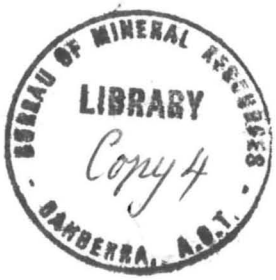
COMMONWEALTH OF AUSTRALIA

DEPARTMENT OF NATIONAL DEVELOPMENT

BUREAU OF MINERAL RESOURCES, GEOLOGY AND GEOPHYSICS

Record 1971/41



ELECTRONIC LOGIC AND COUNTING — A BASIC INTRODUCTION

by

K.J. Seers

Record 1971/41

ELECTRONIC LOGIC AND COUNTING - A BASIC INTRODUCTION

by

K.J. Seers

ELECTRONIC LOGIC AND COUNTING - A BASIC INTRODUCTION

by

K.J. Seers

CONTENTS

FOREWORD

This Record was written primarily to satisfy the need to inform geophysicists and technical officers of the fundamentals of electronic logic used in modern instrumentation. A secondary purpose was to provide a simple text that could be used as a basis for internal qualifying examinations for technical officers in the laboratories of the Bureau of Mineral Resources.

1. ELECTRONIC LOGIC PRINCIPLES

1.1 Introduction

In many electronic systems, expecially those of a digital nature, logic circuits play an important part and ought to be thoroughly understood by those responsible for equipment maintenance. The purpose of this Record is (1) to provide an explanation of the basic logic functions and (2) to describe some circuits used to realize these functions.

1.2 Logic Functions and Truth Tables

The following example shows a simple logical operation.

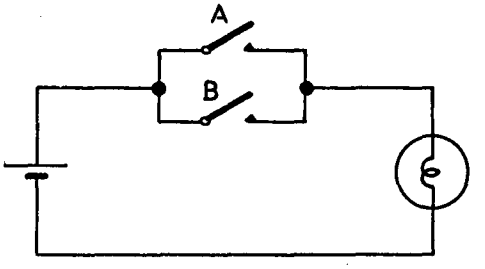


Fig. 1

In Fig. 1, the lamp will light if contact A or contact B (or both) close. This is called an OR operation and is summarized in the following table, called the truth table for the circuit of Fig. 1:

Contact A	Contact B	Lamp
Open	Open	Off
Open	Closed	On
Closed	Open	On
Closed	Closed	On

If the contacts were in series, rather than in parallel, a different truth table would result:

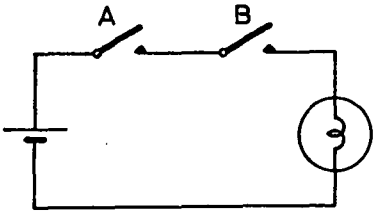


Fig. 2

Contact A	Contact B	Lamp
Open	Open	Off
Open	Closed	Off
Closed	Open	Off
Closed	Closed	On

This is called an AND operation because the lamp is on only if contact A and contact B close.

Any number of contacts could have been used in the foregoing examples, and could have been connected in series, parallel, or series-parallel. In general, each different arrangement would produce a different truth table, for the truth table must show the state of the lamp for every possible combination of contact states. For example, the lamp in Fig. 3 lights if contact C or contacts A and B close. Notice that the truth table for this circuit must have eight rows to allow for all possible contact states.

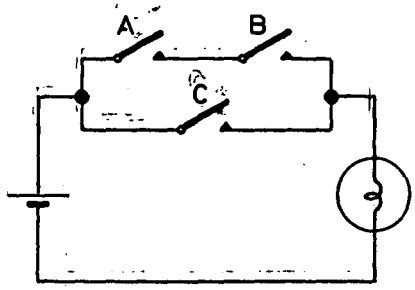


Fig.-3

A	B	C	Lamp
Open	Open	Open	Off
Open	Open	Closed	On
Open	Closed	Open	Off
Open	Closed	Closed	On
Closed	Open	Open	Off
Closed	Open	Closed	On
Closed	Closed	Open	On
Closed	Closed	Closed	On

In general if there are n contacts, the truth table must have 2^n rows.

Of course the application of truth tables is not limited to sets of contacts. Consider the statement: "if the counter is filled and if a sync pulse occurs, the system will re-cycle". The logic implied in the statment corresponds to the logic used in Fig. 2; i.e. "if the counter is filled" corresponds to "if contact A is closed"; "if a sync pulse occurs" corresponds to "if contact B is closed" and "the system will re-cycle" corresponds to " the lamp will light". Because the logic in both cases is the same ("and" logic), it should be possible to represent each case by the same truth table. This can be done if a symbol is chosen to signify the presence of a logical condition or result, and another symbol chosen to represent the absence of such condition or result.

By convention, the symbol 1 is used to signify presence (or "truth") and the symbol 0 is used to signify absence. Hence "contact A is closed" and "the counter is filled" would both be represented by 1: "contact A is open" and "the counter is not filled" would both be represented by 0. If the logical conditions (state of contact A etc.) and logical results (state of lamp etc.) are represented by X, Y, and Z, the following general truth table for the AND logic function results.

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

X, Y represent logical condtions

Z represents logical results.

Fig. 4. AND Truth Table

Fig. 4 states concisely that the logical result (Z) is true, or 1, only if both logical conditions (X,Y) are true. The conditions X and Y are often called the variables or inputs to the logic operation and the result is called the output.

Thus Fig. 4 shows the outputs for all possible input combinations for a two-input AND operation.

Similarly, the generalized truth table for a two-input OR operation is shown in Fig. 5.

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 5. OR Truth Table

The logic AND and logic OR operations together with the NOT (or inversion) operation form the basis of all electronic logic systems.

The NOT function operates on one input only and merely negates (or complements, or inverts) the input state as shown in Fig. 6.

X	Z
0	1
1	0

Fig. 6. NOT Truth Table

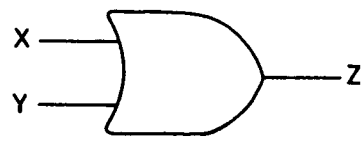
There are two noteworthy points:

(1) The foregoing logic systems are "two-level" systems; i.e. all variables may be uniquely represented by one of two symbols (0 or 1). Theoretically, "multi-level" systems are possible, but they are rare.

(2) The symbols 0 and 1 are only symbols chosen for convenience. Truth tables must not be construed as operations in binary arithmetic.

1.3 Basic Gate Symbols*

Certain symbols are used to represent the various logic operations on flow or block diagrams. The symbol, truth table, and possible circuit are set out below. Note that the symbol for the NOT operation is, for each operation considered so far except for the circle at the input or output, the same as the conventional symbol for an amplifier.



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

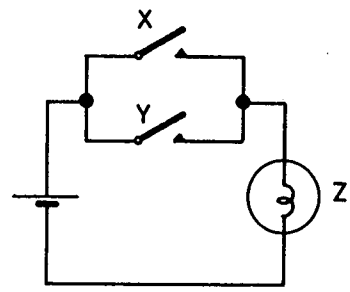
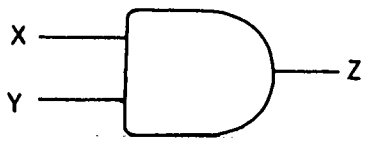


Fig. 7. Two-input OR



X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

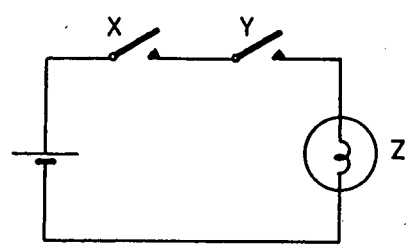
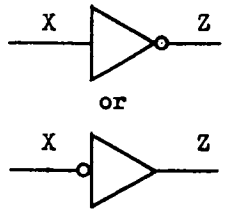


Fig. 8. Two-input AND

---

* Gate symbols used in this text are based on MIL-STD-806B "Graphic Symbols for Logic Diagrams".



X	Z
0	1
1	0

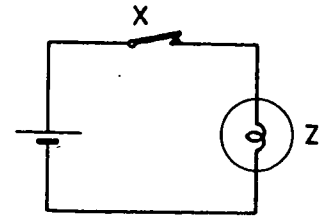


Fig. 9 NOT

In Figs. 7, 8, 9 X = Y = 1 = contact energized,

X = Y = 0 = contact unenergized

Z = 1 = Lamp on

Z = 0 = Lamp off

The circuits that perform logic operations are called gates. The input-output characteristics of a gate are completely specified by its truth table, and the gate may be depicted on a block diagram by the appropriate logic symbol.

1.3.1 Wired Logic Functions

Some specific circuits allow the outputs of two or more gates to be connected together to give a "wired-OR" or "wired-AND" function. The corresponding symbols are shown in Figs. 7a and 8a.

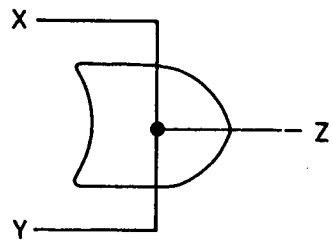


Fig. 7a. Wired-OR

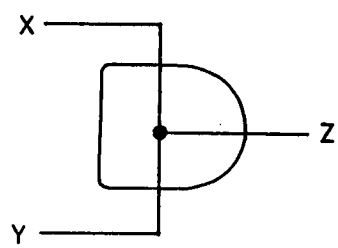


Fig. 8a. Wired-AND

These symbols do not indicate the existence of any circuitry other than the connection of two gate outputs to a common output line. The symbol is drawn around the connection merely to indicate the resultant logic.

There are two combinations of logic functions used so frequently as to warrant separate mention. These give the NOR (not or) and the NAND (not and) function, as shown in Figs. 10 and 11.

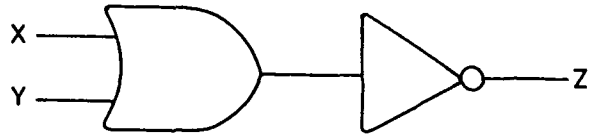


Fig. 10. Two-input NOR

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Fig. 11. Two-input NAND

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

1.4 Positive and Negative Logic

In the majority of electronic logic systems, each of the two logic levels corresponds to one of two specified voltage - or, less often, current - levels.

In the case of a positive-logic system, the higher voltage corresponds to the logic 1 state; whereas for a negative-logic system, the lower voltage corresponds to the logic 1 state. Mixed-logic systems are common. In these, the logic 1 state may correspond to either the higher or the lower voltage, depending on which gate is being considered.

Figs. 12 to 15 show the truth tables re-written in terms of voltage levels (H = higher, L = lower) for both positive and negative logic systems.

X	Y	Z
L	L	L
L	H	H
H	L	H
H	H	H

Fig. 12. Positive two-input OR

X	Y	Z
L	L	L
L	H	L
H	L	L
H	H	H

Fig. 13. Positive two-input AND

X	Y	Z
H	H	H
H	L	L
L	H	L
L	L	L

Fig. 14. Negative two-input OR

X	Y	Z
H	H	H
H	L	H
L	H	H
L	L	L

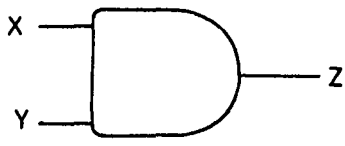
Fig. 15. Negative two-input AND

Examination of Figs. 12 to 15 shows that the truth table for the positive OR function contains the same combinations of levels as does the truth table for the negative AND function; and the truth table for the positive AND function contains the same combinations of levels as does the truth table for the negative OR functions. This illustrates the important and basic fact that the circuit for a positive AND gate may be identical with that for a negative OR gate, and in fact there is no means of distinguishing which function the gate is meant to perform, unless the logic system (positive or negative) is specified. The same is true for the corresponding NAND and NOR functions.

It is obviously desirable that there be no ambiguity in symbolizing a logic function in a block diagram; i.e., a positive OR gate should be distinguishable from a negative AND gate. More than one system has been devised to accomplish this, but the one in general use - and probably the best -

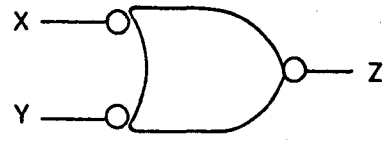
is the following: except for the inverter, only the two basic gate symbols are used (D= or, D= and). However, if, at a gate's input or output, the logic 1 (for whichever function the gate performs) corresponds to the lower voltage level, then a small circle is drawn on that input or output. This results in eight distinct gates as shown in Fig. 16. The gates are grouped in pairs, the same truth table (expressed in voltage levels) applying to each member of a given pair; i.e. the same circuit could be used for each function in the pair.

