CHAPTER V

CONCLUSION

The development of a versatile equipment as the Timer/Counter which can work independently by itself or can be used as a Master/Slave unit controlling other electronic devices is rather complicated and needs great effort. It involves both the digital and linear circuit designs. The latter plays an important role in the comparator section. The choice of the Series 74 digital IC's is based on their availability and wide range of application. The display space and the power consumption is reduced to a minimum by the use of multidigit cluster LED seven-segments display operating in the common anode type. The introduction of multiplexer circuit in the logic design reduces the number of digital IC's in the Timer/Counter remarkably. The precision of the timer depends on the stability of the line frequency which is in many cases better than 0.1% and is high enough for normal operation. The 10 MHz counting speed of the counter portion is high enough for spectroscopy applications. The printout control circuit is not included since the printing peripherals usually require specific printout interface which may include printer drivers of high power. Anyhow the Timer/Counter does provide print output with signals in BCD which can be easily interfaced to most common teletype-

writers and line printers and is even directly compatible to the ORTEC 777 Line Printer and ORTEC 432 A Printout Control.

However the Timer/Counter Still has a disadvantage due to high power consumption. This is due to the fact that the +5V supply is derived from the +12 V from the NIM supply causing a power loss of 8.4 watts. The current drain from the +5 V supply is 1.2 A which is more than 50% of the load that can be delivered by the +12 V supply, causing an unbalance in load distribution and may limit the application of the NIM supply under use. It is recommended that the + 5 V supply should be derived independently from the 110 V AC which is rarely used. A step down transformer in conjuction with bridge rectifier and a proper three-terminal IC regulator is usually sufficient to power the digital IC's in the Timer/Counter.

APPENDICS

APPENDIC A

A.1 BOOLEAN ALGEBRA [11]

A.1.1 Introduction

George Boole (1815-64) introduced an algebra of logic in the mid-1800 that became known as boolean algebra. The present-day application of Boolean algebra to binary functions is credited to a paper by C.E. Shannan in 1938. The entire framework of the calculus of propositions depends on the concept that logical statements can be made and can be designated as true or false. No asignments of intermediate values is allowed. The application of this mathematics to switching and digital design has become the workhorse of the logic designer.

A.1.2 Fundamentals

The fundamentals of logic design are based on a set of postulates and resulting theorem.

Postulate 1a. $X = 1$ if $X \neq 0$.

Postulate 1b. $X = 0$ if $X \neq 1$.

These two postulates form the fundamental definition of a dual-valued variable; that is, if the function X is 1; then it cannot be 0; similarly, if it is 0, it cannot be 1. This occurs with the cancept of true and false.

Postulate 2a.	$0.0 = 0.$
Postulate 2b.	$1 + 1 = 1.$
Postulate 3a.	$1.1 = 1.$
Postulate 3b.	$0 + 0 = 0.$
Postulate 4a.	$1.0 = 0.1 = 0.$
Postulate 4b.	$0 + 1 = 1 + 0 = 1.$
Postulate 5a.	$\bar{0} = 1.$
Postulate 5b.	$\bar{1} = 0.$

A.1.3 Theorems

As in ordinary algebra the letters of the alphabet are used to represent variables, the distinction being that Boolean variables may only have the value 1; and if not 1, then 0. One can define the transistor switch as being in the full ON state or the full OFF state. If the digit 1 is assigned to the full ON state, then digit 0 is assigned to the OFF state. This agrees with the calculus of propositions where by the digits 1 and 0 do not represent numerical values, but rather states or conditions.

Theorem 1a.	$X + 0 = X.$
Theorem 1b.	$X.1 = X.$
Theorem 2a.	$X + 1 = 1.$
Theorem 2b.	$X.0 = 0.$
Theorem 3a.	$X + X = X.$
Theorem 3b.	$X.X = X.$
Theorem 4a.	$\overline{(\bar{X})} = X.$
Theorem 4b.	$\overline{(\bar{X})} = X.$

Theorem 5a.	$X + \bar{X} = 1.$
Theorem 5b.	$X.\bar{X} = 0.$
Theorem 6a.	$X(X+Y) = X.$
Theorem 6b.	$X+XY = X.$
Theorem 7a.	$X(\bar{X}+Y) = XY.$
Theorem 7b.	$X+\bar{X}Y = X+Y.$
Theorem 8a.	$XY+XZ = X(Y+Z).$
Theorem 8b.	$(X+Y)(X+Z) = X + YZ.$
Theorem 9a.	$XY+YZ+\bar{X}Z = XY + \bar{X}Z.$
Theorem 9b.	$(X+Y)(\bar{X}+Z) = XZ + \bar{X}Y.$
Theorem 10a.	$\overline{X+Y+Z+\ \text{---}} = \bar{X}.\bar{Y}.\bar{Z}...$
Theorem 10b.	$\overline{X.Y.Z...} = \bar{X}+\bar{Y}+\bar{Z}\ \text{----}$
Theorem 11a.	$XY + \bar{X}Y = Y.$
Theorem 11b.	$(X+Y)(\bar{X}+Y) = Y.$

The above theorems are extremely useful in algebraic manipulation of Boolean functions. But the implementation of binary functions involves a certain degree of skill in recognizing theorems within equations and the manipulation of these equations by use of the theorems.

Each variable in Boolean algebra can have either of two states, 1 or 0. Thus two variables taken together can have 2^2 or four states. Three variables can be written in 2^3 different ways and n variables can be written in 2^n different ways. This procedure gets quite complex as the number of variables increases. An approach to a systematic listing of all the binary states that

a group of variables can take is a table of combinations or truth table. The truth table is quite useful in analyzing certain functions but it lacks compactness and utility.

A more eloquent and useful presentation of the combination is the Karnaugh map.

A.2 COMBINATION LOGIC

A.2.1 Introduction

Combination logic refers to networks whose output is strictly dependent on its inputs. The mathematical framework of combination logical design is based upon the concept that logical statements can be designated as true (1) or fulse (0). The analysis of such network requires first the writing of this Boolean algebraic equation representation of the network, and then the complete characterization of the output as a result of the possible combinations of the inputs.

A.2.2 LOGIC expressions

It is often desirable to write logic expressions in an easily examinable form. For this purpose logic expressions may be written as:

1. A sum of products, i.e., an OR-ing of AND-ed variables such as

$$AB + AC + \bar{B}C$$

in which the products are terms of the expression or.

2. A product of sums , i.e., an AND-ing of OR-ed variables such as.

$$(P + \bar{Q})\ (Q + R)(P + R)$$

in which the sums are the terms of the expression. The terms

in the expressions above can be expanded to obtain a general form in which each term is a unique combination of all variables involved. A sum of products can be expanded by applying theorems 1b, 3b, 5a. For example.

$$
\begin{aligned}
Y &= AB + AC + \bar{B}C \\
&= A.B.1 + A.C.1 + \bar{B}.C.1 \\
&= AB\,(C + \bar{C}) + AC\,(B + \bar{B}) + \bar{B}C\,(A + \bar{A}) \\
&= ABC + AB\bar{C} + ABC + A\bar{B}C + A\bar{B}C + \bar{A}\bar{B}C \\
&= ABC + AB\bar{C} + A\bar{B}C + \bar{A}\bar{B}C
\end{aligned}
$$

The general form obtained is called the minterm canonical form, sometimes also referred to as the standard sum. Its individual term are usually called minterms. Other names used instead of minterm are minterm canonical term, canonical term, product term and P-term.

A product of sums can be expanded into the general form by applying theorems 1a, 5b For example.

$$
\begin{aligned}
Y &= (P + \bar{Q})(Q + R)(P + \bar{R}) \\
&= (P + \bar{Q} + 0)(Q + R + 0)(P + \bar{R} + 0) \\
&= (P + \bar{Q} + R\bar{R})(Q + R + P\bar{P})(P + \bar{R} + Q\bar{Q}) \\
&= (P + \bar{Q} + R)(P + \bar{Q} + \bar{R})(Q + R + P)(Q + R + \bar{P}), \\
&\quad (P + \bar{R} + Q)(P + \bar{R} + \bar{Q}) \\
&= (P + \bar{Q} + R)(P + \bar{Q} + \bar{R})(P + Q + R)(\bar{P} + Q + R) \\
&\quad (P + Q + \bar{R})(P + \bar{Q} + \bar{R})
\end{aligned}
$$

This is called the maxterm canonical form and sometimes the standard product. The individual terms are usually called

maxterms, other names are maxterm canonical term, sumterm, and S-term.

