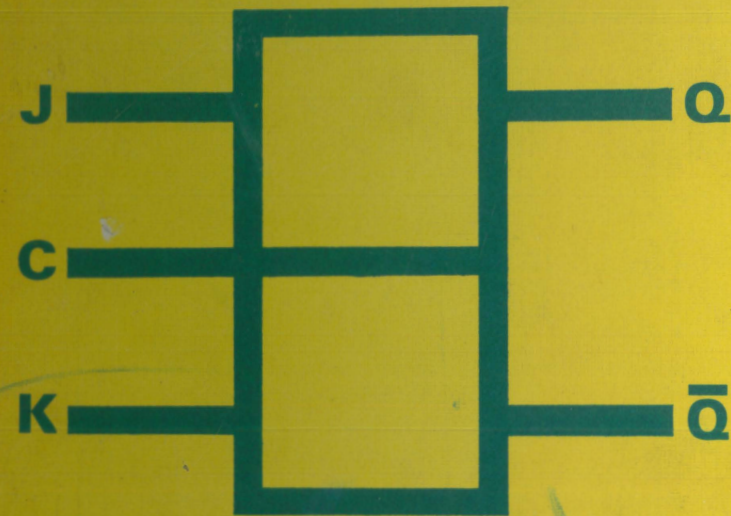


NOEL M. MORRIS

DIGITAL ELECTRONIC CIRCUITS AND SYSTEMS



DIGITAL ELECTRONIC CIRCUITS AND SYSTEMS

Morris

TK
7868
D5
M65

MACMILLAN BASIS BOOKS IN ELECTRONICS

TK Morris, Noel M.
7868 Digital electronic
D5 circuits and systems
M65



DATE DUE

~~TSU~~		
~~Libr. FEB 24 '77~~		
~~TSU~~		
~~Libr. MAR 11 '77~~		
~~ISU~~		
~~Libr. MAY 9 '77~~		
~~TSU~~		
~~Libr. APR 29 '81~~		
~~RT 5/4~~		
~~No Fine~~		
~~TSU~~		
~~Libr. JUN 24 '81~~		
~~ISU~~		
~~Libr. MAY 9 '83~~		
~~RT 5/5~~		
~~TSU~~		
~~Libr. OCT 17 '86~~		
GAYLORD		PRINTED IN U.S.A.

76-03740

© Noel M. Morris 1974

All rights reserved. No part of this publication may be reproduced or transmitted, in any form or by any means, without permission

First published 1974 by
THE MACMILLAN PRESS LTD
London and Basingstoke
Associated companies in New York Dublin
Melbourne Johannesburg and Madras

SBN 333 14862 2

Printed in Great Britain at
The Whitefriars Press Ltd
London and Tonbridge

The paperback edition of this book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out, or otherwise circulated without the publisher's prior consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser

Contents

Foreword	ix
Preface	xi
1 What is Logic?	1
1.1 Gates	1
1.2 Logic signal levels	1
2 Basic Logic Functions	3
2.1 The AND function	3
2.2 The OR function	4
2.3 The NOT function	6
2.4 Complete coin circuit of the vending machine	7
2.5 Negated inputs	8
2.6 The effect of positive and negative logic levels	8
3 NAND and NOR Functions	11
3.1 The NAND function	11
3.2 The NOR function	12
3.3 Why use NAND and NOR gates?	13
3.4 The effect of positive and negative logic on NAND and NOR gates	13
4 Electronic Switches	15
4.1 Semiconductor materials	15
4.2 Semiconductor junction diodes	16
4.3 Charge carrier storage in diodes	18
4.4 The Schottky barrier diode	19
4.5 The bipolar junction transistor (BJT)	19
4.6 A resistor-transistor logic (RTL) NOT gate	22
4.7 'Current-sourcing' and 'current-sinking' gates	24
4.8 Fan-out	25
4.9 Fan-in	26
4.10 Thermal considerations	26

Macmillan Basis Books in Electronics

General Editor:

