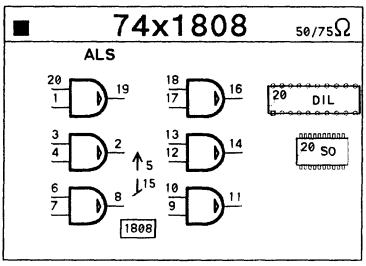


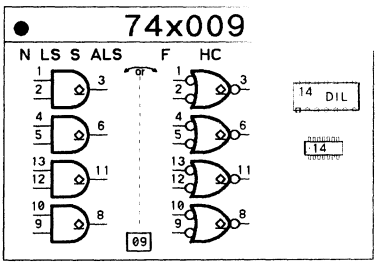
AND gates

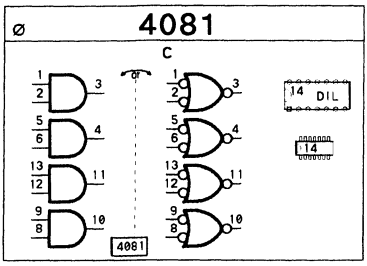


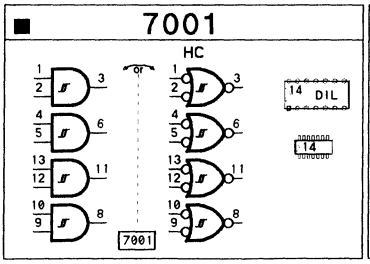


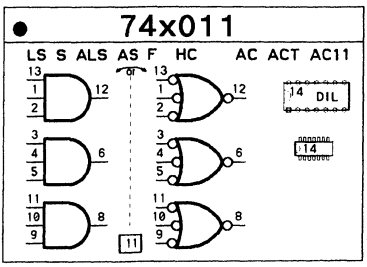


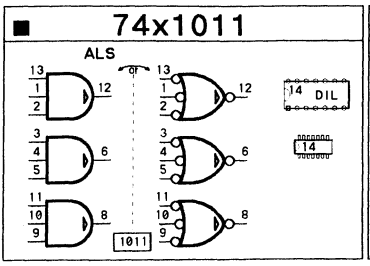


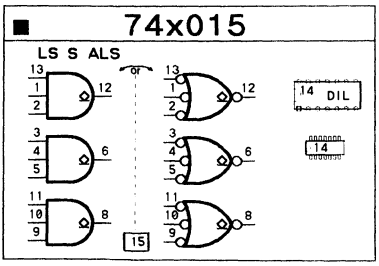




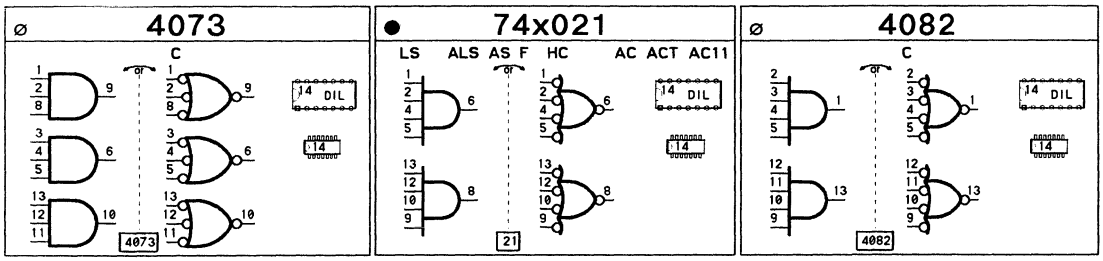




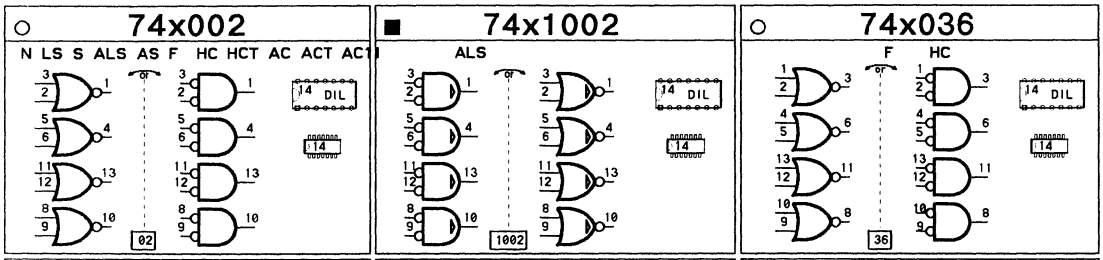


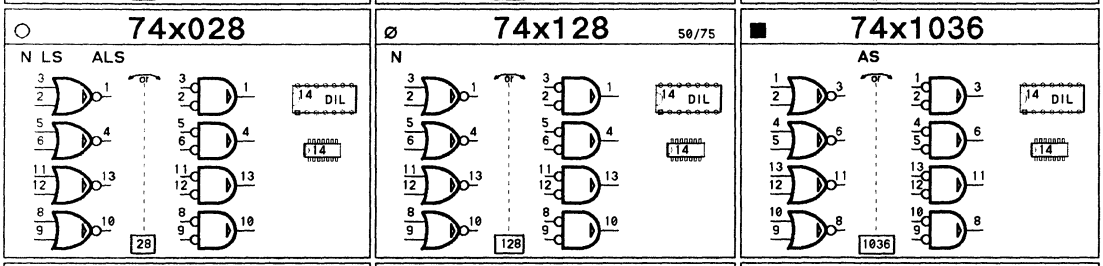


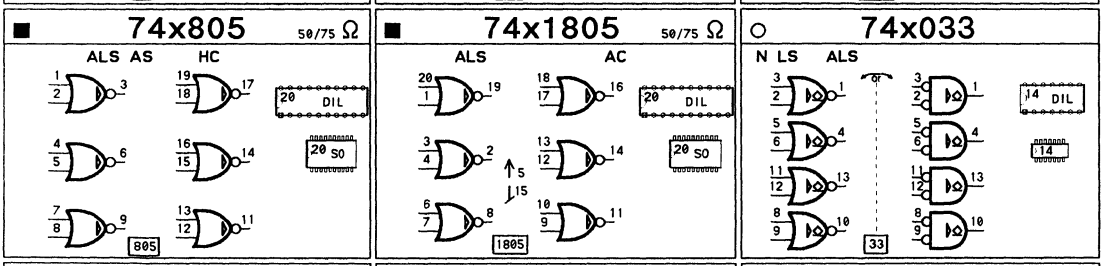
Gates

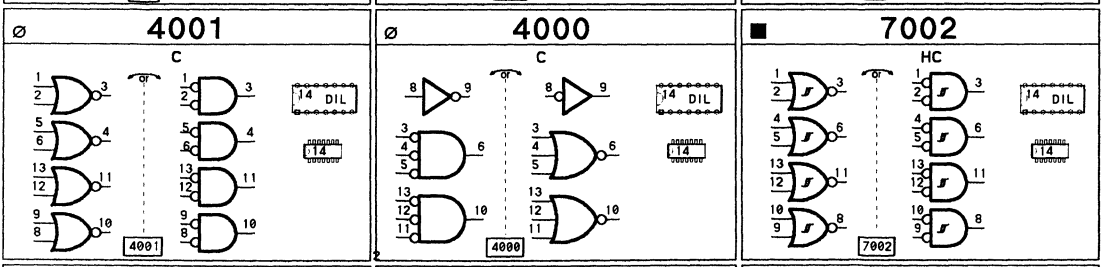


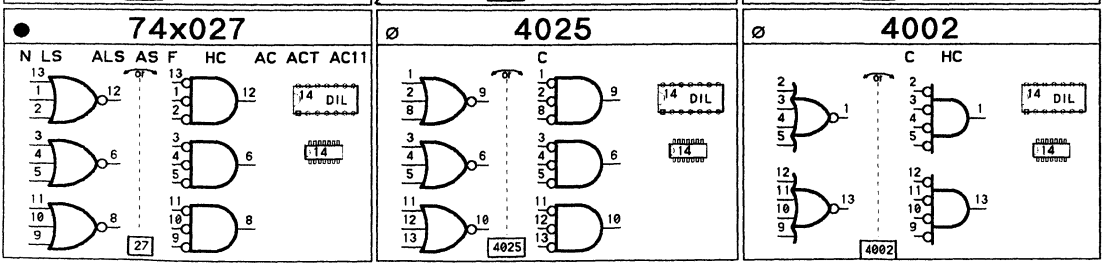
NOR gates



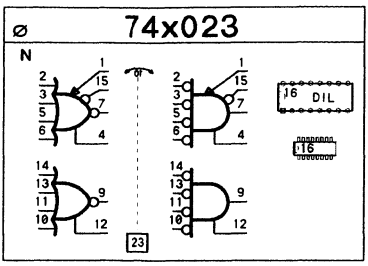


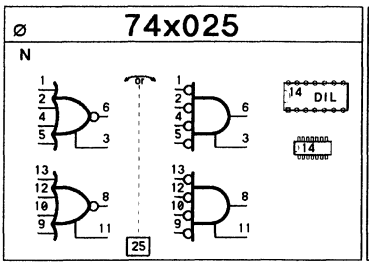


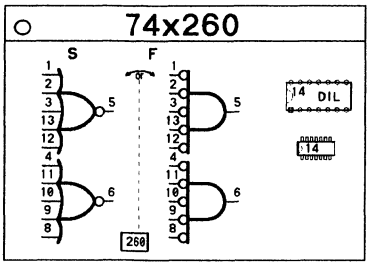


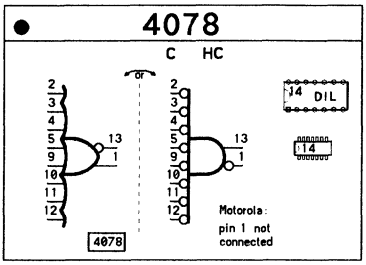


Gates

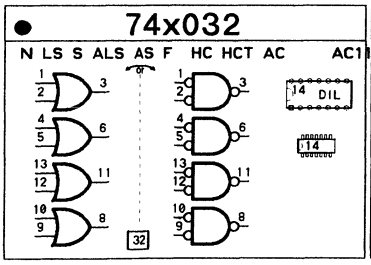


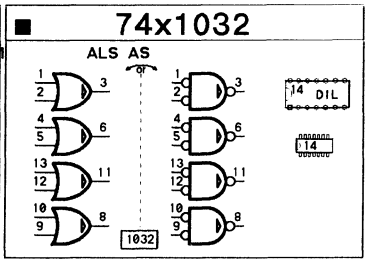


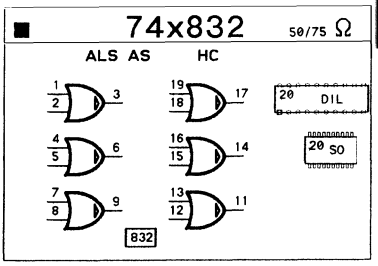


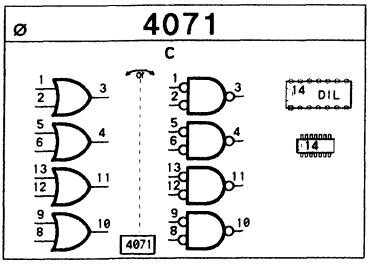


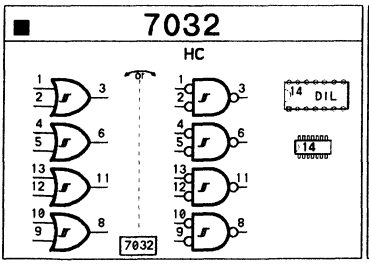
OR gates

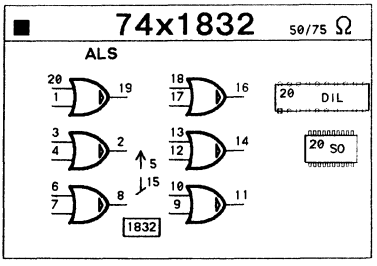


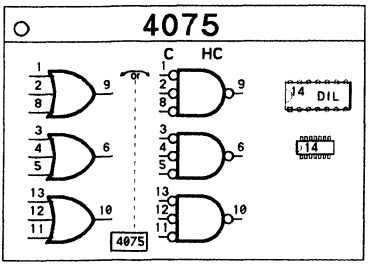


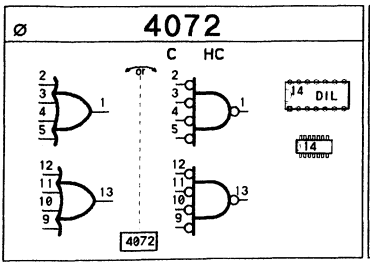


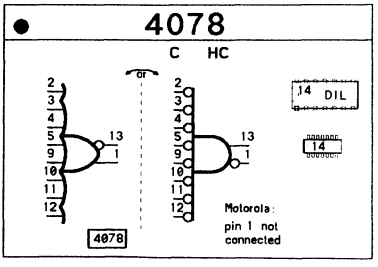






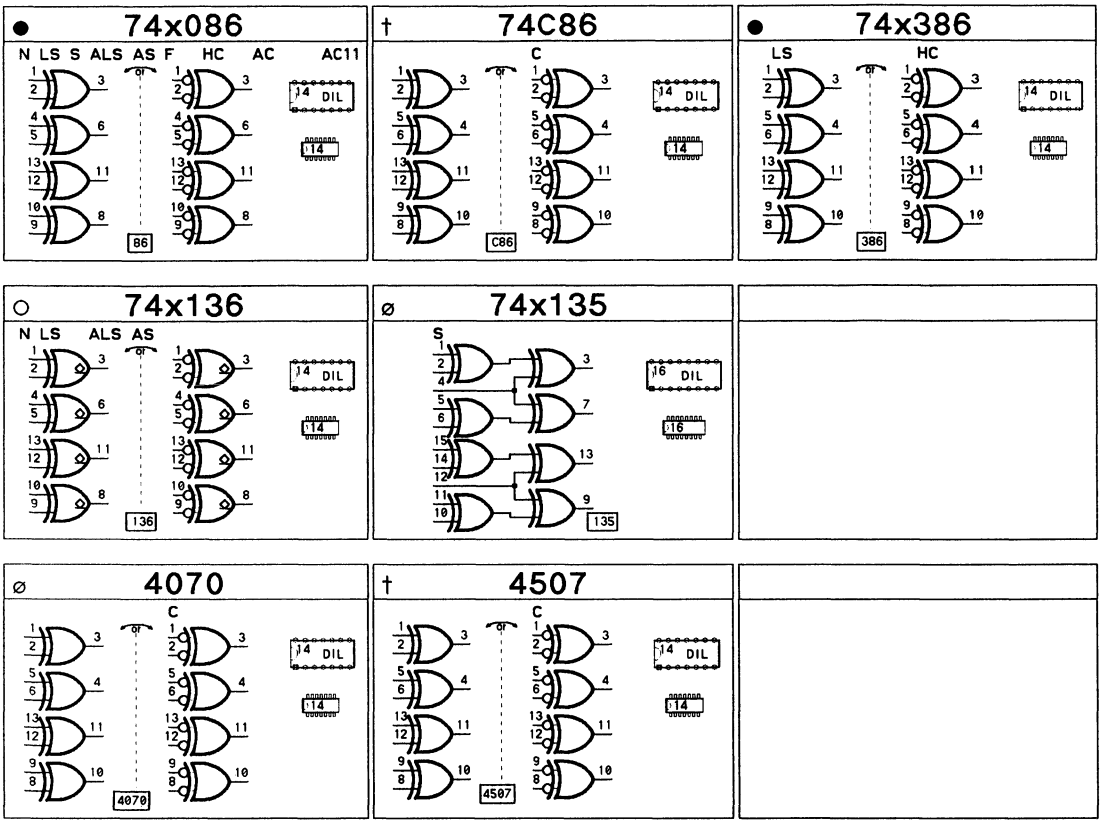






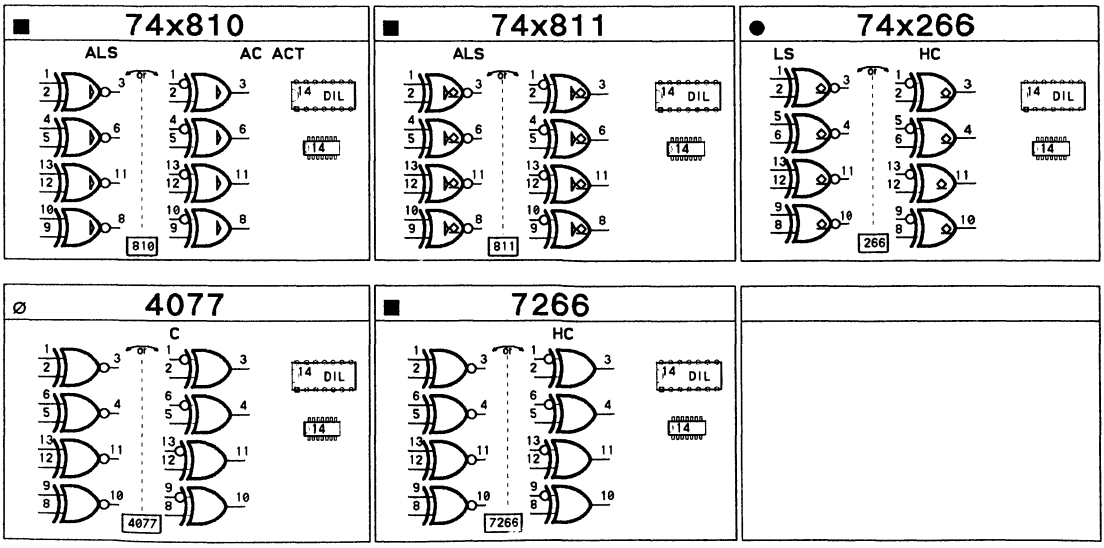
Gates

XOR gates



Gates

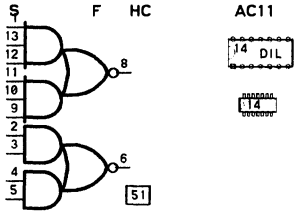
XNOR gates



AND-OR gates

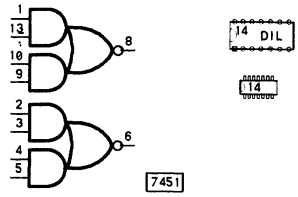
● 74x051

LS S F HC AC11



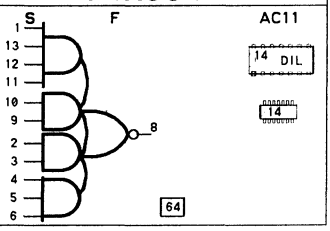
† 7451

N



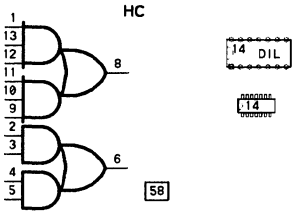
⌀ 74x064

S F AC11



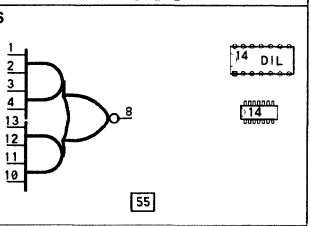
● 74x058

NS HC



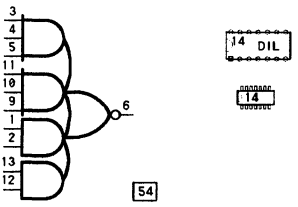
○ 74x055

LS



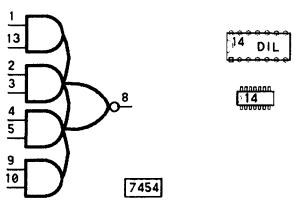
○ 74x054

LS



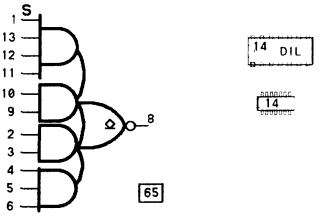
† 7454

N



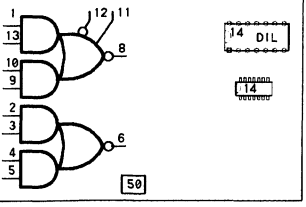
⌀ 74x065

S



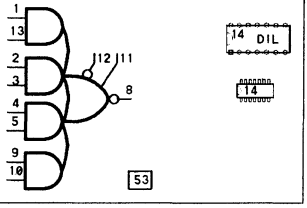
† 74x050

N



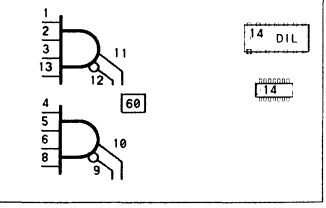
⌀ 74x053

N



† 74x060

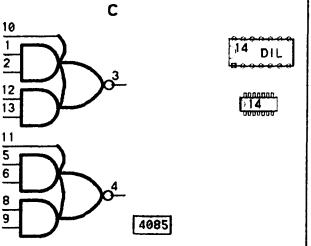
N



† 4085

Toshiba

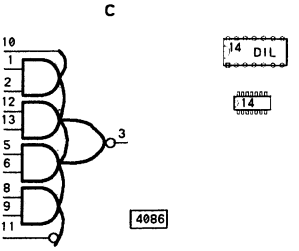
C



† 4086

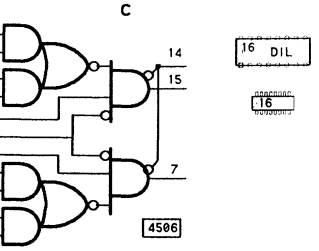
Toshiba

C



† 4506

C

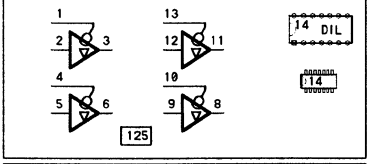


Gates

Three-state drivers

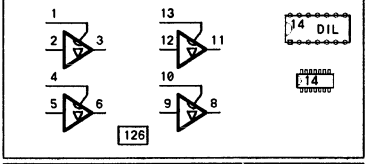
● 74x125

N LS F HC BCT



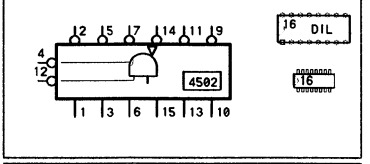
● 74x126

N LS F HC BCT



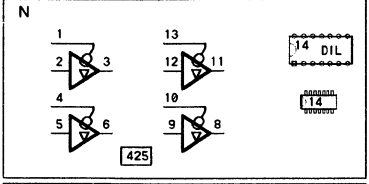
⌀ 74x4502

C



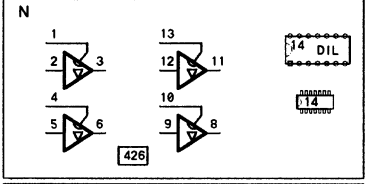
† 74x425

N



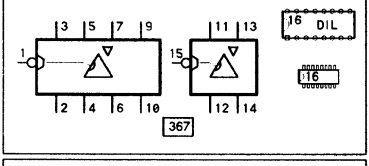
† 74x426

N



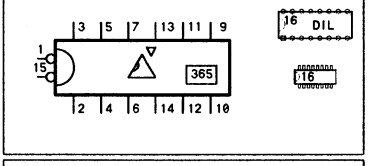
○ 74x367 40097, 81x97, 4503

N LS ALS F HC AC



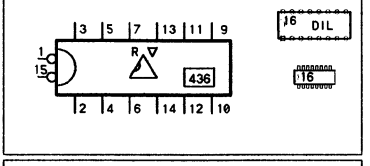
○ 74x365 81x95

N LS ALS F HC



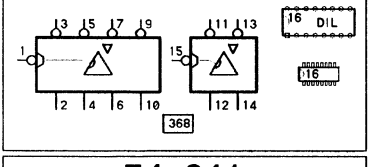
⌀ 74x436

S



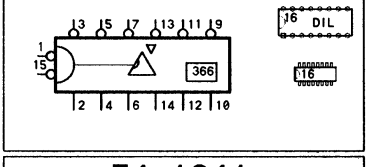
○ 74x368 40098 81x98

N LS ALS F HC AC



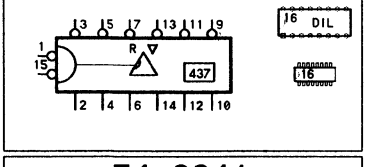
○ 74x366 81x96

N LS ALS F HC



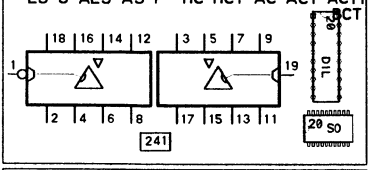
⌀ 74x437

S



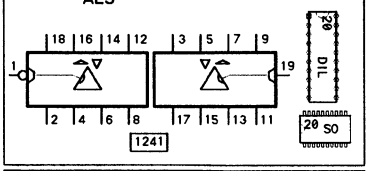
● 74x241 74ALS241-1

LS S ALS AS F HC HCT AC ACT AC11 BCT



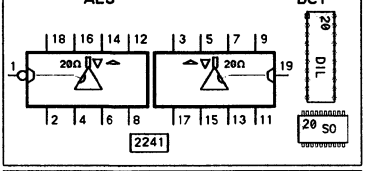
□ 74x1241

ALS



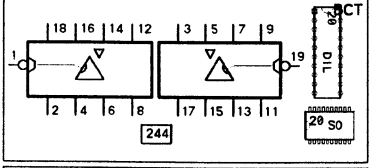
□ 74x2241

ALS BCT



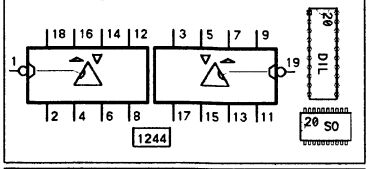
● 74x244 74ALS244-1

LS S ALS AS F HC HCT AC ACT AC11 BCT



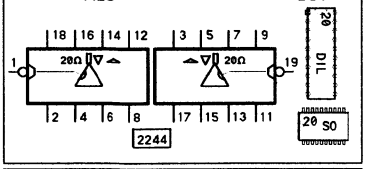
□ 74x1244 74ALS1244-1

ALS



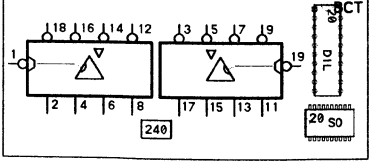
□ 74x2244

ALS BCT



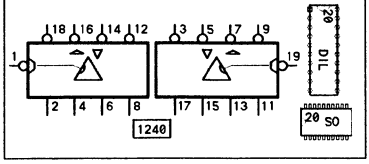
● 74x240 74ALS240-1

LS S ALS AS F HC HCT AC ACT AC11 BCT



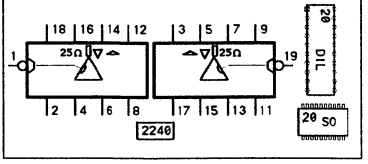
□ 74x1240 74ALS1240-1

ALS

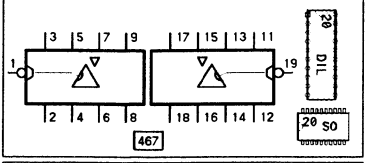


□ 74x2240

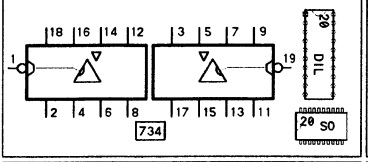
ALS BCT



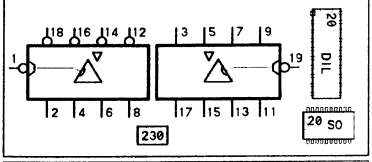
● 74x467
LS ALS



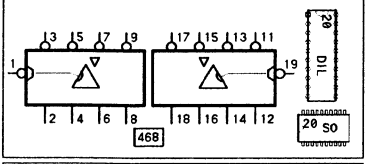
□ 74x734 2966
LS



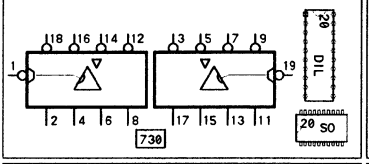
■ 74x230
ALS AS



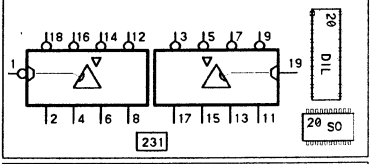
● 74x468
LS ALS



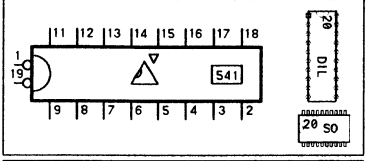
□ 74x730 2965
LS



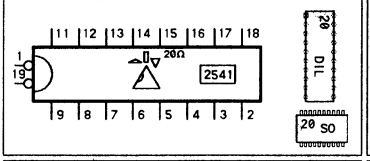
■ 74x231
ALS AS



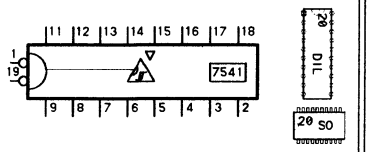
■ 74x541 74ALS541-1
LS ALS F HC HCT AC ACT BCT