Positive AND

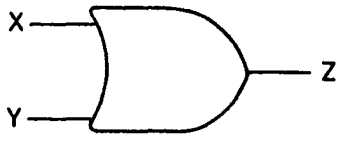


X	Y	Z
L	L	L
L	H	L
H	L	L
H	H	H

Negative OR

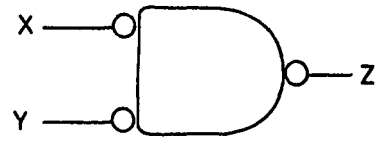


Positive OR

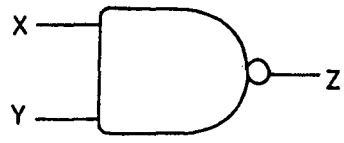


X	Y	Z
L	L	L
L	H	H
H	L	H
H	H	H

Negative AND

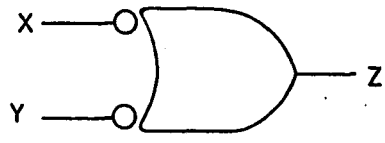


Positive NAND

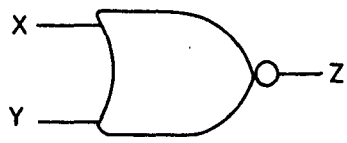


X	Y	Z
L	L	H
L	H	H
H	L	H
H	H	L

Negative NOR



Positive NOR



X	Y	Z
L	L	H
L	H	L
H	L	L
H	H	L

Negative NAND

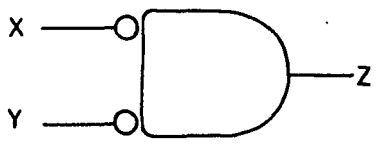


Fig. 16

1.5 Use of Standard Gate

A careful study of Fig. 16 will reveal that any of the functions tabled may be obtained from any other function by inserting inverters at the inputs or output as required. For example, starting with a positive NAND gate, an inverter at the output produces the positive AND function; inverters at each input produce the positive OR function; and inverters at output and inputs produce the positive NOR function.

If one input of the positive NAND gate is held at the higher voltage level, the gate acts as an inverter for the other input. The same applies to the negative NAND gate if one input is held at the lower voltage level.

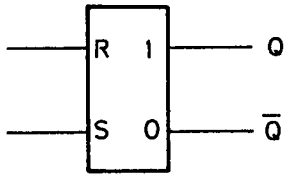
Hence, any of the logic functions so far considered may be realized by the use of NAND gates alone, and although this may not give the most economical solution in terms of hardware, it does have the advantage that the one standard circuit can be used throughout the system.

1.6 The Logic of Flip-flops

The bistable multivibrator, commonly (but erroneously) called the flip-flop, is extensively used in logic circuits. A flip-flop has two complementary outputs (i.e. when one is high the other is low) and up to five separate inputs. It has two stable states - a given output high or low - and will only change from one state to the other when the required input conditions are satisfied. For some input conditions, the flip-flop will not change state at all - hence it is useful as a "memory" element.

There are many types of flip-flops, the main ones being listed hereunder:

1.6.1 The R-S Flip Flop



Rt+1	St+1	Qt+1	$\bar{Q}$t+1	
0	0	1	1	not allowed
0	1	1	0	
1	0	0	1	
1	1	Qt	$\bar{Q}$t	no change

Fig. 17. R-S Flip-Flop. Symbol and Truth Table

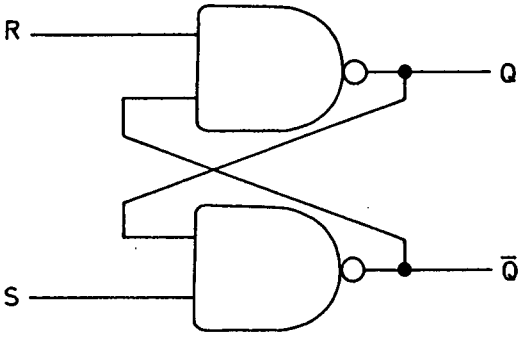
The flip-flop outputs are labelled Q and $\bar{Q}$, the use of the bar being to indicate a complementary output. Notice, however, that complementary outputs do not occur for every input combination. In the case of 0 at each input, both

outputs assume the 1 state. In normal operations this is not allowed, and precautions are taken to ensure that both inputs do not assume the 0 state simultaneously.

The truth table presumes an initial state at some arbitrary time t, and shows the resultant state at a later time, t + 1, when the given input conditions are applied. If both inputs assume the 1 state, there can be no change in the output state; i.e., the state at time t is maintained.

The letters R and S stand for reset and set, the reset state of the flip-flop being defined as that state when output Q is at the 1 level. Hence, to reset the flip-flop, apply a 0 to the R input and a 1 to the S input. Similarly to set the flip-flop, apply a 1 to the R input and a 0 to the S input.

A simple R-S flip-flop may be made by cross-coupling two gates. This is sometimes called a latch circuit and is shown in Fig. 18.

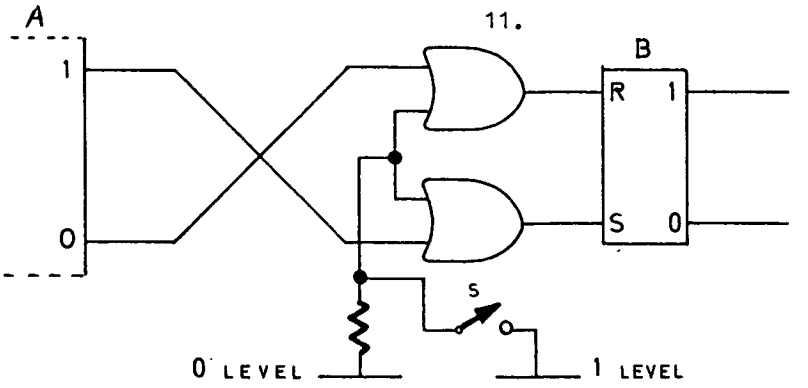


Rt+1	St+1	Qt+1	$\bar{Q}$t+1
L	L	H	H
L	H	H	L
H	L	L	H
H	H	Qt	$\bar{Q}$t

Fig. 18 Latch Circuit and Truth Table

The latch circuit truth table can easily be derived from the truth tables for each positive NAND gate.

One application of the R-S flip-flop is to sample and hold the state of another flip-flop, even though this latter flip-flop may subsequently change state; i.e., the R-S flip-flop "remembers" the state of the sampled flip-flop at the sampling time. This may be achieved by connecting the R and S inputs to the complementary outputs of the sampled flip-flop at the time the sample is required; then switching both R and S inputs to the logic 1 level until a further sample is required. A circuit for this operation is shown in Fig. 19.



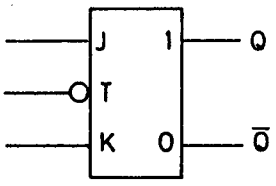
Opening S stores contents of A in B. Closing S allows B to hold its state regardless of changes in A

Fig. 19. Sample and Hold Using R-S Flip-Flop

1.6.2 The J-K Flip-Flop

A disadvantage of the R-S flip-flop is the need to prevent both inputs going low simultaneously. The J-K flip-flop overcomes this disadvantage, and is generally more versatile.

It has three inputs, designated J, K, and T, and will not change state until the voltage level on the T, or trigger, input changes. In positive logic systems a negative going transition at the T input is usually preferred. Once this transition has occurred, the J-K flip-flop will assume a state entirely dependent on the conditions at the J and K inputs. Often there are more than one J and K inputs, the set of J inputs forming an AND gate and similarly for the set of K inputs. The final state of the flip-flop then depends on the output states of the J and K gates, as shown in Fig. 20.



J_t	K_t	Q_{t+1}	$\overline{Q}_{t+1}$
L	L	Q_t	$\overline{Q}_t$
L	H	L	H
H	L	H	L
H	H	$\overline{Q}_t$	Q_t

For negative-going transitions at input T at time t.

Fig. 20. Symbol and Truth Table for J-K Flip-flop

Note that with both J and K inputs low, there is no change of state, but with both J and K inputs high there is always a change of state whenever a trigger input is applied. With the J and K inputs complementary, the action is similar to that of the R-S flip-flop, except that a trigger is required.

1.6.3 The T, or Trigger, or Toggle, or Binary Flip-Flop

This is a variant of the J-K flip-flop, in which the J and K inputs are not accessible and are permanently high. The flip-flop then changes state every time a trigger is applied, i.e. it divides by two, and is used extensively

in counting circuits.

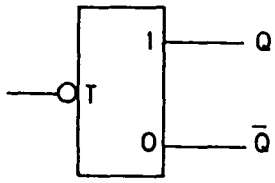
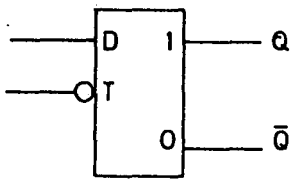


Fig. 21 The T Flip-Flop

1.6.4 The D or Delay Flip-Flop

When a J-K type flip-flop is used for storage (i.e. memory) purposes only, its truth table shows that only one of the J or K states need be specified as the other input is complementary. The D flip-flop may be regarded as a J-K flip-flop with only the J and T inputs accessible, and an internal inverter connected to the K input.



Dt	Qt+1	$\bar{Q}$t+1
H	H	L
L	L	H

For negative-going transition at input at time t.

Fig. 22. The D Flip-flop and Truth Table

1.6.5 Capacitive Coupling at Flip-Flop Inputs

The inputs to flip-flops (as also with gates) may be capacitively coupled. This does not alter the truth tables in any way, although it must be realized that the input conditions are of a transitory nature; i.e., they exist only during the pulse time set by the coupling capacitor and associated circuitry. Because a flip-flop is bistable, a capacitively coupled input results in a stable d.c. output level. The output of capacitively coupled gates, however, can only be a pulse.

1.6.6 Direct Setting and Resetting of Flip-Flops

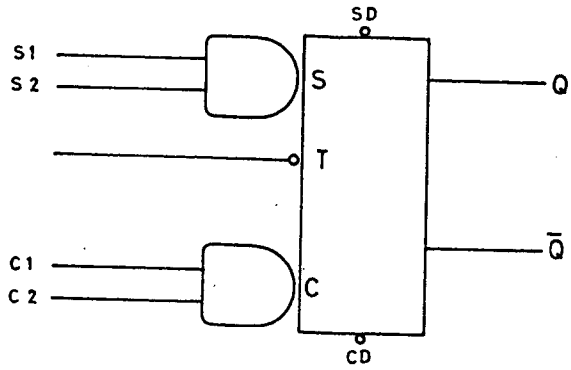
All flip-flops considered so far may be provided with a direct reset or clear input which will turn the flip-flop off; i.e. produce a logical 0 at the 1 output. Similarly a direct set input may be provided which turns the flip-flop on; i.e. produces a logical 1 at the 1 output. Generally these direct (or asynchronous) inputs take precedence over all other inputs, although there may be a preferred level for the toggle input to enable the direct inputs to operate.

1.6.7 The Dual-Rank Flip-Flop

This is a general purpose flip-flop which may be used in lieu of any of the foregoing types. Basically, it consists of two cross-connected flip-flops: a "Master" and a "Slave". There are two synchronous inputs; the "Set" and the "Clear". When the toggle or "Clock" input assumes a given level, say positive, the conditions at the set and clear inputs are stored in the master flip-flop, while the slave is isolated. When the clock input goes negative the master is isolated from the set and clear inputs (so these may then change without affecting the flip-flop state) and the slave is caused to store the contents of the master.

AND gates are usually provided at the synchronous set and clear inputs, and it is the outputs of these gates which are stored by the master flip-flop when the clock pulse goes positive. The term "synchronous" is used of these inputs because they only have effect when a clock pulse occurs.

Clocked flip-flops are usually direct coupled, so that the clock pulse may have any arbitary waveform (sine wave, pulse, sawtooth etc.) provided it passes through the required voltage levels when passing from low to high and vice versa. Asynchronous inputs are also provided, being designated "Clear direct" and "Set direct". These inputs take precedence over all others, provided that, in some cases, the toggle input is high. In some improved clocked flip-flops the condition of the toggle input has no effect on the operation of the asynchronous inputs. Fig. 23 shows the symbol and truth table for such a flip-flop.



Synchronous Entry

(Output changes when T goes low).

S1t	S2t	C1t	C2t	Qt+1
L	X	L	X	Qt
L	X	X	L	Qt
X	L	L	X	Qt
X	L	X	L	Qt
L	X	H	H	L
X	L	H	H	L
H	H	L	X	H
H	H	X	L	H
H	H	H	H	Undetermined

X = H or L

Asynchronous Entry

(Overides synchronous entry and independent of all other inputs)

Cd	Sd	Qt	$\bar{Q}$t
H	H	Qt	Qt
H	L	H	L
L	H	L	H
L	L	H	H

Fig. 23. Clocked Flip-flop

To use the clocked flip-flop as a J-K flip-flop, connect $\bar{Q}$ to S1 and Q to C1. This results in the truth table of Fig. 24, which is seen to be analogous to that of Fig. 20.

S2t	C2t	Qt+1
L	L	Qt
L	H	$\bar{L}$
H	L	H
H	H	$\bar{Q}$t

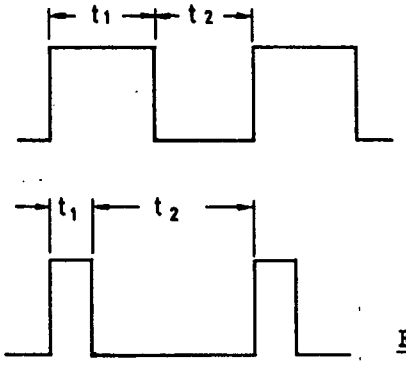
Fig. 24. J-K Mode Operation

In the J-K mode, the clocked flip flop is useful in counting circuits.