A.2.3 Simplification and Minimization

Minimization involves reducing a Boolean algebraic expression to some minimal form. Any minimization tool in Boolean is based on the algebraic theorems. Algebraic reduction of Boolean functions is not easy and require considerable experience judgement and luck. Simplifying logic equations by applying Boolean algebra is possible, but it often becomes a cumbersome and time consuming chore. Therefore, special minimization techniques have been developed to obtain the desired result faster and more accurately. Most common, for a small number of variables is the use of Karnaugh maps.

.2.4 Karnaugh map

The Karnaugh map is an orderly arrangement of squares with assignments such that the difference between any two adjacent squares is a one-variable change. The map must contain a square or cell for each unique combination of variables. A map for two variables must contain four cells, because there are 2^2 different combinations of two variables. A map for three variables must contain 2^3 or eight cells and a map of n variables must contain 2^n cells. An assignment of 1 for an uncomplemented variable and 0 for a complemented variable. Thus, a Karnough map may be considered to be the graphic representation of the minterm canonical form. Each minterm is represented by a cell and the cells are

assembled in an orderly arrangement such that adjacent cell represent minterms which differ by only one variable. For example a Karnaugh map for four variables is shown in Fig. A.1. The diagonal line at the upper left-hand corner of the matrix indicates that the variables A and B are represented by the binary notation across the top of the matrix in the same sequence. The combination of A and B as indicated by the binary notations on top are contained in the column below each assigment. The variables C and D are assigned down the side of the matrix and are contained in the horizontal rows of the matrix.

Because of the special sequence of the binary notations it can be seen that the cell arrangement is such that minterms of adjacent cells are identical except for one variable which appears complemented in one cell and uncomplemented in the adjacent cell. According to this definition of adjacency, the cells at the extreme ends of the same horizontal row are also to be considered adjacent. The same applies to the top and bottom cells for column. As a result, the four corner cells of a Karnaugh map also must be considered to be adjacent.

A logic expression is mapped by entering a 1 in the cells representing minterms that are contained in the expressions, where as a 0 is entered in the cells representing nonpresent minterms. By applying the distributive property ($AB + AC = A(B+C)$) and the theorem 5a and 1b to the minterms of two adjacent cells,

DC \ BA	00	01	11	10
00	$\bar{D}\bar{C}\bar{B}\bar{A}$	$\bar{D}\bar{C}\bar{B}A$	$\bar{D}\bar{C}BA$	$\bar{D}\bar{C}B\bar{A}$
01	$\bar{D}C\bar{B}\bar{A}$	$\bar{D}C\bar{B}A$	$\bar{D}CBA$	$\bar{D}CB\bar{A}$
11	$DC\bar{B}\bar{A}$	$DC\bar{B}A$	$DCBA$	$DCB\bar{A}$
10	$D\bar{C}\bar{B}\bar{A}$	$D\bar{C}\bar{B}A$	$D\bar{C}BA$	$D\bar{C}B\bar{A}$

FIG.A.1 Four-Variable Karnaugh Map with Minterm Designation and Numerical Assignment.

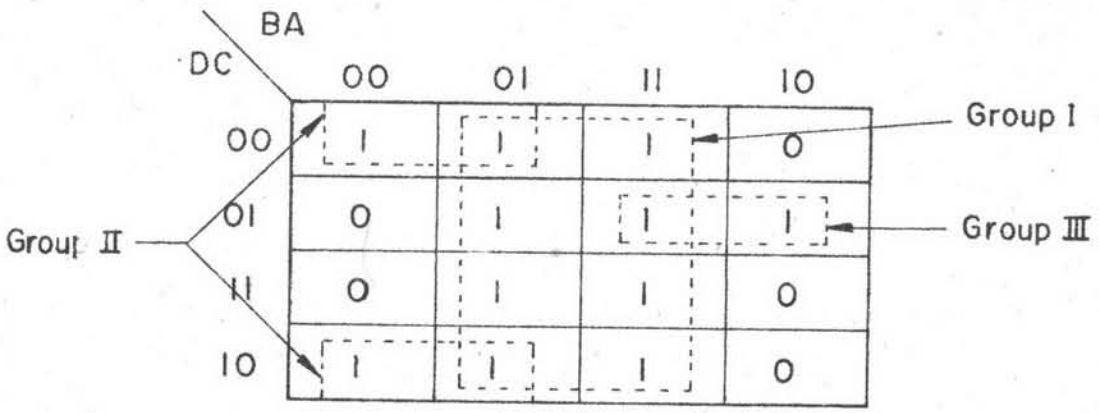


FIG.A.2 Karnaugh Map of Y.

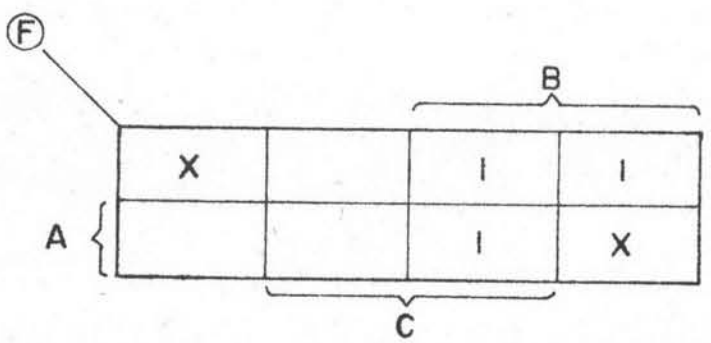


FIG.A.3 K-Map of $F(A,B,C) = ABC+\bar{A}BC+\bar{A}B\bar{C}$.

one variable can be eliminated, because the condition or operation determined by the two minterms is independent of the eliminated variable. The two combined cells may again be combined with two other adjacent cells if the two cell groups are adjacent to each other. Thus a group of four adjacent cells can be combined resulting in the elimination of two variables. This process of combining adjacent groups of combined cells may be continued as far as possible.

In general, the combination of 2^n adjacent cells results in the elimination of N variables. This can also be stated as

$$n = \log_2 m$$

where n = number ofcells eliminated; $n = 0,1,2,3.....$

m = number of cells combined; $m = 1,2,3,8,16.....$

For example the function of

$Y = ABC + AB\bar{D} + A\bar{B}C + A\bar{B}D + AC\bar{D} + \bar{A}\bar{B}\bar{C} + BC\bar{D} + \bar{B}\bar{C}\bar{D}$ in Fig. A.2 can be combined into three groups (I, II, and III). It can be derived that the terms represented by the three groups are

Group I : A

Group II : $\bar{B}\,\bar{C}$

Group III : $B\,C\,\bar{D}$

Thus $Y = ABC + AB\bar{D} + A\bar{B}C + A\bar{B}D + AC\bar{D} + \bar{A}\bar{B}\bar{C} + BC\bar{D} + \bar{B}\bar{C}\bar{D}$

$= A + \bar{B}\,\bar{C} + BC\bar{D}$

A.2.5 Redundancies: Don't-care and Can't-happen conditions.