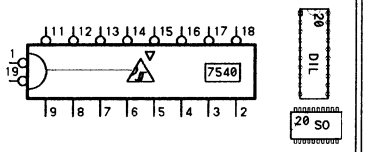
□ 74x2541
ALS



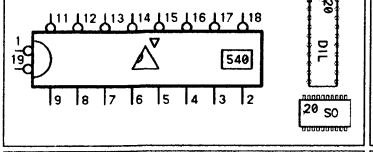
□ 7541 Philips
HC



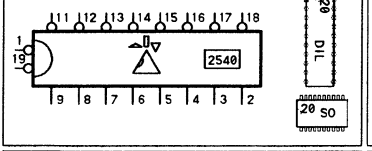
□ 7540 Philips
HC



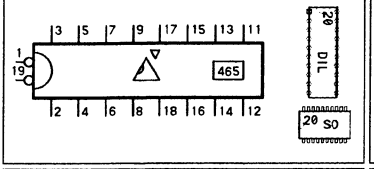
■ 74x540 74ALS540-1
LS ALS F HC HCT AC ACT BCT



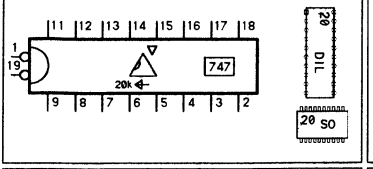
□ 74x2540
ALS



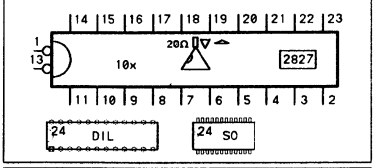
○ 74x465
LS ALS



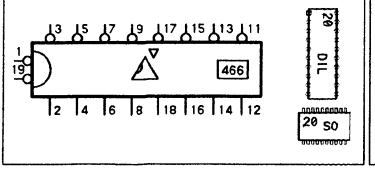
□ 74x747 74ALS747-1
ALS



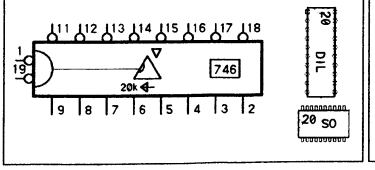
□ 74x2827
BCT



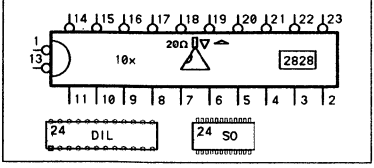
○ 74x466
LS ALS

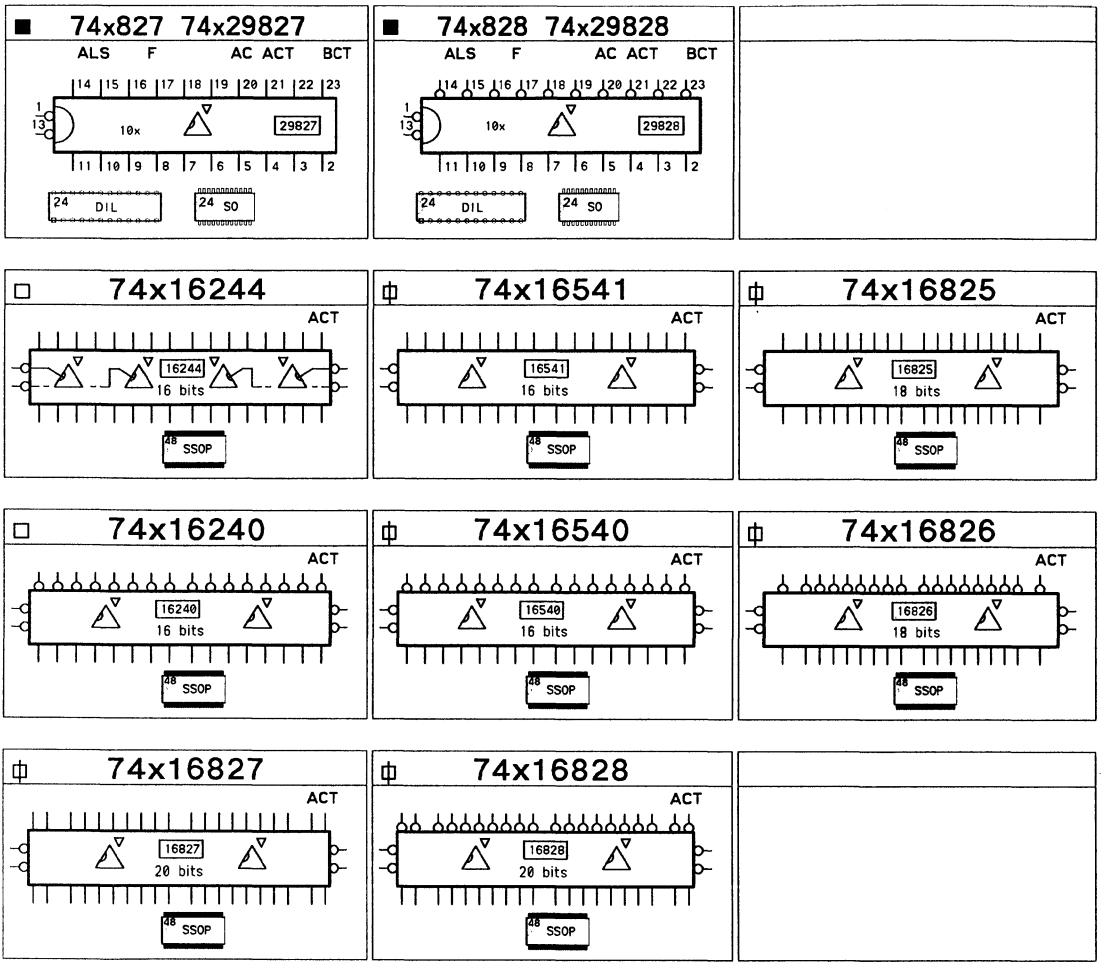


□ 74x746 74ALS746-1
ALS



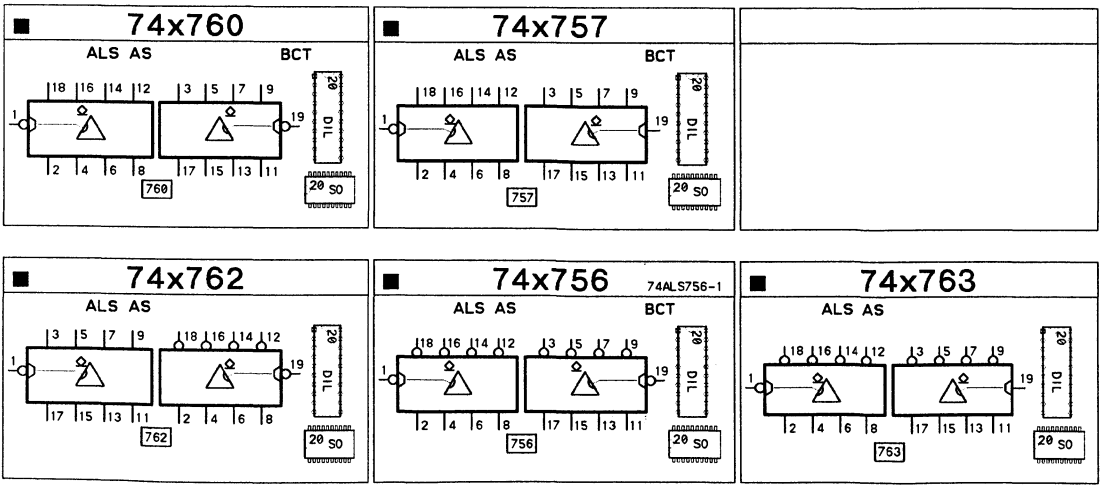
□ 74x2828
BCT



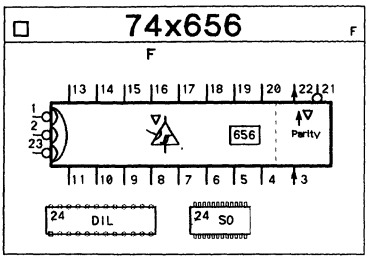


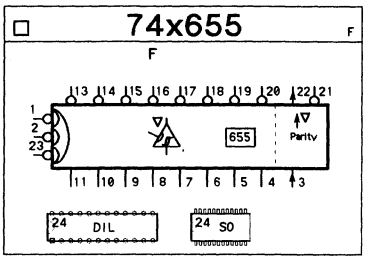
Drivers

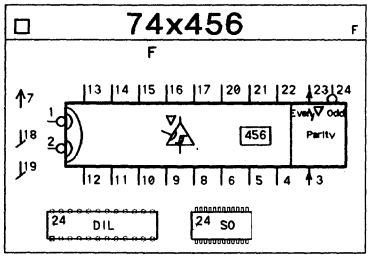
Open-collector drivers

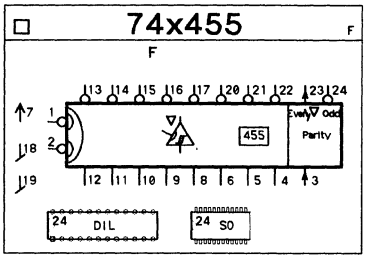


Parity drivers

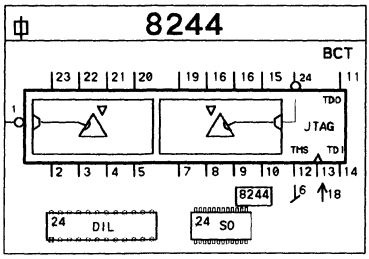


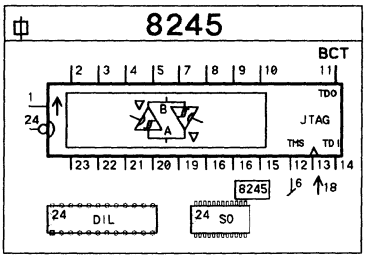


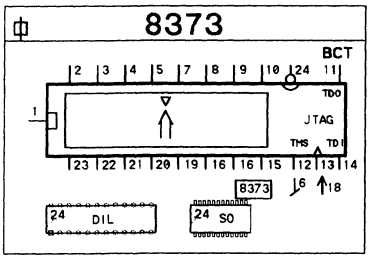


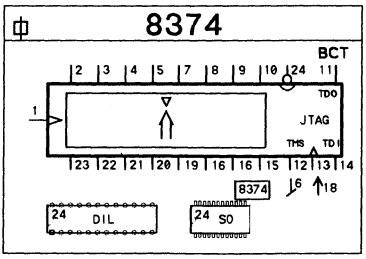


JTAG drivers and registers









Drivers

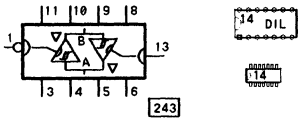
IEEE896 (BTL) drivers

TTL: 3862, 3890, 3892, 3893, 3896, 3897, 3898 (NS)
Fast: 776, 777, 8960, 8961, 8962, 8963 (Signetics)
BicMos: 2030, 2040, 2041, 2042, 2043 (Signetics)

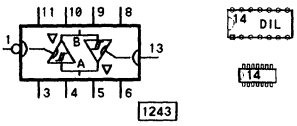
Bidirectionnals three-state drivers

Registered drivers: see page 263

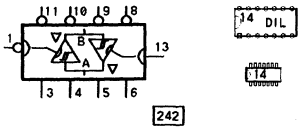
● **74x243** 74ALS243-1
LS ALS AS F HC HCT



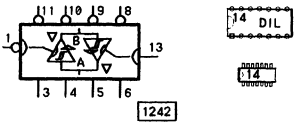
□ **74x1243** NS
ALS



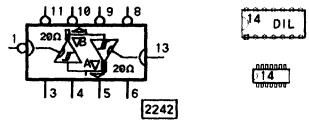
● **74x242** 74ALS242-1
LS ALS AS F HC HCT



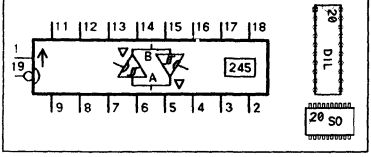
□ **74x1242** 74ALS1242-1
ALS



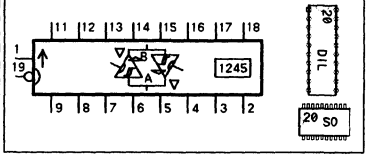
□ **74x2242**
ALS



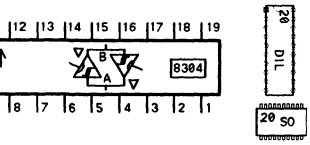
● **74x245** 74ALS245-1
LS ALS AS F HC HCT AC ACT AC11 BCT



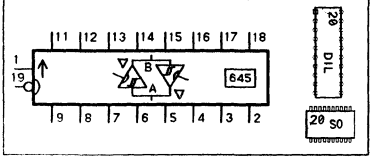
○ **74x1245** 74ALS1245-1
ALS



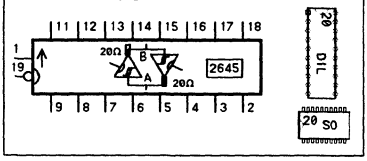
† **8304** NS
S



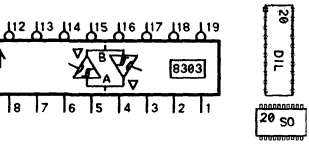
● **74x645** 1645 low power
LS ALS AS HC HCT



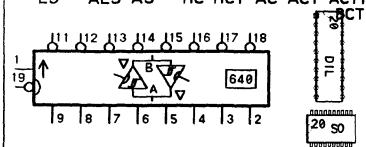
□ **74x2645**
AS



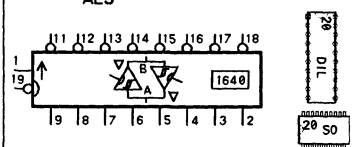
† **8303** NS
S



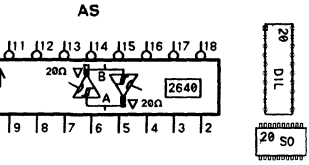
● **74x640** 74ALS640-1
LS ALS AS HC HCT AC ACT AC11 BCT



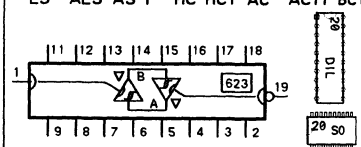
● **74x1640** 74ALS1640-1
ALS



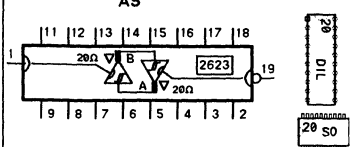
□ **74x2640**
AS



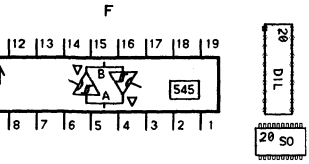
■ **74x623** 74ALS623-1
LS ALS AS F HC HCT AC AC11 BCT



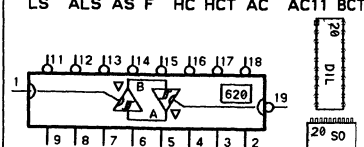
■ **74x2623**
AS



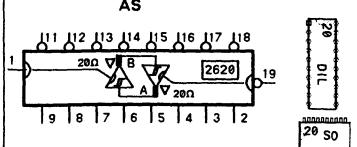
□ **74x545**
F



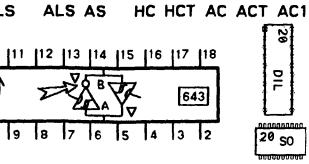
■ **74x620** 74ALS620-1
LS ALS AS F HC HCT AC AC11 BCT

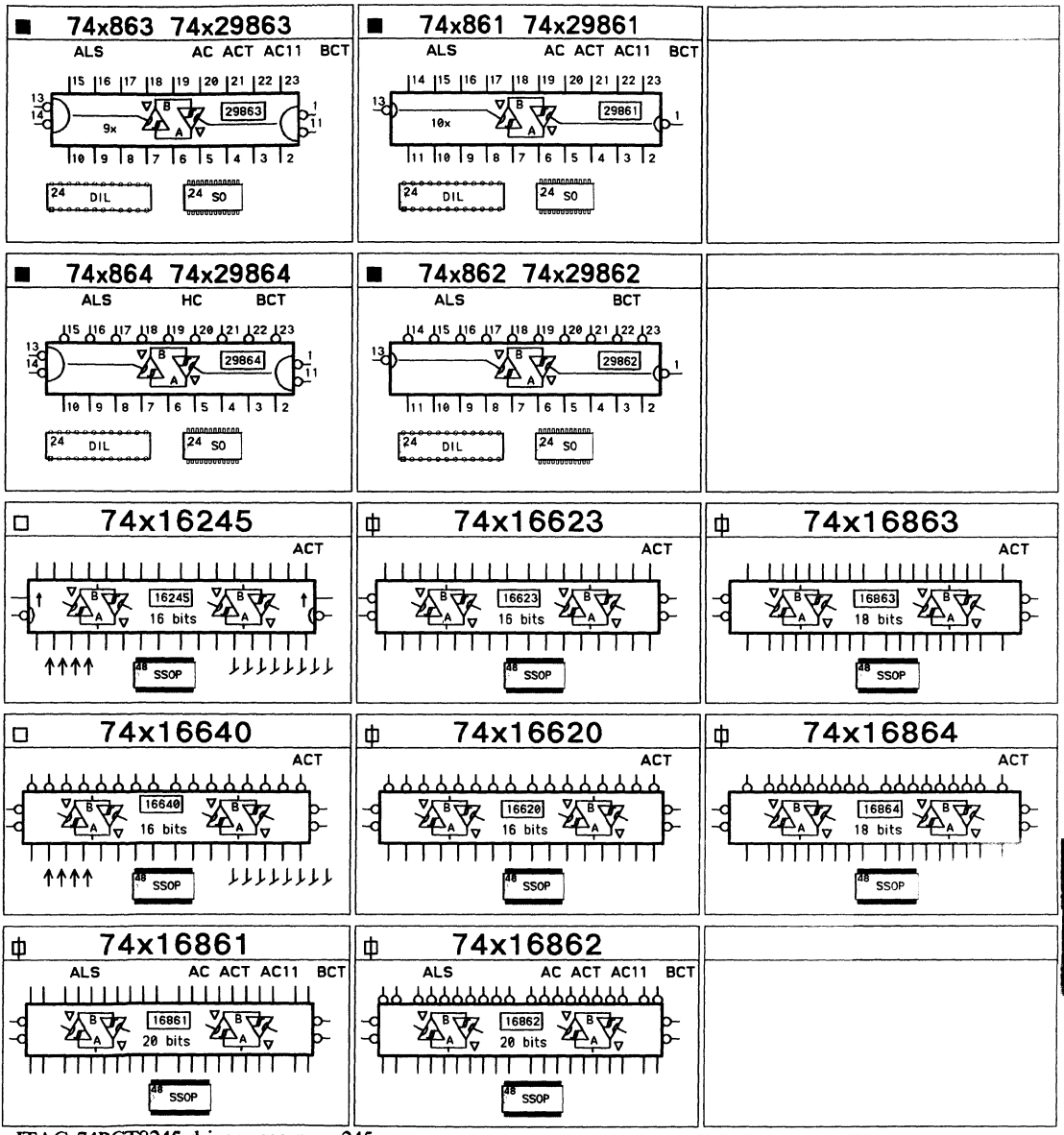


□ **74x2620**
AS



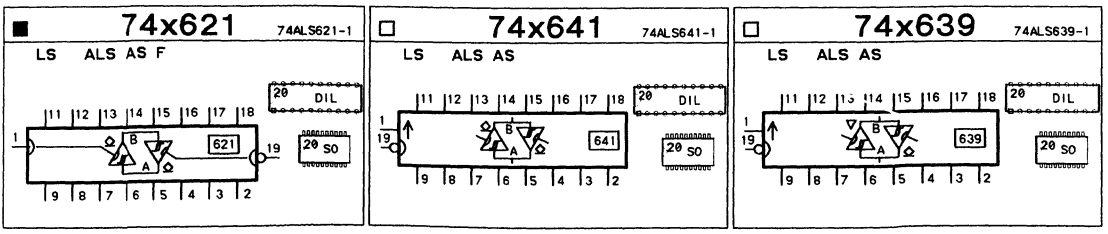
■ **74x643** 74ALS643-1
LS ALS AS HC HCT AC ACT AC11

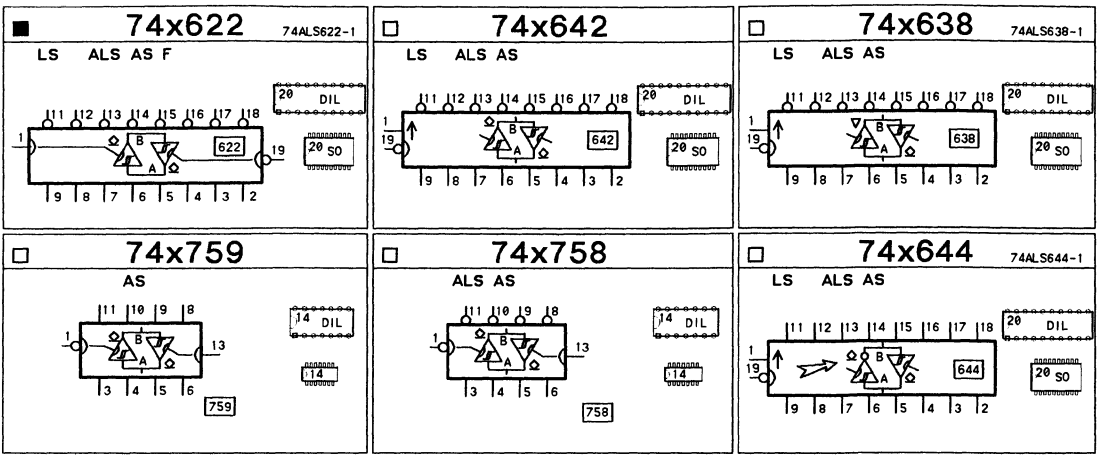




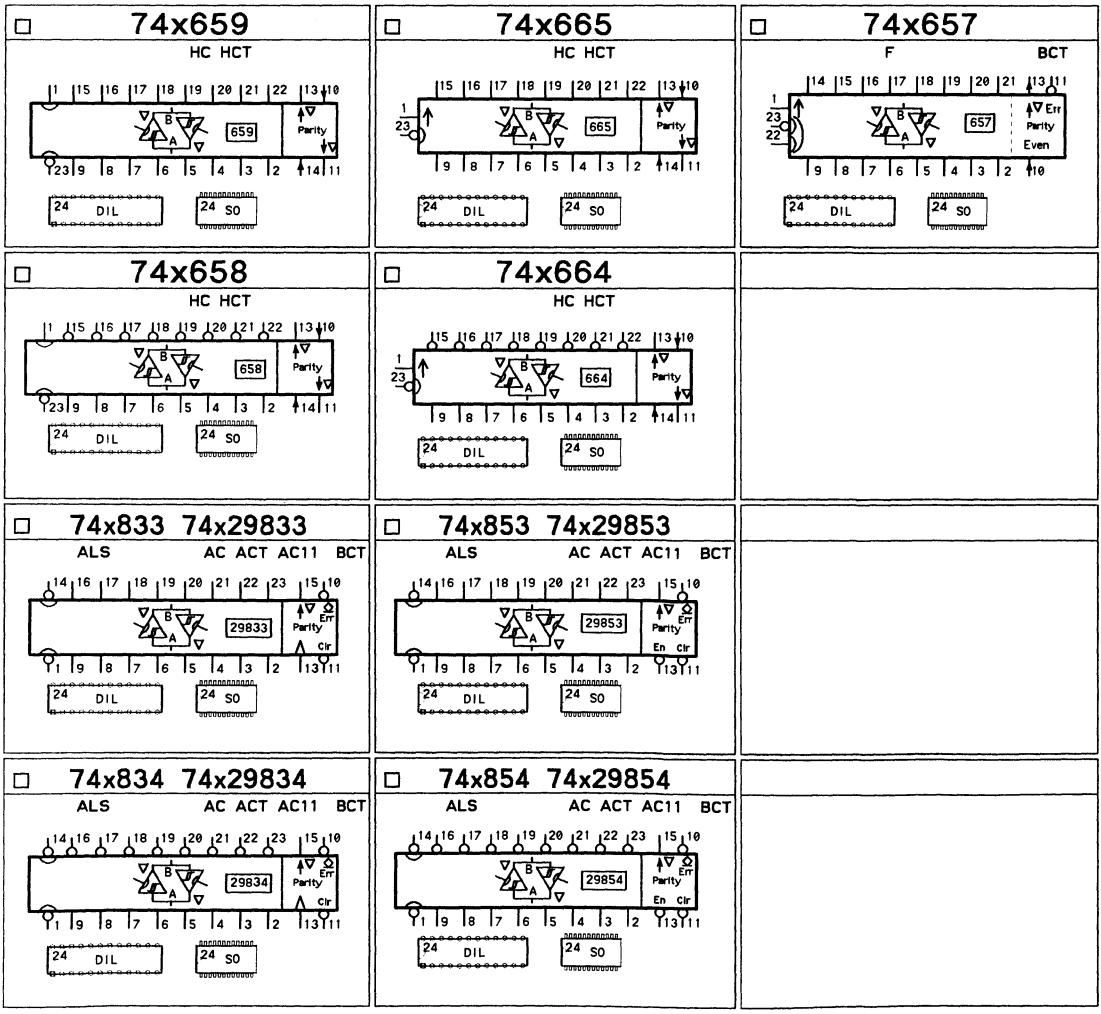
JTAG 74BCT8245 drivers, see page 245

Bidirectionnal open collector drivers





Bidirectionnels drivers with parity

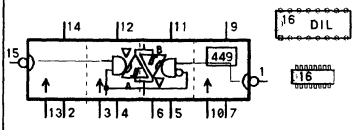


Bidir. drivers

GPIB/IEEE488 drivers

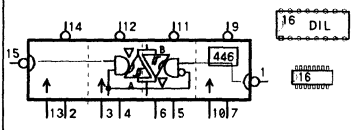
○ 74x449

LS



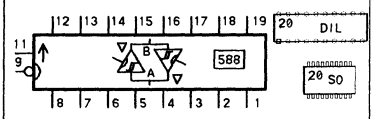
○ 74x446

LS



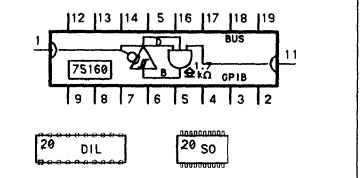
□ 74x588

F



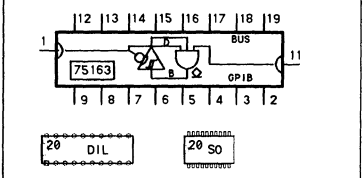
■ 75160

LS



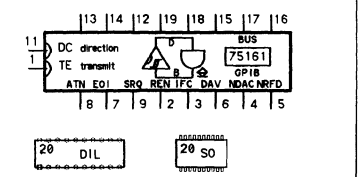
■ 75163

LS



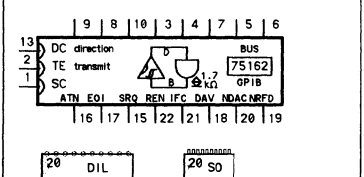
■ 75161

LS



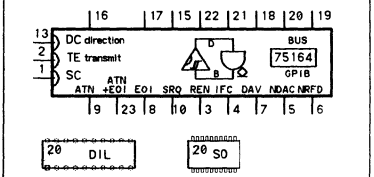
■ 75162

LS



■ 75164

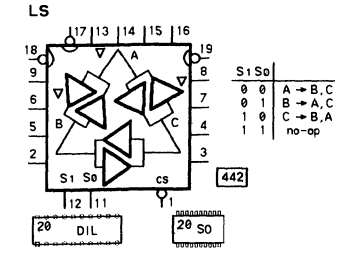
LS



Tridirectionnal drivers

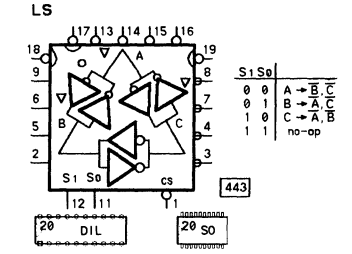
○ 74x442

LS



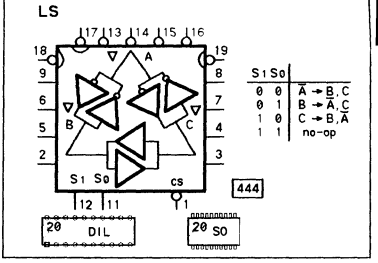
† 74x443

LS



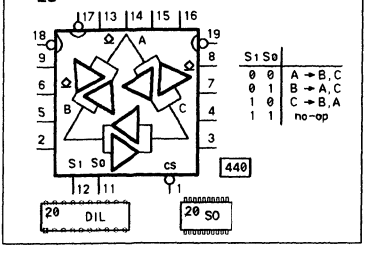
○ 74x444

LS



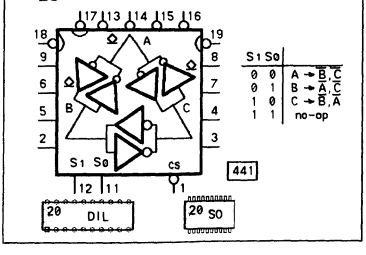
○ 74x440

LS



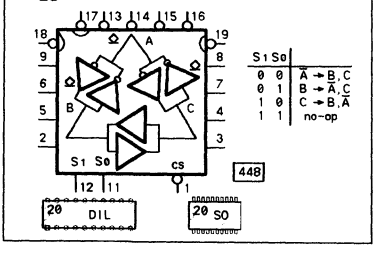
○ 74x441

LS



† 74x448

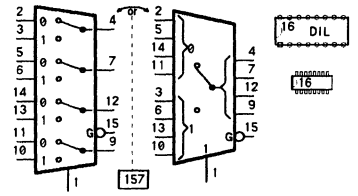
LS



Multiplexers

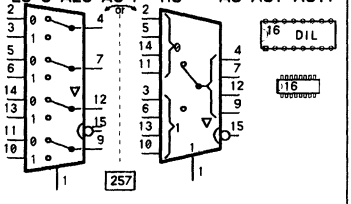
● 74x157

N LS S ALS AS F HC AC ACT AC11



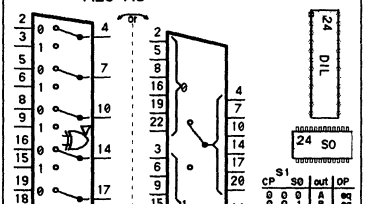
● 74x257

LS S ALS AS F HC AC ACT AC11



■ 74x857

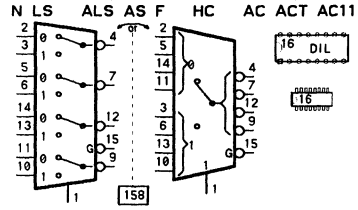
ALS AS



CP	S1	S0	out	OP
0	0	0	A	0
0	0	1	B	0
0	1	0	0	0
0	1	1	0	0
1	0	0	A	1
1	0	1	B	1
1	1	0	0	1
1	1	1	Z	1