1.7 The Astable Multivibrator

Strictly, this is not a logic element, but it is often found in logic systems. In contrast to the bistable multivibrator, which has two stable states, the astable or free-running multivibrator has two quasi-stable states. It remains in one state for time t1 and rapidly changes to the other state, in

which it remains for time t2; it then reverts to the first state, and so on. It is thus a generator of square or rectangular waveforms as shown in Fig. 24.



(a) t1 = t2 Square Waves

(b) t1 ≠ t2 Rectangular Waves

Fig. 24. Astable Multivibrator Waveforms

One application of the astable multivibrator to logic systems is to provide the synchronizing (or clock) waveforms for clocked flip-flops. The clock frequency, f, is $1/(t1 + t2)$.

The astable multivibrator does not require external trigger pulses, although, if desired, it may be locked to an external source having a frequency larger than f.

Complementary outputs are usually available.

1.8 The Monostable Multivibrator

Otherwise known as a univibrator, single-shot, delay, or one-shot, the monostable multivibrator has one stable and one quasistable state. It will remain in its stable state until it receives a trigger signal, when it assumes the quasi-stable state for a time t, after which it reverts to the stable state until the next trigger.

It can thus be used to generate a delay in a logic system, i.e. an output signal occurs t seconds after the input signal is applied. Complementary outputs are usually available. The logic symbol is shown in Fig. 25.

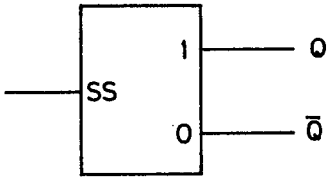


Fig. 25 Monostable Multivibrator

In the stable state the Q output is at logic 0 level.

The input may be direct or capacitively coupled.

2. ELECTRONIC LOGIC CIRCUITS

2.1 Introduction

Logic circuits are classified into types or families, such as diode logic, diode transistor logic, etc., each family having its particular advantages and disadvantages. In choosing a suitable family to perform a given task, the following parameters must be considered:

a) Operating Speed. This is usually defined as the time taken for the output to reach a specified level after the input signal has passed through another specified level, for a specified set of operating conditions. The levels specified at the input and output are known as the logic threshold levels for the specified set of operating conditions.

This is illustrated in Fig. 2.1

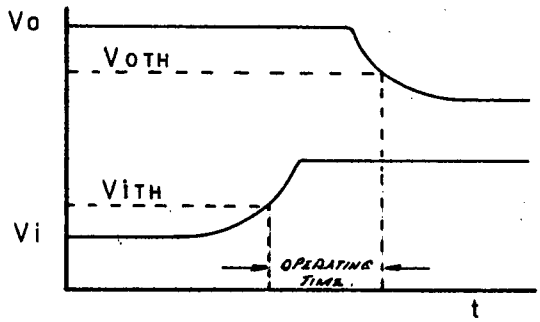


Fig. 2.1

Logic Operating Time

The operating time is a measure of the switching delays in the circuit under consideration and may be as low as a few nanoseconds (10^{-9} seconds) for some families or as high as a few microseconds for other families.

b) Noise Immunity. This is a difficult parameter to specify completely, as a circuit can be affected by noise on power supply lines and subsidiary input lines as well as the input line under consideration. The noise immunity at logic inputs is a function of the difference between input and output logic threshold levels, and is usually expressed in volts.

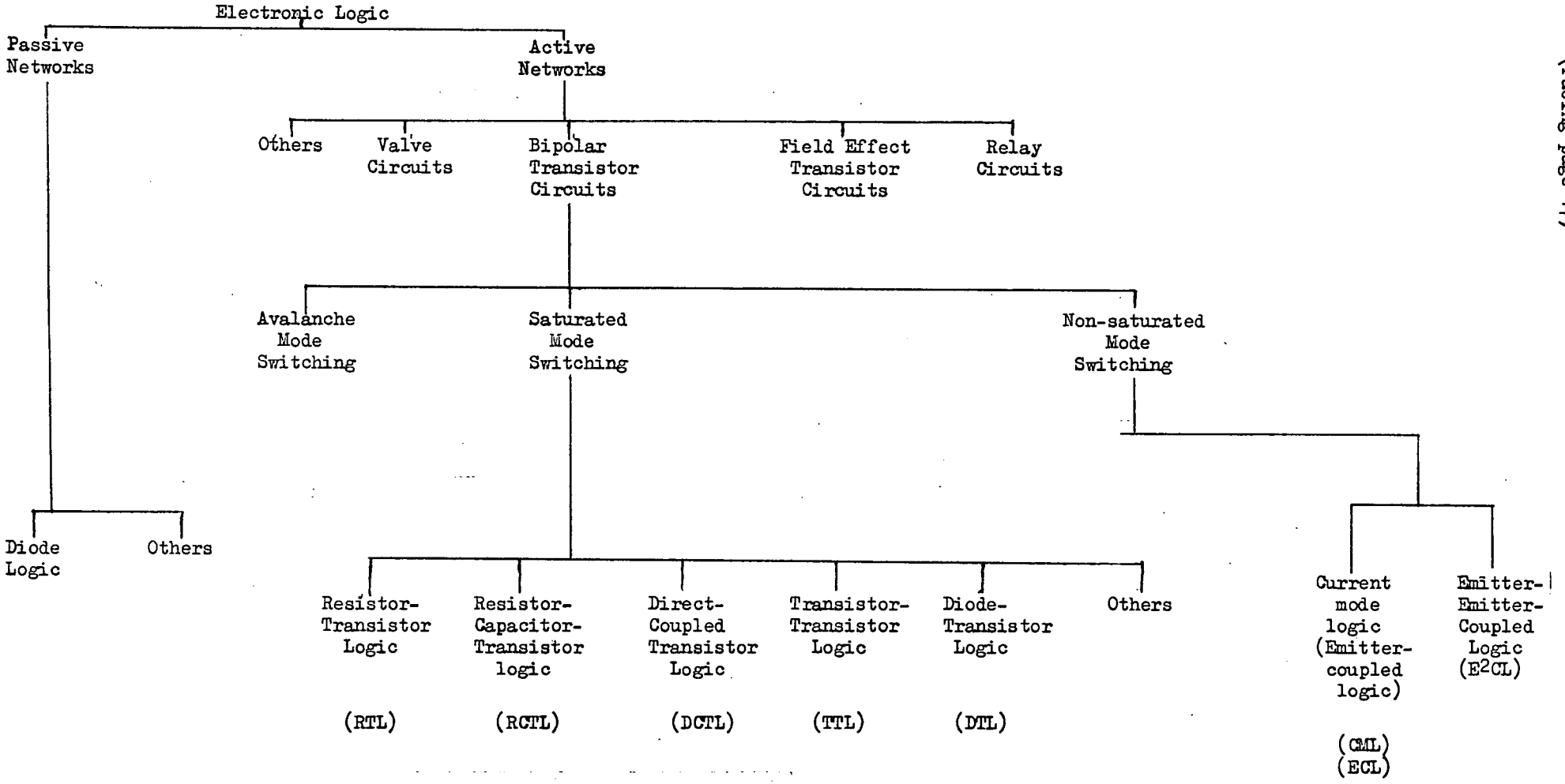


Fig. 2.2 Logic Family Classification

(Facing page 17)

c) Fan-out. The fan-out of a logic circuit is a measure of the ability of its output to drive other circuits in the same family. The fan-out is usually expressed as the number of gate inputs which can be driven reliably under worst case conditions.

d) Power Dissipation. In equipment designed for portable operation, the power dissipation per gate must be as low as possible to conserve battery charge.

e) Cost. If high noise immunity, high fan-out etc. are required, the cost is usually commensurate. Thus some compromise in performance may be necessary for reasonable cost. Although a logic family is usually chosen by optimizing the foregoing parameters, further optimizing within a given family is usually possible. For example, operating speed and noise immunity are usually improved by restricting the fan-out; power dissipation may be reduced at the expense of operating speed. Such "Trade-offs" are often used to obtain the most reliable performance.

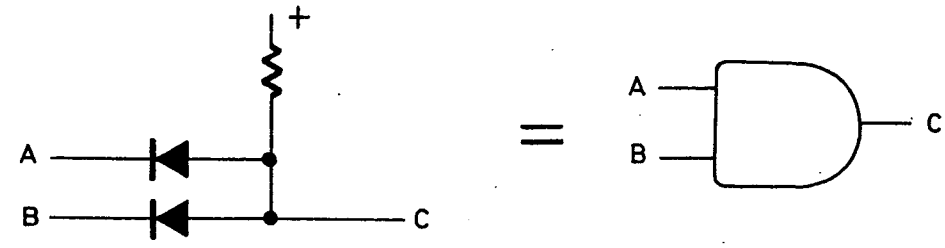
2.2 Circuit Configurations

There are many ways of implementing logic functions using electronic components; however, this discussion is restricted to circuits using semiconductor diodes and/or bipolar transistors, in association with linear passive components (resistors, capacitors etc.). Figure 2.2 classifies the various logic families according to the type of circuit used. Families under the heading of Passive Networks (mainly diode logic) do not contain active devices in their circuits; hence the available output power is always less than the input power. This means that the fan-out is low and that when gates are cascaded, the logic levels become progressively less well defined.

On the other hand, Active Networks can supply a relatively high output power without degradation of logic levels. The two branches of bipolar transistor logic to be discussed are:(1) saturated mode switching; and (2) non-saturated mode switching. Of these the latter is superior in operating speed, but has lower noise immunity and higher power dissipation.

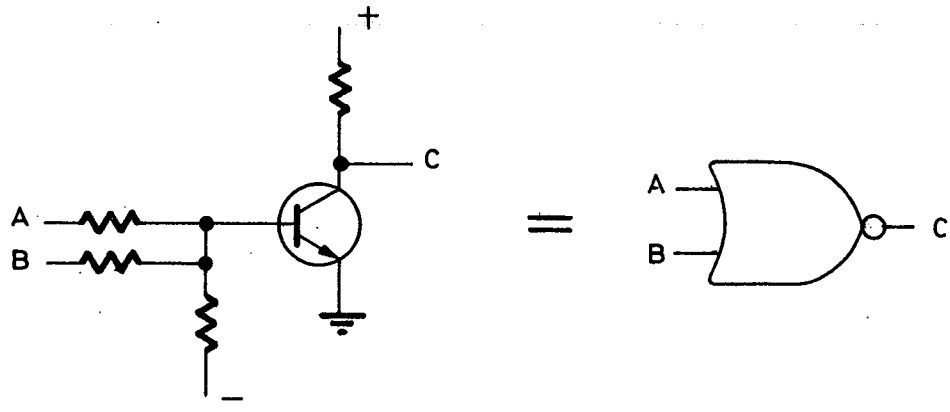
Simple circuit examples of a two input gate will now be given for each logic family.

2.2.1 Diode Logic (DL)



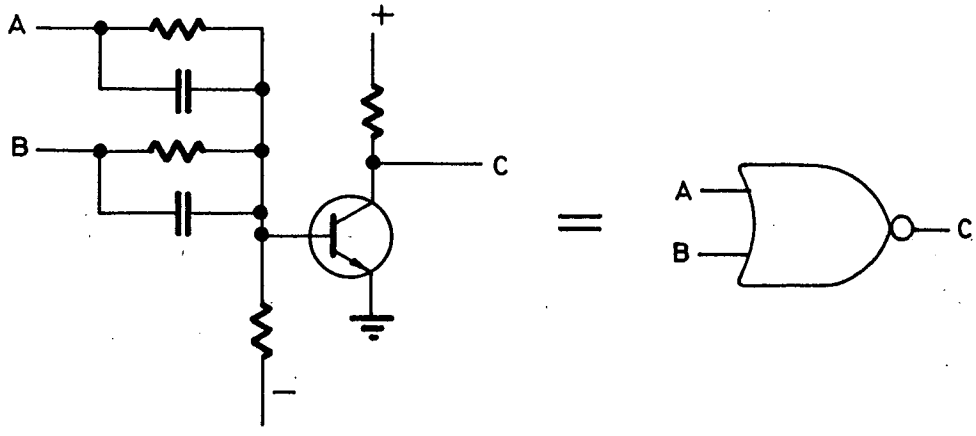
The disadvantages of this passive circuit have already been mentioned.

2.2.2 Resistor Transistor Logic (RTL)



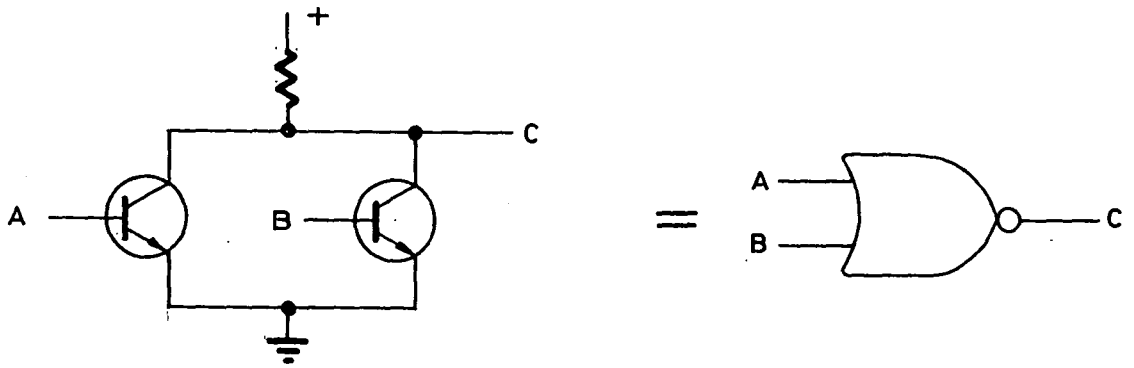
This circuit has low speed, low fan-out, and poor noise immunity.