In many applications, however, there are input combinations that, for some reason or other, do not require a specific logical output value. These cases, commonly called redundancies, fall into two general categories don't care input conditions wherein the output value of the function is of no concern to the designer and can't-happen input conditions that because of some constraint can never appear at the input to the combinational device. Combinational problems that have either don't care or can't happen inputs are said to be incompletely specified. The redundancies are useful in simplifying the combinational logic that is required to implement the Boolean expression, since the output values that are produced in the presence of a redundant input combination can be freely chosen as TRUE or FALSE, depending on which values are most useful in simplifying the logical systems Redundancies are normally plotted as an X on a K-map, with the X being freely used as either a 0 or a 1 (or as both) during the grouping process. For example, let $F(A,B,C) = ABC + \bar{A}BC + \bar{A}B\bar{C}$ and the input conditions $\bar{A}\bar{B}\bar{C}$ and $AB\bar{C}$ can never occur. The K-map for F can be shown in Fig. A.3 Without the use of the redundancy ABC the expression for F would be

$$F = \bar{A}\,B + BC$$

But if ABC is used as 1 the grouping of F becomes.

$$F = B$$

A.2.6 Pratical Logic Gates

A.2.6.1 The most common logic gates are

1. NAND Gate. The NAND gate is AND followed by a NOT operation. The symbol, logic expression, and truth table for a 2-input NAND gate are shown in Fig . A.4. According to De Morgan's theorem, this may be written as.

$$Y = \overline{AB} = \bar{A} + \bar{B}$$

From this expression it can be seen that a single NAND gate performs an OR operation on its complemented input signal.

2. NOR Gate. The NOR gate is an OR followed by a NOT operation. The symbol logic expression and truth table are given in Fig. A.5

3. NOT (inverter). The inverter performs the NOT operation. The symbol, logic rxpression and truth table are shown in Fig. A.6

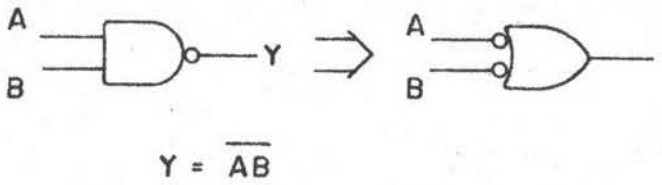
A.2.6.2 Analysis of interconnected array of logic gates:

According to the consequences of De Morgan's theorem for NAND and NOR gates, the logic expression of the output signal of an interconnected array of inverting gates can be stated in general as follows.

1. Consider the gate from which the output signal will be obtained as the first (odd) level of inversion, the preceding gate as the second (even) level, etc.

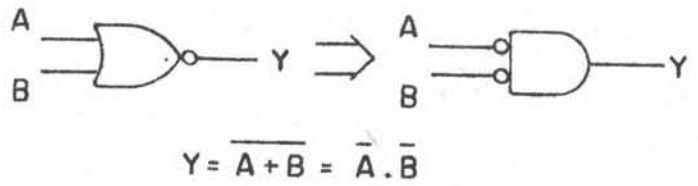
2. Consider all NAND gates in odd levels to perform the OR operation.

3. Consider all NAND gates in even levels to perform the AND operation.



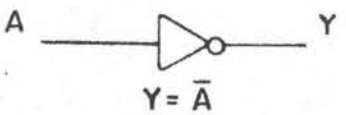
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

FIG.A.4 NAND Gate.



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

FIG.A.5 NOR Gate.



A	Y
0	1
1	0

FIG.A.6 Inverter.

4. Consider all NOR gates in odd levels to perform the AND operation.

5. Consider all NOR gates in even level to perform the OR operation.

6. All input variables entering gates in odd levels should appear complemented in the logic expression of the output signal.

7. All input variables entering gates in even levels should appear uncomplemented in the logic expression of the output signal

For example the logic expressions for Y_1 and Y_2 in Fig. A.7 a can be easily simplified by using the above methods.

A.2.7 DOT - AND/ORING:

Dot-AND/ORing (wired-OR, dot-OR or collector-dotting) is the most universally used way of getting information from peripheral equipment into a central station through an input bus as shown in Fig. A.8. The central station will send out a device number, which goes to all the peripheral devices. When this device number picks out device No1, that device decodes this device number so that the x input to each of the gates shown goes High. If the other input on any gate is also High, then the output goes Low, pulling that bus line Low, Meanwhile, the inputs to the bus lines from all the other peripheral units are disabled because their x inputs are all low (preventing the turn-on of their output

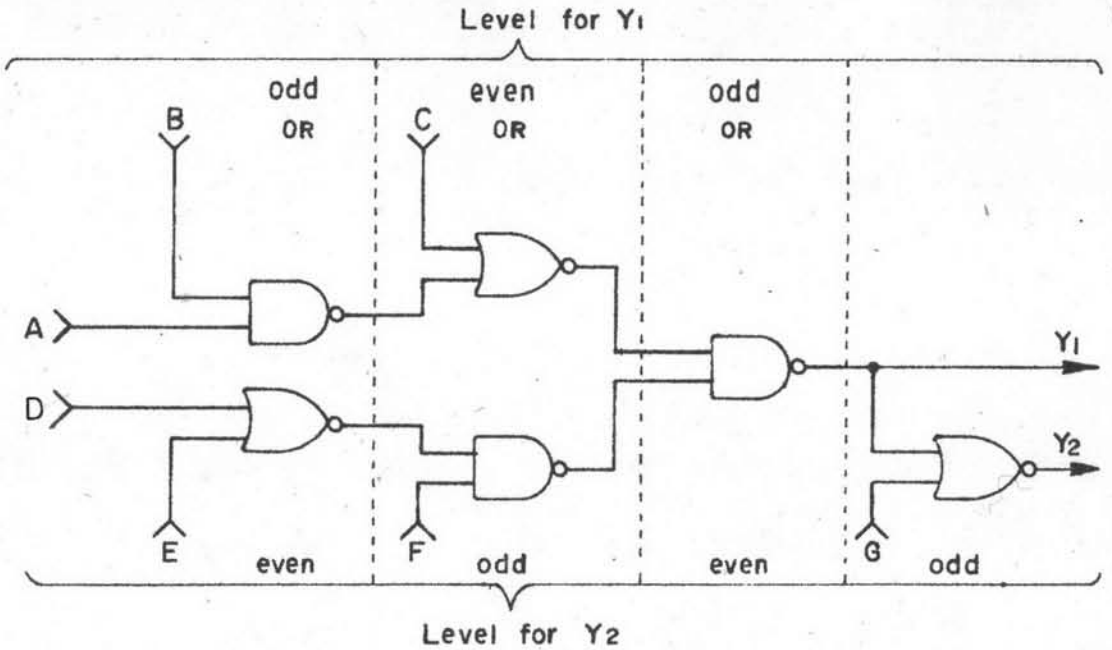


FIG.A.7 (a) Interconnected Array of Inverting Gates.

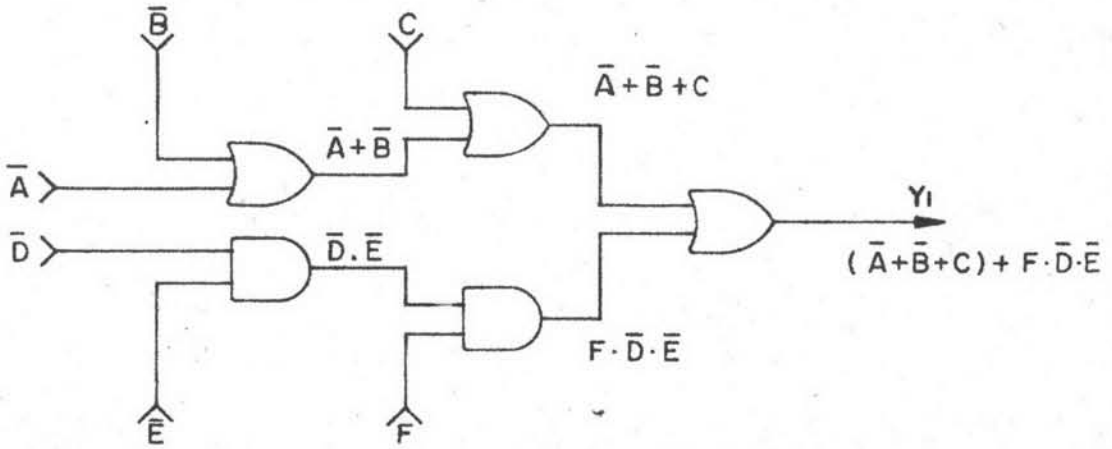


FIG.A.7 (b) Equivalent Logic Diagram for Y_1.