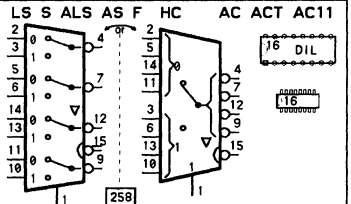
● 74x158

N LS ALS AS F HC AC ACT AC11



● 74x258

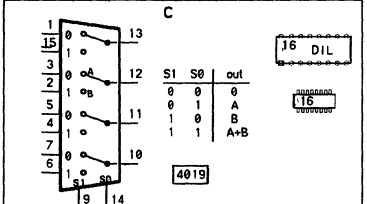
LS S ALS AS F HC AC ACT AC11



⌀ 4019

Toshiba

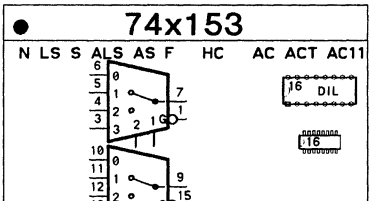
C



S1	S0	out
0	0	0
0	1	A
1	0	B
1	1	A+B

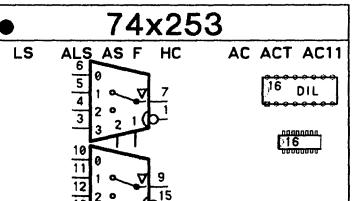
● 74x153

N LS S ALS AS F HC AC ACT AC11



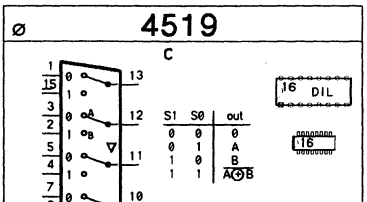
● 74x253

LS ALS AS F HC AC ACT AC11



⌀ 4519

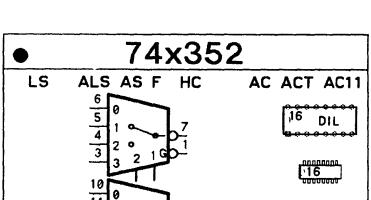
C



S1	S0	out
0	0	0
0	1	A
1	0	B
1	1	A⊕B

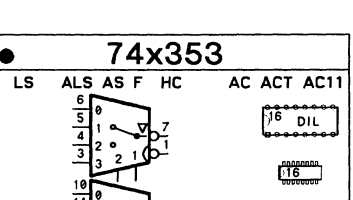
● 74x352

LS ALS AS F HC AC ACT AC11



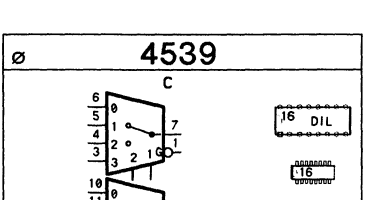
● 74x353

LS ALS AS F HC AC ACT AC11



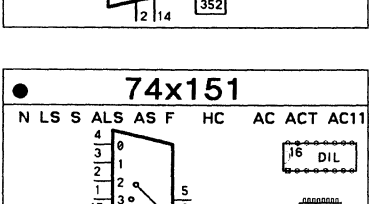
⌀ 4539

C



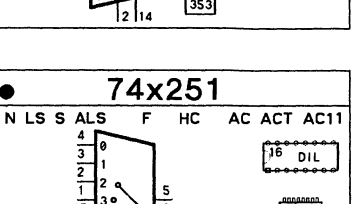
● 74x151

N LS S ALS AS F HC AC ACT AC11



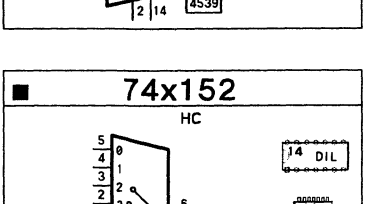
● 74x251

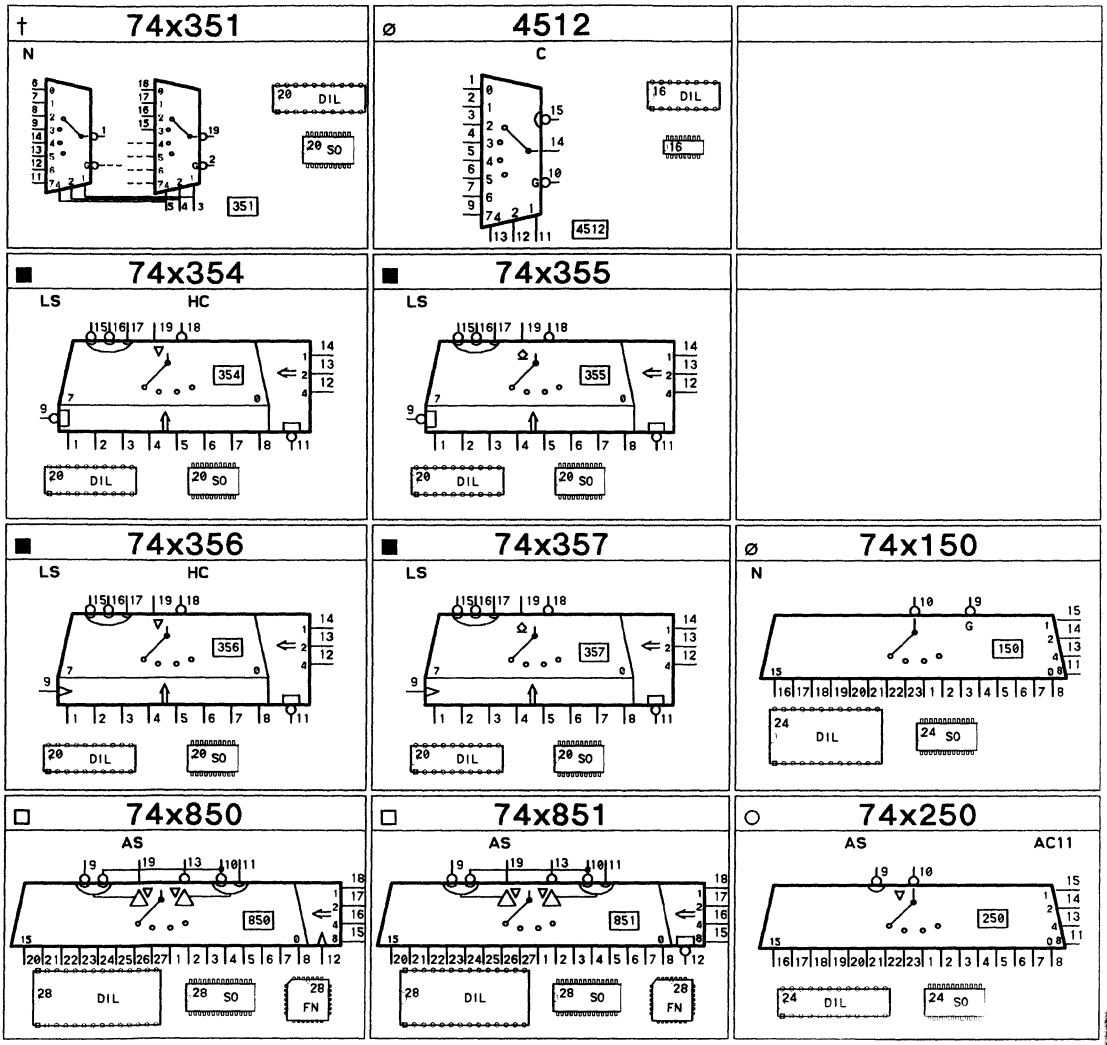
N LS S ALS F HC AC ACT AC11



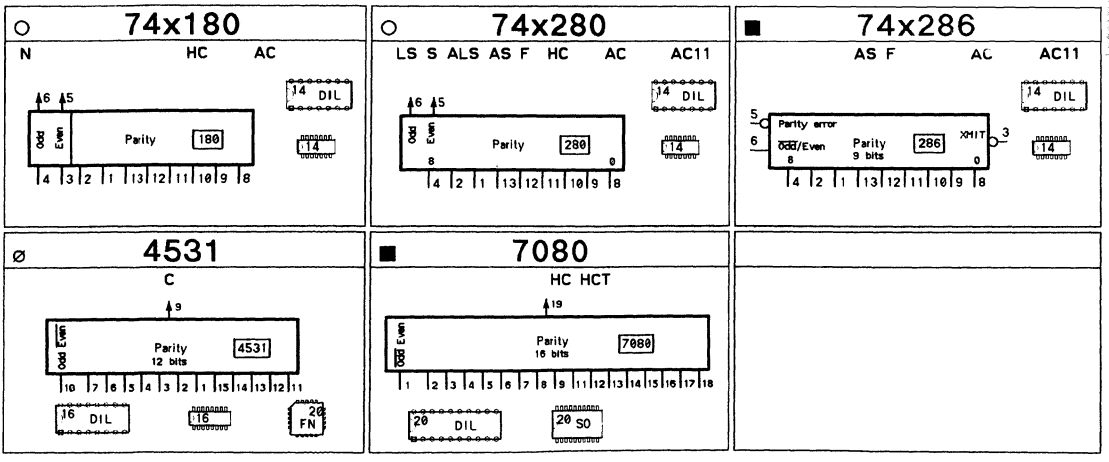
■ 74x152

HC





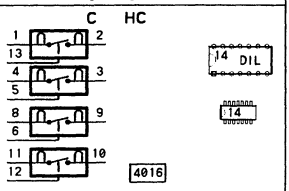
Parity generators (drivers with parity pages 255, 258)