Noel M. Morris, *Principal Lecturer, North Staffordshire Polytechnic*

LINEAR ELECTRONIC CIRCUITS AND SYSTEMS: *G. D. Bishop*

P015

Digital Electronic Circuits and Systems

Noel M. Morris

Principal Lecturer

North Staffordshire Polytechnic

MACMILLAN

12 Shift Registers and Ring Counters

A register is simply an array of flip-flops used for the storage of binary data, and a **shift register** is one which is designed so that the data may be 'shifted' along the register in either direction, that is, either to the right or to the left. A **ring counter** is a shift register which is connected in the form of a continuous ring, the input to the 'beginning' of the ring being a logical function of the signals at one or more points in the register.

12.1 A serial-input, serial-output shift register

Two basic circuits for shift registers are shown in figure 12.1. In the circuits shown, the data to be stored in the register is shifted into the register in a serial fashion from the left-hand end, and is shifted out at the right-hand end. Let us consider the operation of figure 12.1(a).

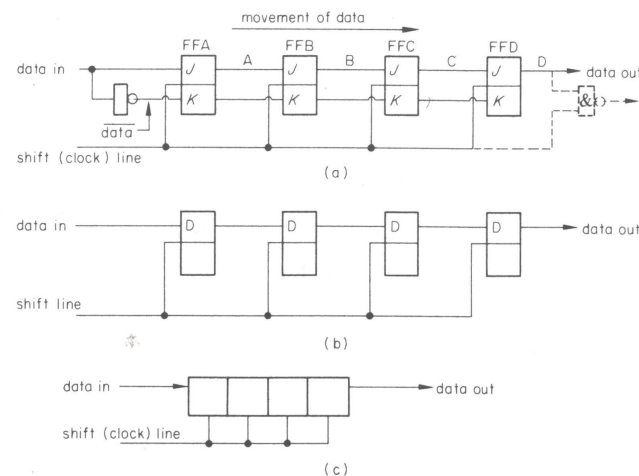


Fig. 12.1 Serial-input, serial-output shift registers

Assume that outputs *A*, *B*, *C*, and *D* are initially zero. The data presented at the input line is logically inverted by the NOT gate, so that complementary signals are presented to the *J* and *K* lines of the first flip-flop. Suppose now that a '1' signal is applied to the DATA IN line and that a pulse is applied to the clock (SHIFT) line of the register. At the end of the clock pulse the '1' is transferred to output *A*. Meanwhile, outputs *B*, *C*, and *D* remain at '0', since the input signals applied to those flip-flops were '0' during the period that the clock pulse was at the '1' level. The result of the above operation is shown in the first two rows of table 12.1. The '1' injected above is marked with an asterisk in table 12.1 to indicate its movement through the register.

Table 12.1

clock pulse	signal on data line	state of outputs			
		<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
initial condition		0	0	0	0
1	1*	1*	0	0	0
2	0	0	1*	0	0
3	0	0	0	1*	0
4	1	1	0	0	1*
5	1	1	1	0	0
6	1	1	1	1	0

one complete shift cycle

Thus we have shifted a '1' from the data line into FFA. Also we have shifted the '0' formerly stored in FFA into FFB, as well as shifting the '0' in FFB into FFC, and the '0' in FFC into FFD. Hence *all* the information stored in the register has been transferred one step to the right.

If, now, the signal on the data line is reduced to zero and another shift pulse is applied, then this '0' is shifted into FFA at the end of the pulse. This change is brought about by shifting a '1' into the $\bar{A}$ output and at the same time the 1* is shifted into FFB.

Since the shift register has four stages, it can store a binary **word** of four bits, and requires four clock pulses to shift a new word into the register. Hence, after four clock pulses we have (i) serially shifted out of FFD the word 0000, and (ii) serially shifted in the word 1001. In the register in figure 12.1(a), we say that the information has been **shifted up** or **shifted to the right**.

We can continue to shift data into and out of the register, and table 12.1 is continued beyond one cycle to show the effect of shifting the first two 1's of the next word into the store. This has the effect of shifting the two right-hand bits stored at the end of the first cycle of the register.