2.2.3 Resistor Capacitor Transistor Logic (RCTL)



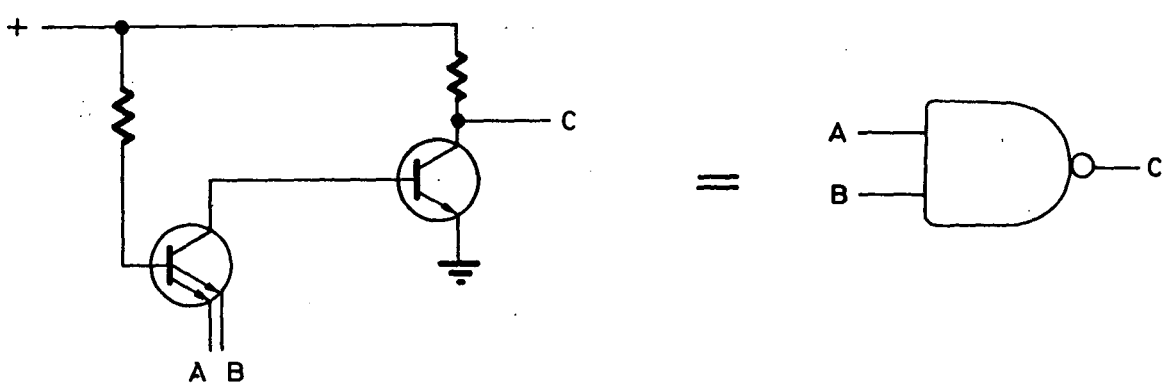
Capacitors have been added to increase operating speed, but noise immunity is reduced.

2.2.4 Direct Coupled Transistor Logic (DCTL)



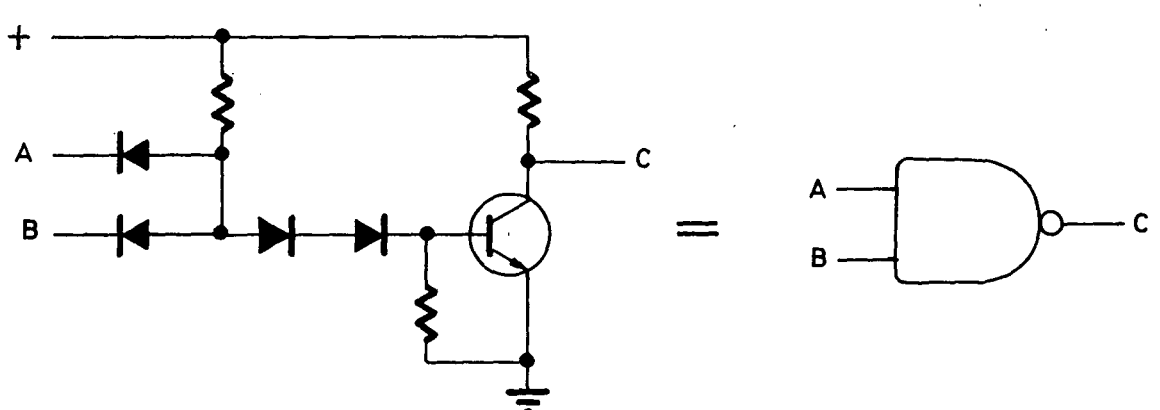
This circuit has low power dissipation and is cheap to produce, but the small differential between logic levels ($\sim\frac{1}{2}$ volt) gives very poor noise immunity. Fan-out is also poor.

2.2.5 Transistor Transistor Logic (TTL)



This circuit uses a multiple-emitter transistor and possesses higher noise immunity and fan-out than DCTL. Operating speed is also high.

2.2.6 Diode Transistor Logic



This is basically a DL circuit followed by an amplifier. The series diodes provide a higher noise immunity than can be obtained in any other circuit discussed so far. Speed is medium; fan-out is high, and power dissipation low. This is probably the most popular general purpose logic family.

An additional feature is the ease of obtaining the "wired-OR" function (Section 1.3.1) by simply bridging the outputs of two or more gates. The symbol for this is shown as wired-AND because an AND function is performed with reference to the positive logic at the gate inputs. However, for the negative logic produced at the gate outputs an OR function is performed.

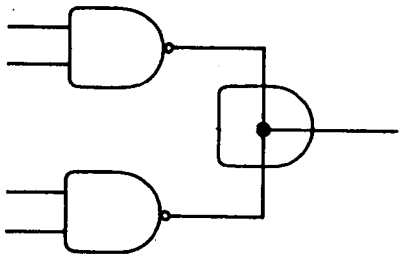
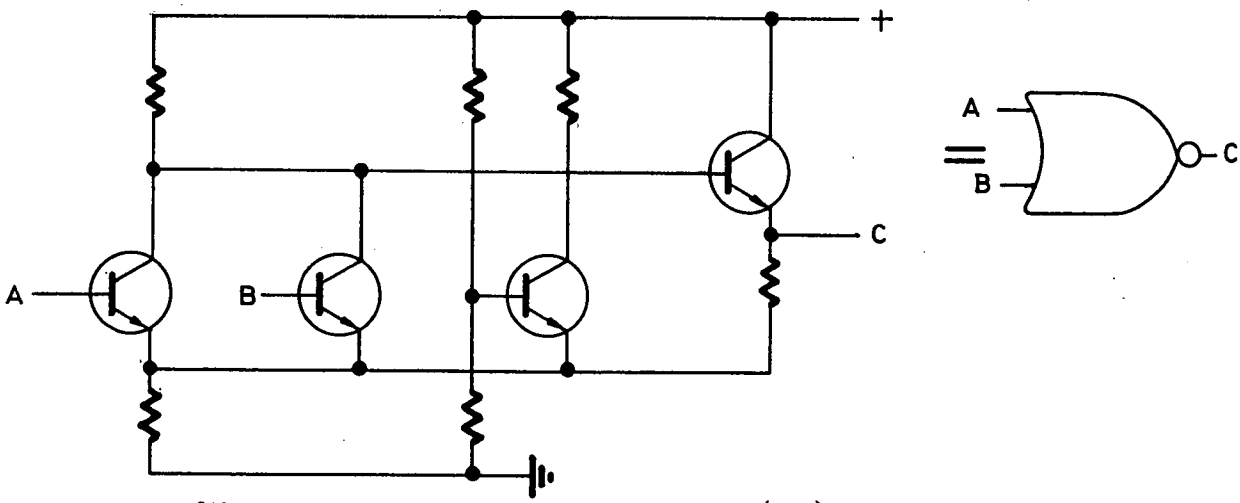


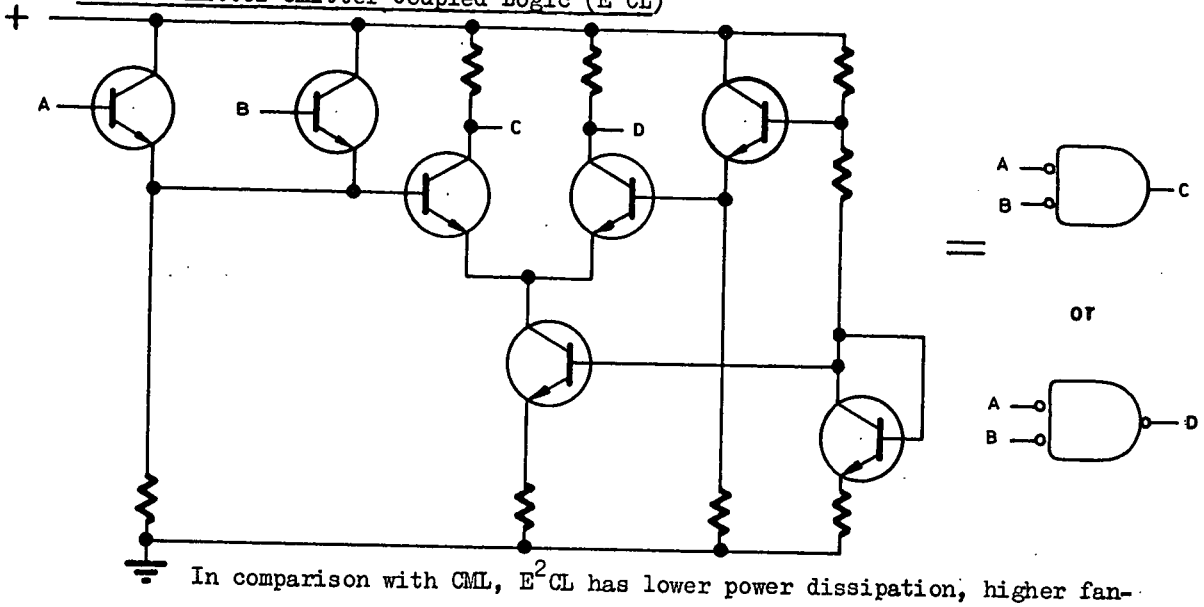
Fig. 2.2 Wired-OR connection of DTL gates

2.2.7 Current Mode Logic (CML)



Otherwise known as emitter coupled logic (ECL) this circuit is very fast (~ 10 ns). Fan out is moderate; noise immunity is low, and power dissipation high. Because neither logic level is ground, it is sometimes difficult to drive other types of circuits. Complementary outputs can be made available.

2.2.8 Emitter-emitter Coupled Logic (E^2CL)



In comparison with CML, E^2CL has lower power dissipation, higher fan-out and faster speed (~ 2 ns). Noise immunity is still low, and matching to other circuits difficult.

3. LOGIC ALGEBRA AND MAPPING

3.1 Introduction

In 1847 a system of algebra for handling two-valued logic problems was invented by George Boole. In 1938 Claude Shannon applied this algebra to switching circuits. Boolean algebra is used extensively in simplifying logic problems and consequently simplifying logic circuits. Although this is primarily the designer's domain, circuit functions are often expressed algebraically on data sheets etc; hence some knowledge of the principles involved is desirable for servicing and operating.

3.2 Representation of AND, OR, and NOT

Suppose we have two logic variables X and Y. To indicate a logic AND operation between these variables we write X.Y (or simply XY). The logic OR operation is written X+Y. Not X is written $\overline{X}$.

As before, when a variable is in its "true" logic state it is denoted by 1; when in the "false" logic state it is denoted by 0. Thus if both X and Y are 1, the AND and OR operations become 1.1 and 1+1 respectively. Again it is emphasized that these are not expressions in binary arithmetic in which 1+1 = 10, rather than 1 as in Boolean algebra.

3.3 Postulates and Theorems

The Postulates and Theorems of Boolean algebra are tabulated below without comment. The theorems are each presented in two forms known as duals.

The dual of a Boolean expression is obtained by changing all .'s to +'s; changing all +'s to .'s, changing all 1's to 0's, and changing all 0's to 1's.

3.3.1 Postulates

$X = 1$ or else $X = 0$.

$1.1 = 1$	$0 + 0 = 0$
$1.0 = 0.1 = 0$	$0 + 1 = 1 + 0 = 1$
$0.0 = 0$	$1 + 1 = 1$
$\bar{1} = 0$	$\bar{0} = 1$.

3.3.2 Theorems

1.	$0.X = 0$	$1 + X = 1$
2.	$1.X = X$	$0 + X = X$
3.	$XX = X$	$X + X = X$
4.	$X\bar{X} = 0$	$X + \bar{X} = 1$
5.	$XY = YX$	$X + Y = Y + X$
6.	$XYZ = X(YZ) = (XY)Z$	$X+Y+Z = X+(Y+Z) = (X+Y)+Z$.
7.	$\overline{XY....Z} = \bar{X} + \bar{Y} + ... + \bar{Z}$	$\overline{X + Y + ... + Z} = \bar{X}\bar{Y}....\bar{Z}$

Theorem 7 is known as De Morgan's Theorem which is generalized in Theorem 8.

8. $\bar{f}(X,Y, ...Z, \cdot, +) = f(\bar{X},\bar{Y},.....\bar{Z},+,\cdot)$

9.	$XY + XZ = X(Y+Z)$	$(X+Y)(X+Z) = X+YZ$
10.	$XY + X\bar{Y} = X$	$(X+Y)(X+\bar{Y}) = X$
11.	$X + XY = X$	$X(X + Y) = X$
12.	$X + \bar{X}Y = X+Y$	$X(\bar{X}+Y) = XY$
12'.	$zx + z\bar{x}y = ZX + ZY$	$(Z+X)(Z+\bar{X}+Y) = (Z+X)(Z+Y)$
13.	$XY + \bar{X}Z+YZ = XY+\bar{X}Z$	$(X+Y)(\bar{X}+Z)(Y+Z) = (X+Y)(\bar{X}+Z)$
14.	$XY+\bar{X}Z = (X+Z)(\bar{X}+Y)$	$(X+Y)(\bar{X}+Z) = XZ+\bar{X}Y$
15.	$X.f(X,\bar{X}, Y,...,Z)=X.f(1,0,Y,...Z)$	$X+f(X,\bar{X},Y,...,Z)= X+f(0,1,Y,...Z)$
16.	$f(X,\bar{X},Y,...Z)=X.f(1,0,Y,..Z) + \bar{X}.f(0,1,Y,...Z)$	
	$f(X,\bar{X},Y,...Z) = \{X+f(0,1,Y,...,Z)\}\{\bar{X} + f(1,0,Y,...Z)\}$	

All the foregoing theorems may be derived from the basic postulates. Their validity may be checked by drawing up truth tables, e.g. for Theorem 10,

X	Y	XY	$X\bar{Y}$	$XY+X\bar{Y}$
0	0	0	0	0(=X)
0	1	0	0	0(=X)
1	0	0	1	1(=X)
1	1	1	0	1(=X)

3.4 Applications

The two-input ordinary (or inclusive) OR function gives a logic 1 output when either or both inputs are 1. There is another OR function known as the exclusive OR, which gives a logic one output when either but not both inputs are 1.