Analog multiplexers

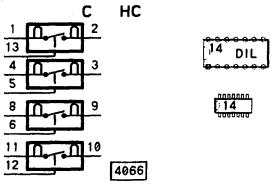
† **4016**

C HC



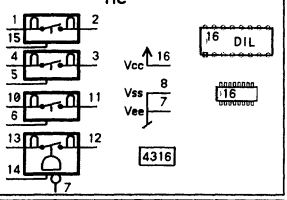
○ **4066**

C HC



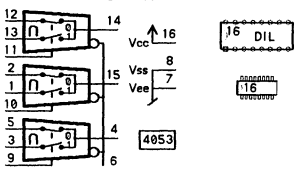
○ **4316**

HC



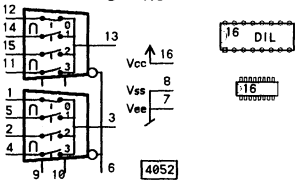
○ **4053**

C HC



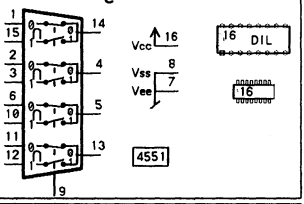
○ **4052**

C HC



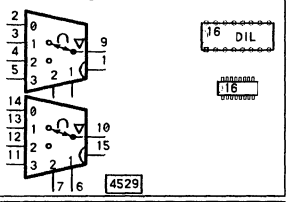
○ **4551**

C



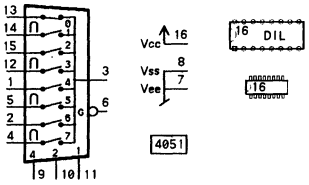
⌀ **4529**

C



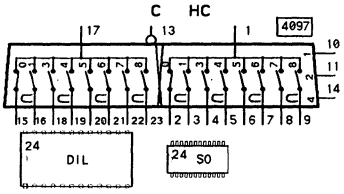
○ **4051**

C HC



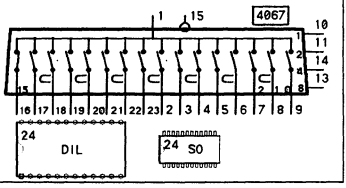
⌀ **4097**

C HC



⌀ **4067**

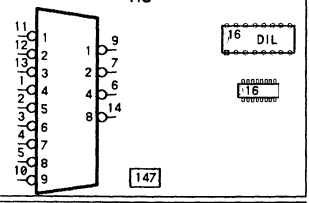
C HC



Priority encoders

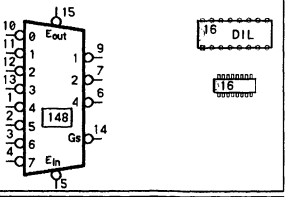
○ **74x147**

N LS HC



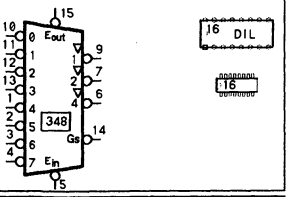
● **74x148**

N LS F HC



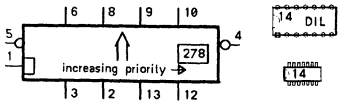
● **74x348**

LS



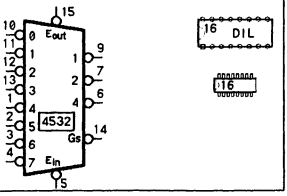
⌀ **74x278**

N



○ **4532**

C

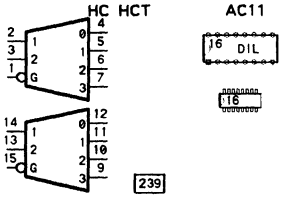


Multiplexers

Decoders

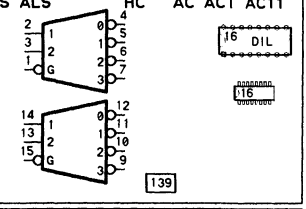
■ 74x239

HC HCT AC11



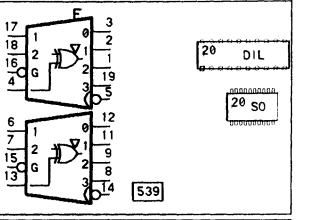
● 74x139

LS S ALS HC AC ACT AC11



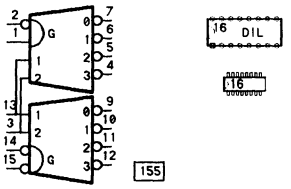
■ 74x539 25LS2539

F



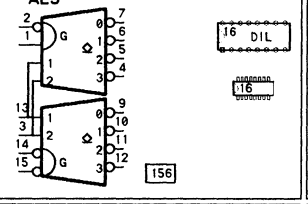
⌀ 74x155

N LS ALS



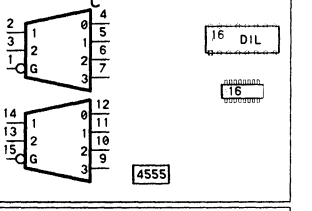
⌀ 74x156

N LS ALS



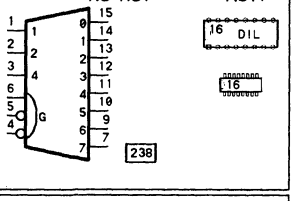
⌀ 4555

C



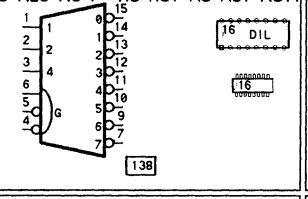
■ 74x238

HC HCT AC11



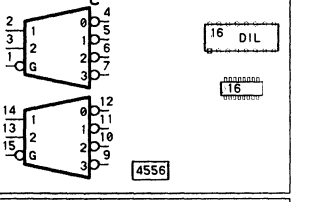
● 74x138

LS S ALS AS F HC HCT AC ACT AC11



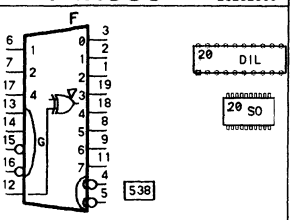
⌀ 4556

C



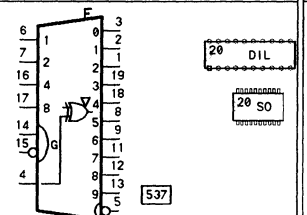
■ 74x538 25LS2538

F



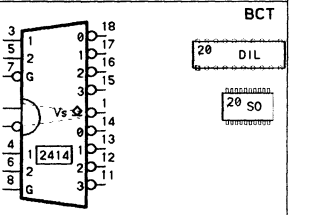
■ 74x537 25LS2537

F



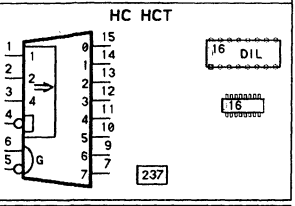
□ 74x2414

BCT



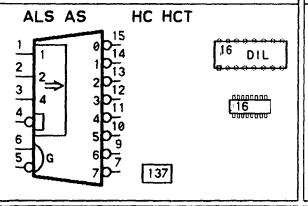
■ 74x237

HC HCT



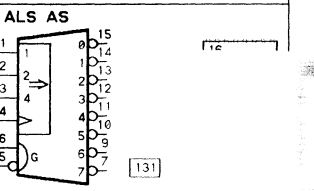
■ 74x137

LS ALS AS HC HCT



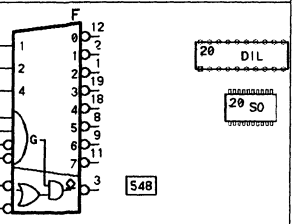
■ 74x131

ALS AS



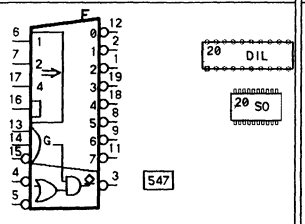
■ 74x548 z8148 F

F



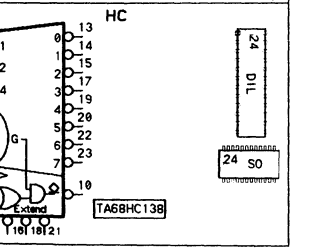
■ 74x547 F

F



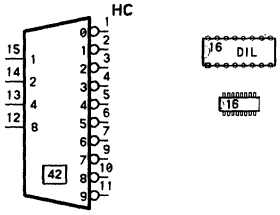
□ 68HC138 Texas Array

HC



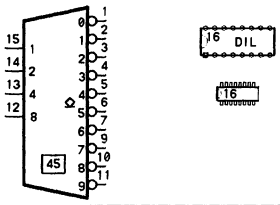
● 74x042

N LS HC



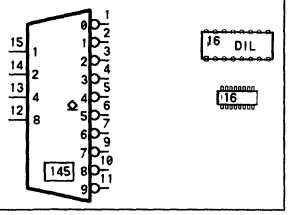
○ 74x045 80mA 30V

N



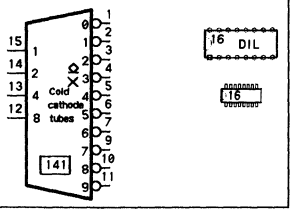
○ 74x145 80mA 15V

N LS



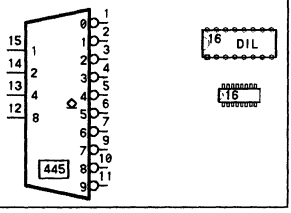
⌀ 74x141 60V

N



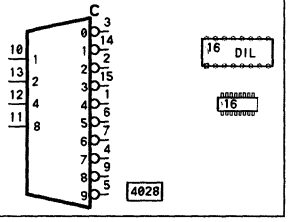
○ 74x445 80mA 7V

LS



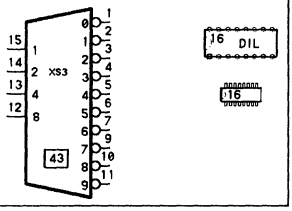
⌀ 4028

C



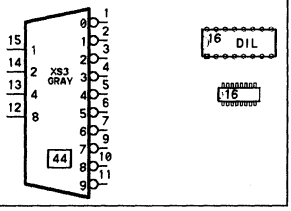
† 74x043 XS3

N



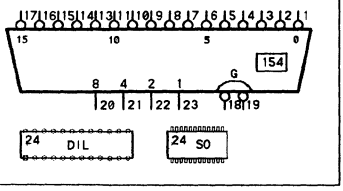
† 74x044 Gray

N



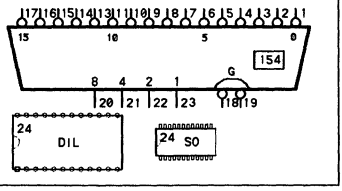
● 74x154

ALS HC



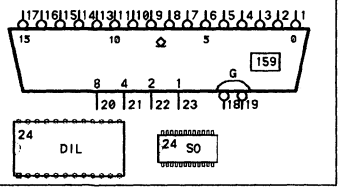
† 74154

N



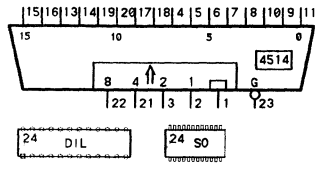
⌀ 74x159

N



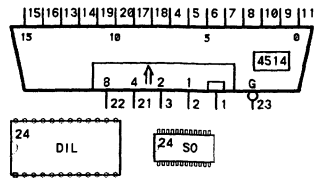
○ 4514

HC



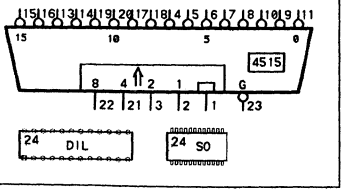
† 4514

C



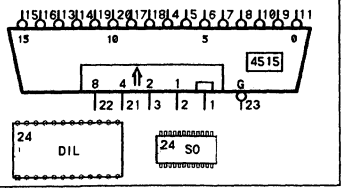
○ 4515

HC



† 4515

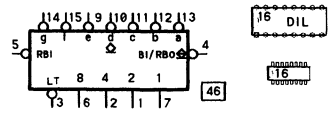
C



7-segment decoders

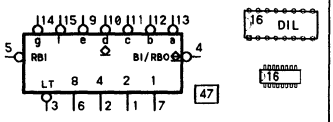
∅ 74x046 30V

N



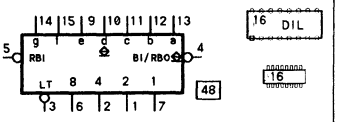
● 74x047 15V

N LS



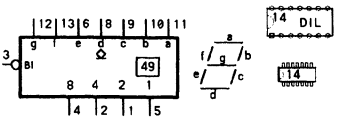
● 74x048

N LS C



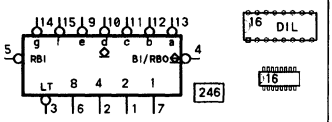
○ 74x049

LS



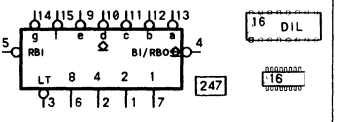
† 74x246 30V

N



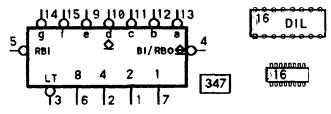
○ 74x247 15V

N LS



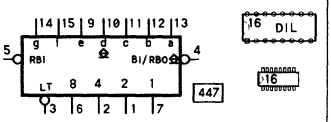
† 74x347

LS



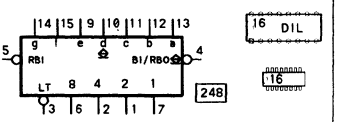
† 74x447

LS



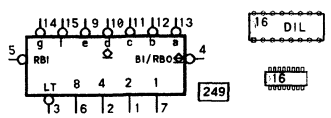
○ 74x248

LS



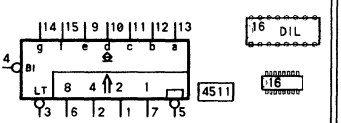
† 74x249

N



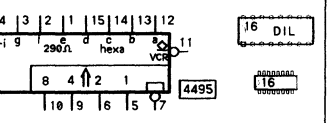
○ 4511

C HC



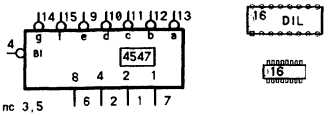
○ 4495

C



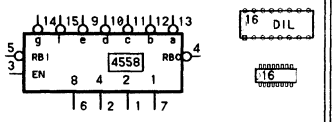
† 4547

C



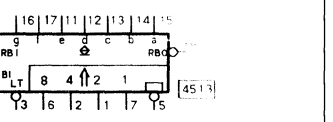
† 4558

C



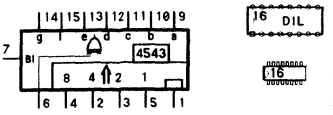
† 4513

C



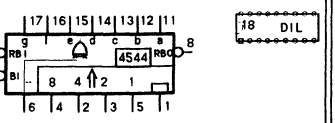
○ 4543

C



† 4544

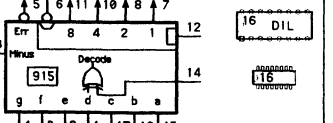
C



† 915

NS

C

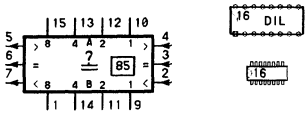


Counters with 7-segments decoder, see page 285

Comparators

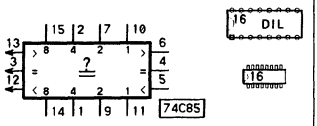
○ 085

N LS S F HC



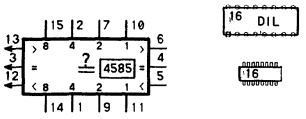
† 74C85

C



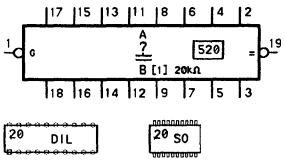
† 4585

C



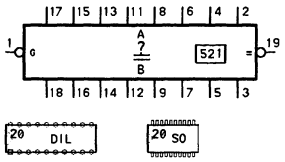
■ 520

ALS F AC ACT AC11



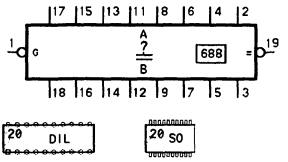
■ 521

ALS F AC ACT AC11



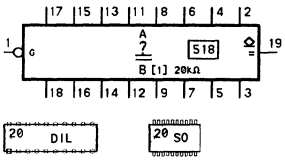
■ 688

LS ALS HC HCT



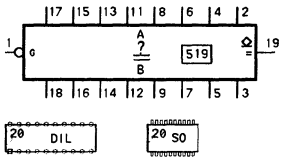
□ 518

ALS F



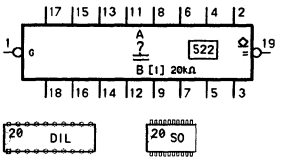
□ 519

ALS F



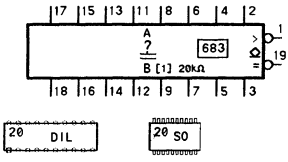
□ 522

ALS



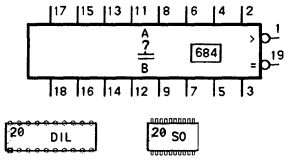
□ 683

LS



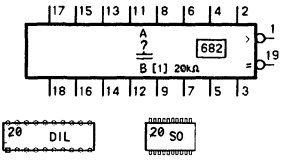
□ 684

LS HC



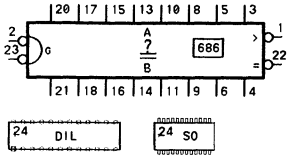
□ 682

LS HC



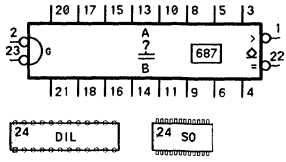
■ 686

LS HC



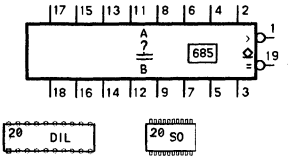
□ 687

LS



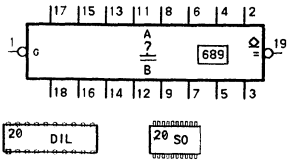
□ 685

LS



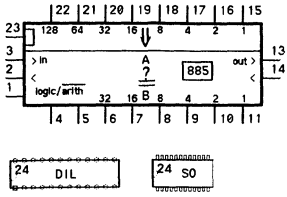
□ 689

LS ALS



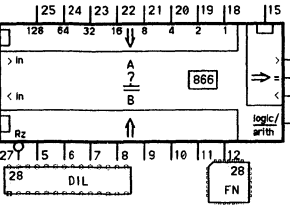
□ 885

AS

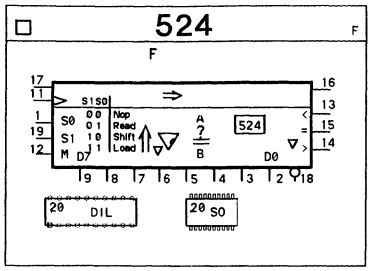


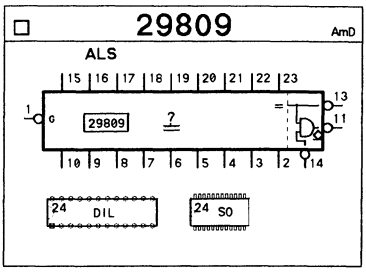
□ 866

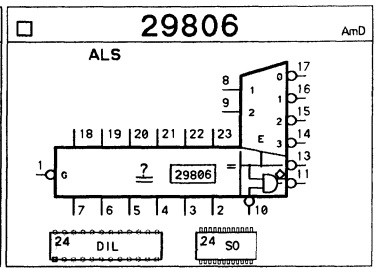
AS



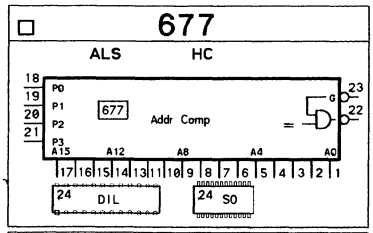
Arithmetic

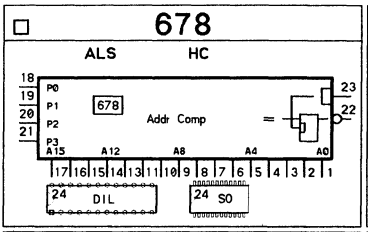


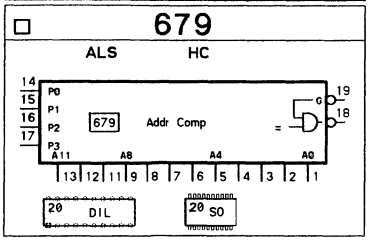


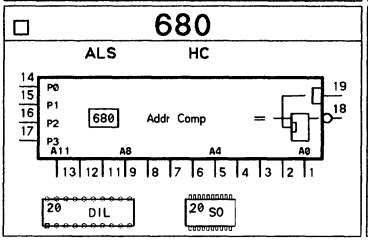


Address comparators

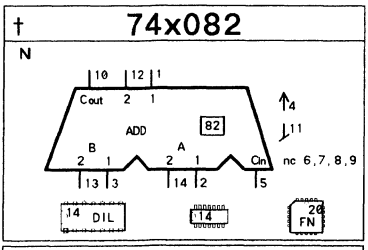


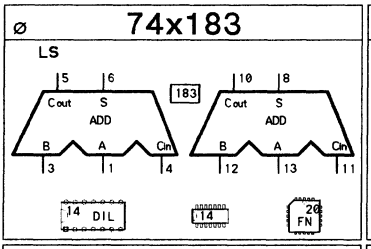


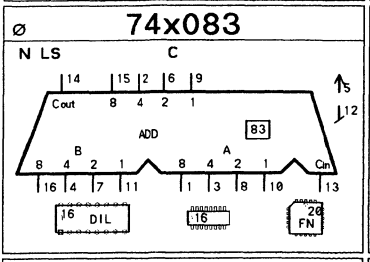


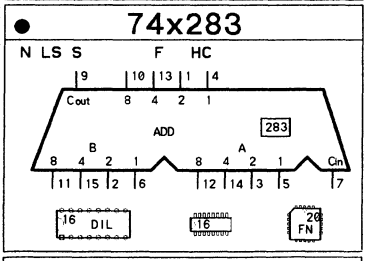


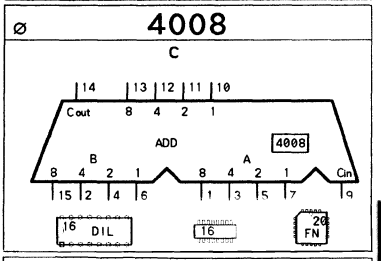
Arithmetic units

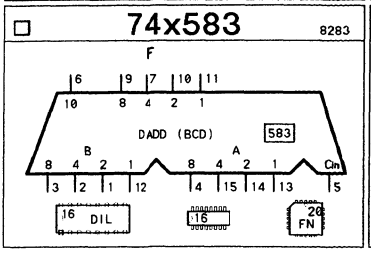


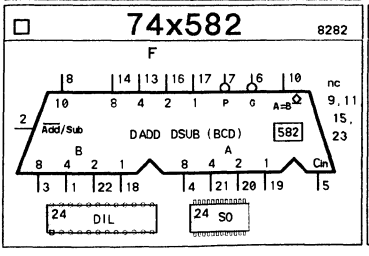


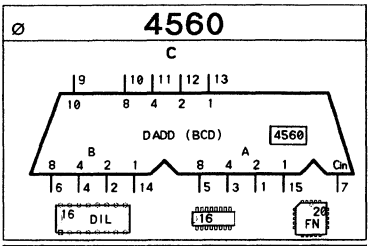


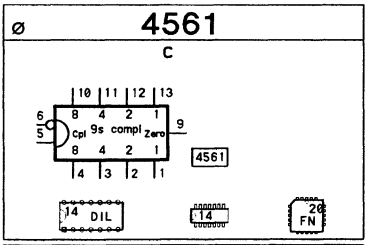


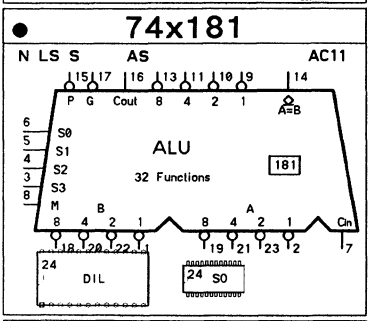


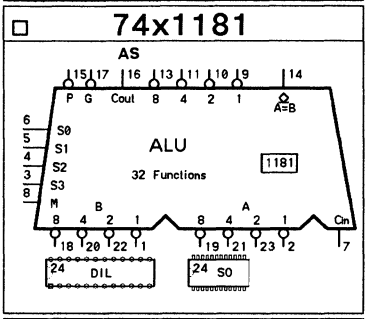


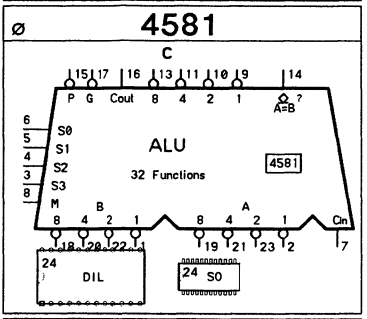


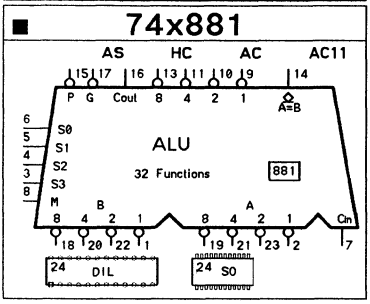


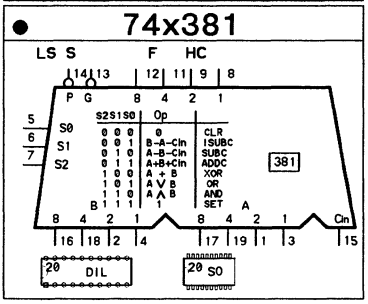


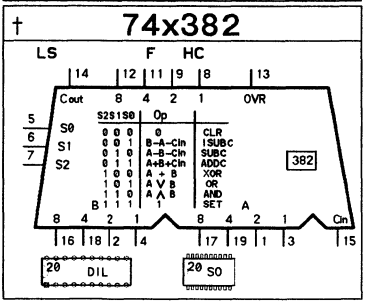




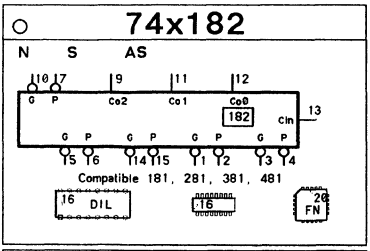


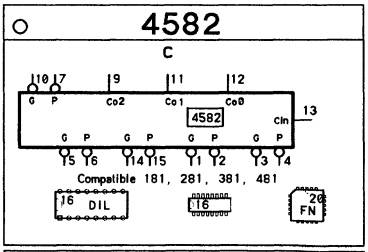


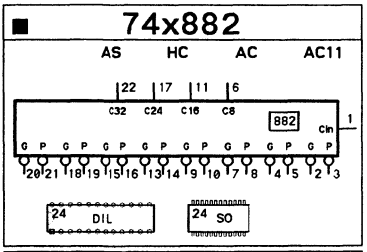


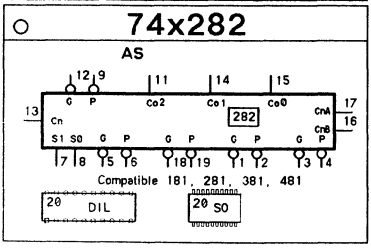


Carry look ahead









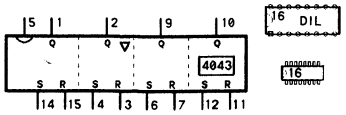
Arithmetic

ALU with accumulator: 74LS681, 74S281, 74S481, 74S482, 2901, 29000
Serial Adders: 74LS385, 74F385, 14032, 14038
Serial multipliers: 74F784, 74LS384
Parallel multipliers: 74LS261, S274, 284, 285, 275
Shifters: 74F350, 74LS681
Code converters: 74484,74485

RS-type flip-flops

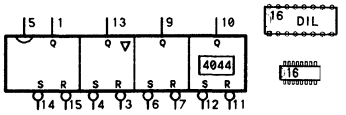
ø 4043

C



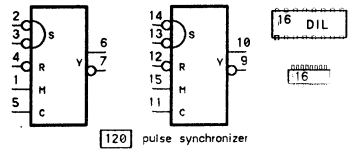
ø 4044

C



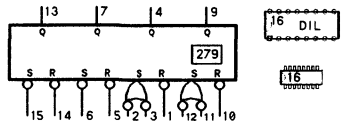
ø 74x120

N



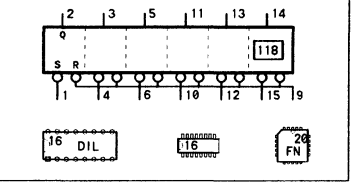
ø 74x279

N LS



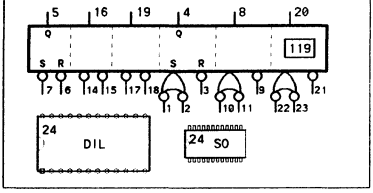
ø 74x118

N



ø 74x119

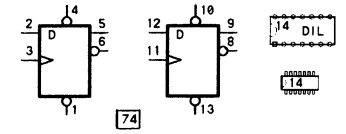
N



D-type flip-flops

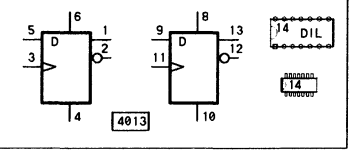
● 74x074

N LS S ALS AS F C HC HCT AC ACT AC11



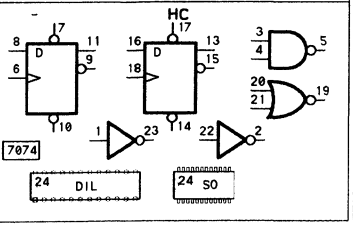
○ 4013

C



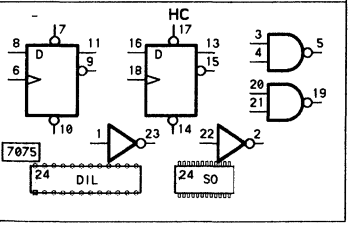
■ 74x7074

- HC



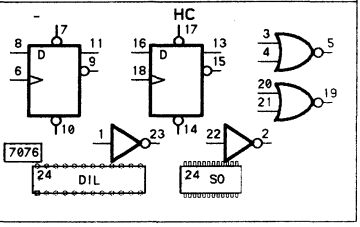
● 74x7075

- HC



● 74x7076

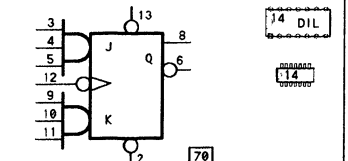
- HC



JK-type flip-flops

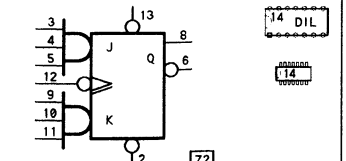
† 74x070

N



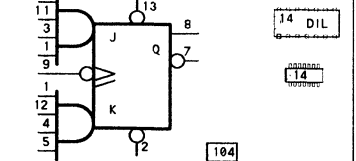
† 74x072

N



† 74x104

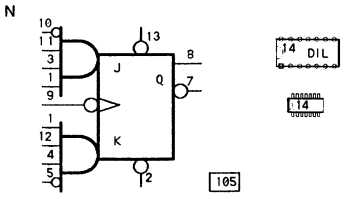
N



Flip-flops

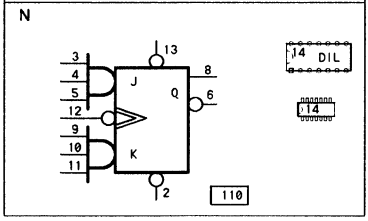
† **74x105**

N



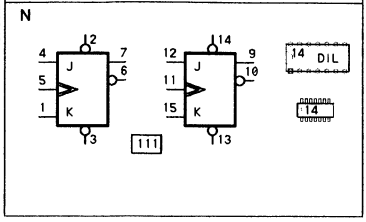
† **74x110**

N



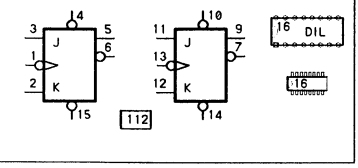
ø **74x111**

N



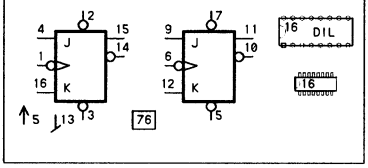
● **74x112**

LS S ALS F HC AC AC11



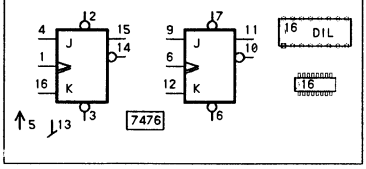
○ **74x076**

LS HC



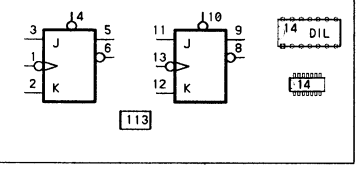
† **7476**

N



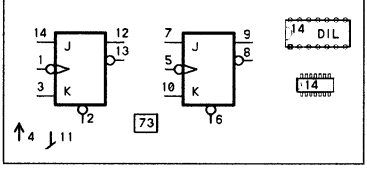
● **74x113**

LS S ALS F HC



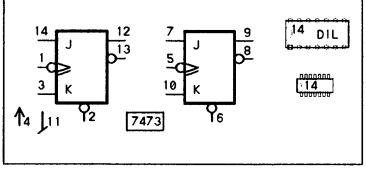
○ **74x073**

LS HC HCT



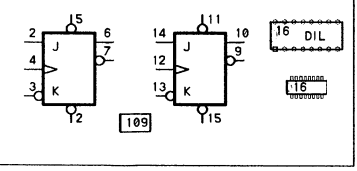
† **7473**

N



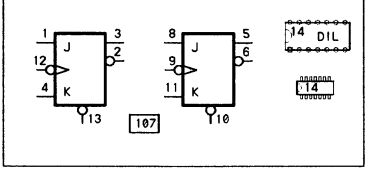
● **74x109**

N LS ALS AS F HC AC ACT AC11



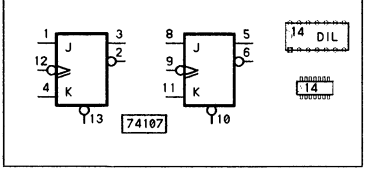
○ **74x107**

LS HC



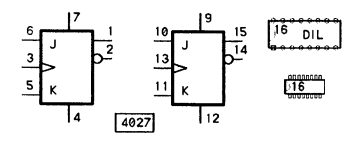
† **74107**

N



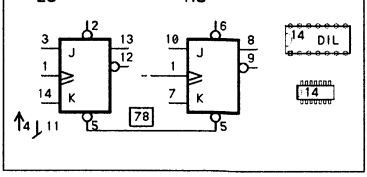
○ **4027**

C



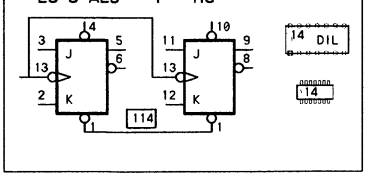
○ **74x078**

LS HC



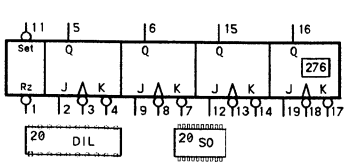
● **74x114**

LS S ALS F HC



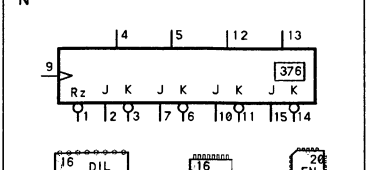
ø **74x276**

N



ø **74x376**

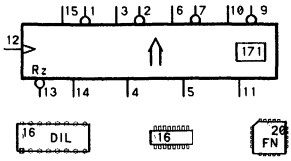
N



Parallel registers (latches page 273)

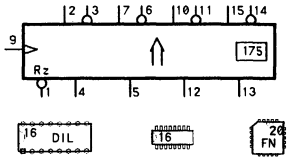
† 74x171

LS



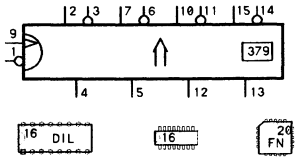
● 74x175

N LS S ALS AS F HC AC ACT AC11



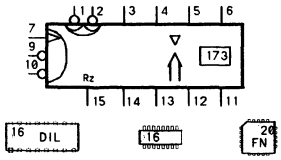
● 74x379

LS F HC AC ACT AC11



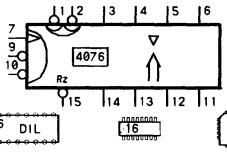
○ 74x173

N LS HC



⌀ 4076

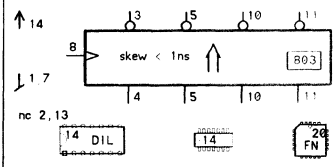
C



□ 74x803

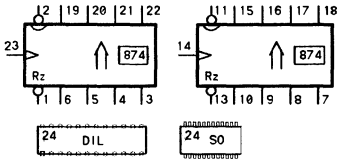
Motorola

F



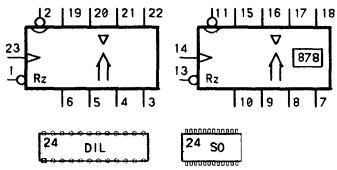
■ 74x874

ALS AS AC AC11



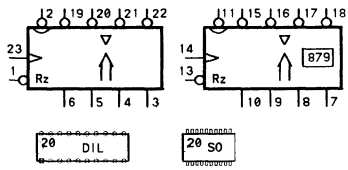
■ 74x878

ALS AS



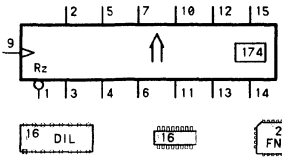
■ 74x879

ALS AS



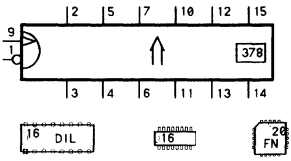
● 74x174

N LS S ALS AS F HC AC ACT AC11



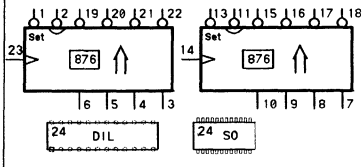
○ 74x378

LS F HC AC ACT AC11



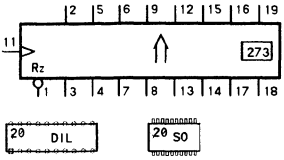
□ 74x876

ALS AS



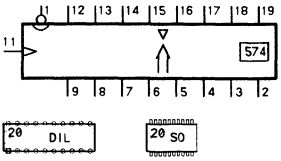
● 74x273

N LS ALS F HC AC ACT



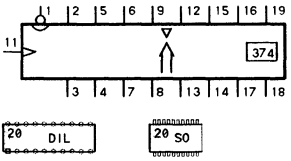
● 74x574

ALS AS F HC HCT AC ACT BCT



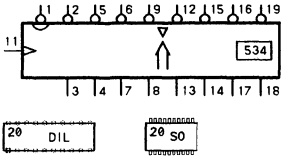
● 74x374

LS S ALS AS F HC HCT AC ACT AC11 BCT



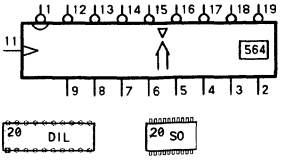
● 74x534

ALS AS F HC HCT AC ACT AC11 BCT



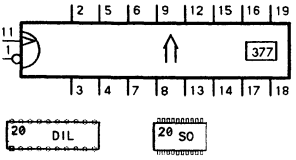
● 74x564

ALS F HC HCT AC ACT BCT



■ 74x377

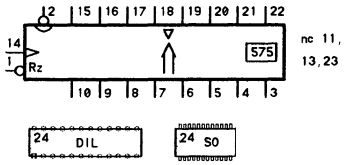
LS F HC AC ACT AC11



JTAG register 74BCT8374, see page 245

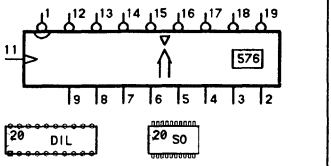
● 74x575

ALS AS



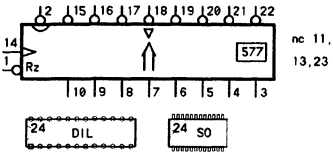
● 74x576

ALS AS



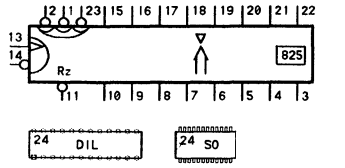
● 74x577

ALS AS



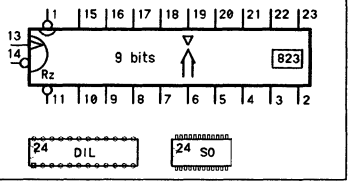
■ 74x825 74x29825

ALS AS F AC ACT AC11



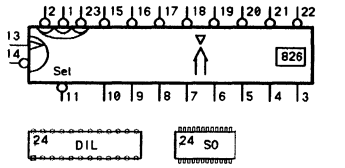
■ 74x823 74x29823

ALS AS F AC ACT AC11 BCT



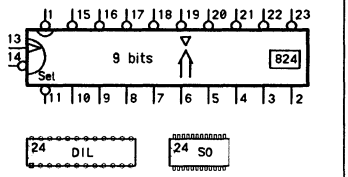
■ 74x826 74x29826

ALS AS AC ACT AC11



■ 74x824 74x29824

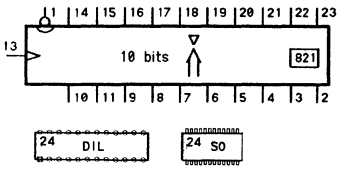
ALS AS AC ACT AC11 BCT



Note: With 74AS826, 74AS824, T.I. document "reset" for pin 11, but inputs are inverted