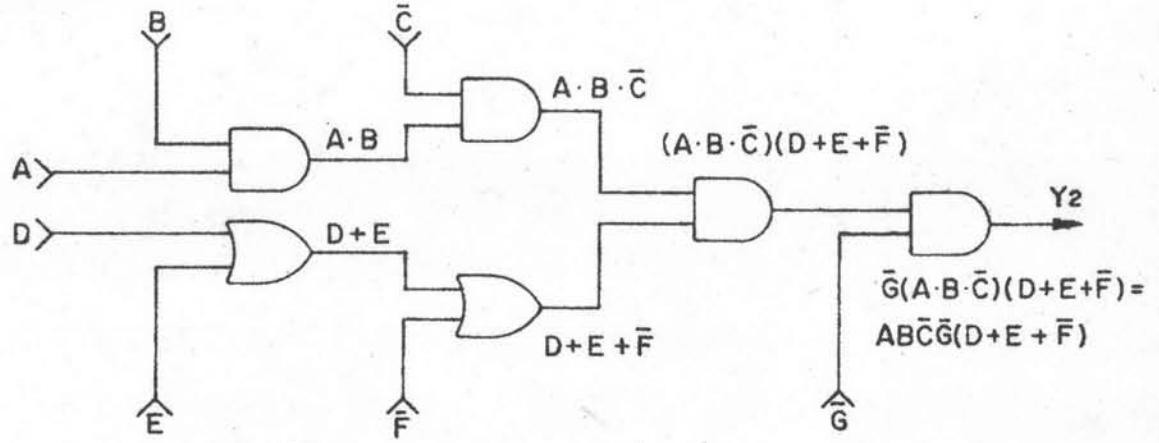


FIG.A.7 (C) Equivalent Logic Diagram for Y_2.

transistors). By ORing the m input from each of n peripherals in this way the number of input wires to the central station is reduced from m x n to just m. Fig. A.8 Shown Dot/ORing on an input bus to a central station.

Consider two NAND gates connected in parallel as shown in Fig. A.9 a and b. The output F is pulled Low if either gate is turned ON and since the output will be turned ON if and only if both inputs to the gate go High therefore the expression for output F should be.

$$F = \overline{AB + CD}$$

From this expression the equivalences can be shown in Fig. A.10 a and b

The following steps should be considered in simplifying any circuit with logic having Dot-AND/OR capability:

1. Design a circuit, forgetting about the Dot-AND/OR capability.

2. Look at the resulting circuit. Anywhere there are two or more gates whose output go only into one gate, a Dot-AND/OR simplification is possible, tie the output of these gates together and replace the gate they feed with an inverter.

3. Remove any inverter pairs which result, provided the circuit does not use the signal between the inverters.

As an example the circuit of Fig.A.11a can be simplified by applying the above procedure to Fig. A.11c.

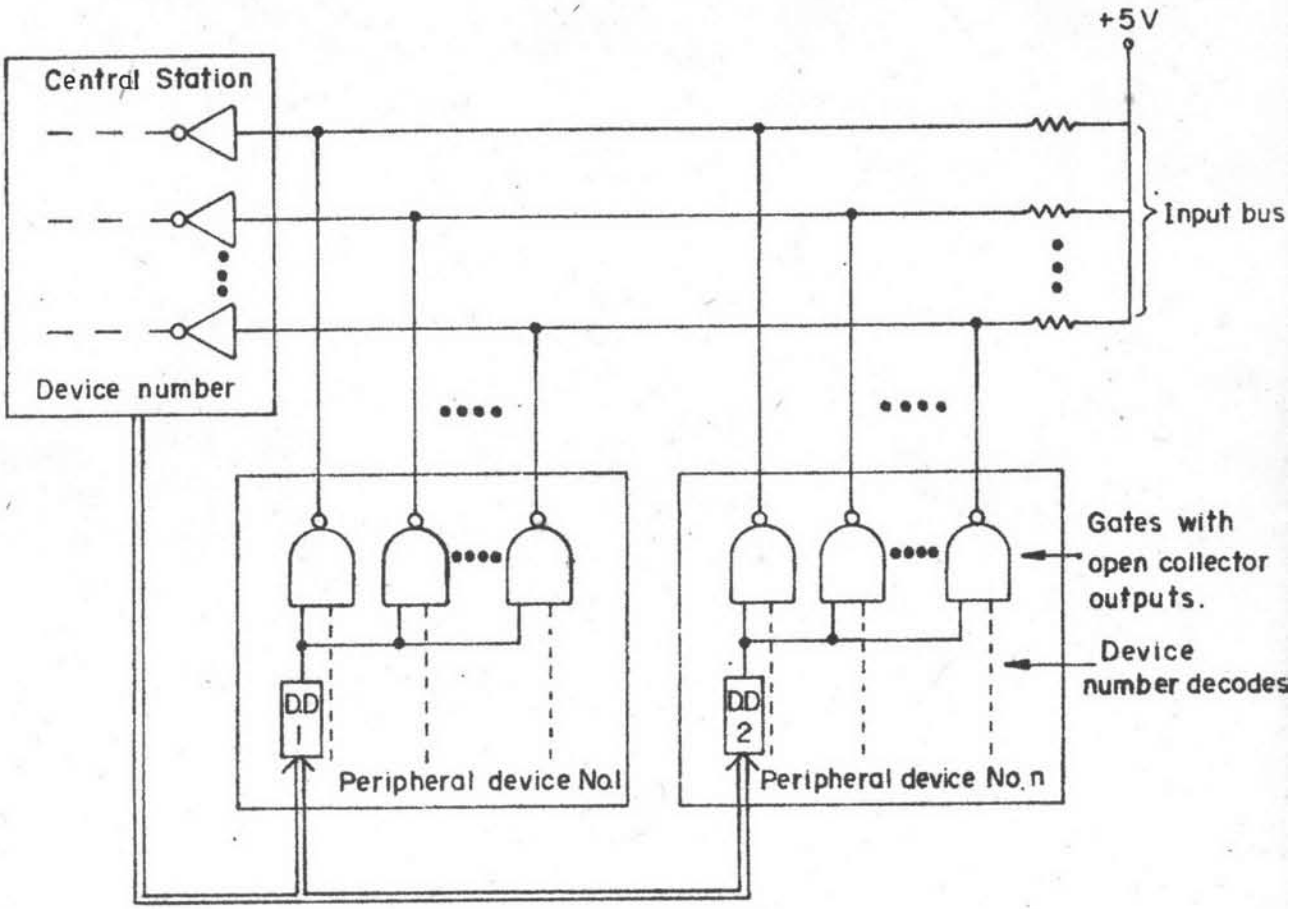


FIG.A.8 Dot-AND/ORing on an Input Bus to a Central Station

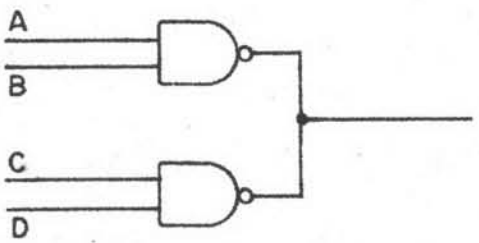


FIG.A.9 (a) Tying outputs of two gates together.

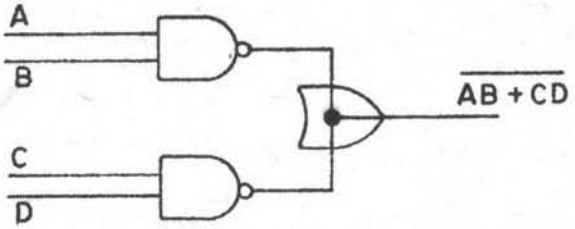


FIG.A.9 (b) Symbolism for do-ORing.