The truth table and algebraic expression are shown in Fig. 3.1.

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

$Z = X\bar{Y} + \bar{X}Y$

Fig. 3.1 Exclusive OR Function

A straightforward way of implementing this operation is shown in Fig. 3.2

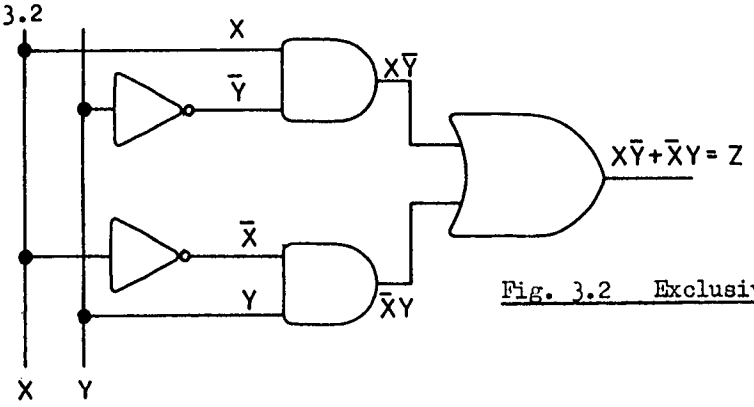


Fig. 3.2 Exclusive OR Implementation

However, the expression may be simplified :

$$X\bar{Y} + \bar{X}Y$$

$$= (X+Y)(\bar{X}+\bar{Y}) \text{ by Theorem 14, (substituting } \bar{Y} \text{ for Y, and Y for Z)}$$

$$= (X+Y)\,\overline{XY} \text{ by Theorem 7.}$$

This results in the logic diagram of Fig. 3.3, which uses one less inverter.

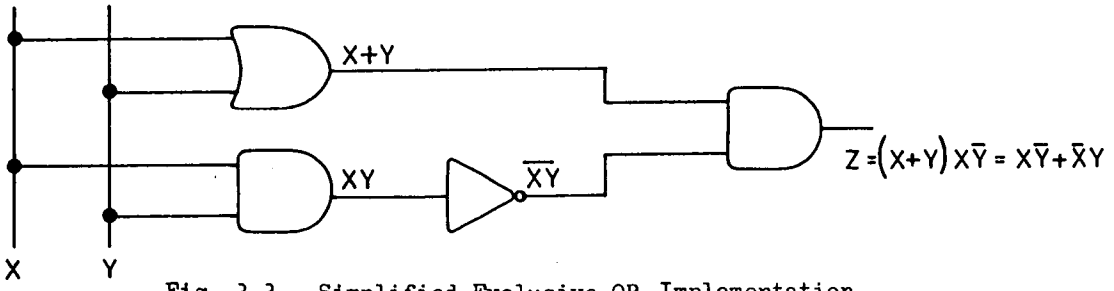


Fig. 3.3 Simplified Exclusive OR Implementation

As an example of a more difficult problem, consider the proposition:

A circuit will give an output pulse only if:

1, the supply is greater than 10 volts, the temperature is less than $50^{\circ}C$ and the test switch is off;

or 2, the supply is greater than 10 volts, the temperature is less than $50^{\circ}C$ and a sync pulse is present;

or 3, the supply is less than 10 volts, the temperature is less than $50^{\circ}C$ and the test switch is on;

or 4, the test switch is off and a sync pulse is present;

or 5, the temperature is less than $50^{\circ}C$ and a sync pulse is absent.

The problem is to express this proposition in its simplest terms.

A = Supply greater than 10 volts.

B = Temperature less than $50^{\circ}C$.

C = Test switch off.

D = Sync pulse present.

Then the circuit will give an output pulse if

1, ABC

or 2, ABD

or 3, $\bar{A}B\bar{C}$

or 4, CD

or 5, $B\bar{D}$

i.e. the logical expression for the circuit to give an output pulse is:

$$ABC + ABD + \bar{A}B\bar{C} + CD + B\bar{D}$$

$$= ABC + AB + \bar{A}B\bar{C} + CD + B\bar{D} \quad \text{(Theorem 12', } ABD + B\bar{D}\text{)}$$

$$= AB + \bar{A}B\bar{C} + CD + B\bar{D} \quad \text{(Theorem 11).}$$

$$= AB + B\bar{C} + CD + B\bar{D} \quad \text{(Theorem 12', } AB + \bar{A}B\bar{C}\text{)}$$

$$= AB + B\bar{C} + CD + B\bar{D} + BC \quad \text{(Theorem 13 in reverse).}$$

$$= AB + B + CD + B\bar{D} \quad \text{(Theorem 10, } BC + B\bar{C}\text{)}$$

$$= B + CD + B\bar{D} \quad \text{(Theorem 11)}$$

$$= B + CD \quad \text{(Theorem 11).}$$

Thus the circuit gives an output pulse if:

1, the temperature is less than $50^{\circ}C$

or 2, the test switch is off and the sync pulse is present.

The required logic can be realized by two gates instead of the 6 gates and 3

inverters implicit in the original expression.

3.5 Simplification by Mapping

A more powerful method of simplifying a logic expression is the mapping method introduced by Veitch in 1952 and improved by Karnaugh in 1953. It is not proposed to discuss the details of this method here, but in essence, the method consists of drawing up a chart, or Karnaugh map, which has provision for entering all possible logic combinations of the variables under consideration. If simplification is possible, groups of 2^n (where n is an integer greater than zero) adjacent entries will appear. Each group can be represented by only one logic combination.

Much of the simplicity of this method is lost if the total number of variables is greater than four.

4. BINARY ARITHMETIC AND ELECTRONIC COUNTING

4.1 Number Systems

The ordinary or decimal number system is a number system to the base 10. It is composed of 10 different digits (0 to 9), and numbers greater than 9 are represented by allocating powers of 10 as multiplicands associated with the position occupied by each digit in the number, e.g. $47286 = 4 \times 10^4 + 7 \times 10^3 + 2 \times 10^2 + 8 \times 10^1 + 6 \times 10^0$

$= 40000 + 7000 + 200 + 80 + 6.$

Other number systems may be similarly defined. For a base n, there will be n different digits, and the multiplicand associated with the position of a given digit in a number is raised to the appropriate power. For example the number expressed as 43 to the base 10 becomes 61 when expressed to the base 7, i.e. $6 \times 7^1 + 1 \times 7^0$.

The following table shows numbers from 0 to 10 expressed to bases 10, 7 and 2.

Base 10	Base 7	Base 2
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	10	111
8	11	1000
9	12	1001
10	13	1010

The rules of arithmetic can be applied to number systems to any base. Two additions are shown below, the first in base 5, the second in base 2. Numbers in brackets are decimal equivalents.

Base 5		Base 2	
33	(18)	111	(7)
14	(9)	110	(8)
102	(27)	1101	(13)

The number system to the base 2 is called the binary system, and is particularly suited to electronic manipulation because there are only two digits. As in the case of electronic logic, the on state of a gate or flip-flop may designate 1, the off state 0. A binary digit, whether 0 or 1, is called a bit.

4.2 Simple Counting Codes

In the fields of electronic control and computing, the counting of pulses and the digitizing of analogue information are basic operations. In these operations numbers must be represented by the states of flip-flop outputs etc., and as well as straight binary representation, there are many other counting codes, some of which have distinct advantages. However, in all codes of interest there are only two different bits: 0 and 1.

4.2.1 Natural Binary Code

This code expresses numbers in their normal binary form - the least significant digit has a multiplicand of 1; the next digit has a multiplicand of 2, the next of 4, and the n th digit has a multiplicand of 2^{n-1}, so that the binary number 1011 is the equivalent of

$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$. Thus if we think of the most significant digit in this number as having a "weight" of 8, the next as having a weight of 4, the next of 2, and the least significant digit as having a weight of 1, the decimal equivalent is easily found by adding the weights when the bits are 1 and ignoring the weights where the bits are 0. Thus the decimal equivalent of 1011 is 8 + 2+1 = 11. For numbers up to 10, this may be verified by the table in Section 4.1.

4.2.2 Natural Binary Coded Decimal

Very large binary numbers are tedious to convert to decimal equivalents. One method of overcoming this objection (at the expense of more bits) is to express each digit of a decimal number in its binary form. Any decimal digit (0 to 9) can be expressed in binary by using a maximum of four bits; so that a number having three decimal digits would have three groups of four bits in the natural binary decimal code: e.g. 374 becomes

0011 0111 0100.

To facilitate visual decoding the weights may be written or imagined above each group as follows.

Weights	8421	8421	8421
NBCD	0011	0111	0100
Decimal	3	7	4

4.2.3 Other BCD Codes

Two other common BCD codes are tabled below. These differ from NBCD in the values of the weights assigned to each digit. These codes are the outcome of certain counting methods to be described later.

Decimal	BCD 2421	BCD 4221
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0110
5	0101	0111
6	0110	1100
7	0111	1101
8	1110	1110
9	1111	1111

4.2.3. The Excess 3 Code

This is yet another BCD code but it used the middle ten of the first 16 possible combinations of four bits in the ordered binary sequence, with the result that the code for each number is the equivalent binary number plus three. One advantage of this code when used in computers is that the 9's complement of a number is simply found by complementing all bits.

Decimal	Excess -3 8421
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

4.2.5 Cyclic Codes - Reflected Binary

It is often advantageous to use a code in which consecutive numbers differ in only one bit. Such a code is particularly useful for minimising errors from encoding discs. Cyclic codes are sometimes called "unit distance" codes.

One of these codes is the reflected binary, or Gray, code. In this code all bits, except for the most significant, are reflected about the mid-point as shown.

Decimal	Reflected Binary (no weights)	
0	0000	
1	0001	
2	0011	
3	0010	
4	0110	
5	0111	
6	0101	
7	0100	
- - - - - - - - -	- - - - - - - - - -	- - Reflection
8	1100	
9	1101	
10	1111	
11	1110	
12	1010	
13	1011	
14	1001	
15	1000	

The reflected binary code may be used to obtain a so-called reflected BCD code and a reflected Excess-3 code in exactly the same manner as the corresponding non-reflected codes are obtained from normal binary. One striking feature of the reflected Excess -3 code is the ease of obtaining the 9's complement: simply complement the most significant bit.

4.2.6 Ring Counter Codes

In a ring or shift counter, the ON state shifts from one stage to the next on receiving each input pulse. The stages are connected in a ring, giving a code as set out below for a 10 stage counter.

Decimal	Ring Counter 0 1 2 3 4 5 6 7 8 9
0	1 0 0 0 0 0 0 0 0 0
1	0 1 0 0 0 0 0 0 0 0
2	0 0 1 0 0 0 0 0 0 0
3	0 0 0 1 0 0 0 0 0 0
4	0 0 0 0 1 0 0 0 0 0
5	0 0 0 0 0 1 0 0 0 0
6	0 0 0 0 0 0 1 0 0 0
7	0 0 0 0 0 0 0 1 0 0
8	0 0 0 0 0 0 0 0 1 0
9	0 0 0 0 0 0 0 0 0 1

4.2.7 Twisted Ring Counter Code

This type of counter functions in a similar manner to the ring counter but uses only half the number of stages.

Decimal	Twisted Ring Counter (no weights)
0	00000
1	00001
2	00011
3	00111
4	01111
5	11111
6	11110
7	11100
8	11000
9	10000

4.3 Error Checking Codes

Codes have been devised which allow each number to be checked for possible errors, such as may occur if the circuit producing one of the bits fails. More complex codes are able to indicate which bit or bits are in error and initiate correction.

For such codes the following equation holds:

$$M - 1 = C+D \qquad (C \leq D)$$

where C = number of bits in error that can be corrected

D = " " " " " " " " detected.

M = minimum distance of the code, i.e. the minimum number of bits, in any character, which must be changed to generate another character of the same code. Two of the simpler codes, (minimum distance 2) of this type follow:

4.3.1 The Biquinary Code

This code uses seven bits to encode numbers from 0 to 9. Two of these bits are always 1's and the remaining five bits are 0's. If an auxilliary counter is used to count the 0's and 1's in each number it is possible to detect non-cancelling errors.

Decimal	Biquinary
0	0100001
1	0100010
2	0100100
3	0101000
4	0110000
5	1000001
6	1000010
7	1000100
8	1001000
9	1010000

4.3.2 BCD Code with Parity

By adding a redundant bit to the normal BCD code, a single error detecting code can be formed. This bit is called a parity bit. When "even parity" is used, the extra bit is chosen to make the number of 1's in the character even. For "odd parity", the number of 1's is made odd.