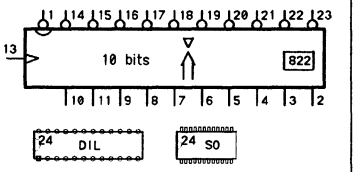
■ 74x821 74x29821

ALS AS F AC ACT AC11 BCT



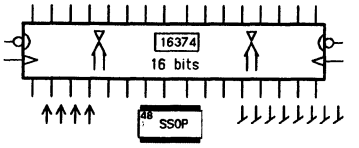
■ 74x822 74x29822

ALS AS AC ACT AC11 BCT



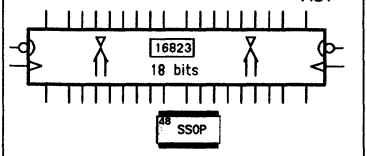
□ 74x16374

ACT



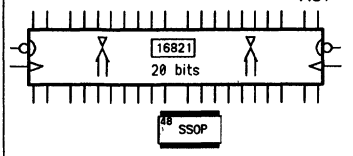
⊕ 74x16823

ACT



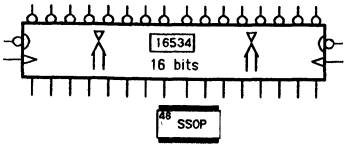
⊕ 74x16821

ACT



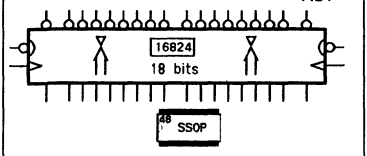
⊕ 74x16534

ACT



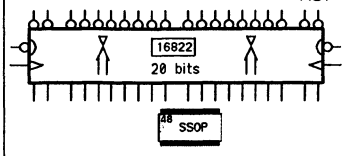
⊕ 74x16824

ACT



⊕ 74x16822

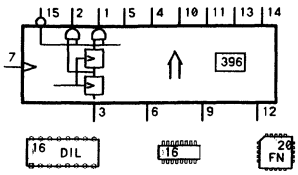
ACT



Dual-rank registers

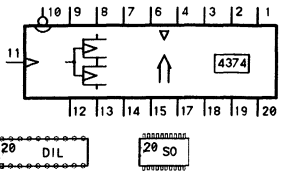
⌀ 74x396

LS



□ 74x4374

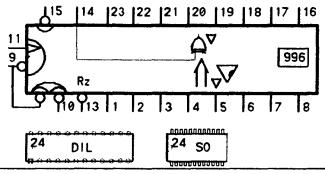
AS



Read-back registers (latches page 265)

□ 74x996 74ALS996-1

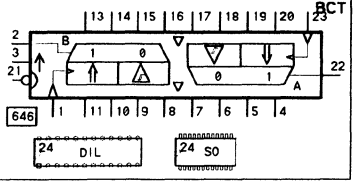
ALS



Bidirectionnal drivers with registers (latches page 267)

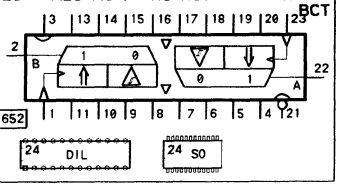
● 74x646 74ALS646-1

LS ALS AS F HC HCT AC ACT AC11 BCT



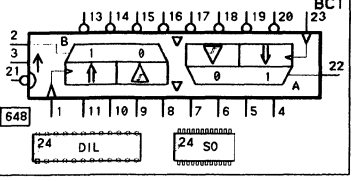
● 74x652 74ALS652-1

LS ALS AS F HC HCT AC11 BCT



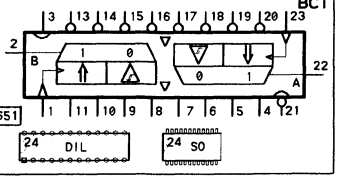
● 74x648 74ALS648-1

LS ALS AS F HC HCT AC AC11 BCT



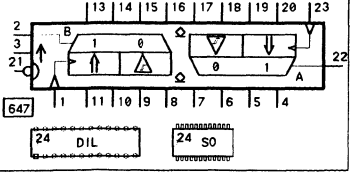
● 74x651 74ALS651-1

LS ALS AS HC HCT AC11 BCT



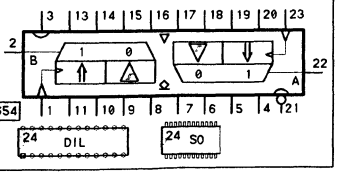
■ 74x647 74ALS647-1

LS ALS



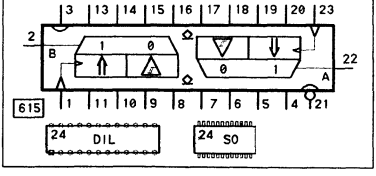
■ 74x654 74ALS654-1

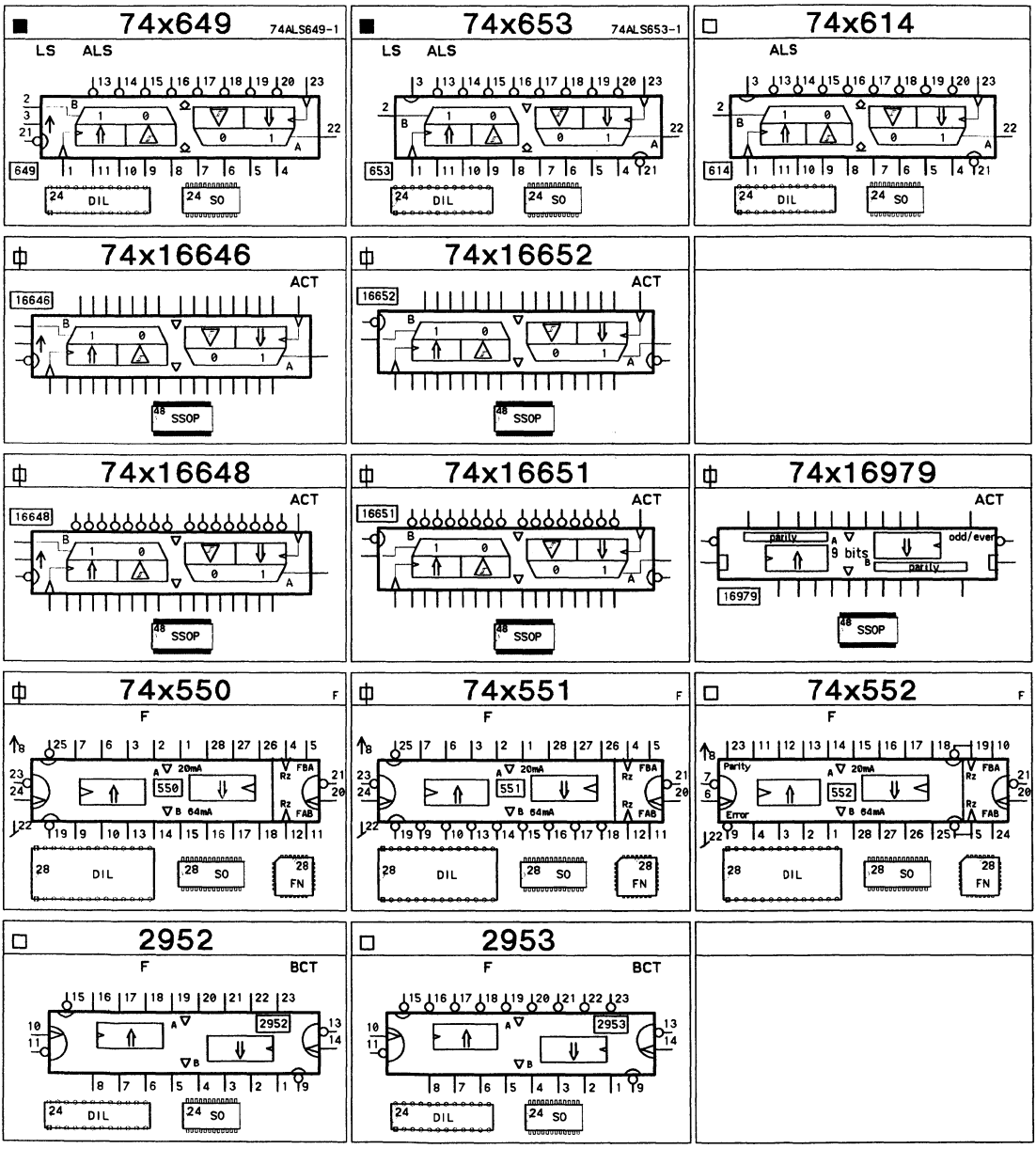
LS ALS



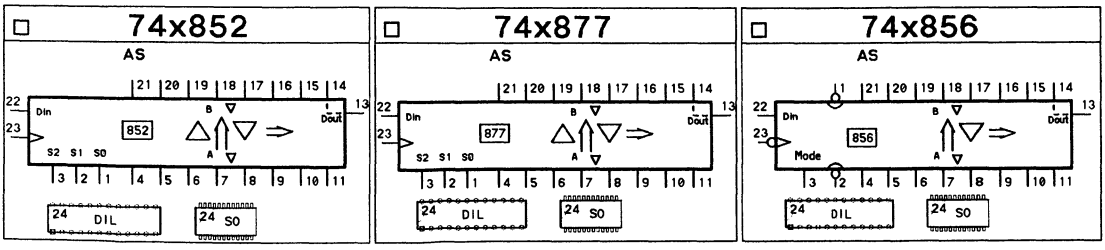
■ 74x615 74ALS615-1

ALS





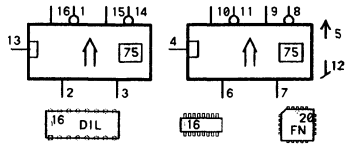
Universal serial-parallel registers



Latches

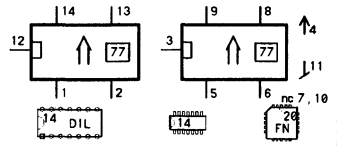
○ 74x075

N LS HC



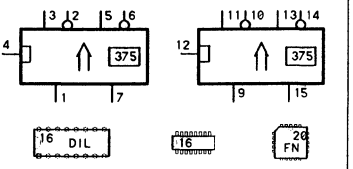
○ 74x077

HC



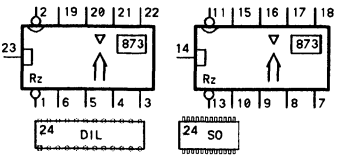
○ 74x375

LS HC -



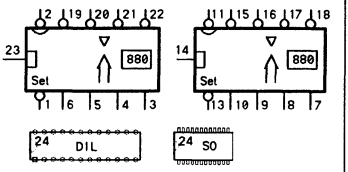
□ 74x873

ALS AS



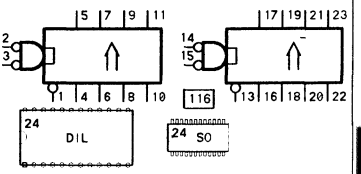
□ 74x880

ALS AS



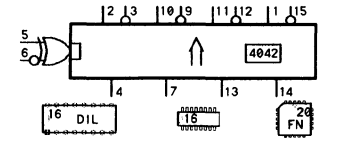
⌀ 74x116

N



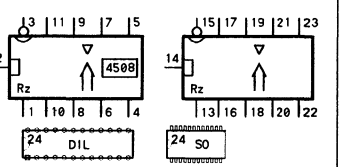
† 4042

C



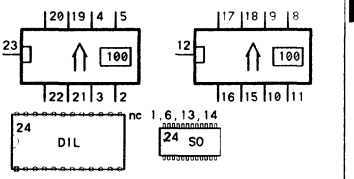
† 4508

C



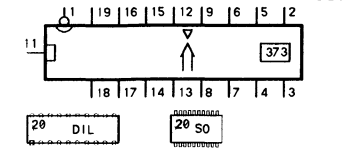
† 74x100

N



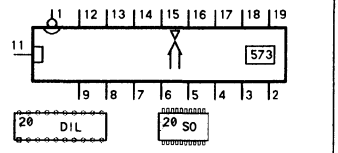
● 74x373

LS S ALS AS F HC HCT AC ACT AC11 BCT



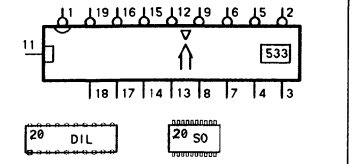
■ 74x573

ALS AS F HC HCT AC ACT BCT



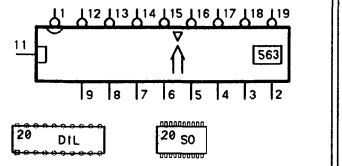
■ 74x533

ALS AS F HC HCT AC ACT AC11 BCT



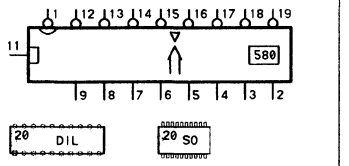
■ 74x563

ALS F HC HCT AC ACT BCT



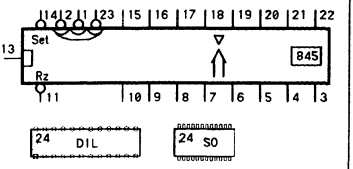
□ 74x580

ALS AS



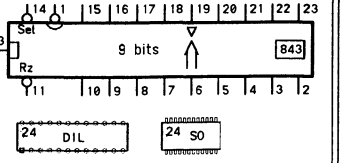
■ 74x845 74x29845

ALS AS F AC ACT AC11 BCT



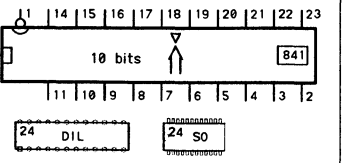
■ 74x843 74x29843

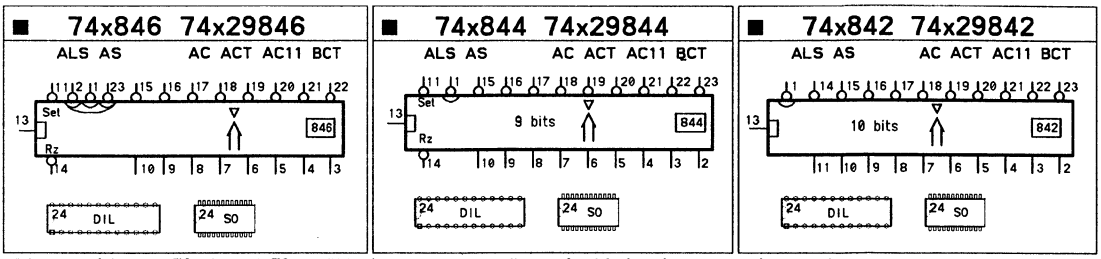
ALS AS F AC ACT AC11 BCT



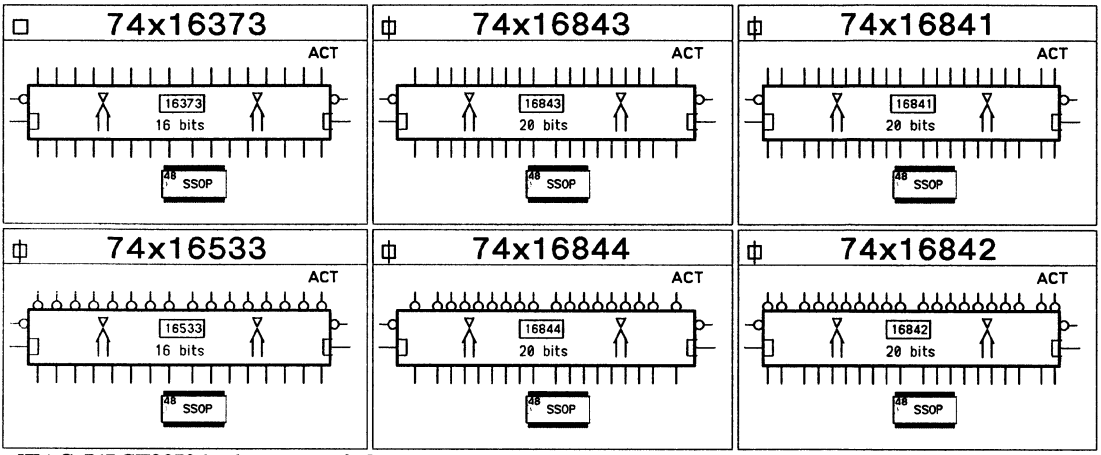
■ 74x841 74x29841

ALS AS F AC ACT AC11 BCT



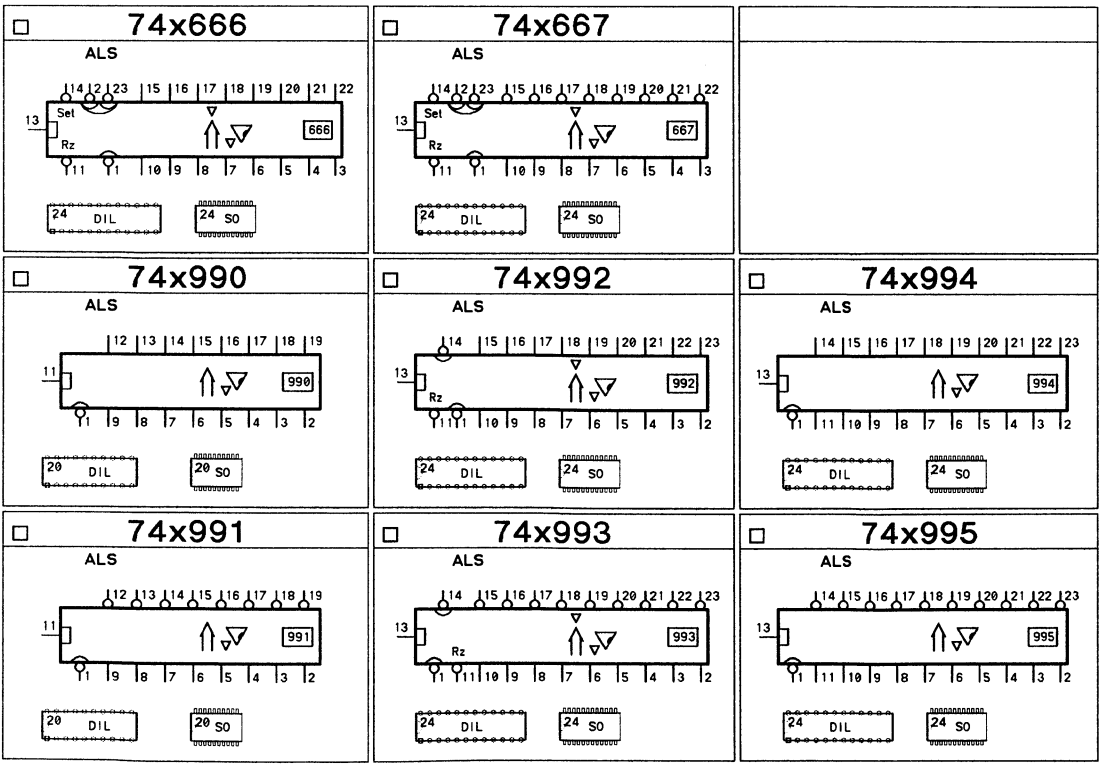


Note: With 74AS846, 74AS844, T.I. document "reset" on pin 11, but inputs are inverted