In the form of circuit shown in figure 12.1(a), the output from FFD will be a continuous signal, either a '1' or a '0', for the whole of the clock cycle. If we need the output in the form of a pulse train, then it can be obtained by AND-gating the

output of FFD with the clock pulse as shown in broken lines in figure 12.1(a). The interconnections between the flip-flops can be simplified by the use of D-type elements, as shown in figure 12.1(b). These flip-flops have internal NOT gates which generate the **DATA** information internally.

A symbol used to represent serial registers is shown in figure 12.1(c).

12.2 A reversible serial shift register

The data stored in the registers described above can only be shifted to the right. Some applications call for registers which can shift data either to the right or to the left at the command of a signal applied to a control line. The circuit in figure 12.2 is the basis of one type of reversible shift register. In this circuit, a '1' applied to the control line enables G1 to operate and it also inhibits G2. This permits data in the flip-flop to the *left* of FFB (FFA in figure 12.1) to be entered into FFB. When a '0' is applied to the right/left control line, G2 is opened and G1 is inhibited. This allows data stored in the flip-flop to the *right* of FFB (FFC in figure 12.1) to be shifted into FFB. Readers will note that the AND-OR combination in figure 12.2 can be replaced directly by three NAND gates (also see figure 7.6).

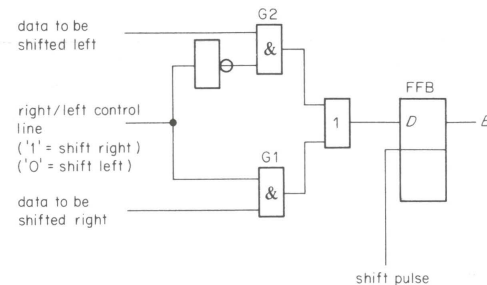


Fig. 12.2 One form of input arrangement to a stage of a reversible serial-input shift register

A three-stage reversible shift register using the circuit in figure 12.2 is shown in figure 12.3. When shifting right, data is fed in from the left-hand input into FFA, and is shifted serially into FFB, and then into FFC. When shifting left, data is fed serially into FFC from the right-hand input, and then is transferred into FFB and, finally, into FFA.

Once again, if the output is to be in the form of a pulse train, the appropriate outputs must be AND-gated with the clock pulse.

12.3 Parallel read-out

In some applications it is necessary to simultaneously observe the states of all the stages, that is, the output is required in parallel form. In the registers described so

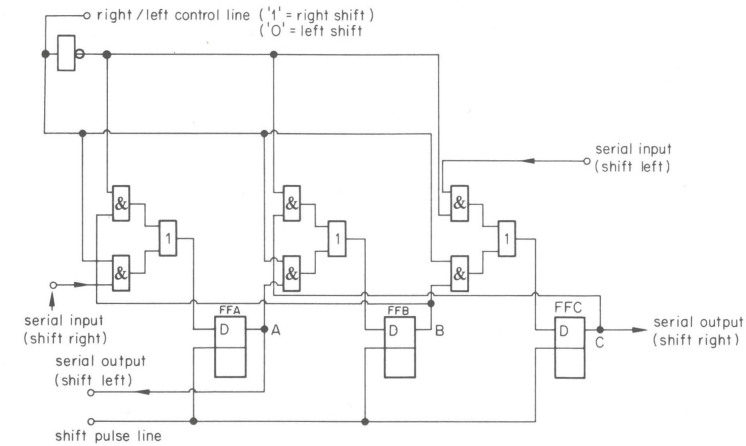


Fig. 12.3 A reversible shift register

far, this can be achieved if we employ a circuit in which the *Q* output of each flip-flop is available. In an IC pack this has the effect of limiting the maximum register length since, in the case of figure 12.1, we need a minimum of five pins on the IC pack, one each for the data in, data out, clock, power supply, and earth lines. In a 14-pin DIL pack there remain nine other connections for the additional input, output and control functions.