When data formed in this way are processed, the computer can be programmed to detect single errors by checking the parity.

Decimal	Even Parity BCD 8421P	Odd Parity BCD 8421P
0	00000	00000
1	00011	00010
2	00101	00100
3	00110	00111
4	01001	01000
5	01010	01011
6	01100	01101
7	01111	01110
8	10010	10011
9	10010	10011

4.4 Counting Circuits

Nearly all electronic counters use bistable elements, the flip-flop being the most popular - at least up to the tens of megahertz rate. Tunnel diodes and various forms of trigger diodes are also used but are not common.

The number of flip-flops used in a counter depends on its "modulo" number. This is the number of pulses that must be applied to the counter's input to produce a repetition of the output state. If the counter is modulo n and the next greater power of 2 is m (when n may equal m) then the minimum number of flip-flops required is m. For example a modulo 10, or decade, counter requires at least four flip-flops.

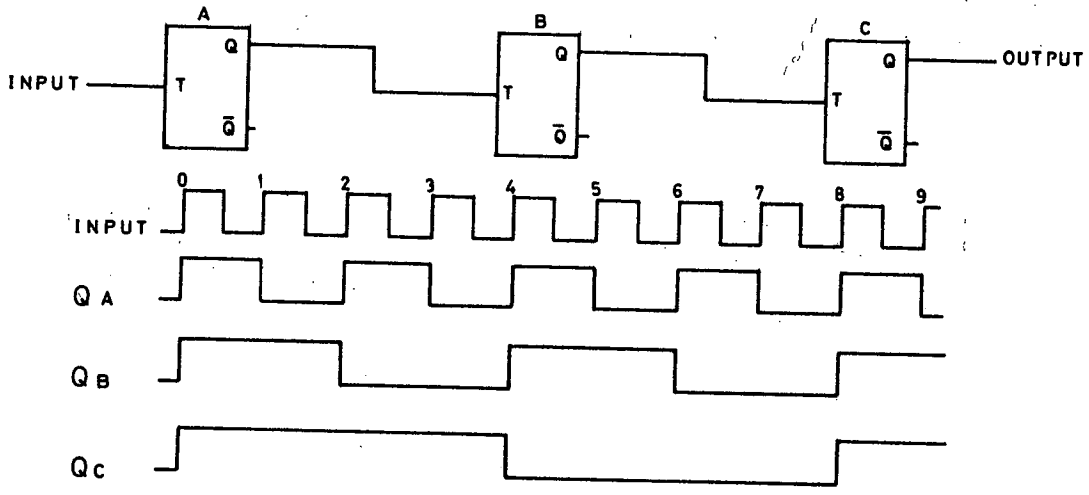
The method of applying the input signal classifies counters into two groups : (i) the asynchronous or ripple counter, in which the input signal is applied to the first flip-flop only, and the change of state of this flip flop provides a signal for the second flip-flop and so on; (ii) the synchronous counter in which the input signal is applied to all flip-flops, any possible change of state being determined by the state of the previous flip-flop prior to the application of each input signal pulse.

The asynchronous counter is usually simpler, but the synchronous counter does not suffer from "propagation delay" and is therefore faster. Also the asynchronous counter having a modulo other than a power of two must use feedback which, at high counting rates, can be troublesome. For these reasons synchronous counters are preferred for high-speed counting.

Outputs from each flip flop provide the bits (0 or 1) making up the coded count. If desired, these outputs may be fed to decoding circuits for the purpose of displaying the count in decimal form, or for initiating various operations when the count reaches pre-determined states.

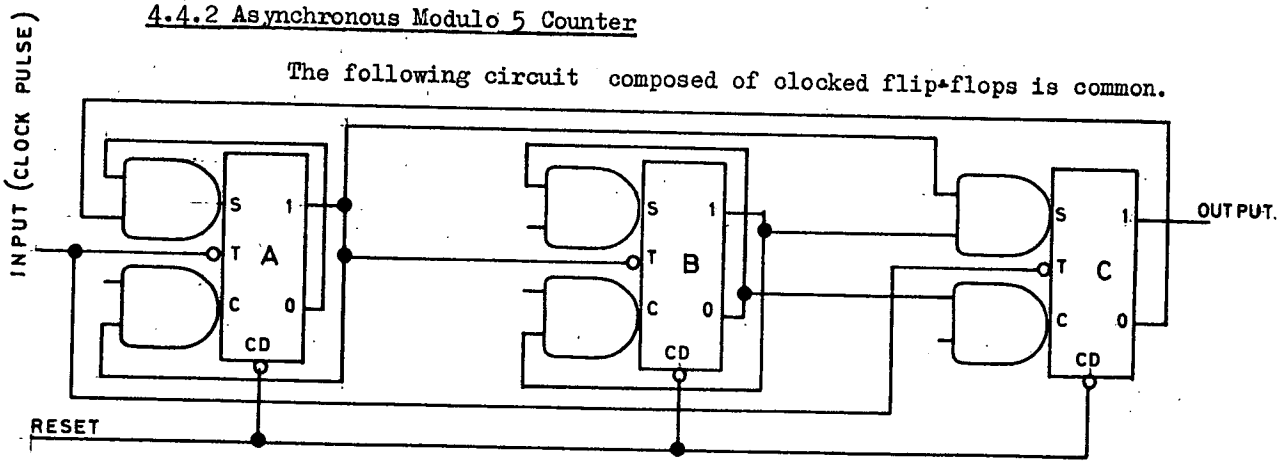
4.4.1 Asynchronous Binary Counters

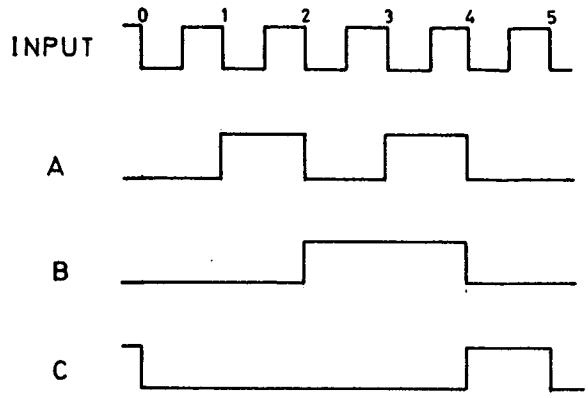
To make a counter which counts to 2^n, simply cascade n flip flops. The following counter will count to 8 (2^3).



4.4.2 Asynchronous Modulo 5 Counter

The following circuit composed of clocked flip-flops is common.



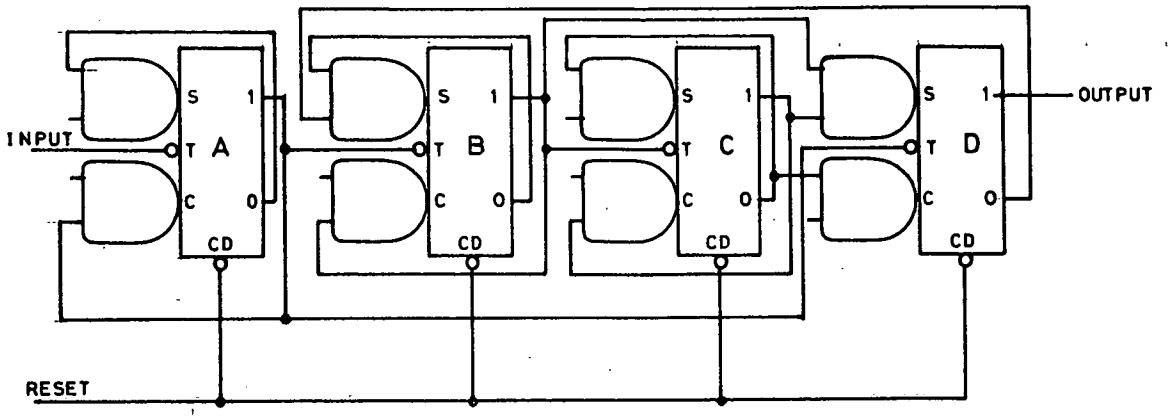


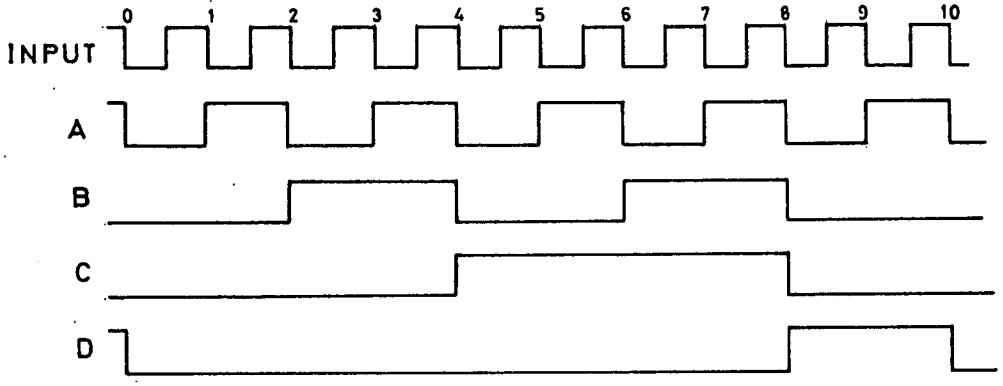
IN	A (1)	B (2)	C (4)
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	0	0	0

In the reset or zero state, all flip-flops have their 1 output low. Thus flip-flop A will operate as a normal binary except when inhibited by feedback from flip-flop C after the fourth input pulse. B is a binary, driven by negative-going transitions from A. The set and clear inputs on C are connected to the 1 and 0 outputs respectively of B. An additional inhibiting input is connected from the output of A to the set input of C so that C cannot be activated by the clock pulse until both A and B have been high simultaneously; i.e. C does not change state until the fourth clock pulse. Because A is inhibited after C changes state, A does not alter on the fifth clock pulse so that all outputs are again in the low state and the cycle starts again. When examining circuits containing clocked flip-flops it is important to remember that the state of a flip-flop is determined by the state of the set and clear inputs during the time the toggle input is high, but the flip-flop does not change to this state until the toggle input goes low (see Section 1-6-7).

4.4.3 Asynchronous Decade Counter

A decade (modulo 10) counter is easily made from the foregoing scale of five by connecting a simple binary (scale of two) flip-flop to the input. This is one of the most common counting circuits; flip-flops A, B, C and D having weights 1, 2, 4, 8 form a natural binary coded decimal counter.





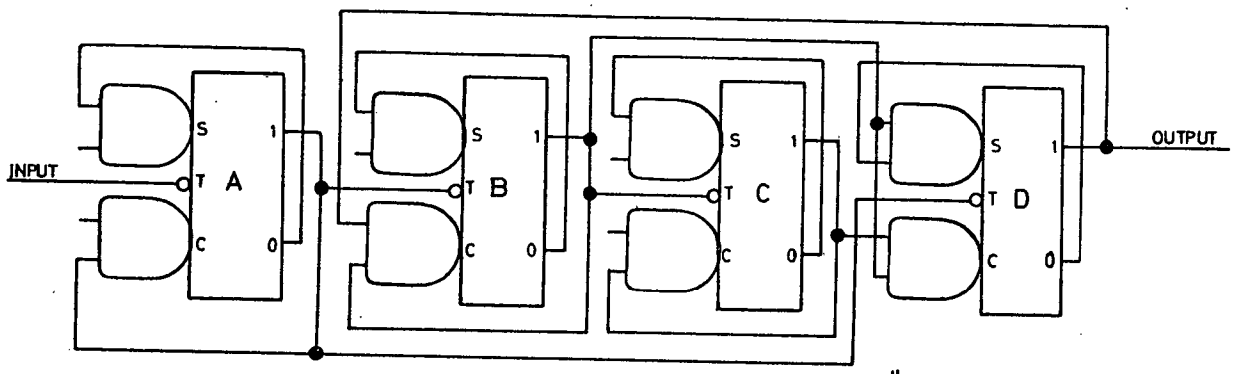
IN	A (1)	B (2)	C (4)	D (8)
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	0	0	0

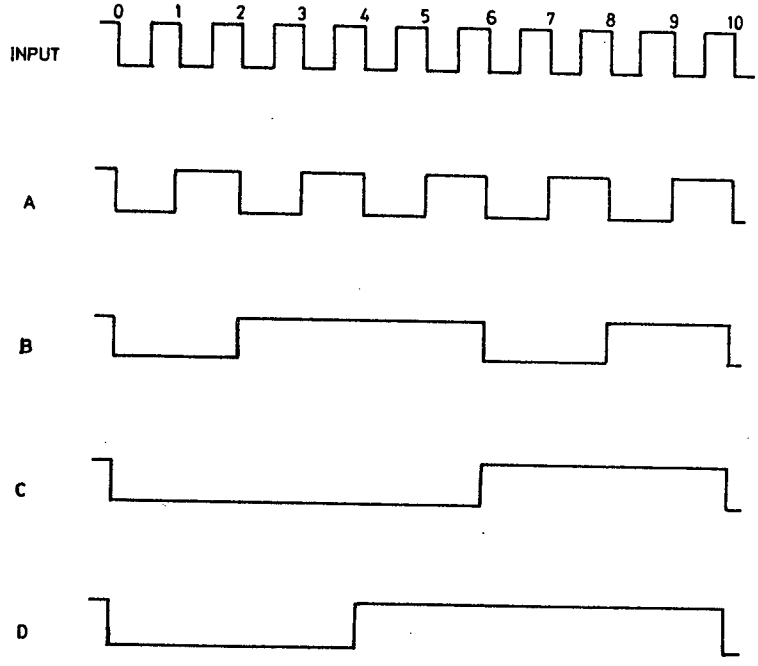
4.4.4 General Asynchronous Counters

Counting to any number (other than 2^n) can be accomplished either by suitable feedback connections as for the modulo 5 counter, or by decoding either a binary or a cascade of BCD counters to give a reset pulse at the number required. The latter method is perfectly general but is limited to relatively slow count rates because of propagation delays. The feedback method can only be used for a few numbers and is not general.