JTAG 74BCT8373 latch, see page 245

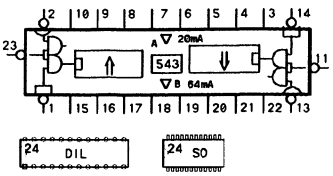
Read back latches



Bidirectionnal latches

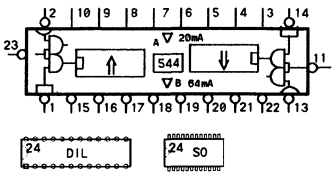
□ 74x543

F BCT



□ 74x544

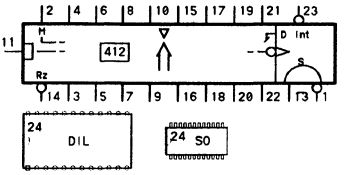
F BCT



Registers with interrupt logic

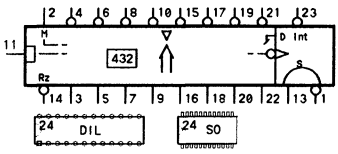
† 74x412 Intel 8212 F

S F



□ 74x432 F

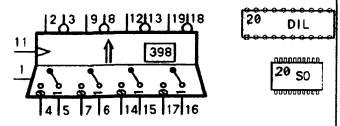
F -



Registers/latches with multiplexer

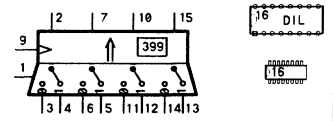
○ 74x398

LS F HC AC ACT



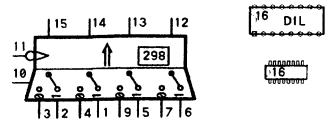
○ 74x399 AMD 26S09

LS F HC AC ACT



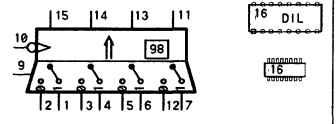
○ 74x298

N LS AS F HC



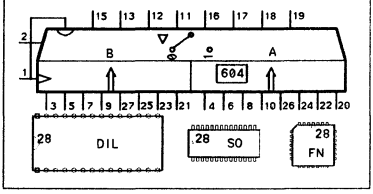
† 74x098

N



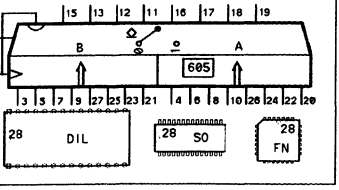
ø 74x604

LS F HC



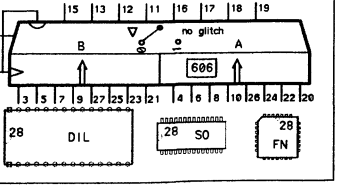
† 74x605

LS



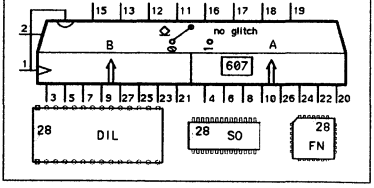
ø 74x606

LS



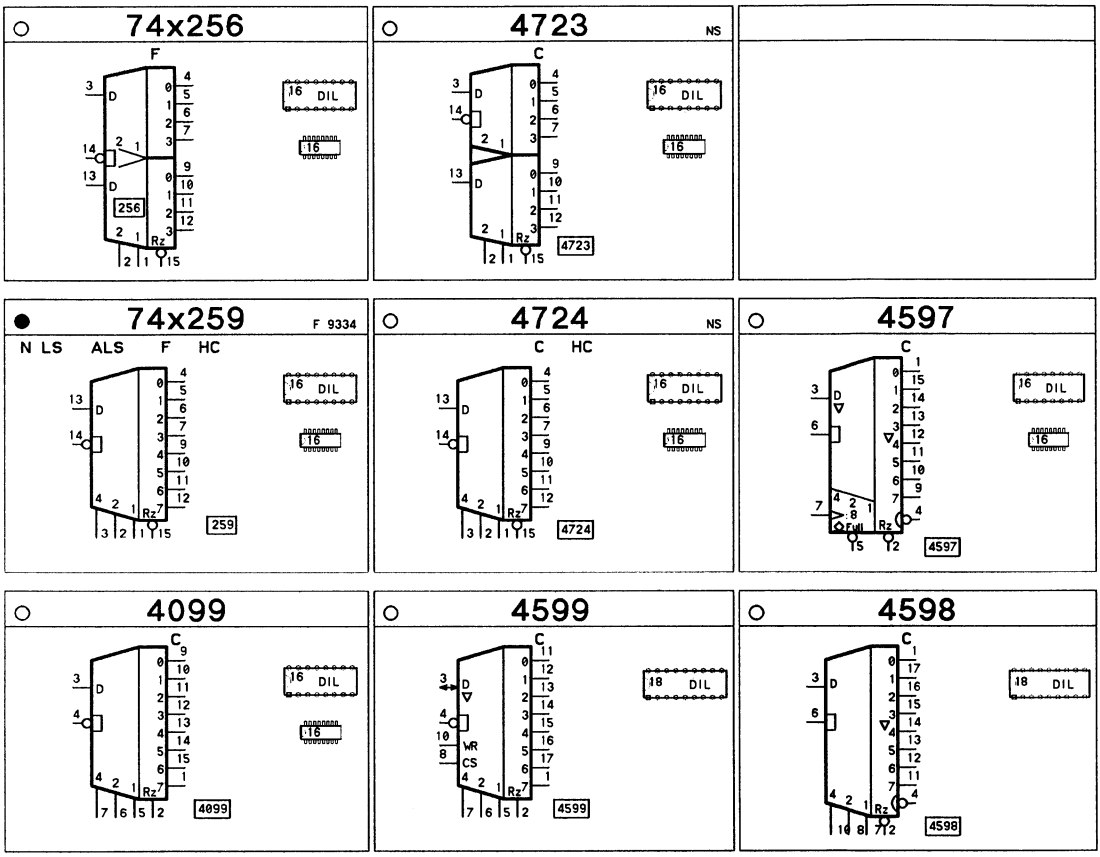
ø 74x607

LS



Latches

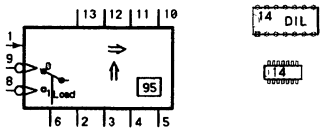
Addressable latches



Serial registers

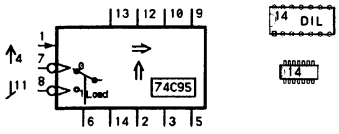
ø 74x095

N LS AS C



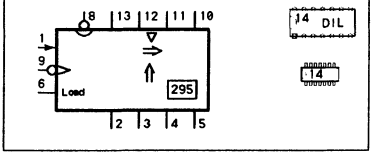
† 74C95

C



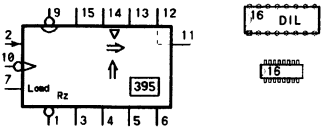
ø 74x295

LS



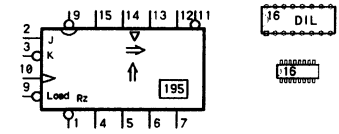
○ 74x395

LS F



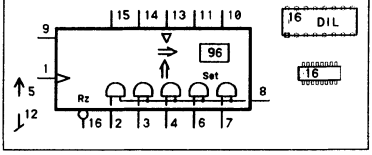
○ 74x195

N LS S AS F HC



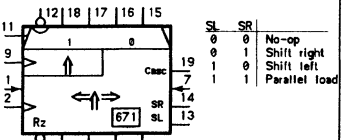
ø 74x096

N LS



□ 74x671

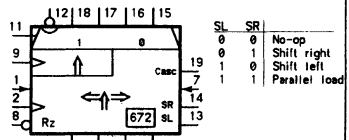
LS



SL	SR	
0	0	No-op
0	1	Shift right
1	0	Shift left
1	1	Parallel load

□ 74x672

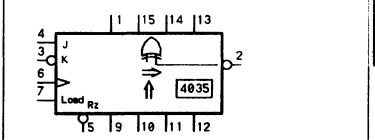
LS



SL	SR	
0	0	No-op
0	1	Shift right
1	0	Shift left
1	1	Parallel load

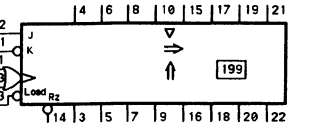
ø 4035

C



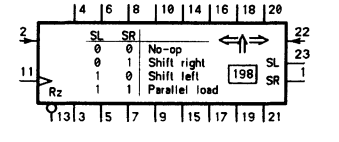
ø 74x199

N



ø 74x198

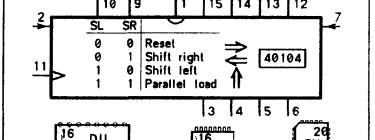
N



SL	SR	
0	0	No-op
0	1	Shift right
1	0	Shift left
1	1	Parallel load

□ 74x40104 RCA Philips

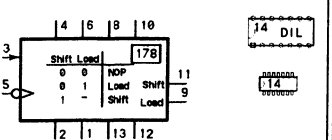
HC



SL	SR	
0	0	Reset
0	1	Shift right
1	0	Shift left
1	1	Parallel load

ø 74x178

N

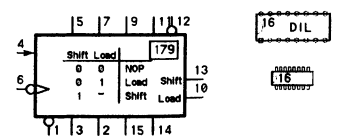


Shift	Load	
0	0	NOP
0	1	Load
1	–	Shift

Shift Load

ø 74x179

N

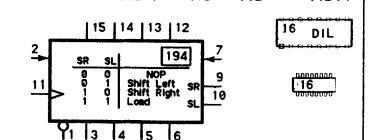


Shift	Load	
0	0	NOP
0	1	Load
1	–	Shift

Shift Load

● 74x194

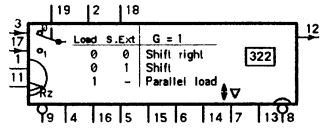
N LS S AS F HC AC AC11



SR	SL	
0	0	NOP
0	1	Shift Left
1	0	Shift Right
1	1	Load

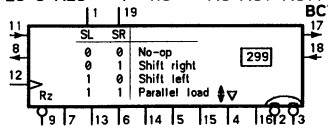
● 74x322

LS F HC



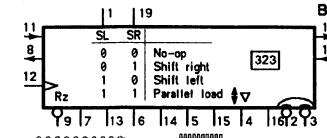
● 74x299

LS S ALS F HC AC ACT AC11 BCT



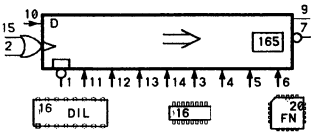
● 74x323

LS ALS F HC AC ACT AC11 BCT



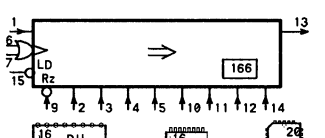
● 74x165

N LS ALS HC



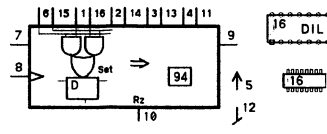
● 74x166

N LS ALS HC



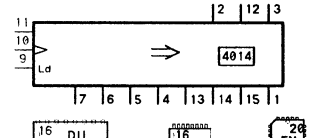
† 74x094

N



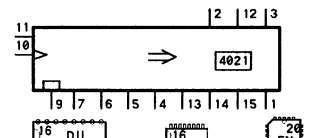
⌀ 4014

C



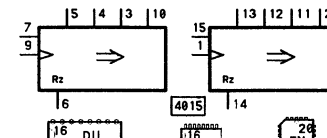
⌀ 4021

C



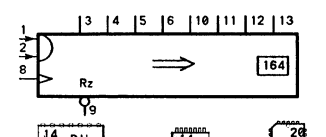
⌀ 4015

C HC



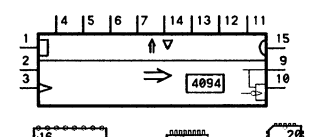
● 74x164

N LS ALS F HC



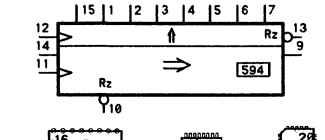
● 4094

C HC



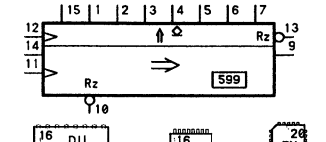
■ 74x594

LS HC



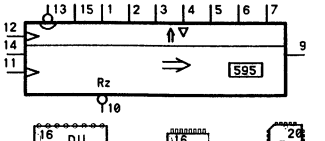
○ 74x599

LS



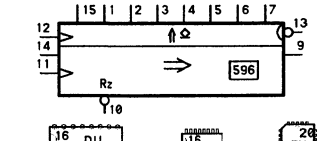
● 74x595

LS HC



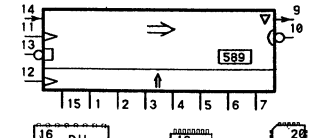
○ 74x596

LS



□ 74x589

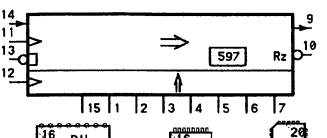
HC

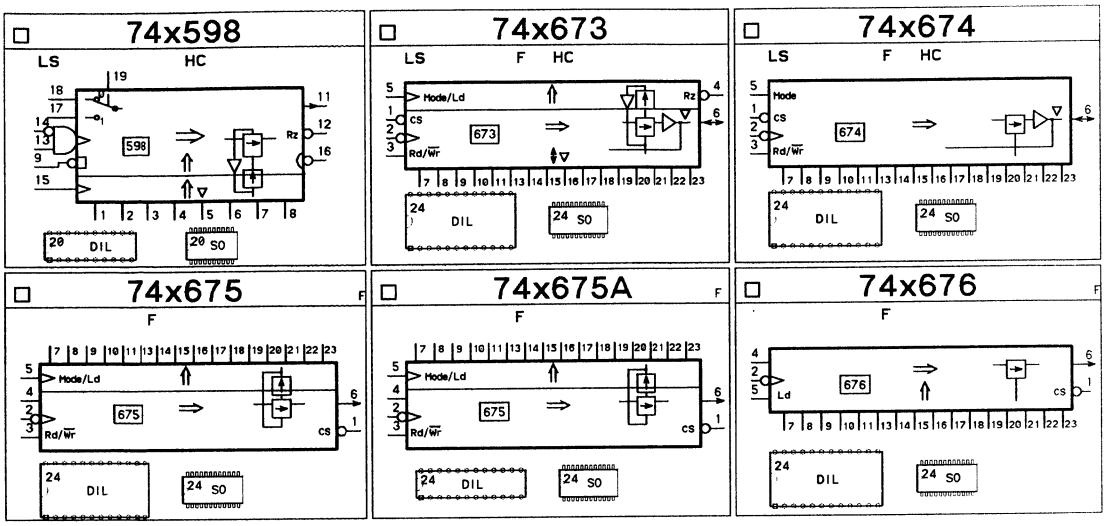


■ 74x597

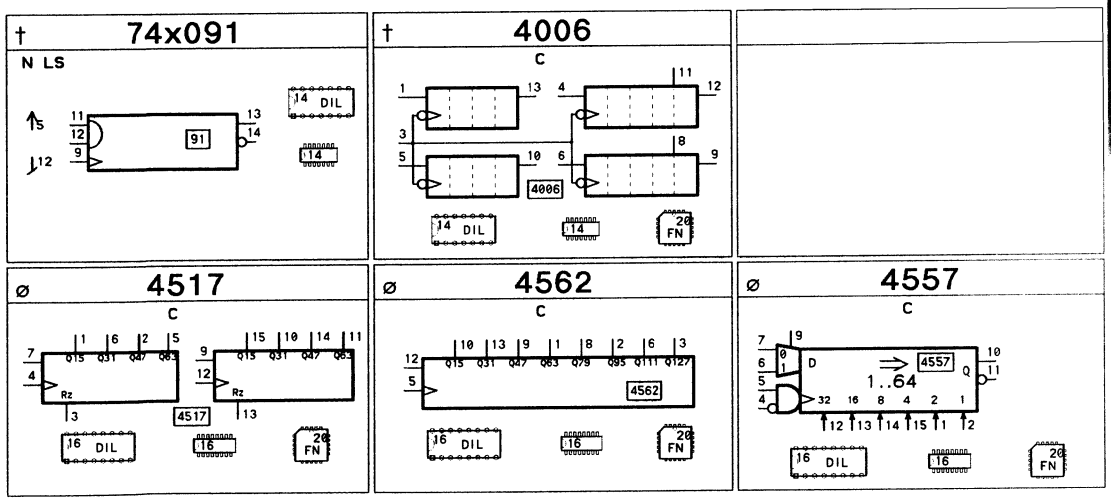
Philips 7597

LS HC

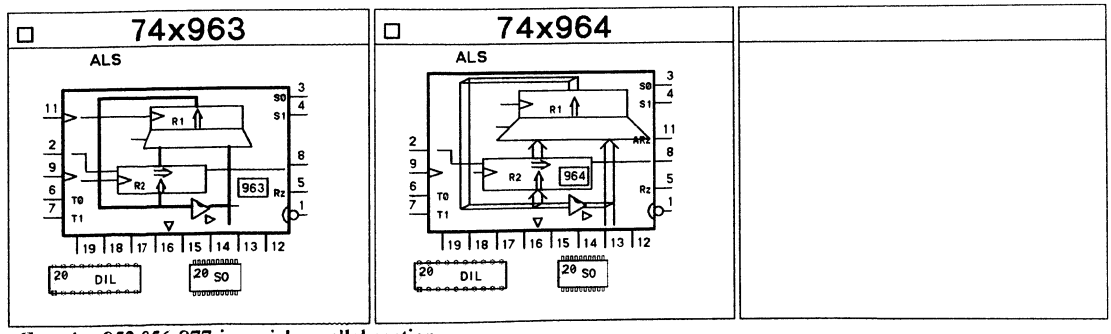




Serial-only registers



Special registers

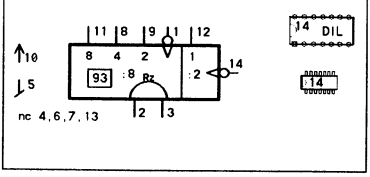


See also 852 856 877 in serial-parallel section

Asynchronous counters

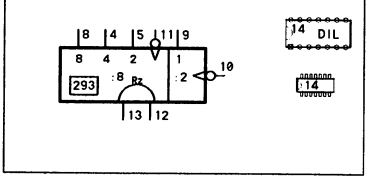
74x093

N LS C



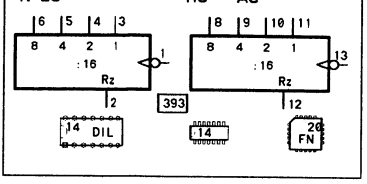
74x293

N LS



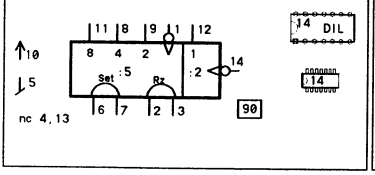
74x393

N LS HC AC



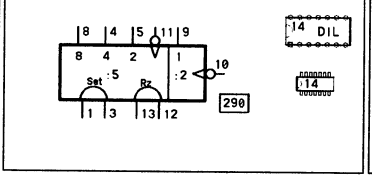
74x090

N LS C



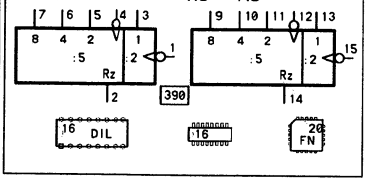
74x290

N LS



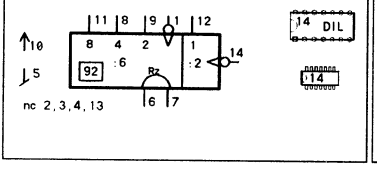
74x390

N LS HC AC



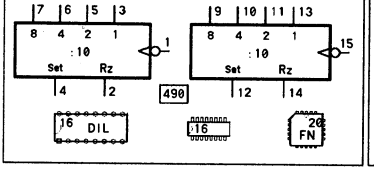
74x092

N LS



74x490

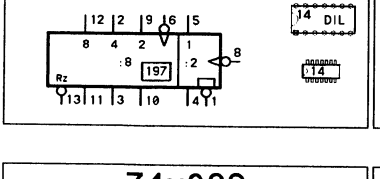
N LS HC



74x197

74177

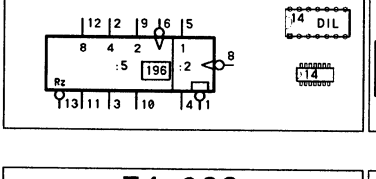
N LS S



74x196

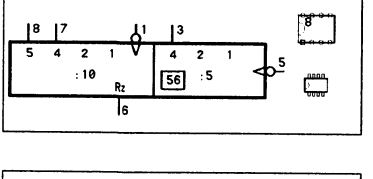
74176

N LS S



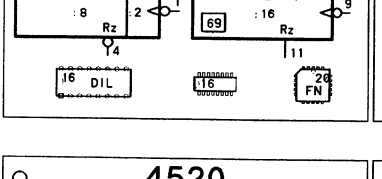
74x056

LS



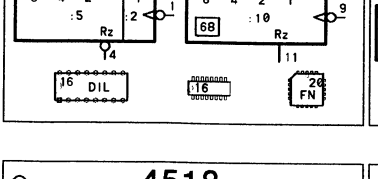
74x069

LS



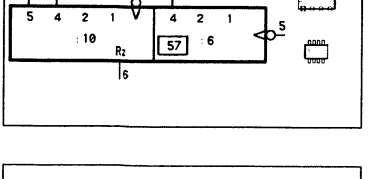
74x068

LS



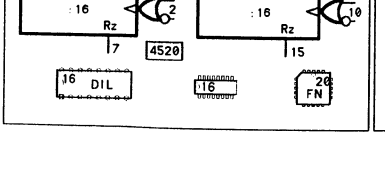
74x057

LS



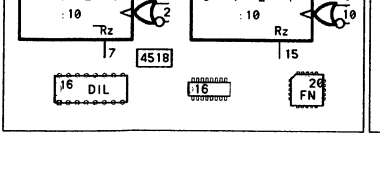
4520

C HC

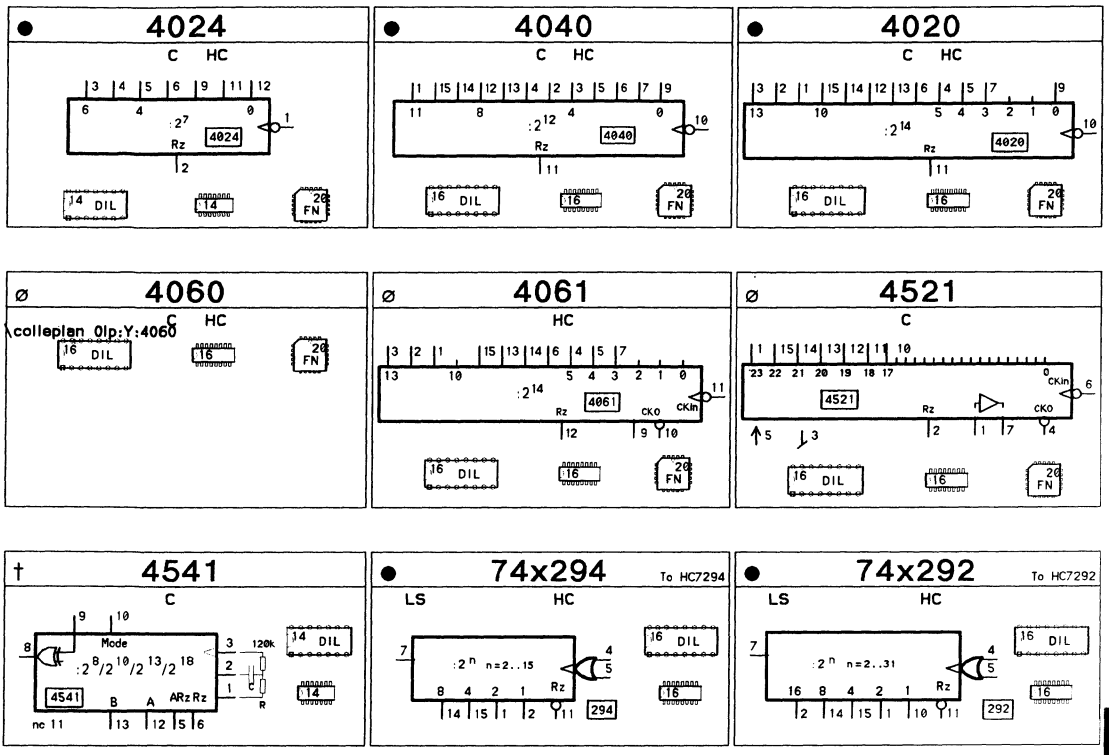


4518

C HC

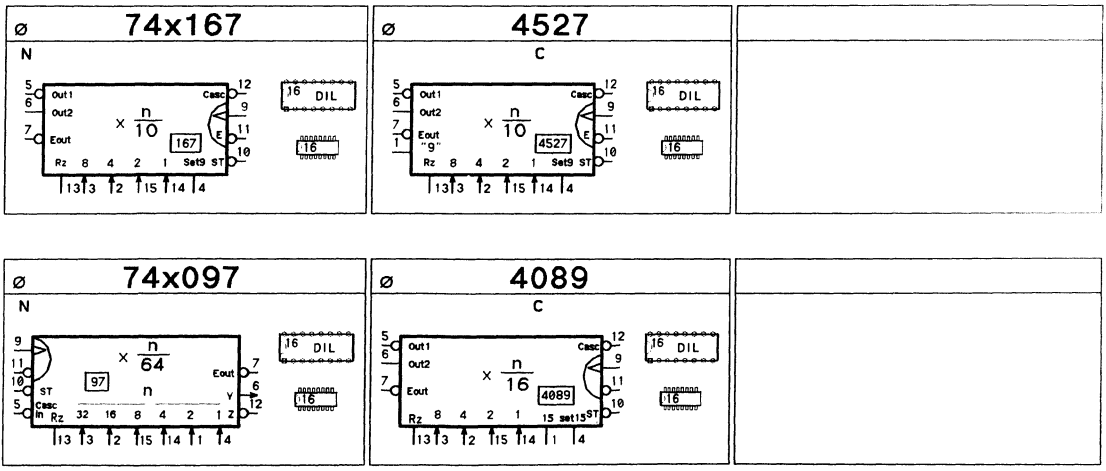


Counters



Counters

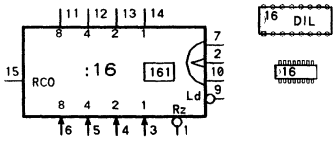
Rate multipliers



Synchronous counters

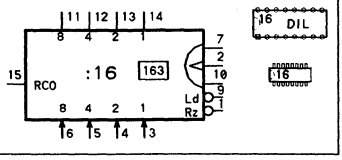
● 161

N LS ALS AS F HC AC ACT AC11



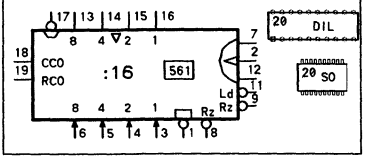
● 163

N LS S ALS AS F HC AC ACT AC11



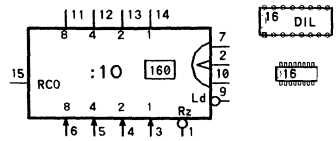
● 561

ALS



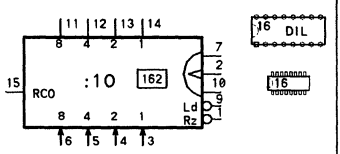
● 160

N LS ALS AS F HC AC AC11



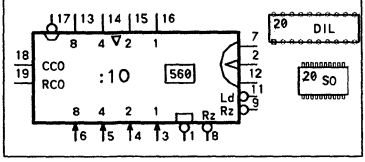
● 162

N LS S ALS AS F HC AC AC11



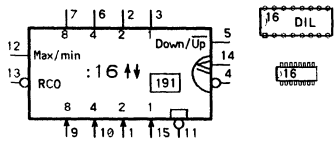
● 560

ALS



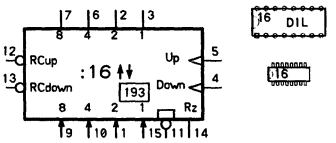
● 191

N LS ALS F HC AC AC11



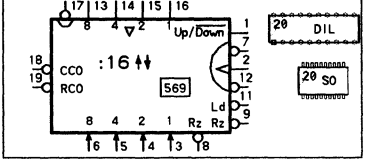
● 193

N LS ALS F HC AC AC11



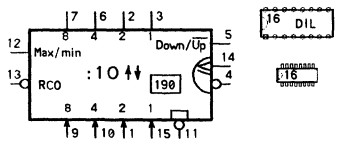
● 569

ALS F AC AC11



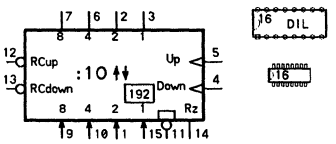
● 190

N LS ALS F HC AC AC11



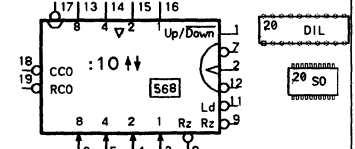
● 192

N LS ALS F HC AC AC11



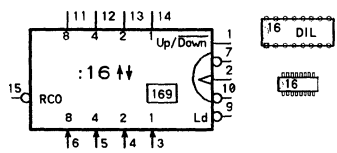
● 568

ALS F AC AC11



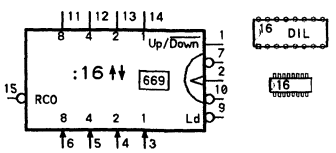
● 169

LS S ALS AS F AC AC11



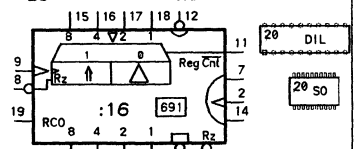
■ 669

LS



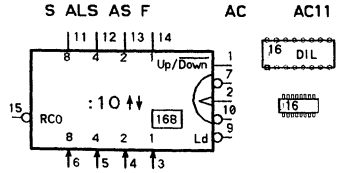
■ 691

LS HC



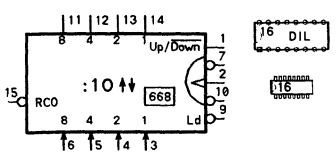
○ 168

S ALS AS F AC AC11



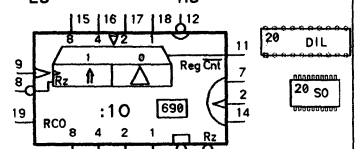
■ 668

LS



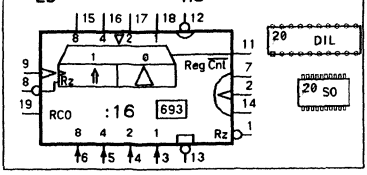
■ 690

LS HC



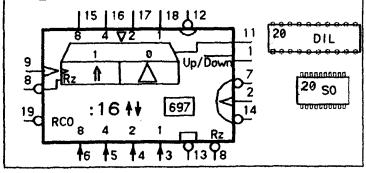
□ 693

LS HC



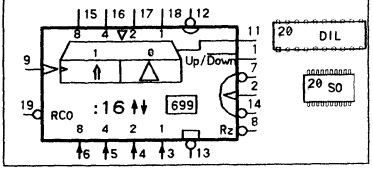
□ 697

LS HC



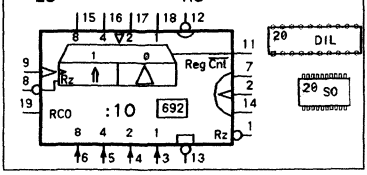
□ 699

LS HC



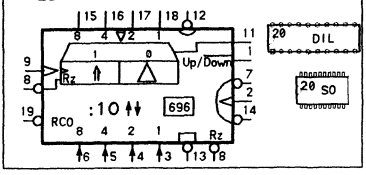
□ 692

LS HC



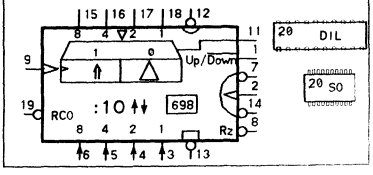
□ 696

LS HC



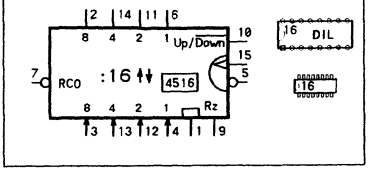
□ 698

HC



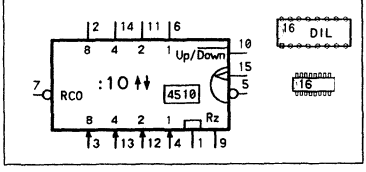
● 4516

C HC HCT



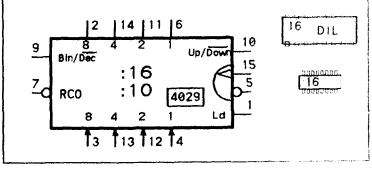
○ 4510

C HC



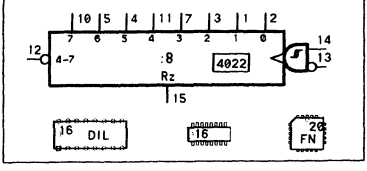
⌀ 4029

C



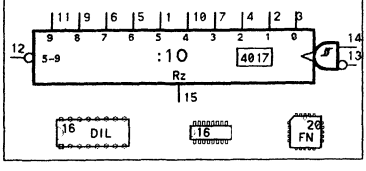
● 4022

C HC



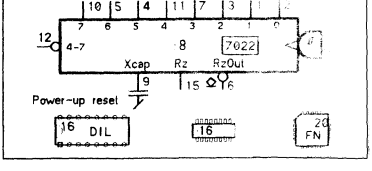
● 4017

C HC



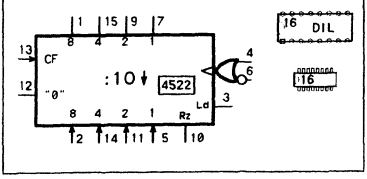
□ 7022

HC



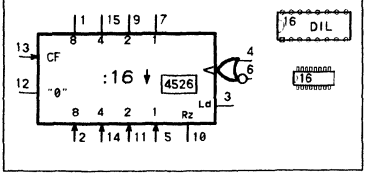
⌀ 4522

C



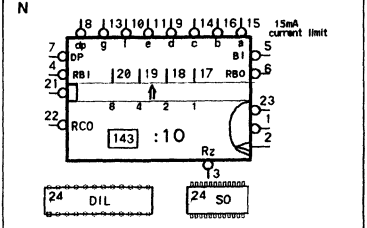
⌀ 4526

C



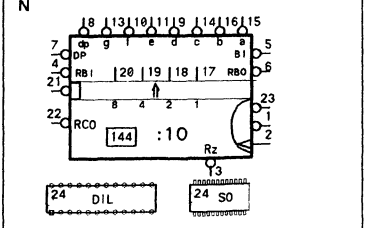
† 143

N

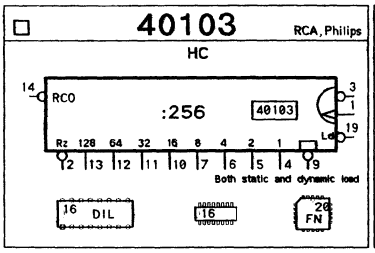


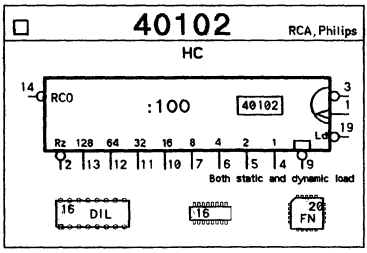
† 144

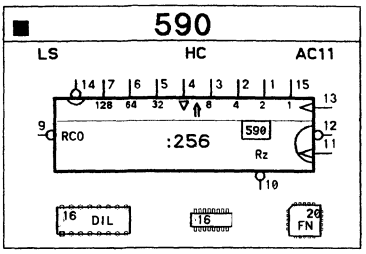
N

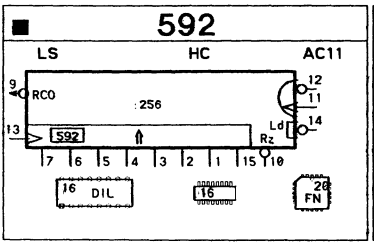


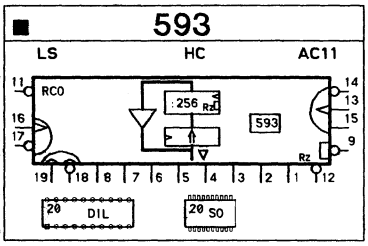
Counters

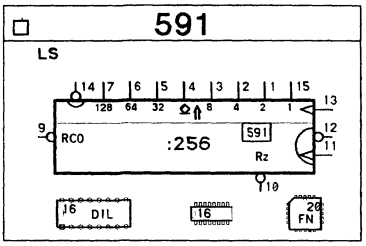

40103
RCA, Philips
HC
RCO
:256
40103
Rz 128 64 32 16 8 4 2 1
Ld
Both static and dynamic load
16 DIL
16
20 FN

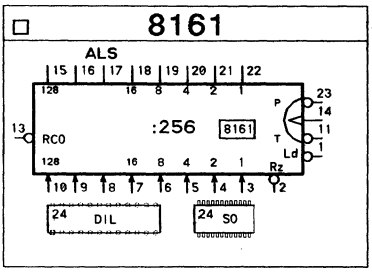

40102
RCA, Philips
HC
RCO
:100
40102
Rz 128 64 32 16 8 4 2 1
Ld
Both static and dynamic load
16 DIL
16
20 FN

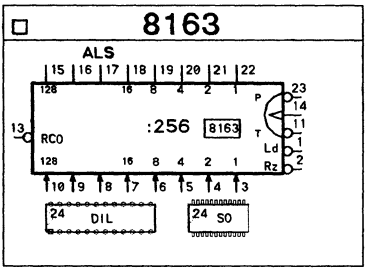

590
LS
HC
AC11
RCO
:256
590
Rz
16 DIL
16
20 FN

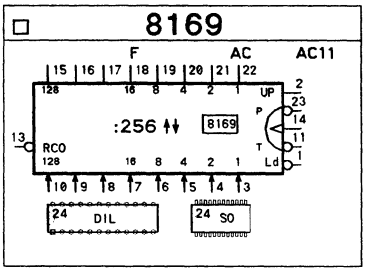

592
LS
HC
AC11
RCO
:256
592
Ld
Rz
16 DIL
16
20 FN

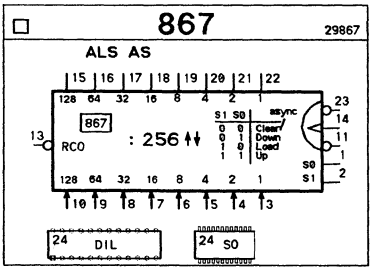

593
LS
HC
AC11
RCO
:256
593
Rz
20 DIL
20 SO

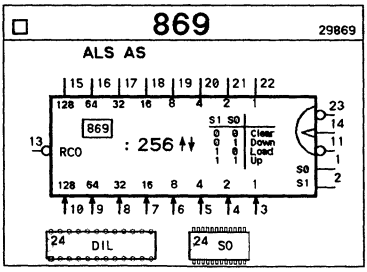

591
LS
RCO
:256
591
Rz
16 DIL
16
20 FN

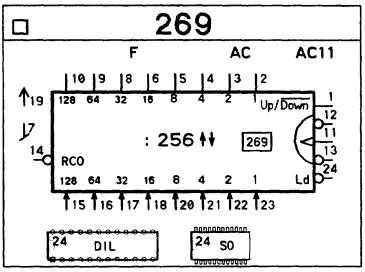

8161
ALS
RCO
:256
8161
P
T
Ld
Rz
24 DIL
24 SO

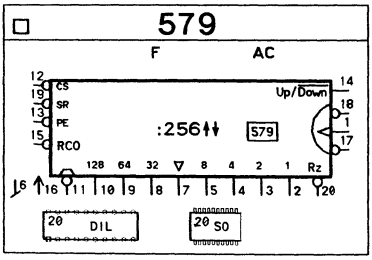

8163
ALS
RCO
:256
8163
P
T
Ld
Rz
24 DIL
24 SO

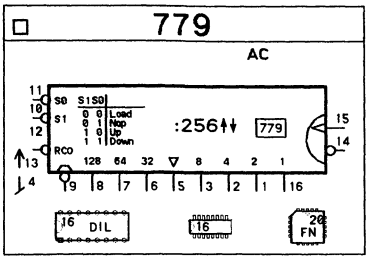

8169
F
AC
AC11
UP
RCO
:256
8169
P
T
Ld
24 DIL
24 SO

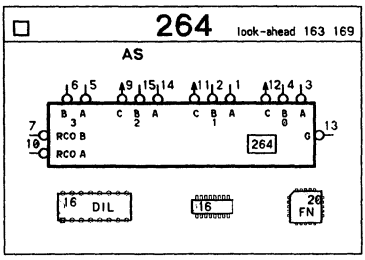

867
29867
ALS AS
867
RCO
: 256
S1 S0
0 0 Clear
0 1 Down
1 0 Load
1 1 Up
async
S0
S1
24 DIL
24 SO