12.4 Parallel-input shift registers

As with binary counters, data can be preset into the register in a parallel mode. A logic network for carrying out parallel entry of data was described in section 11.5 and can be used in connection with shift registers.

12.5 Dynamic shift registers

A unique property of the MOSFET is the facility that it can be used as a dynamic memory element, as was described in section 8.10. When used in conjunction with other MOSFET's, simple and reliable shift registers can be constructed in monolithic IC form, the basis of one type being shown in figure 12.4.

The memory elements are the inverter stages G1, G2, G3, et cetera, together with the parasitic gate capacitors C1, C2, C3, et cetera. A complete shift register stage comprises two of these memory elements, together with two MOST switches which are turned ON and OFF by signals ϕ_1 and ϕ_2 . The control lines ϕ_1 and ϕ_2 are energised by a **two-phase pulse supply**, the timing of ϕ_1 and ϕ_2 being such that the two switches associated with each memory are never closed simultaneously.

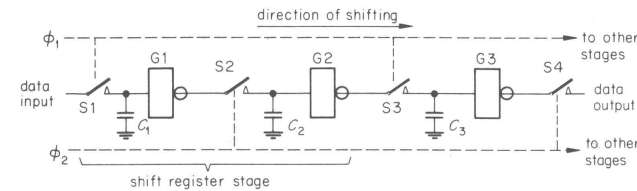


Fig. 12.4 The basis of a MOS dynamic shift register

When ϕ_1 is energised by a '1' signal, switch S1 is closed and the data on the input line is transferred to capacitor C1. Let us suppose that the data signal is logic '1'. This signal charges C1, and the inverting action of G1 causes the output of that gate to be '0'. At the end of the ϕ_1 pulse S1 opens, and a logic '1' signal is applied to line ϕ_2 . This causes S2 to close, and the '0' signal at the output of G1 discharges C2. The inverting action of G2 causes the output of that gate to be logic '1'. Hence, in a complete cycle of events ϕ_1 and ϕ_2 become logic '1' in turn, and cause the input data to be transferred into the first stage of the register.

During the period of time that input data is transferred into G1, the data stored in G2 is transferred into G3. Clearly, signals ϕ_1 and ϕ_2 cause all the data stored in the register to be shifted to the right.

A diagram of one section of one form of dynamic shift register is shown in figure 12.5. In this circuit TR1 replaces S1 in figure 12.4, TR4 replaces S2, and TR7 replaces S3. Gate G1 in figure 12.4 is replaced in figure 12.5 by TR2 and TR3, and G2 is replaced by TR5 and TR6.

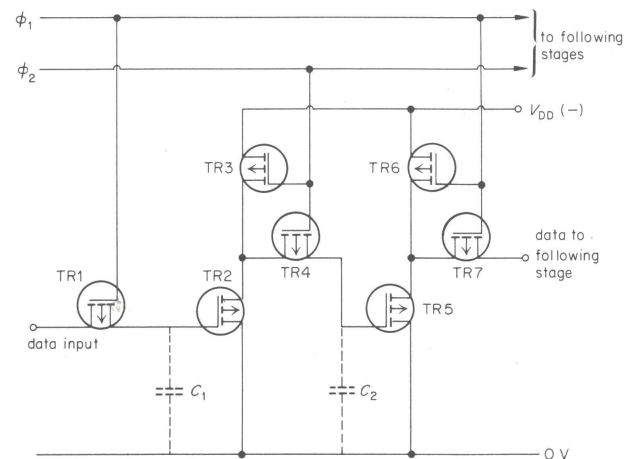


Fig. 12.5 One stage of a dynamic shift register

12.6 Ring counters

A ring counter is simply a shift register whose input is derived directly from its output, in the manner shown in figure 12.6.

Let us assume that stage A of figure 12.6(a) initially stores a '1', and that all other outputs are zero. The first shift pulse applied to the counter causes the '1' to move from stage A to stage B, and the 0's in stages B and C to move into stages C and D, respectively. Due to the feedback connection, the '0' in D is fed back into stage A. In this way the single '1' circulates continuously around the register in the manner shown in figure 12.6(b).