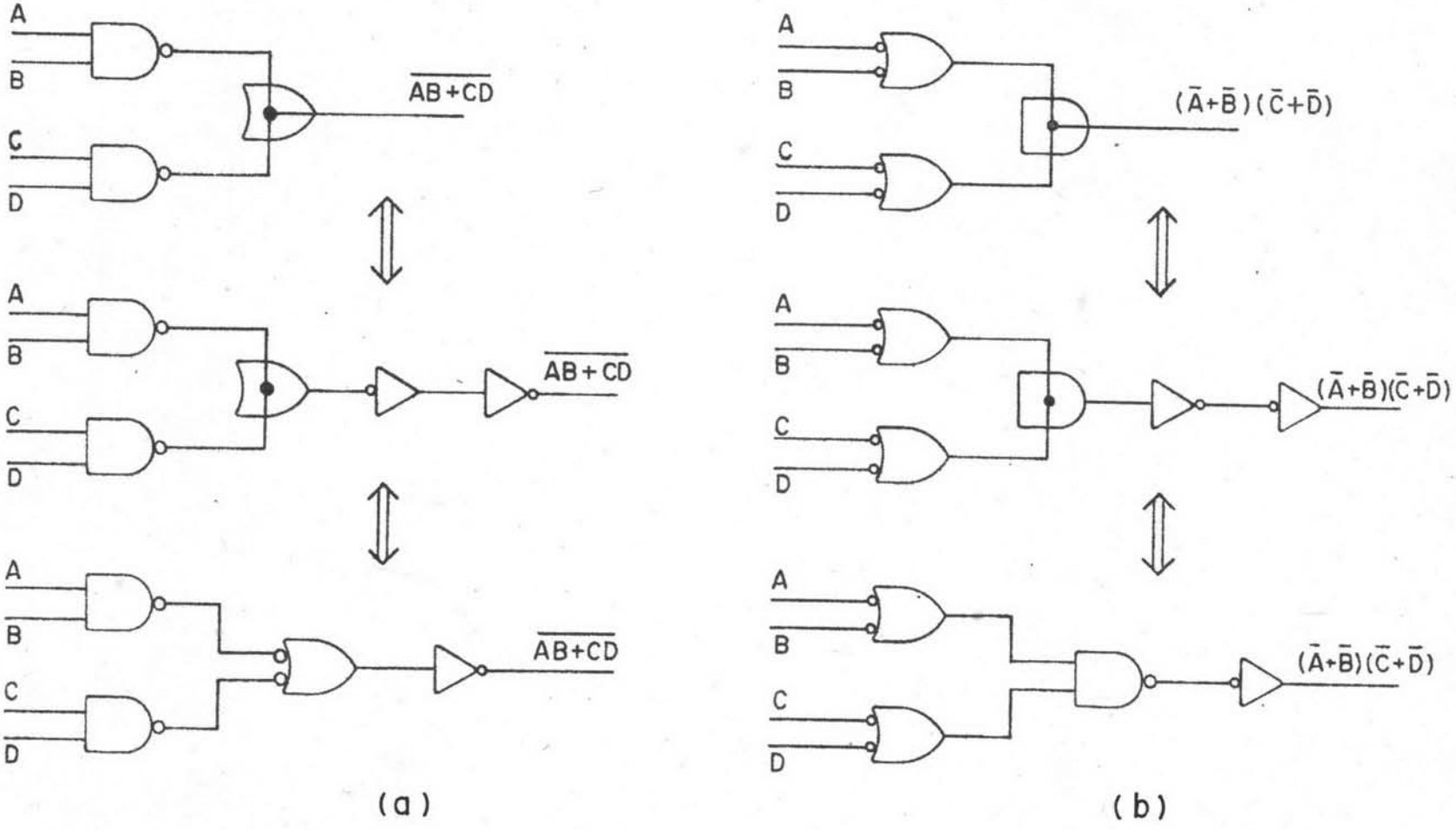


FIG.A.10 Equivalent Circuit for Dot-AND/ORing.

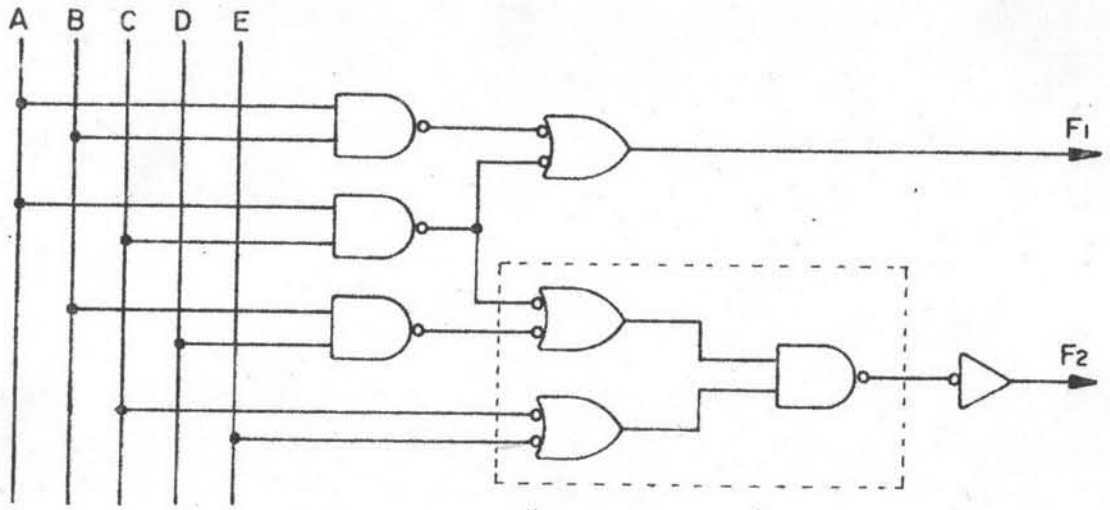


FIG.A.II(a) Initial Circuit.

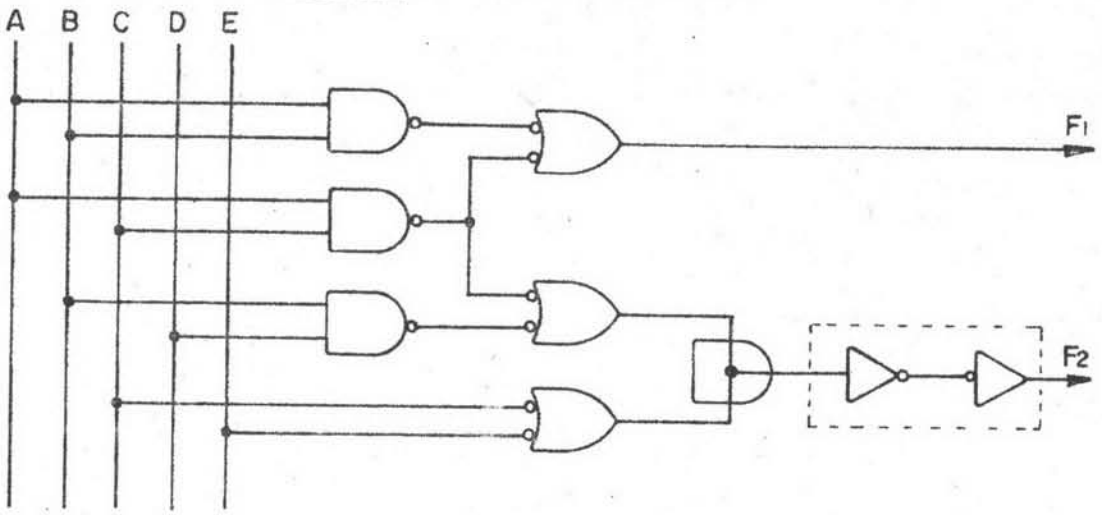


FIG.A.II (b) Circuit after Dot-ANDing.