869
29869
ALS AS
869
RCO
: 256
S1 S0
0 0 Clear
0 1 Down
1 0 Load
1 1 Up
S0
S1
24 DIL
24 SO


269
F
AC
AC11
Up/Down
RCO
: 256
269
Ld
24 DIL
24 SO


579
F
AC
CS
SR
PE
RCO
Up/Down
:256
579
Rz
20 DIL
20 SO


779
AC
S0 S1 S0
0 0 Load
0 1 Nop
1 0 Up
1 1 Down
S1
RCO
:256
779
16 DIL
16
20 FN


264
look-ahead 163 169
AS
RCO B
RCO A
264
16 DIL
16
20 FN

One-shots

ø 74x121

N 2k

ΔT = 0.7 R C

nc 2,8,12,13

16 DIL

○ 74x122

N LS 10k

ΔT = 0.7 R C

nc 10,12

16 DIL

ø 74x422

LS 10k

ΔT = 0.33 R C

14 DIL

● 74x123

N LS

ΔT = 0.28 R C

16 DIL

● 74x423

LS HC

ΔT = 0.33 R C

16 DIL

● 74x221

N LS HC

ΔT = 0.7 R C

16 DIL

ø 8601

Sig 9601

N

ΔT = 0.3 R C

16 DIL

ø 8602

Sig 9602

S

ΔT = 0.3 R C

14 DIL

† 4528

C

ΔT = 0.98 R C

16 DIL

○ 4538

C HC

ΔT = 0.98 R C

16 DIL

† 4548

C

ΔT = 0.9 R C

16 DIL

retriggerable with appropriate wiring

Oscillators

ø 74x320

LS

16 DIL

ø 74x321

LS

16 DIL

● Osc Quartz EXO-3

HC

Osc quartz f=12 à 20MHz

Exo-3

Si G=0 pas d'oscillation

● 555

● 556

○ Osc Quartz

LS

100kHz à 100MHz

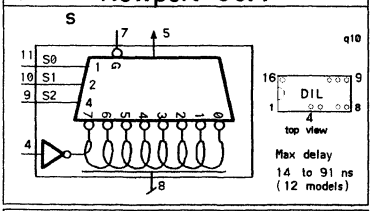
Osc quartz

top view

One-shots

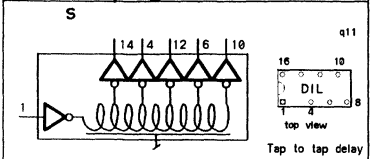
Delay elements

Newport 60A



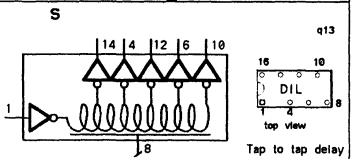
Max delay
14 to 91 ns
(12 models)

Sprague 3600



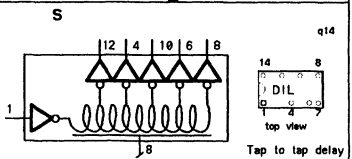
Tap to tap delay
10, 20, 50ns
(3 models)

Pulse Eng 213xx



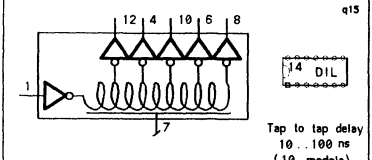
Tap to tap delay
5..100 ns
(7 models)

Pulse Eng 214xx



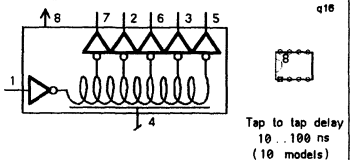
Tap to tap delay
5..100 ns
(7 models)

Dallas DS1000



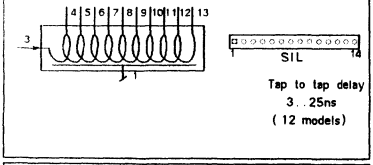
Tap to tap delay
10..100 ns
(10 models)

Dallas DS1000M



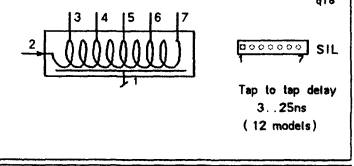
Tap to tap delay
10..100 ns
(10 models)

Newport 20A



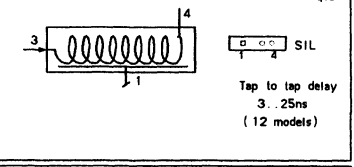
Tap to tap delay
3..25ns
(12 models)

Newport 21A



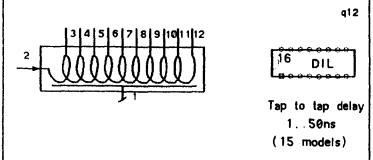
Tap to tap delay
3..25ns
(12 models)

Newport 22A



Tap to tap delay
3..25ns
(12 models)

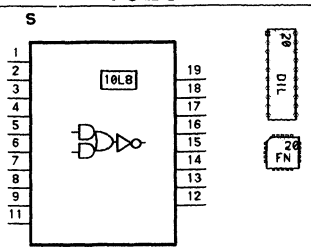
Pulse Eng 211xx



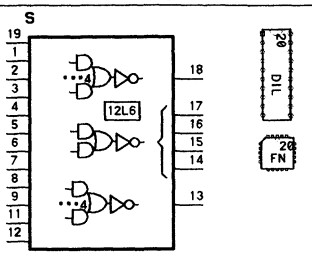
Tap to tap delay
1..50ns
(15 models)

PLDs

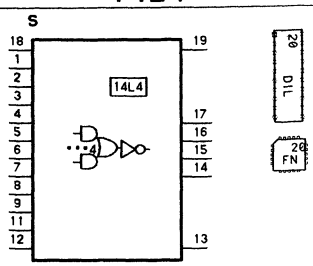
10L8



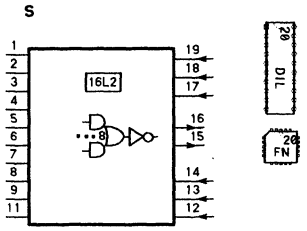
12L6



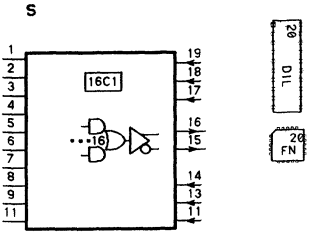
14L4



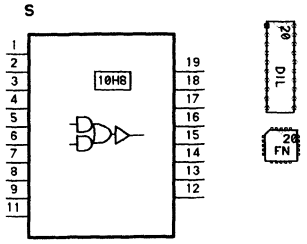
16L2



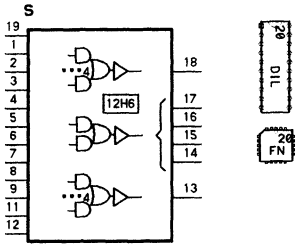
16C1



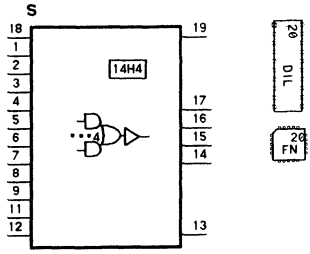
10H8



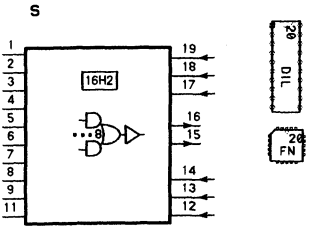
12H6



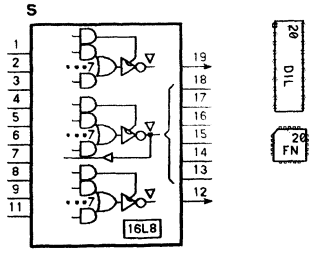
14H4



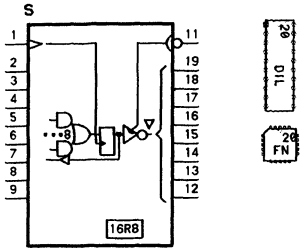
16H2



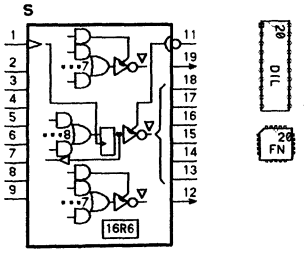
16L8



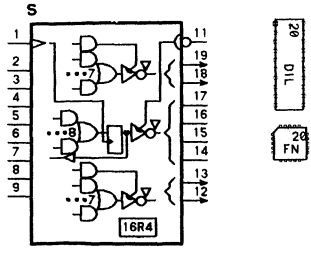
16R8



16R6



16R4



■ 16P8

S

20 DIL

20 FN

16P8

■ 18P8

S

20 DIL

20 FN

18P8

Ø 16X4

S

20 DIL

20 FN

16X4

† 16A4

S

20 DIL

20 FN

16A4

■ 16RP8

S

20 DIL

20 FN

16RP8

■ 16RP6

S

20 DIL

20 FN

16RP6

■ 16RP4

S

20 DIL

20 FN

16RP4

● 16V8 Lattice AMD

S AC

20 DIL

20 FN

16V8

● EP310 Altera (Intel=6C31)

S

20 DIL

20 FN

EP310

S R

● EP320 Altera (Intel=6C32)

S

20 DIL

20 FN

EP320

S R

Macrocell GAL 16V8

CK

étage m+1

I/O

Feedback

16V8 20V8 OE

étage m-1

Macrocell Altera 320

CK Set EP310

I/O

Reset

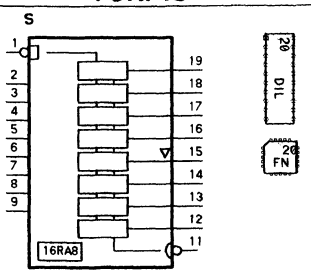
Macrocell Altera 320

CK EP320

I/O

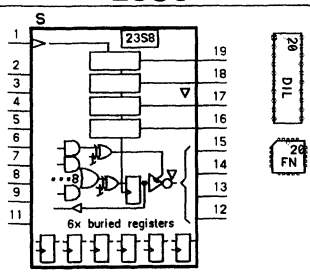
OE

16RA8

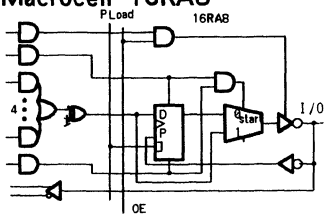


23S8

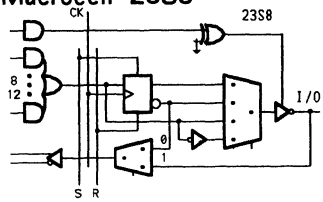
AMD



Macrocell 16RA8

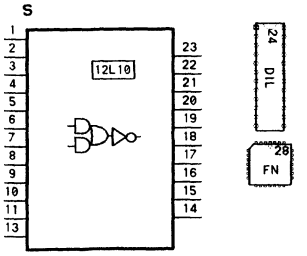


Macrocell 23S8

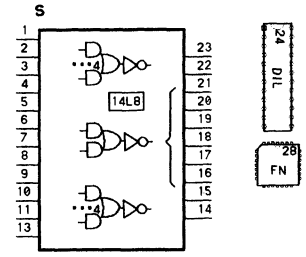


PLD (24 pins)

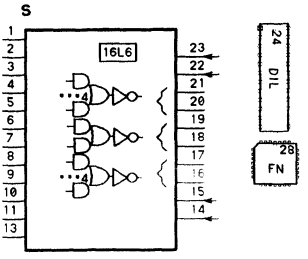
12L10



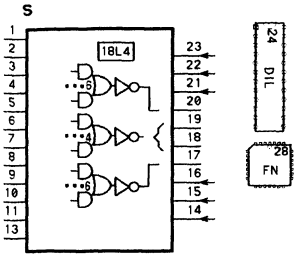
14L8



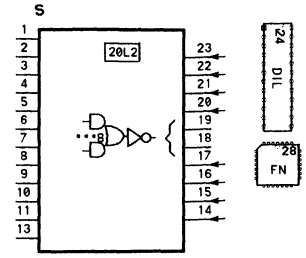
16L6



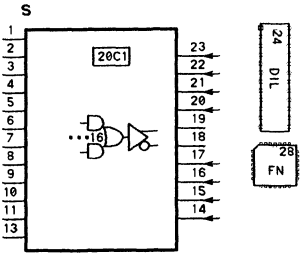
18L4



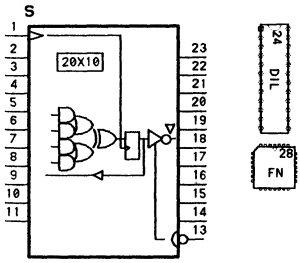
20L2



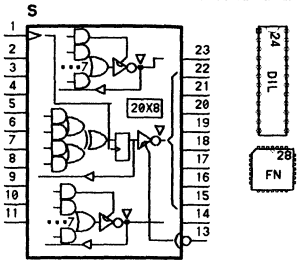
20C1



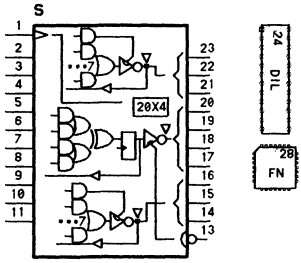
○ 20X10



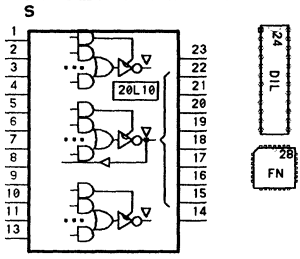
○ 20X8



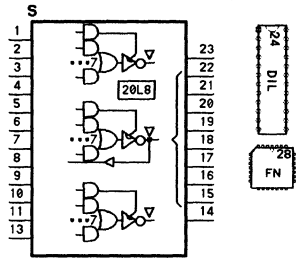
○ 20X4



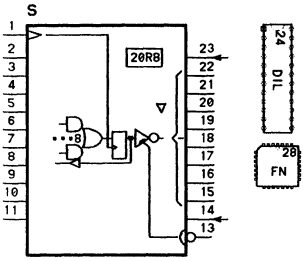
○ 20L10



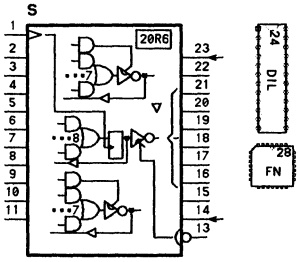
● 20L8



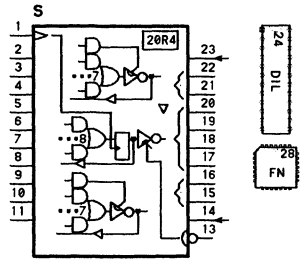
● 20R8



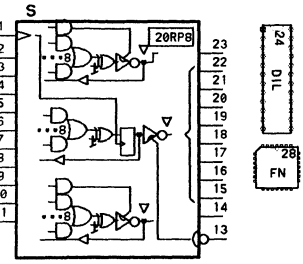
● 20R6



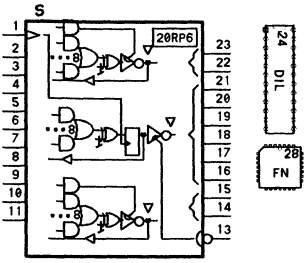
● 20R4



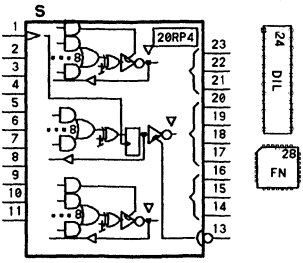
□ 20RP8



□ 20RP6



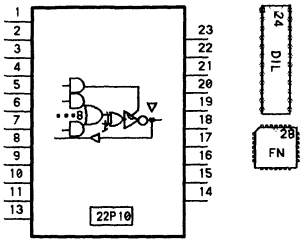
□ 20RP4



PLDs

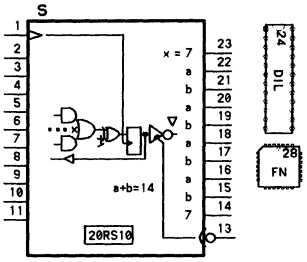
□ 22P10

S



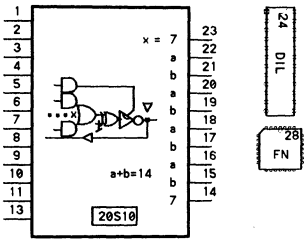
□ 20RS10

S



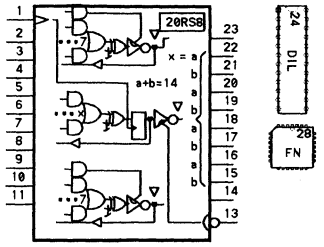
⌀ 20S10

S



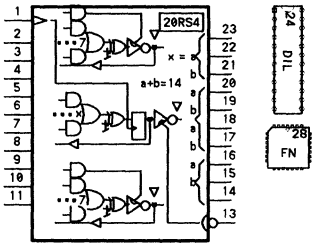
⌀ 20RS8

S



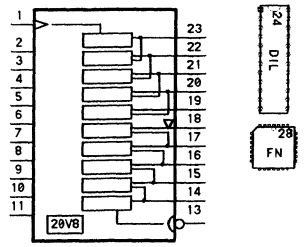
⌀ 20RS4

S



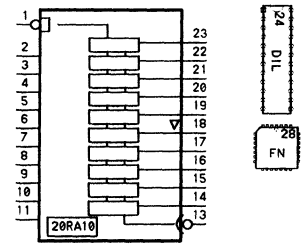
● 20V8 Lattice AMD

S AC



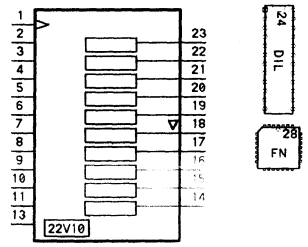
■ 20RA10

S

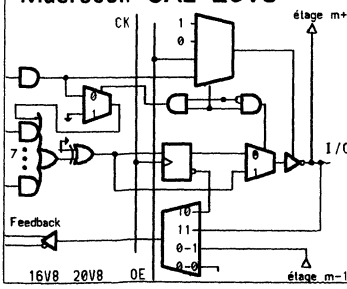


■ 22V10

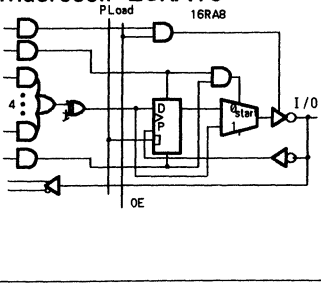
S AC



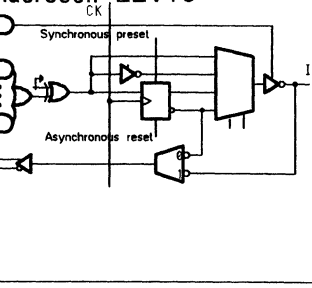
Macrocell GAL 20V8



Macrocell 20RA10



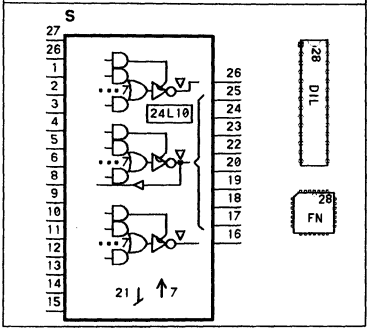
Macrocell 22V10



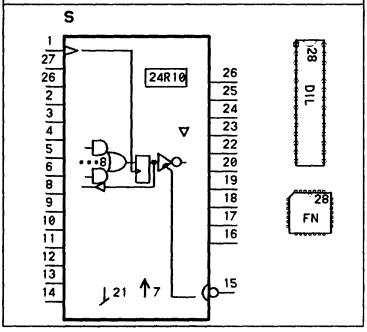
PLDs

PLD (28 pins)

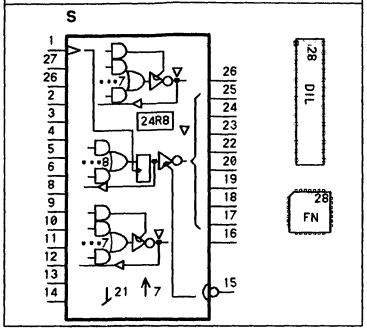
24L10



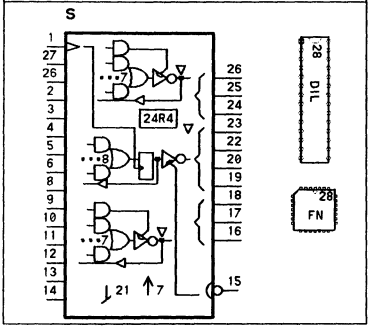
24R10



24R8



24R4



Index of circuits

000 NAND, 235
001 NAND, 235
002 NOR, 238
003 NAND, 235
004u Inverter, 231
004 Inverter, 231
005 Inverter, 231
006 Inverter, 231
007 Buffer, 232
008 AND, 237
009 AND, 237
010 NAND, 236
011 AND, 237
012 NAND, 236
013 NAND, 236
014 Inverter, 231
015 AND, 237
016 Inverter, 231
017 Buffer, 232
018 NAND, 236
019 Inverter, 231
020 NAND, 236
021 AND, 238
022 NAND, 236
023 NOR, 239
024 NAND, 235
025 NOR, 239
026 NAND, 235
027 NOR, 238
028 NOR, 238
030 NAND, 236
031 Gates, 234
032 OR, 239
033 NOR, 238
034 Buffer, 232
035 Buffer, 233
036 NOR, 238
037 NAND, 235
038 NAND, 235
039 NAND, 235
040 NAND, 236
042 Decoder, 254
043 Decoder, 254
044 Decoder, 254
045 Decoder, 254
046 7-seg, 255
047 7-seg, 255
048 7-seg, 255
049 7-seg, 255
050 AND-OR, 241
051 AND-OR, 241
053 AND-OR, 241
054 AND-OR, 241
055 AND-OR, 241
056 Async-counter, 272
057 Async-counter, 272
058 AND-OR, 241
060 AND-OR, 241
063 Buffer, 233
064 AND-OR, 241
065 AND-OR, 241
068 Async-counter, 272
069 Async-counter, 272
070 JK-flip-flop, 259
072 JK-flip-flop, 259
073 JK-flip-flop, 260
074 D-flip-flop, 259
075 Latch, 265
076 JK-flip-flop, 260
077 Latch, 265
078 JK-flip-flop, 260
082 ALU, 257
083 ALU, 257
085 Comparator, 256
086 XOR, 240
090 Async-counter, 272
091 Serial-only, 271
092 Async-counter, 272
093 Async-counter, 272
094 Serial-reg, 270
095 Serial-reg, 269
096 Serial-reg, 269
097 Rate-mult, 273
098 Mux+reg, 267
100 Latch, 265
104 JK-flip-flop, 259
105 JK-flip-flop, 260
107 JK-flip-flop, 260
109 JK-flip-flop, 260
110 JK-flip-flop, 260
111 JK-flip-flop, 260
112 JK-flip-flop, 260
113 JK-flip-flop, 260
114 JK-flip-flop, 260
116 Latch, 265
118 RS-flip-flop, 259
119 RS-flip-flop, 259
120 RS-flip-flop, 259
121 One-shot, 277
122 One-shot, 277
123 One-shot, 277
125 3s-driver, 242
126 3s-driver, 242
128 NOR, 238
131 Decoder, 253
132 NAND, 235
133 NAND, 237
134 NAND, 237
135 XOR, 240
136 XOR, 240
137 Decoder, 253
138 Decoder, 253
139 Decoder, 253
140 NAND, 236
141 Decoder, 254
143 Sync-counter, 275
144 Sync-counter, 275
145 Decoder, 254
147 Priority-enco, 252
148 Priority-enco, 252
150 Mux, 251
151 Mux, 250
152 Mux, 250
153 Mux, 250
154 Decoder, 254
155 Decoder, 253
156 Decoder, 253
157 Mux, 250
158 Mux, 250
159 Decoder, 254
160 Sync-counter, 274
161 Sync-counter, 274
162 Sync-counter, 274
163 Sync-counter, 274
164 Serial-reg, 270
165 Serial-reg, 270
166 Serial-reg, 270
167 Rate-mult, 273
168 Sync-counter, 274
169 Sync-counter, 274
171 Parallel-reg, 261
173 Parallel-reg, 261
174 Parallel-reg, 261
175 Parallel-reg, 261
178 Serial-reg, 269
179 Serial-reg, 269
180 Parity, 251
181 ALU, 258
182 Carry, 258
183 ALU, 257
190 Sync-counter, 274
191 Sync-counter, 274
192 Sync-counter, 274
193 Sync-counter, 274
194 Serial-reg, 269
195 Serial-reg, 269
196 Async-counter, 272
197 Async-counter, 272
198 Serial-reg, 269
199 Serial-reg, 269
221 One-shot, 277
230 3s-driver, 243
231 3s-driver, 243
237 Decoder, 253
238 Decoder, 253
239 Decoder, 253
240 3s-driver, 242
241 3s-driver, 242
242 3s-bidir, 246
243 3s-bidir, 246
244 3s-driver, 242
245 3s-bidir, 246
246 7-seg, 255
247 7-seg, 255
248 7-seg, 255
249 7-seg, 255
250 Mux, 251
251 Mux, 250
253 Mux, 250
256 Addr-latch, 268
257 Mux, 250
258 Mux, 250
259 Addr-latch, 268
260 NOR, 239
264 Sync-counter, 276
265 Gates, 234
266 XNOR, 240
269 Sync-counter, 276
273 Parallel-reg, 261
276 JK-flip-flop, 260
278 Priority-enco, 252
279 RS-flip-flop, 259
280 Parity, 251
282 Carry, 258
283 ALU, 257
286 Parity, 251
290 Async-counter, 272
292 Async-counter, 273
293 Async-counter, 272
294 Async-counter, 273
295 Serial-reg, 269
298 Mux+reg, 267
299 Serial-reg, 270
320 Oscillator, 277
321 Oscillator, 277
322 Serial-reg, 270
323 Serial-reg, 270
347 7-seg, 255
348 Priority-enco, 252
351 Mux, 251
352 Mux, 250
353 Mux, 250
354 Mux, 251
355 Mux, 251
356 Mux, 251
357 Mux, 251
365 3s-driver, 242
366 3s-driver, 242
367 3s-driver, 242
368 3s-driver, 242
373 Latch, 265
374 Parallel-reg, 261