4.4.5 Other Asynchronous Decade Counters

As mentioned in Section 4.2.3, weighting systems other than 1-2-4-8 are possible in BCD counting. The following circuit shows a 1-2-4-2 system.

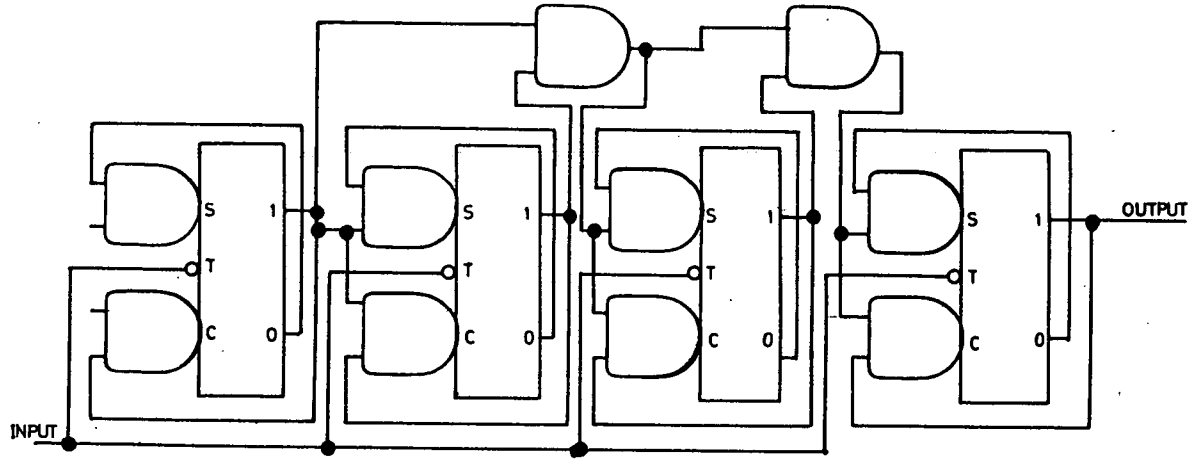




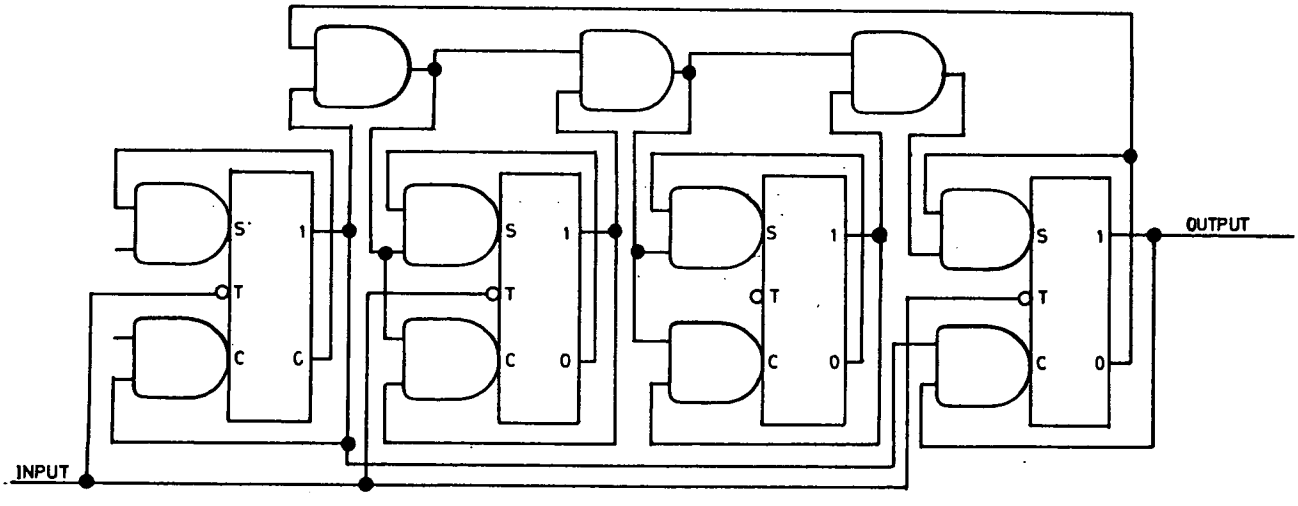
In	A (1)	B (2)	C (4)	D (2)
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	1	0	1
5	1	1	0	1
6	0	0	1	1
7	1	0	1	1
8	0	1	1	1
9	1	1	1	1
10	0	0	0	0

4.4.6 Synchronous Binary Counters

In synchronous counters each flip-flop changes state when triggered by the clock pulse, and therefore does not suffer from propagation delay while waiting for the preceding stage to provide a trigger as in the case of asynchronous counters. The state of preceding stages must be taken into account, however, to ensure that each flip-flop triggers at the correct time. In a natural binary counter a given stage should not change state unless all preceding stages are in the 1 state. This is achieved by gating the outputs of all preceding stages into both the set and clear inputs of the given stage. A four-stage counter is shown.

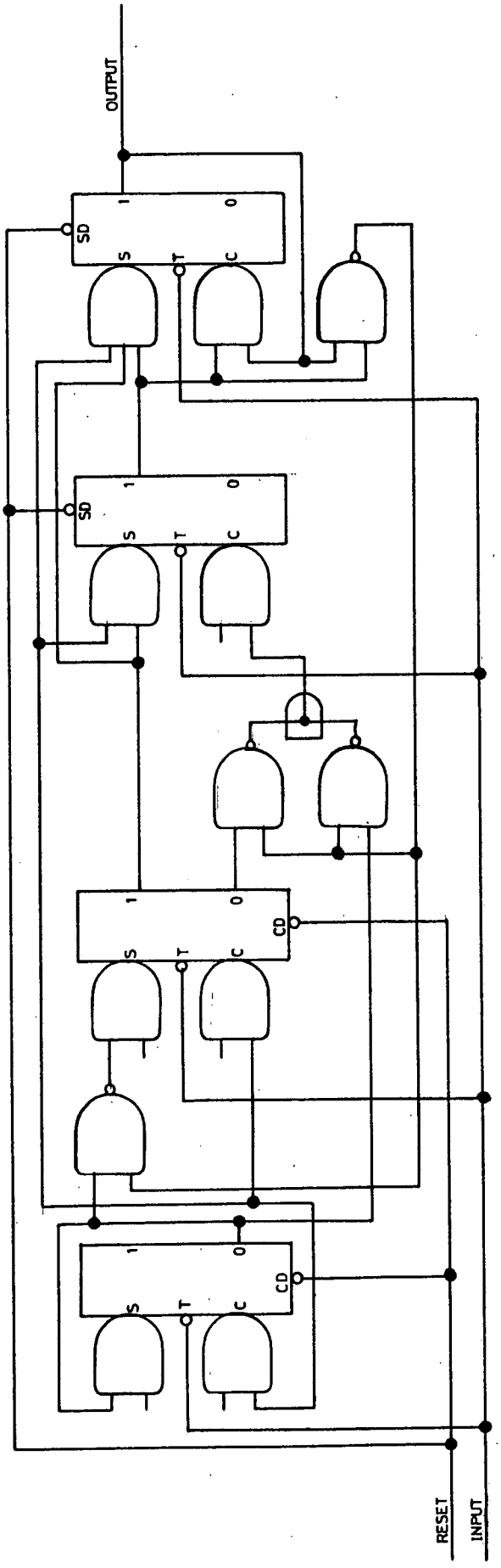


4.4.7 Synchronous Natural BCD Decade



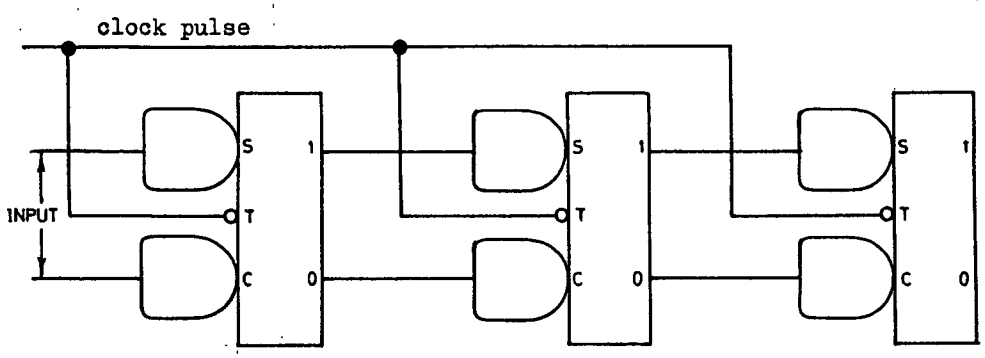
4.4.8 Synchronous Excess-3 Counter

Section 4.2.4 describes the Excess-3 Code. The following circuit will achieve this code.



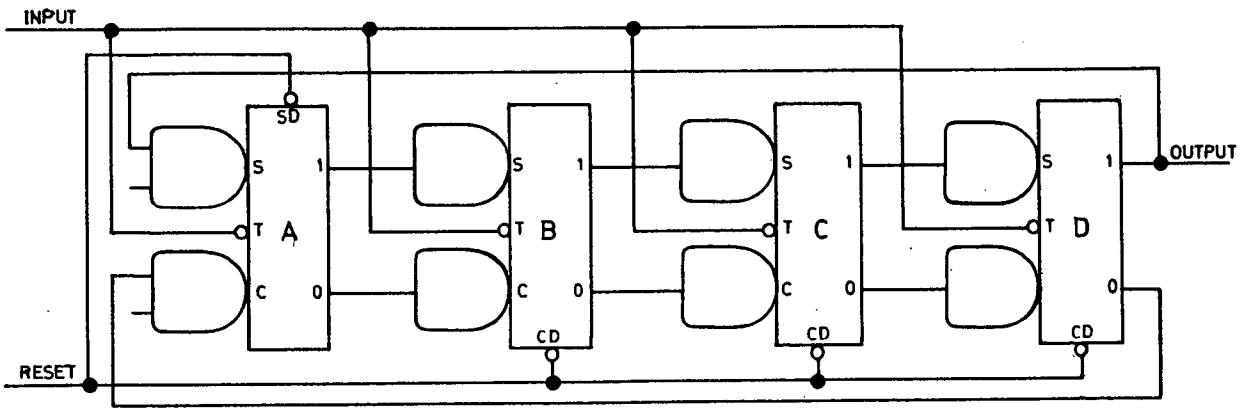
4.4.9 Shift Registers

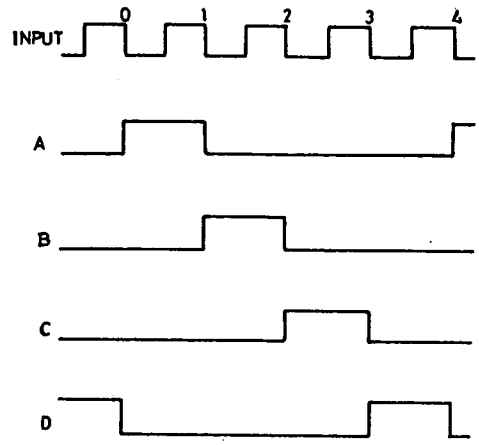
Although not a counter in itself, a shift register can easily be turned into a counter, so a short explanation will be given. A shift register is a chain of flip-flops so connected that when a clock pulse is simultaneously applied to all stages each flip-flop assumes the state of the preceding flip-flop. This is the method by which binary numbers are serially written into computer systems. The operation is simply obtained with the following circuit.