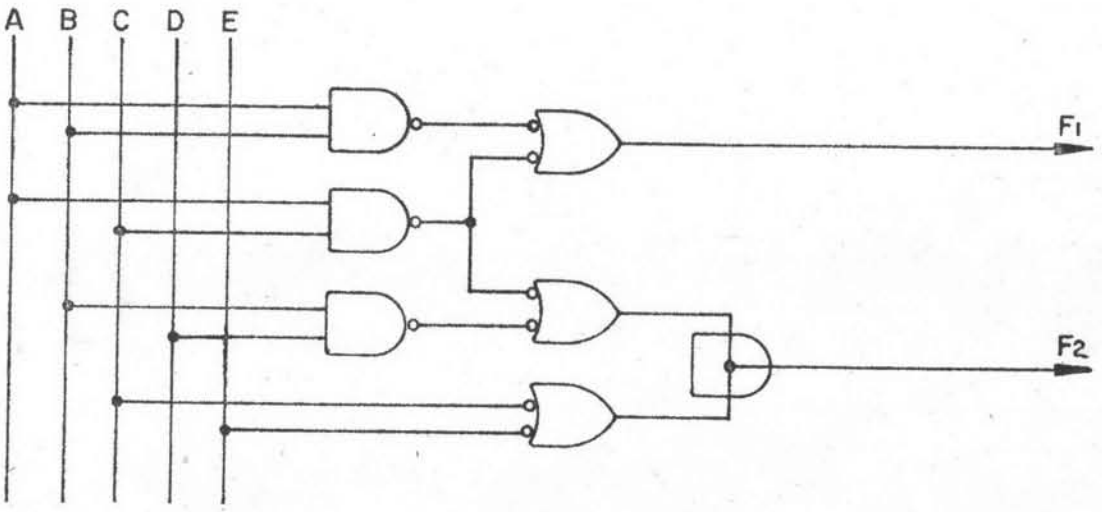


FIG.A.II (c) Final Circuit after Removing Inverter Pair.

A.2.8 BCD-To-SEVEN SEGMENT DECODER

Many devices have been developed for displaying the output signals that are produced by digital systems. One of the more recent developments is the seven-segment display unit which can indicate all the decimal digits and several alphabetic characters. A typical arrangement for the segments is shown in Fig.A.12a.

There is some latitude in selecting the segmented representations for the various digits. For example, the digit 1 can be represented by segments b and c or by segments e and f. The b-c representation is used here in. The representation for 6 is also somewhat open. The c,d,e, f and g segments must be ON, but the a segment can be either ON or OFF. Similarly don't care segments occur with the d segment for a 9 and the segment for a 7 as shown in the truth table of Fig. A.12 b.

By applying the combination design process for segment a the simplified expression for segment a is obtained as shown in Fig. A.12c. Continuing the same process the simplified Sum-OF-Products forms for the other six segments are given as

$$c = B + \overline{C} + D$$

$$d = \overline{B}\,\overline{D} + B\overline{C}D + \overline{B}C + C\overline{D}$$

$$e = \overline{B}\,\overline{D} + C\overline{D}$$

$$f = A + B + \overline{C}\overline{D}$$

$$g = A + C\overline{D} + (B + C)$$

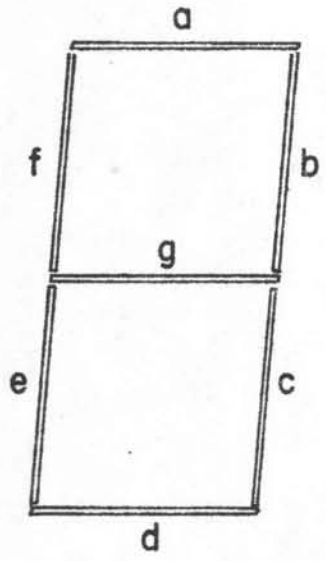


FIG. A.12(a) Typicial Seven-Segment Display.

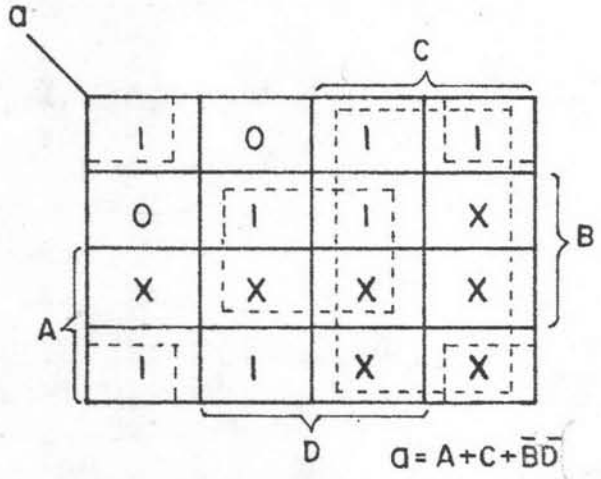


FIG.A.12(c) K-Map Minimization of the a Output Function.

Inputs				Outputs							Symbol
A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	X	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	X	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	X	0	1	1	9

FIG.A.12 (b) Truth Table of the BCD to Seven-Segment Decoder.

A.3 SEQUENTIAL SYSTEMS

A.3.1 Introduction

In many logical systems the outputs from the system cannot be determined from knowledge of the present input values alone. Rather, the past history of the input and output signal must be included in their determination. Such systems are commonly described by the term sequential system which are generally characterized as having two properties:

1. There is at least one feedback path from the output of the system to the input of the system.

2. The system has a memory capability for holding past of information, so that previous input and output values cannot be used in determining the current (and future) output signals. In general the system can be shown in Fig. A.13.

A.3.2 Sequential System Description

A.3.2.1 Inputs, outputs and system states

The details of the generalized system of Fig.A.13 is shown in Fig.A.14. All the inputs to and outputs from the system are assumed to be Boolean in native and may fall into two classes, level or pulse signals. The combination portion of the system receives two sets of input signals: the primary input variables and the secondary input variables. The secondary input variables are taken from the memory portion of the sequential system and are sometimes called the present-state variables

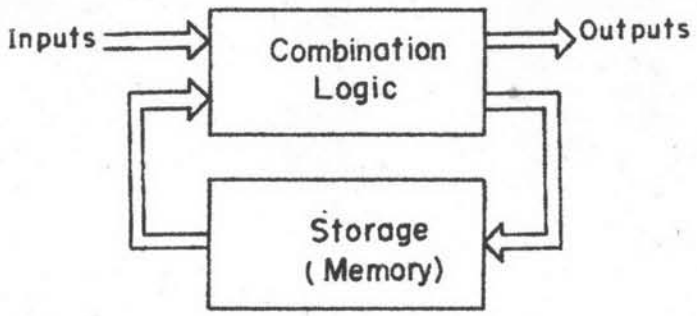


FIG.A13 A Generalized Sequential System.

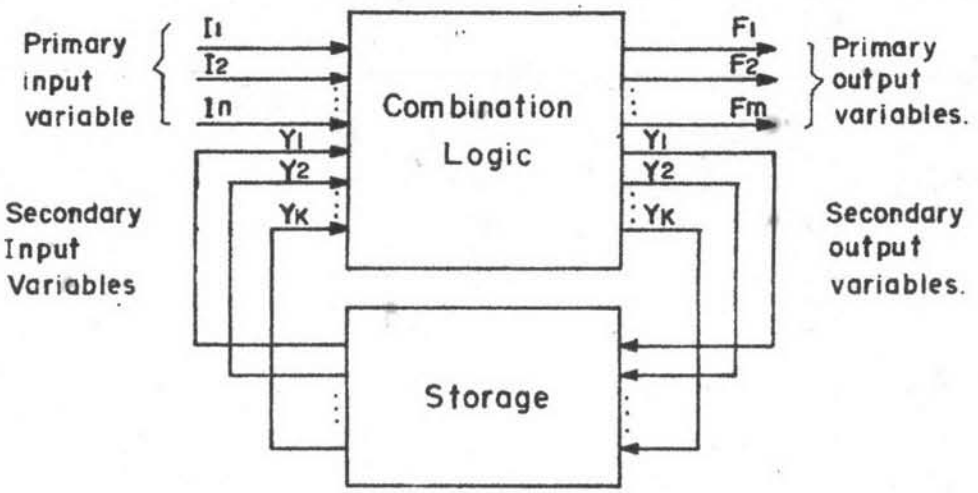


FIG.A14 Details of a Generalized Sequential System.