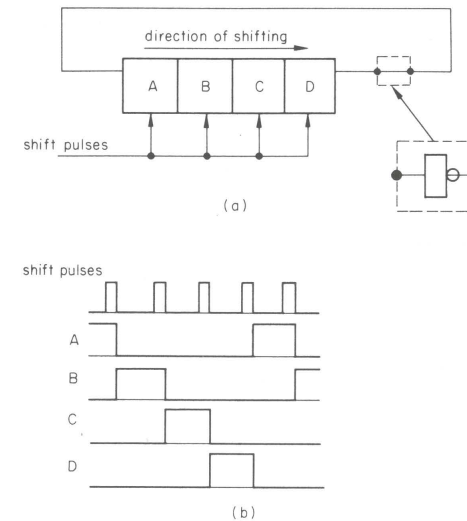


Fig. 12.6 A ring counter

A feature of the ring counter is that the value of the number stored is easily decoded, since when a '1' appears at output A it is equivalent to a count of zero, when it appears at B it is equivalent to a count of unity, when at C it is equivalent to a count of two, and when at D three pulses have been applied. This means that the **cycle length** of the code generated is four for a four-stage shift register. A ring counter which operates in a decimal code would need ten flip-flops.

As we have seen in the chapter on counters, it is possible to generate a cycle length of $2^4 = 16$ with four binary stages. Evidently, although the ring counter provides us with a simple method of 'reading' the number stored, it is uneconomic in its use of flip-flops.

Starting conditions in a ring counter

When the ring counter is first switched on, there is no guarantee that stage A will contain a '1' and that the other stages will contain zeros. A method sometimes adopted in circuits using this type of counter is to correct the situation at the end of the first complete cycle of operations by including an additional gate in the feedback network. The operation will be correct after the first cycle. The operation of such a circuit is outlined below.

We have seen that a '1' must be fed into stage A following the condition that the outputs of stages A, B, C, and D are *all* zero. If we use the feedback arrangement in figure 12.7(a) in which the input of stage A is energised by a NOR gate whose inputs are A, B, C, and D, then a '1' will be fed into A following the condition that $A = B = C = D = 0$. Alternatively we can use the feedback arrangement in figure 12.7(b).

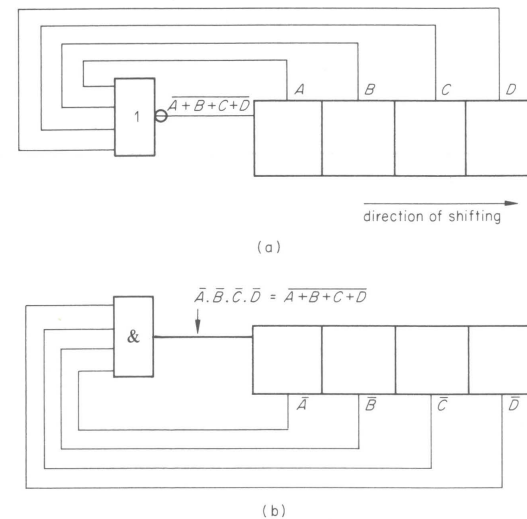


Fig. 12.7 Methods of ensuring the correct starting conditions in a ring counter

A twisted ring counter

By including the NOT gate shown in the inset to figure 12.6(a) in the feedback loop of the counter, the input to stage A is the logical function $\bar{D}$. This has the effect of increasing the cycle length to eight, that is, the code cycle length is double that of an otherwise equivalent ring counter. Offset against this is the fact that it is slightly more difficult to decode into decimal the binary value stored in the counter. The

code sequence generated by the counter, known as a **twisted ring counter** due to the inversion in the feedback loop, commencing with the group 0000, is given in table 12.2.

Table 12.2

decimal value	A	B	C	D
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
	0	0	0	0

In this case the logical complement of D is fed into A, so that A becomes '1' following the condition the D is '0', and becomes '0' following the condition that D is '1'.