4.4.10 Ring Counters

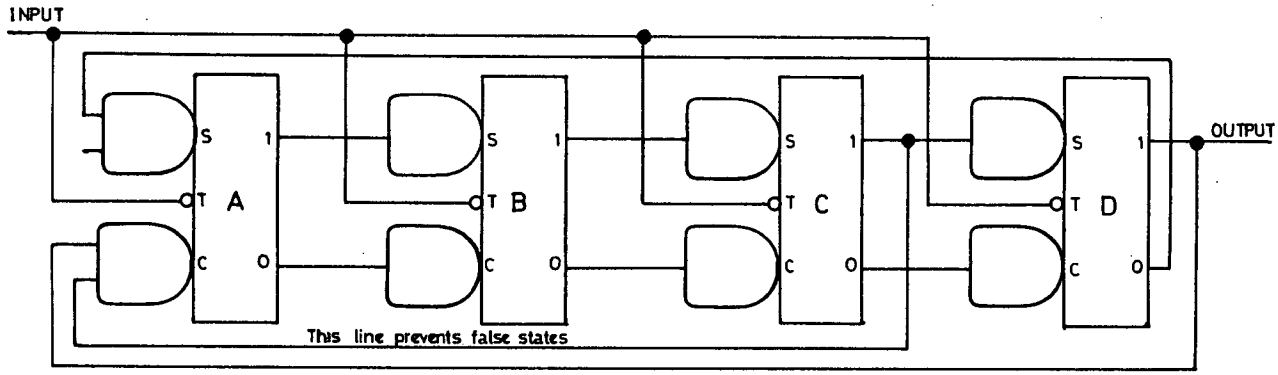
If the output of a shift register is connected back to the input, a ring counter results. If a 1 is set into one flip-flop and 0 into all others, each clock pulse will move the 1 one stage on around the ring. The counter modulo is simply the number of stages used. Outputs from each flip-flop provide already decoded signals. The following circuit shows a modulo 4 counter.

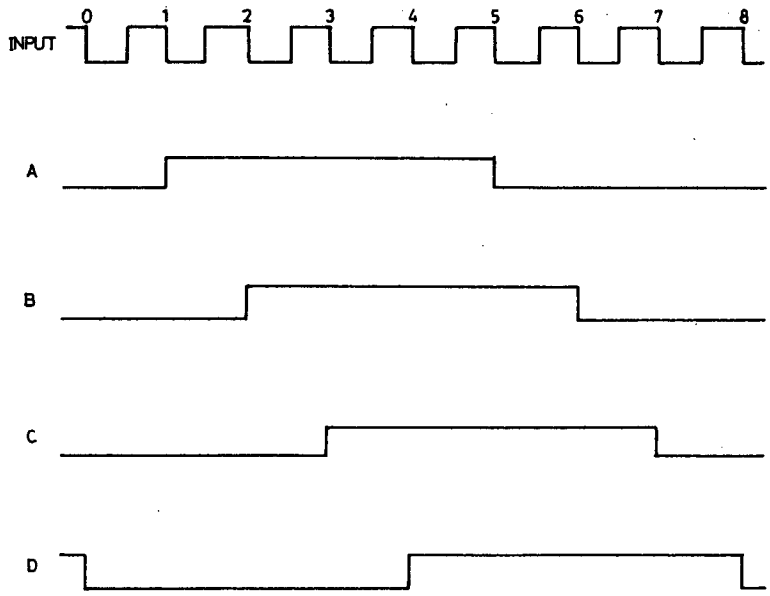




In	A	B	C	D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0
5	0	1	0	0

A more economical ring counter is obtained by crossing over the lines connecting output to input. This twisted-ring counter has a modulo of twice the number of stages used. The following circuit shows a modulo 8 counter.





In	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
8	0	0	0	0

4.4.11 Other Counters

There are a large number of counter circuits for various applications, but the main types have been covered. Mention should be made, however, of the so-called bi-quinary and qui-binary counters, which combine synchronous and asynchronous stages to form decade counters. Noteworthy is the fact that some counting circuits can easily be made reversible by gating complementary triggers to each stage.

The type of counter to be used in a given application depends on the counting rate, the code required, and the amount of decoding necessary.

4.5 Decoding

Decoding of a counter is performed by arranging a system of gates to give outputs whenever the count reaches a desired number or numbers. For example the fourth and seventh pulses into a decade counter may be required to initiate some operation each time the decade goes through its cycle. Two gating circuits would be required - one to give an output on the fourth pulse and no other, and one to give an output on the seventh pulse only. Again, complete decoding of a decade counter may be required to operate a decimal numerical display, such as Nixie tube, so as to give a visual decimal indication of the state of the counter. For this application ten gates are required - one for each digit 0 to 9.

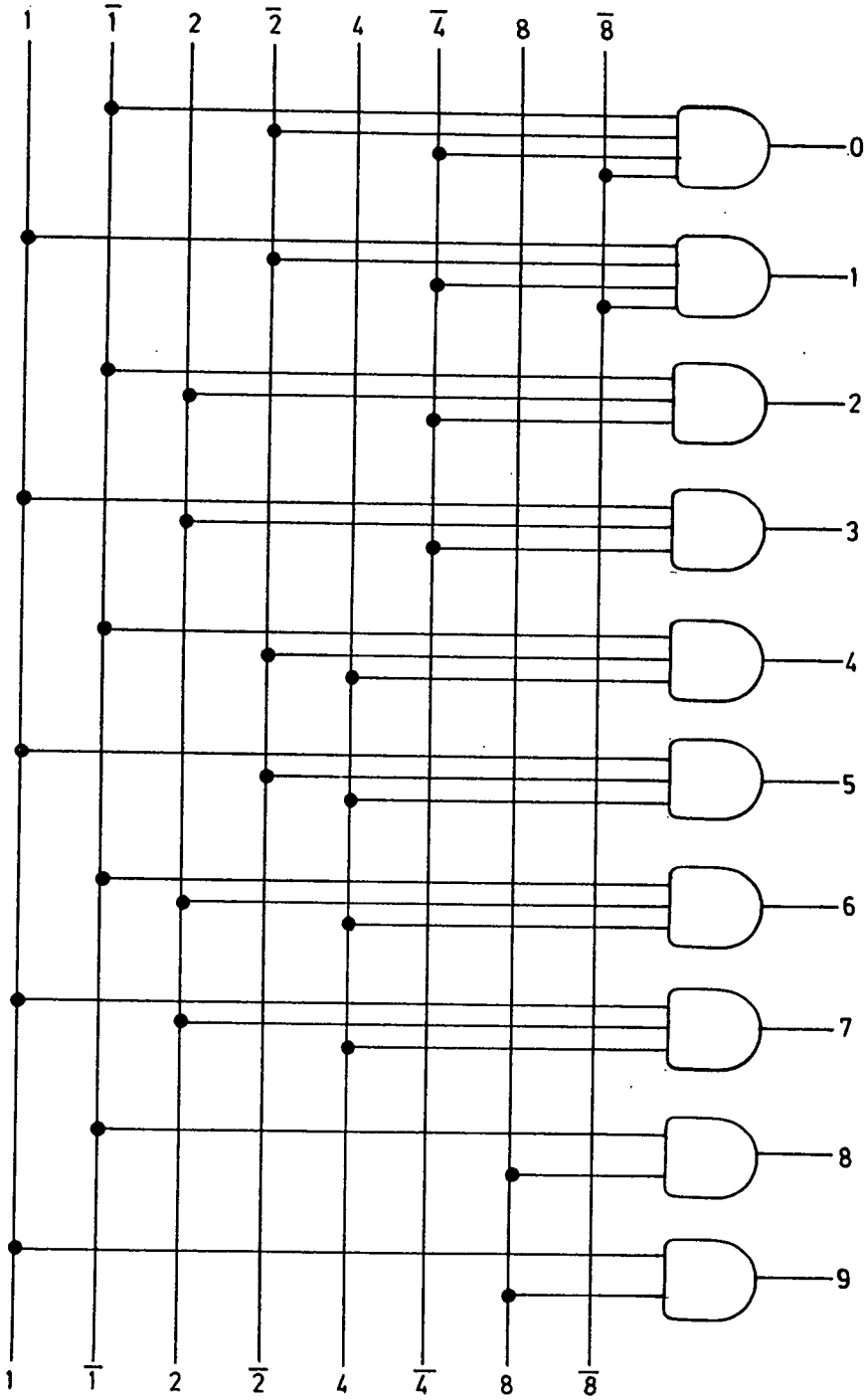
Gates can also be used to encode from one counting system to another; e.g. natural binary to Gray code, or BCD to natural binary. The design of any such gating system requires a relatively simple application of truth tables, or Boolean algebra, or Karnaugh maps.

A commonly found example is the complete decoding of BCD to decimal:

Decimal Number	Formation from BCD
0	$\overline{1}\ \overline{2}\ \overline{4}\ \overline{8}$
1	$1\ \overline{2}\ \overline{4}\ \overline{8}$
2	$\overline{1}\ 2\ \overline{4}\ \overline{8}$
3	$1\ 2\ \overline{4}\ \overline{8}$
4	$\overline{1}\ \overline{2}\ 4\ \overline{8}$
5	$1\ \overline{2}\ 4\ \overline{8}$
6	$\overline{1}\ 2\ 4\ \overline{8}$
7	$1\ 2\ 4\ \overline{8}$
8	$\overline{1}\ \overline{2}\ \overline{4}\ 8$
9	$1\ \overline{2}\ \overline{4}\ 8$

Notice that the numbers 2 to 7 are not really dependent on the state of the 8 flip-flop because the various combinations of 1, 2, 4, and their complements do not occur elsewhere in the table. Thus three-input gates are required to decode these numbers. Numbers 8 and 9 are the only ones to require a true 8, and the states of the 2 and 4 flip-flops do not change. Hence two-input gates are required for these numbers. Four-input gates are necessary to decode 0 and 1.

The following diagram depicts the decoding arrangements.



4.6 Arithmetic Operations

By suitable application of logic principles, it is possible to design gating circuits which will add or subtract numbers stored in flip-flops and place the sum or difference in another set of flip-flops or register. It is not proposed to go into circuit details as these are readily available in any book dealing with digital computer ciruits.

Multiplication and division may also be performed by successive additions or subtractions or, in some cases, by shifting digits along shift registers.

In fact, nearly all mathematical operations can be reduced to sequences of adding and shifting - hence the wide range of application of the digital computer.

5. MICROCIRCUITS AND THEIR APPLICATIONS

In recent years semiconductor technology has progressed to the point where complete circuits, rather than discrete transistors, can be fabricated (or "integrated") on one semiconductor chip. This has reduced the size of a NAND circuit, for example, to such an extent that four two-input NAND gates can be fitted into a package smaller than a postage stamp, and give reliable operation to temperatures up to 125°C. The main limitation on size reduction with these packages is the physical package itself (usually epoxy, silicone, or ceramic) and the space required to weld the connecting leads to the chip. As a result, another type of integrated circuit has resulted in which many flip-flops and gates are interconnected to give circuits such as decade counters, BCD decoders, and shift registers. The integration of various circuits to give a functional element such as a counter is known as "Medium Scale Integration" (MSI). This is but a prelude to "Large Scale Integration" (LSI), in which complete equipments, such as desk computers, will be integrated into a single package.

As well as logic circuits, various linear circuits such as operational amplifiers, audio and i.f. amplifiers, and voltage comparators have been released in integrated form, but these will not be dealt with here.

The most common packages for integrated circuits are the "Cerpak", the "Dual In-line", and the metal can. The "Cerpak" or "Flatpak" is a ceramic package approximately 6 mm square and 1 mm thick, having up to 16 flat ribbon leads spaced around all four sides. This is the package currently preferred by military authorities. The "Dual In-line" package is a ceramic, epoxy, or silicone package approximately 19 mm x 6 mm x 5 mm and having two rows of pins (up to 8 per side) suitable for printed circuit mounting. This package is favoured by industrial users and is the one mostly used in BMR equipment. Metal can packages are wimilar to the JEDEC TO-5 transistor package, but have up to 12 leads.

5.1 Microcircuit Logic Families

The major semiconductor manufacturers provide a range of integrated logic families usually including RTL, TTL, DTL, and E^2CL. Of these, TTL and DTL are currently the most popular for logic speeds up to 1 MHz. TTL circuits are superior at higher speeds, while for the very fast logic found in modern

computers E^2CL is used. There is also an increasing use of insulated gate field effect transistor logic in integrated circuits, particularly in MSI and LSI applications.

In any given family, e.g. DTL, it is possible to have sub-families such as low-power DTL and medium-power DTL. Even further sub-division is found according to the size of the pull-up resistor at the gate outputs; Thus one speaks of 2k or 6k medium power DTL, the 2k or 6k resistor determining operating speed or fan-out. As mentioned previously variations in one characteristic of a family can only occur if "trade-offs" in other characteristics are made. Thus for each sub-family the manufacturer lays down loading rules for reliable operation. Usually data sheets give loading rules for combining sub-families and occasionally loading rules are given for combining different families. For example DTL and TTL families can be made compatible by proper choice of loading rules. This results from the fact that both families act as current sinks (i.e. current flows into the gate outputs) when gate outputs are in the low state (assuming NPN transistors), but very little current is passed when in the high state.

Thus the various families may be classified into the following groups: current sinking logic (DTL and TTL), current sourcing logic (RTL and RCTL) and current mode logic (ECL and E^2CL). Current mode logic drives current in both output states. Families in each of these groups are compatible so long as the appropriate loading rules are observed.

The whole field of integrated logic circuits is a rapidly developing one and there will doubtless be further families appearing in the future.

5.2 Handling of Microcircuits

If properly handled, a microcircuit is probably ten times as reliable as the discrete circuit it replaces. To ensure reliable operation, the following precautions should be observed:

(1) Do not exceed any of the manufacturer's ratings.

(2) Do not subject the device to severe mechanical shock.

(3) Do not strain connecting leads, e.g. by bending close to package.

(4) Do not subject the device to excessive temperatures when soldering.

(5) Ensure that the soldering iron tip is earthed to avoid applying large 50-Hz voltages to the device.