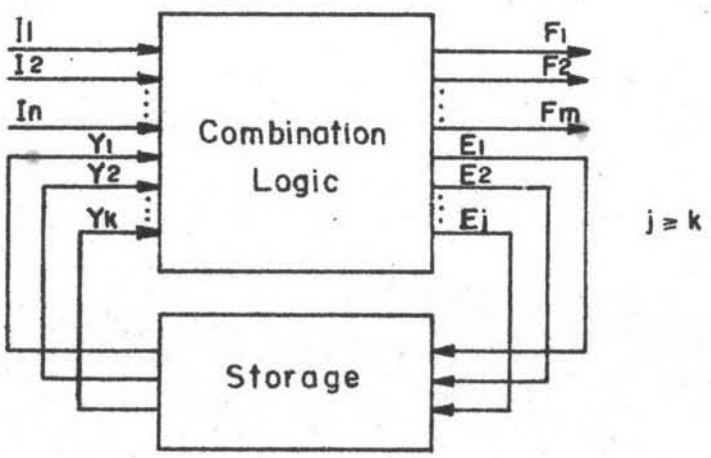


FIG.A15 Generalized Sequential System that Uses Excitation Variables.

The values of the secondary input variables are a reflection of the past history of the sequential system's operation. The values of the primary input variables define the current input condition one of 2^n different possible combinations. Taken together the two sets of inputs are said to define the total present state of the system with 2^{n+k} different total present states being possible.

The outputs from the combinational portion of a sequential system are also separated into two categories:

1. The primary output variables ($F_1, F_2 \ldots F_m$) are the signals that are sent from the sequential system into its associated environment and may operate other systems.

2. The secondary output variables ($Y_1, Y_2 \ldots Y_k$) which are returned to the memory portion of the sequential system. These signals describe the new values of the information that will be stored in the memory when the next cycle of the system's operation takes place. Thus, the secondary output signals are said to describe the next state of the sequential system.

The transfer from present state to next state takes place in one of the two ways, depending on the type of memory that is used by the sequential system. If the memory is provided through delay, the present-state-to-next state change takes place when the end of the delay period is reached. If, on the other hand, the memory is provided by storage devices, the state change occurs whenever the control signals to the storage devices cause the

new values of the state variables to replace the old.

It should be noted that many types of storage devices cannot operate directly from the signals. They require a different set of control signals, commonly called excitation variables. These variables are determined from the total present state. The use of excitation variables eliminates the need for explicitly determining the values of the Y variables, giving rise to the modified sequential system organization that is shown in Fig. A.15.

A.3.2.2 State Tables and State Diagrams

The relationships among present-state variables, primary input variables next state (or excitation) variables, and primary output variables that describe the behavior of a sequential system can be specified using state tables and state diagrams which can be best illustraled using the simple sequential system as shown in Fig. A.16.

The system has one secondary input variable when combined with two primary inputs giving out eight different total present states. The values of the next-state variable Y, and the primary output variable F for all total present states can be arranged in a tabular form called state table as shown in Fig. .17

The system is said to be stable when all the next-state variables agree with their present-state counterpart, thus no change in state will take place when the end of the delay period is reached. Usually the stable states are indicated on

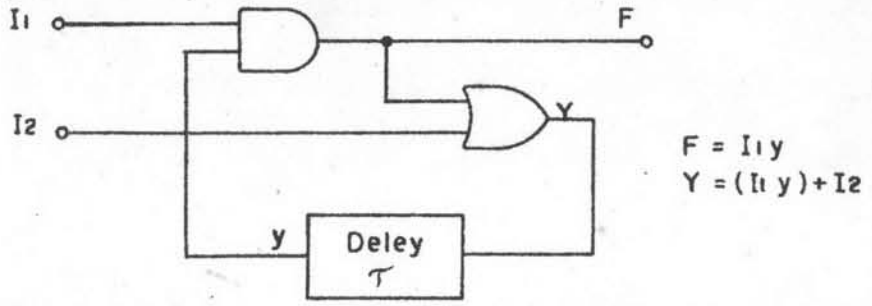


FIG.A.16 A Simple Sequential System.

Present state	Next state Y	Output F
y	I_1I_2=00 01 10 11	I_1I_2=00 01 10 11
0	⓪ 1 ⓪ 1	0 0 0 0
1	0 ① ① ①	0 0 1 1

FIG.A.17 State Table for the System of Fig.A.16

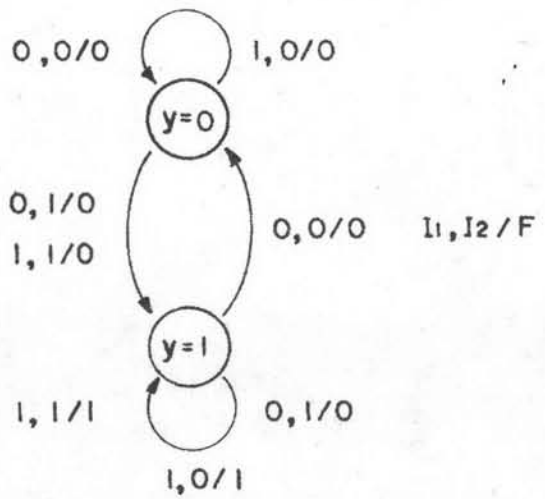


FIG.A.18 A State Diagram Describing the System of Fig.A.16.

the state table by circling their next-state location. If the next-state variable doesn't agree with the present-state counterparts, the system is in an unstable state. For example consider the total present state $yI_1\ I_2 = 100$, at the and of the delay period y will change from TRUE to FALSE. This change will move the system to the stable total present state $yI_1\ I_2 = 000$.

Another method for describing the behaviour of a sequential system is the use of a state diagram. This method presents a pictorial representation of the present state/next-state sequences that apply to the sequential device.

Each possible present state is represented by a circle, with the values of the present state variables given within the circle. State-to-state transfers are indicated by arrows. The primary input conditions that correspond to each transfer are indicated beside each arrow, along with the value of the output variable that coresponds to that present-state/primary input combination. Arrows that begin and terminate on the same state indicate the stable state. As an example the state diagram of Fig.A.16 is shown in Fig.A.18.

A.3.2.3 State Assignments

In the design of sequential system different present states are assigned by distinct names or symbols rather than by specific combinations of the present state variables. Varying the assignments will usually affect the complexity of the combinational portion of the sequential system. A good state assignment

is one that reduces the amount of combination logic that is required to implement the sequential operation.

A.3.2.4 Fundamental-Mode Operation

A system is said to be operating in the fundamental mode if and only if its primary input variables are never allowed to change in logical value unless the system is in a stable state. Thus it can be implemented by allowing only one of the input variables to change at a time and by separating successive input changes by a minimum time pereid that is sufficiently long to allow the system to settle into a stable state between the changes. If the above condition cannot be met special interface circuits called synchronizers are required.

A.3.2.5 Pulse-Mode Operation

Pulse-mode operation requires that all changes in the internal state of the sequential system take place only in response to a pulse signal. The system then requires storage devices (rather than delay alone) as their memory and the stored data that represents the present state must remain unchanged between pulses no matter how long the period between pulse inputs becomes. Usually the basic arrangement of a pulse-mode sequential system is shown in Fig.A.19.