Twisted ring counters are often known as **Johnson code counters**, and the code in table 12.2 is a 4-bit Johnson code. The correct logical conditions for a Johnson code sequence can be generated after one complete cycle by energising the input to stage C from a network which develops the logical function $B \cdot (A + C)$, rather than by energising it directly by signal B. Readers may like to test their skill by verifying this fact.

12.7 Chain code generators

The binary sequence which is generated by a chain code generator has no apparent logical pattern and, in fact, appears to be a random sequence of binary numbers. The code is generated by a shift register whose input is derived either from a gate or from a network which develops a more or less complicated logical function of the outputs from the register.

A basic form of chain code generator is shown in figure 12.8, and comprises a 4-bit shift register whose input is supplied by the output from a NOT-EQUIVALENT gate. The input is fed in serially at the left-hand end, and the output may either be fed out in a serial or in a parallel mode. The code sequence generated by figure 12.8 is given in table 12.3, and we see from the table that a '1' is fed into stage A following the condition that $C \neq D$.

Assuming that the cycle commences with $A = 1$, and all other outputs are equal to zero then, since $C \equiv D$, a '0' is fed into stage A when the first shift pulse is applied. Once more, after the first clock pulse, $C \equiv D$ so that a second '0' is fed into stage A by the second clock pulse. After this shift pulse condition $C \neq D$ exists, and

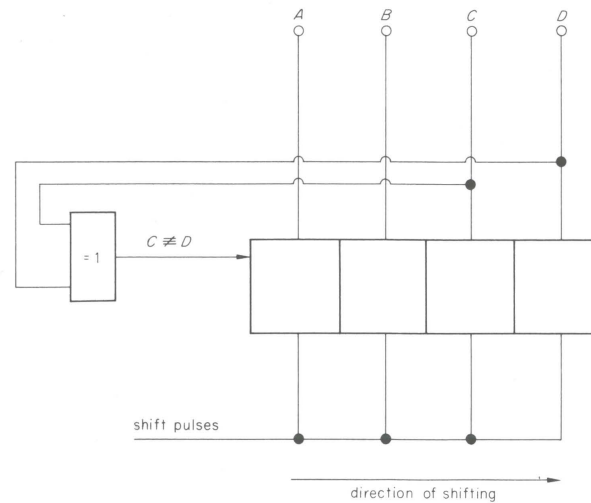


Fig. 12.8 A chain code generator

Table 12.3

shift pulse	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
initial condition	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	1	0	0	1
4	1	1	0	0
5	0	1	1	0
6	1	0	1	1
7	0	1	0	1
8	1	0	1	0
9	1	1	0	1
10	1	1	1	0
11	1	1	1	1
12	0	1	1	1
13	0	0	1	1
14	0	0	0	1

a '1' is returned to the input of the register. This '1' appears at the output of stage A after the third clock pulse has been completed. The code sequence of the counter is completed after fifteen shift pulses have been applied.

This type of shift register counter utilises the flip-flops to greater effect than counters described hitherto since fifteen of the sixteen possible code combinations are used. The code group missing in table 12.3 is the combination 0000; if the outputs assume the 'all zero's' condition at the instant of switch-on, then there is no apparent movement of data since the zeros continuously circulate through the counter. In counters of this type it is necessary to include additional logic to prevent the 'all zero's' condition arising.

Had we used an equivalence gate in the feedback loop of figure 12.8, we should have generated a code sequence of length fifteen which includes the 'all 0's' condition, but which excludes the 'all 1's' state. Readers may like to verify this fact.

Since the pattern generated by the network is not a truly random pattern, we call it a **pseudo-random binary sequence** (PRBS). Applications of PRBS generators range from illumination control (for example, Christmas tree lighting) to electronic system testing.

The cycle length generated by chain code generators depends not only on the number of stages over which the feedback is applied, but also on the way in which the signal is derived. The maximum code length for N stages is $(2N-1)$, which is 7 for three stages, 15 for four stages, 31 for five stages, and so on.