375 Latch, 265
376 JK-flip-flop, 260
377 Parallel-reg, 261
378 Parallel-reg, 261
379 Parallel-reg, 261
381 ALU, 258
382 ALU, 258
386 XOR, 240
390 Async-counter, 272
393 Async-counter, 272
395 Serial-reg, 269
396 Dual-rank, 263
398 Mux+reg, 267
399 Mux+reg, 267
412 Reg.with-inter.logic, 267
422 One-shot, 277
423 One-shot, 277
425 3s-driver, 242
426 3s-driver, 242
432 Reg.with-inter.logic, 267
436 3s-driver, 242
437 3s-driver, 242
440 tridir-driv, 249
441 tridir-driv, 249
442 tridir-driv, 249
443 tridir-driv, 249
444 tridir-driv, 249
445 Decoder, 254
446 488-driv., 249
447 7-seg, 255
448 tridir-driv, 249
449 488-driv., 249
455 Pairity-driv, 245
456 Pairity-driv, 245
465 3s-driver, 243
466 3s-driver, 243
467 3s-driver, 243
468 3s-driver, 243
490 Async-counter, 272
518 Comparator, 256
519 Comparator, 256
520 Comparator, 256
521 Comparator, 256
522 Comparator, 256
524 Comparator, 257
533 Latch, 265
534 Parallel-reg, 261
537 Decoder, 253
538 Decoder, 253
539 Decoder, 253
540 3s-driver, 243
541 3s-driver, 243
543 Bidir-latch, 267
544 Bidir-latch, 267
545 3s-bidir, 246
547 Decoder, 253
548 Decoder, 253
550 Transceiver, 264
551 Transceiver, 264
552 Transceiver, 264
560 Sync-counter, 274
561 Sync-counter, 274
563 Latch, 265
564 Parallel-reg, 261
568 Sync-counter, 274
569 Sync-counter, 274
573 Latch, 265
574 Parallel-reg, 261
575 Parallel-reg, 262
576 Parallel-reg, 262
577 Parallel-reg, 262
579 Sync-counter, 276
580 Latch, 265
582 ALU, 257
583 ALU, 257
588 488-driv., 249
589 Serial-reg, 270
590 Sync-counter, 276
591 Sync-counter, 276
592 Sync-counter, 276
593 Sync-counter, 276
594 Serial-reg, 270
595 Serial-reg, 270
596 Serial-reg, 270
597 Serial-reg, 270
598 Serial-reg, 271
599 Serial-reg, 270
604 Mux+reg, 267
605 Mux+reg, 267
606 Mux+reg, 267
607 Mux+reg, 267
614 Transceiver, 264
615 Transceiver, 263
620 3s-bidir, 246
621 OC-bidir, 247
622 OC-bidir, 248
623 3s-bidir, 246
638 OC-bidir, 248
639 OC-bidir, 247
640 3s-bidir, 246
641 OC-bidir, 247
642 OC-bidir, 248
643 3s-bidir, 246
644 OC-bidir, 248
645 3s-bidir, 246
646 Transceiver, 263
647 Transceiver, 263
648 Transceiver, 263
649 Transceiver, 264
651 Transceiver, 263
652 Transceiver, 263
653 Transceiver, 264
654 Transceiver, 263
655 Pairity-driv, 245
656 Pairity-driv, 245
657 Parity-bidir, 248
658 Parity-bidir, 248
659 Parity-bidir, 248
664 Parity-bidir, 248
665 Parity-bidir, 248
666 Read-back-latch, 266
667 Read-back-latch, 266
668 Sync-counter, 274
669 Sync-counter, 274
671 Serial-reg, 269
672 Serial-reg, 269
673 Serial-reg, 271
674 Serial-reg, 271
675 Serial-reg, 271
675A Serial-reg, 271
676 Serial-reg, 271
677 Addr-comp, 257
678 Addr-comp, 257
679 Addr-comp, 257
680 Addr-comp, 257
682 Comparator, 256
683 Comparator, 256
684 Comparator, 256
685 Comparator, 256
686 Comparator, 256
687 Comparator, 256
688 Comparator, 256
689 Comparator, 256
690 Sync-counter, 274
691 Sync-counter, 274
692 Sync-counter, 275
693 Sync-counter, 275
696 Sync-counter, 275
697 Sync-counter, 275
698 Sync-counter, 275
699 Sync-counter, 275
730 3s-driver, 243
734 3s-driver, 243
746 3s-driver, 243
747 3s-driver, 243
756 OC-driver, 244
757 OC-driver, 244
758 OC-bidir, 248
759 OC-bidir, 248
760 OC-driver, 244
762 OC-driver, 244
763 OC-driver, 244
779 Sync-counter, 276
803 Parallel-reg, 261
804 NAND, 235
805 NOR, 238
808 AND, 237
810 XNOR, 240
811 XNOR, 240
821 Parallel-reg, 262
822 Parallel-reg, 262
823 Parallel-reg, 262
824 Parallel-reg, 262
825 Parallel-reg, 262
826 Parallel-reg, 262
827 3-s-driver, 244
828 3-s-driver, 244
832 OR, 239
833 Parity-bidir, 248
834 Parity-bidir, 248
841 Latch, 265
842 Latch, 266
843 Latch, 266
844 Latch, 266
845 Latch, 265
846 Latch, 265
850 Mux, 251
851 Mux, 251
852 Universal-reg, 264
853 Parity-bidir, 248
854 Parity-bidir, 248
856 Universal-reg, 264
857 Mux, 250
861 3s-bidir, 247
862 3s-bidir, 247
863 3s-bidir, 247
864 3s-bidir, 247
866 Comparator, 256
867 Sync-counter, 276
869 Sync-counter, 276
873 Latch, 265
874 Parallel-reg, 261
876 Parallel-reg, 261
877 Universal-reg, 264
878 Parallel-reg, 261
879 Parallel-reg, 261
880 Latch, 265
881 ALU, 258
882 Carry, 258
885 Comparator, 256
901 Inverter, 231
902 Buffer, 233
903 Inverter, 231
904 Buffer, 233
906 Buffer, 233
907 Buffer, 233
914 Inverter, 231
915 7-seg, 255
963 Special-reg, 271
964 Special-reg, 271
990 Read-back-latch, 266
991 Read-back-latch, 266
992 Read-back-latch, 266
993 Read-back-latch, 266
994 Read-back-latch, 266
995 Read-back-latch, 266
996 Read-back-reg, 263
1000 NAND, 235
1002 NOR, 238
1003 NAND, 235
1004 Inverter, 231

1005 Inverter, 231
1008 AND, 237
1010 NAND, 236
1011 AND, 237
1020 NAND, 236
1032 OR, 239
1034 Buffer, 232
1035 Buffer, 233
1036 NOR, 238
1181 ALU, 258
1240 3s-driver, 242
1241 3s-driver, 242
1242 3s-bidir, 246
1243 3s-bidir, 246
1244 3s-driver, 242
1245 3s-bidir, 246
1804 NAND, 235
1805 NOR, 238
1808 AND, 237
1832 OR, 239
2240 3s-driver, 242
2241 3s-driver, 242
2242 3s-bidir, 246
2244 3s-driver, 242
2414 Decoder, 253
2519 Parallel-reg, 262
2530 3s-driver, 243
2531 3s-driver, 243
2620 3s-bidir, 246
2623 3s-bidir, 246
2640 3s-bidir, 246
2645 3s-bidir, 246
2827 3s-driver, 243
2828 3s-driver, 243
2952 Transceiver, 264
2953 Transceiver, 264
4000 NOR, 238
4001 NOR, 238
4002 NOR, 238
4006 Serial-only, 271
4008 ALU, 257
4009 Inverter, 231
40102 Sync-counter, 276
40103 Sync-counter, 276
40104 Serial-reg, 269
40106 Inverter, 232
40107 NAND, 235
4010 Buffer, 232
4011 NAND, 235
4012 NAND, 236
4013 D-flip-flop, 259
4014 Serial-reg, 270
4015 Serial-reg, 270
4016 Analog-mux, 252
4017 Sync-counter, 275
4019 Mux, 250
4020 Async-counter, 273
4021 Serial-reg, 270
4022 Sync-counter, 275
4023 NAND, 236
4024 Async-counter, 273
4025 NOR, 238
4027 JK-flip-flop, 260
4028 Decoder, 254
4029 Sync-counter, 275
4035 Serial-reg, 269
4040 Async-counter, 273
4042 Latch, 265
4043 RS-flip-flop, 259
4044 RS-flip-flop, 259
4049 Inverter, 231
4050 Buffer, 232
4051 Analog-mux, 252
4052 Analog-mux, 252
4053 Analog-mux, 252
4060 Async-counter, 273
4061 Async-counter, 273
4066 Analog-mux, 252
4067 Analog-mux, 252
4068 NAND, 236
4069 Inverter, 231
4070 XOR, 240
4071 OR, 239
4072 OR, 239
4073 AND, 238
4075 OR, 239
4076 Parallel-reg, 261
4077 XNOR, 240
4078 NOR, 239
4078 OR, 239
4081 AND, 237
4082 AND, 238
4085 AND-OR, 241
4086 AND-OR, 241
4089 Rate-mult, 273
4093 NAND, 235
4094 Serial-reg, 270
4097 Analog-mux, 252
4099 Addr-latch, 268
4316 Analog-mux, 252
4374 Dual-rank, 263
4495 7-seg, 255
4501 Gates, 234
4502 3s-driver, 242
4504 Buffer, 233
4506 AND-OR, 241
4507 XOR, 240
4508 Latch, 265
4510 Sync-counter, 275
4511 7-seg, 255
4512 Mux, 251
4513 7-seg, 255
4514 Decoder, 254
4515 Decoder, 254
4516 Sync-counter, 275
4517 Serial-only, 271
4518 Async-counter, 272
4519 Mux, 250
4520 Async-counter, 272
4521 Async-counter, 273
4522 Sync-counter, 275
4526 Sync-counter, 275
4527 Rate-mult, 273
4528 One-shot, 277
4529 Analog-mux, 252
4530 Gates, 234
4531 Parity, 251
4532 Priority-enco, 252
4538 One-shot, 277
4539 Mux, 250
4541 Async-counter, 273
4543 7-seg, 255
4544 7-seg, 255
4547 7-seg, 255
4548 One-shot, 277
4551 Analog-mux, 252
4555 Decoder, 253
4556 Decoder, 253
4557 Serial-only, 271
4558 7-seg, 255
4560 ALU, 258
4561 ALU, 258
4562 Serial-only, 271
4572 Gates, 234
4581 ALU, 258
4582 Carry, 258
4583 Inverter, 232
4584 Inverter, 232
4585 Comparator, 256
4597 Addr-latch, 268
4598 Addr-latch, 268
4599 Addr-latch, 268
4723 Addr-latch, 268
4724 Addr-latch, 268
4929 Gates, 234
4931 NAND, 236
7001 AND, 237
7002 NOR, 238
7003 NAND, 235
7006 Gates, 234
7008 Gates, 234
7022 Sync-counter, 275
7032 OR, 239
7074 D-flip-flop, 259
7075 D-flip-flop, 259
7075 D-flip-flop, 259
7080 Parity, 251
7266 XNOR, 240
7340 Parallel-reg, 262
74107 JK-flip-flop, 260
74154 Decoder, 254
7451 AND-OR, 241
7454 AND-OR, 241
7473 JK-flip-flop, 260
7476 JK-flip-flop, 260
74C85 Comparator, 256
74C86 XOR, 240
74C95 Serial-reg, 269
75160 488-driv., 249
75161 488-driv., 249
75162 488-driv., 249
75163 488-driv., 249
75164 488-driv., 249
7540 3s-driver, 243
7541 3s-driver, 243
8003 NAND, 235
8161 Sync-counter, 276
8163 Sync-counter, 276
8169 Sync-counter, 276
8244 JTAG driver, 245
8245 JTAG driver, 245
8273 JTAG latch, 245
8274 JTAG register, 245
8303 3s-bidir, 246
8304 3s-bidir, 246
8601 One-shot, 277
8602 One-shot, 277
9014 Inverter, 232
9015 Buffer, 233
9114 Inverter, 232
9115 Buffer, 233

16240 3s-driver, 244
16244 3s-driver, 244
16245 3s-bidir, 247
16373 Latch, 266
16374 Parallel-reg, 262
16533 Latch, 266
16534 Parallel-reg, 262
16540 3s-driver, 244
16541 3s-driver, 244
16620 3s-bidir, 247
16623 3s-bidir, 247
16640 3s-bidir, 247
16646 Transceiver, 264
16648 Transceiver, 264
16651 Transceiver, 264
16652 Transceiver, 264
16821 Parallel-reg, 262
16822 Parallel-reg, 262
16823 Parallel-reg, 262
16824 Parallel-reg, 262
16825 3s-driver, 244
16826 3s-driver, 244
16826 3s-driver, 244
16827 3s-driver, 244
16841 Latch, 266
16842 Latch, 266
16843 Latch, 266
16844 Latch, 266
16861 3s-bidir, 247

16862 3s-bidir, 247
16863 3s-bidir, 247
16864 3s-bidir, 247
16979 3s-bidir, 264

29806 Comparator, 257
29809 Comparator, 257
29821 Parallel-reg, 262
29822 Parallel-reg, 262
29823 Parallel-reg, 262
29824 Parallel-reg, 262
29825 Parallel-reg, 262
29826 Parallel-reg, 262
29827 3-s-driver, 244
29828 3-s-driver, 244
29833 Parity-bidir, 248
29834 Parity-bidir, 248
29841 Latch, 265
29842 Latch, 266
29843 Latch, 266
29844 Latch, 266
29845 Latch, 265
29846 Latch, 265
29853 Parity-bidir, 248
29854 Parity-bidir, 248
29861 3s-bidir, 247
29862 3s-bidir, 247
29863 3s-bidir, 247
29864 3s-bidir, 247
68HC138 Decoder, 253

1-gate circuits

4S01F NOR, 234
4S11F NAND, 234
4S30F XOR, 234
4S66F Transf-gate, 234
4S69F Inverter, 234
4S71F OR, 234
4S81F AND, 234
4S84F Inverter, 234
4SU69F Inverter, 234
7S00F NAND, 234
7S02F NOR, 234
7S04F Inverter, 234
7S08F AND, 234
7S32F OR, 234
7SU04F Inverter, 234

PLDs

10H8 PLD, 279
10L8 PLD, 279
12H6 PLD, 279
12L10 PLD, 281
12L6 PLD, 279
14H4 PLD, 279
14L4 PLD, 279
14L8 PLD, 281
16A4 PLD, 280
16C1 PLD, 279
16H2 PLD, 279
16L2 PLD, 279
16L6 PLD, 281
16L8 PLD, 279
16P8 PLD, 280
16R4 PLD, 279
16R6 PLD, 279
16R8 PLD, 279
16RA8 PLD, 281
16RP4 PLD, 280
16RP6 PLD, 280
16RP8 PLD, 280
16V8 PLD, 280
16X4 PLD, 280
18L4 PLD, 281
18P8 PLD, 280
20C1 PLD, 281
20L10 PLD, 282
20L2 PLD, 281
20L8 PLD, 282
20R4 PLD, 282
20R6 PLD, 282
20R8 PLD, 282
20RA10 PLD, 283
20RP4 PLD, 282
20RP6 PLD, 282
20RP8 PLD, 282
20RS10 PLD, 283
20RS4 PLD, 283
20RS8 PLD, 283
20S10 PLD, 283
20V8 PLD, 283
20X10 PLD, 282
20X4 PLD, 282
20X8 PLD, 282
22P10 PLD, 283
22V10 PLD, 283
23S8 PLD, 281
24L10 PLD, 284
24R10 PLD, 284
24R4 PLD, 284
24R8 PLD, 284
EP310 PLD, 280
EP320 PLD, 280

Delay lines

Dallas DS1000M, 278
Dallas DS1000, 278
Newport 20A, 278
Newport 21A, 278
Newport 22A, 278
Newport 60A, 278
Pulse Eng 211xx, 278
Pulse Eng 213xx, 278
Pulse Eng 214xx, 278
Sprague 3600, 278

Quartz oscillators

Osc Quartz EXO-3, 277
Osc Quartz, 277

Index of terms

"4000" family, 12
"74" family, 7
68000 decoder, 125
AC Advanced CMOS, 7
active filters, 148
active state, 24
adders, 41, 56
addressable latches, 206, 210
AIEE-91, 99
ALS Advanced Low-power Schottky, 7
ALUs Arithmetic Logic Unit, 41
amplified output, 45
analog comparator, 149
analog oscillator, 95
analog switches, 50
analog-digital converters, 150
AND, 26
angular encoder, 144
anticoincidence flip-flop, 61
application specific integrated circuits, 34
arithmetic logic units, 56
AS Advanced Schottky, 7
ASICs, 34
asserted, 24
associative access, 105
asynchronous counters, 88
asynchronous loading, 83
asynchronous reset, 83
asynchronous systems, 64
availability, 165
average life expectation, 164
barrel shifters, 57
bed of nails, 163
bias current, 147
bidirectional drivers, 49
BILBO Built-In Logic Block Observer, 172
binary values, 23
bipolar transistor, 3
boundary scan, 169
break-down, 164
buffers, 45
buried, 135
bus termination, 47
bytes, 105
capacitors, 1
capture IR state, 171
carry in, 56
carry look-ahead, 57
carry out, 56, 74
center pining, 20
CCD Charge Coupled Device, 113
chip select, 106
circuit testers, 163
Ck, 85
Cklnhibit, 85
clear, 59
clock (inhibiting the), 81
clock pulse, 66
CMOS technology, 4
CMOS-TTL compatibility, 13
COB Chip On Board, 19
colour code, 2
CAS Column Address Strobe, 114
column, 114
combinational, 25
CMRR Common Mode Rejection Ratio, 147
common-control blocks, 101
common-output element, 101
comparators, 41, 55
contact bounce, 142
controllability, 167
corner pining, 20
counter (PLD), 131
counter by 2^n, 189
counters with registers, 91
coverage rate, 165
crosstalk, 30
crystal oscillators, 96
cycle samplers, 77
D flip-flops, 68
damp contact, 141
Darlington transistors, 154
data lockout, 72
data selectors, 50
deasserted, 24
debugging, 161
decoder, 41, 51, 207
decoupling capacitors, 9
delay lines, 94
demultiplexer, 41, 51, 207
DeMorgan's theorem, 25

digital filtering, 140
DIL, 19
DIN41859 standard, 99
diode logic, 5
diodes, 2
direct, 24
divide by 2^n counter, 88
double buffers, 196
double flip-flops, 75, 198
double synchronization, 71
drift, 147
drivers, 41
driving motors, 155
DTL Diode Transistor Logic, 6
DIL Dual In Line, 19
dual ported memory, 106, 113
dynamic characteristics, 5
dynamic logic, 14
dynamic parallel registers, 206
dynamic RAM, 113
dynamic SR flip-flops, 73
ECL, 15
EDAC error detection and correction, 54
edge triggered, 72
EEPROM Electrically Erasable PROMs, 108, 110
electro-mechanical relays, 155
electrostatic discharges, 18
elementary arbiter, 61
enable, 41, 46
encoders, 41, 207
EPLD Erasable PLD, 121, 136
EPROM Erasable PROM, 107
EDC Error Detection and Correction, 54
F Fast, 7
fall through, 116
fall times, 5
fan-in, 6, 8
fan-out, 6, 8
faulty voltage supply, 30
FPGA Field Programmable Gate Arrays, 137
FPLA Field Programmable Logic Array, 121
FIFO First In First Out, 105, 116, 201, 116
Flash memories, 108, 110
flip-flop, 59
floating Gate MOS, 107
FPGA, 137
FPLAs, 121
function generators, 163
functional pseudo-inverting, 28
GaAs, 15
GALPAT galloping pattern, 176
galvanic insulation, 139
gate arrays, 33
gate, 25, 41, 42
GAL Generic Array Logic, 121, 136
glitches, 30, 164, 180
gold circuit, 165
Gray-binary conversion, 144
ground planes, 177
HDL Hardware Design Languages, 166
HC High speed CMOS, 7
high (state), 23
hold time, 69
IEC-617, 99
IFLs, 121
implicit access, 105
inactive, 24
quadrature encoder, 145
inductances, 2
infant mortality, 164
initial state, 168
instrumentation amplifiers, 149
IFL Integrated Fuse Logic, 121
inverted, 3, 24
inverting circle, 24
IrUpdate state, 171
JK flip-flop, 71
JTAG Joint Test Action Group, 169
Karnaugh maps, 37
keyboards, 143
kilo, 105
kilobytes, 105
latch, 62, 84, 206
LCA Logic Cell Array, 137
LCD Liquid Crystal Displays, 153
LFSR Linear Feedback Shift Register, 172
LIFO Last In First Out, 116
line, 23
linear registers, 92
lines, 25
load, 62
logic analyzer, 162
logic probe, 162
logic signal, 30
logic tests, 165
low load input, 44
low (state), 23
LS Low-power Schottky, 7
LT Lamp Test, 52
macrocell, 135
matched, 149
mega, 105
megabytes, 105
metastability, 69
metastable state, 61
microcontroller outputs, 152
MIL-STD806b standard, 99
monostable multivibrator, 97
MOS inverter, 4
MOS technology, 4

MOS transistor, 4
MTBF Mean Time Between Failures, 165
MTBM Mean Time Between Metastables, 70
MTTF Mean Time To Failure, 165
MTTR Mean Time To Repair, 165
multi-phase clock, 67
multiplexing LCD, 154
multiplexors, 41, 50, 207
multiplier, 57
NAND, 27
negative logic, 23
NMOS, 14
noise filtering, 139
noise immunity, 12
noise, 147
non retriggerable, 97
non volatile RAM, 106, 111
NPN, 3
observability, 167
OC Open Collector, 10
off-line, 165
offset current, 147
on-line, 165
OTPROM One Time PROMs, 108
one's catcher, 75
one-shot, 97
op-amp, 146
open collector drivers, 46
open collector outputs, 208
open collector, 10, 45
open drain, 13, 45
open emitter, 45
open source, 13, 45
operational amplifier, 146
optical coupler, 139
oscilloscope, 162
OE Output Enable, 107, 112
output inhibition, 41
overvoltages, 139
Page-addressed EPROM, 110
PAL® Programmable Array Logic, 117
parallel registers, 84
parametric tests, 165
parity generator, 41, 54
PSA Pattern Signature Analysis, 172
permanent, 164
phantom key, 143, 215
photo-, thermo- or piezo-resistors, 141
photodiode, 141
pin grids, 20
PISO Parallel In Serial Out, 80
PNP, 3
positive logic, 23
power drivers, 45
power-up reset, 98
pre-ageing, 164
precharacterized, 33
precharge time, 115
preset, 59
printed circuits, 22
priority encoders, 53
probe card, 163
program mode, 108
PLA Programmable Logic Array, 119, 121
PLD Programmable logic device, 35, 117, 122
programmable logic device, 35
programmable read-only-memory, 36
Programmable ROM, 107
PROM programmer, 107
PROM Programmable Memory, 36
propagation time, 5, 30, 69
propagation times,
protective diodes, 16
pseudo-random counters, 91
PRPG Pseudo-Random Pattern Generation, 172
pull-down resistor, 1, 45
pull-up resistor, 1, 45
pulse sampler, 77
pulse triggered, 72
PWM Pulse Width Modulation, 157
pure asynchronous systems, 92
quantified delay, 94
quasi-synchronous, 67
random access, 105
rate multiplier, 184
RC networks delay, 94
RCO Ripple Count Out, 81, 89
ROM Read only memories, 106
read pulse, 112
read write memories, 106
read-back registers, 85
real signal, 30
real time clock, 186
redundant covering term, 65
reed type relays, 150
reflection on long lines, 30
refresh, 113
register banks, 113
RTL Resistor Transistor Logic, 6
RTL Register Transfer Languages, 166
reliability, 164
reset 68000, 128
reset, 59
resistors, 1
ring counter, 91, 181
RBI Ripple Blanking Input, 52
RBO Ripple Blanking Output, 52
rise time, 5, 30
RAS Row Address Strobe, 114
row, 114

run-test/idle, 171
SAM Serial Access Memory, 115
sample and hold, 150
scan path, 168
scanned keyboard, 143
Schmitt trigger, 44
Schottky diodes, 2, 3
sea of gates, 34
selects, 46
sensors, 141
serial, 105
serial-parallel registers, 85
set, 59
set-up time, 69
signals, 25
signature analyzer, 163
SIPO, 80
skew, 73
slew rate, 147
SMD, 20
socket time, 165
SPICE, 166
stack, 105
static characteristics, 5
static relay, 155
static signal, 65
static SR flip-flop, 60
stick drawing, 31
subtractors, 56
synchronous counter, 89
synchronous loading, 83
synchronous reset, 83
T flip-flops, 73
TAB tape automated bonding, 19
TAP Test Access Port, 171
Test Clock, 169
Test costs, 165
Test Data In, 169
Test Data Out, 169
test logic reset, 171
test mode select, 169
test points, 168
test vector, 164
three-state gates, 46
three-state outputs, 207
three-state, 11
time, 5
transfer gate, 4
transitory, 164
transparent registers, 206
trickle, 90
tridirectional drivers, 49
triggerable, 97
TTL Transistor-Transistor-Logic, 6
ultraviolet PROM, 107
unidirectional drivers, 48
universal registers, 86
UV EPROM eraser, 107
V-MOS or MOS FET, 154
VCO Voltage Controlled Oscillator, 95
very large scale integrated circuit, 31
volatile memories, 106
voltage detector, 99
voltage-frequency conversion, 152
Wheatstone bridge, 151
wired AND, 10
wired OR, 10
worst case, 6, 30
WE Write Enable, 110
write pulse, 112
Zener diodes, 2
ZIL Zig-zag In Line, 19