A.3.3 Flip-flop

A device that exhibits two different stable states is extremely useful as a memory element in a binary system. Any electrical circuit that has this characteristic falls into the

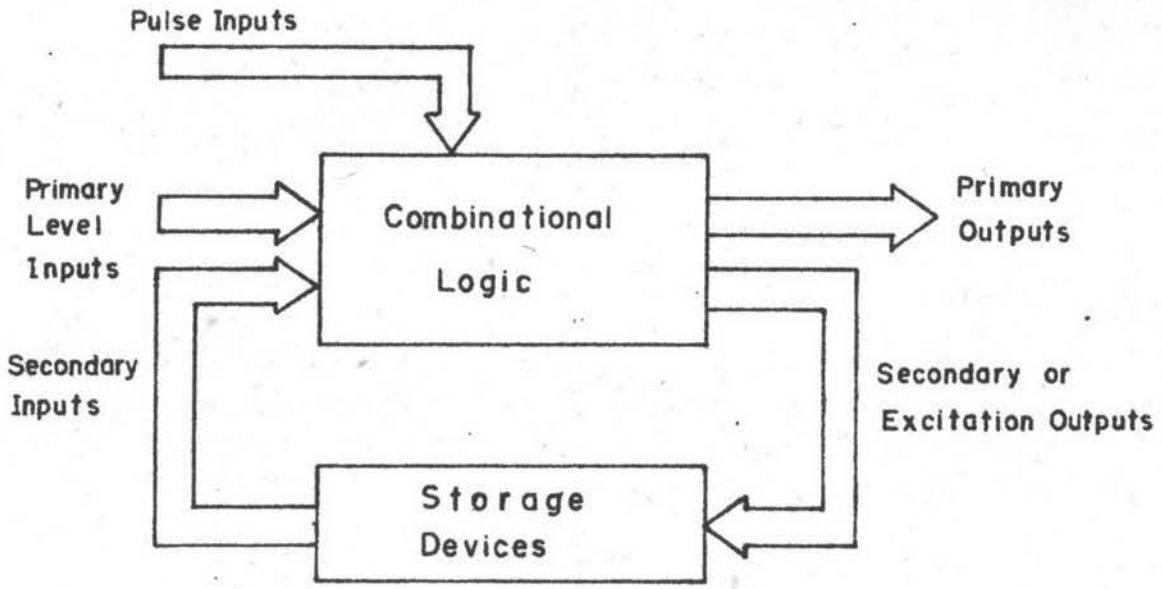


FIG.A.19 A Pulse-Mode Sequential System.

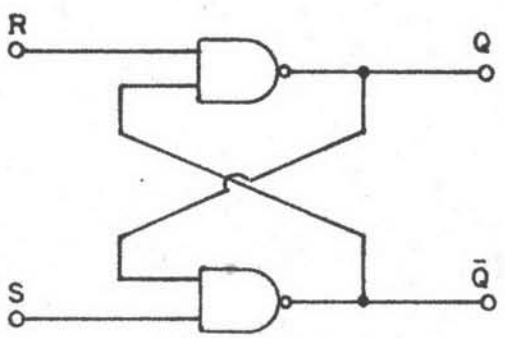


FIG.A.20 R-S Nand Latch.

category of devices commonly known as flip-flops. The basic types of flip-flops are the D,T,R-S and J-K. The flip-flops can have either synchronous clocking or asynchronous inputs. A synchronous data input is one which does not cause an immediate change in the output, it requires the occurence of another input called the clock pulse to generate the change of state. The clock input may be one of the general types: d-c or edge-triggered, a-c coupled and master-slave. If the flip-flops require no clock pulse to change state, they are called asynchronous clocking and the example of which is shown in Fig. A.20.

A.3.3.1 R-S Flip-flop

Clock R-S flip-flop is obtained by applying two steering gates to a basic latch circuit. The circuit, truth table and symbol are shown in Fig. A.21. The gate N_1 and N_2 form a latch whereas N_3 and N_4 are the control or steering gates which programe the state of the flip-flop after the pulse appears.

Note that when the Q output is in the logical 0 state and the S data input is at a 0 level, it does not matter whether the R input is a 0 or 1. The Q output at Q_{n+1} will always be a logical 0. Similarly if Q_n is a logical 1 and the R input is a 0, regardless of the state of the S input the output Q_{n+1} is a logical 1. The only factor that limits the use of R-S flip-flop in actual applications is the indeterminate condition of the output when R = S = 1. Since the output is not predictable, it is necessary to avoid this input condition when

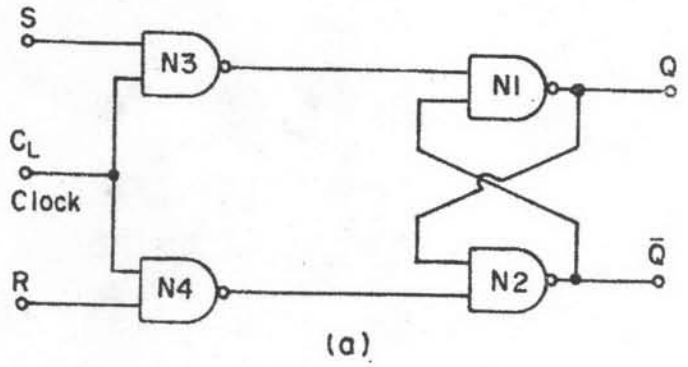
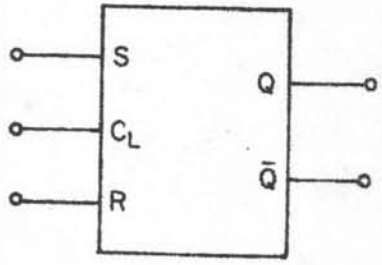


FIG.A.21 a) An R-S Clocked Flip-flop.

Sn	Rn	Q_{n+1}
0	0	Qn
1.	0	1
0	1	0
1	1	?

(b)

b) Truth Table.



(c)

c) Logic Symbol.

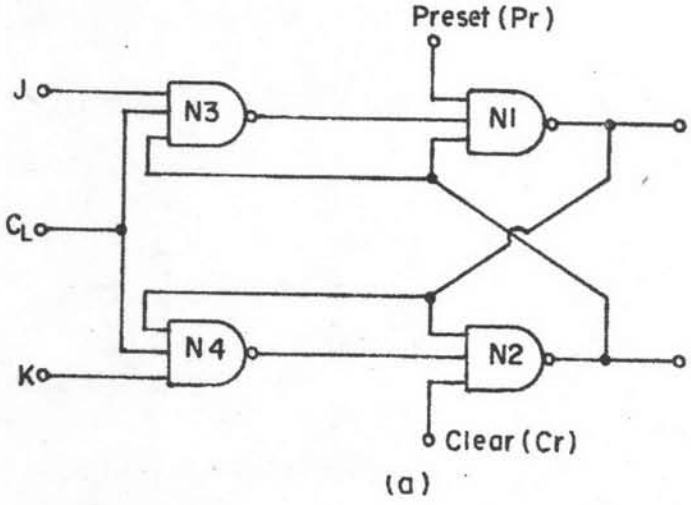
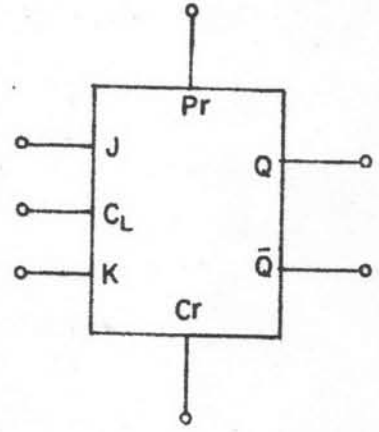


FIG.A.22 a) A J-K Flip-flop.

J	K	Q_{n+1}
0	0	Qn
0	1	0
1	0	1
1	1	$\bar{Q}n$

(b)

b) Truth Table.



(c)

c) Logic Symbol.

clocking. When it cannot be avaided, a J-K flip-flop is used.

A.3.3.2 J-K Flip-flop

The J-K flip-flop has two data inputs J and K, and only a single clock input. Many J-K types have the internal gates and the inputs to provide multiple J and K inputs, thereby eliminating the need for external gates in many applications. Fig. A.22 shows a J-K flip-flop with two asynchronous control input (preset and clear). The truth table is identical to the R-S type ($J = S, K = R$) with one exception for $J = K = 1$, the output changes state.

However, because of the feedback connection Q ($\bar{Q}$) at the input to K (J), the input will change during the clock pulse ($C_L = 1$) if the output changes state. This change takes place after a time interval Δt equal to the propagation delay through two NAND gates in series; hence for $J = 1, K = 1, C_L = 1$ the output will oscillate back and forth between 0 and 1. At the end of clock pulse the value of Q is ambiguous. This situation just described is called a race-around condition . To prevent the above phenomena, loop delay must be much less than the width of the clock or using Master-Slave type.

A.3.3.3 The Master-Slave J-K Flip-flop

The master-slave J-K flip-flop shown in Fig A.23 is a cascading of two S-R flip-flops with feedback from the output of the second (called the slave) to the input of the first (called the master). For $J = K = 1$, J(K) is anded with Q ($\bar{Q}$) from the

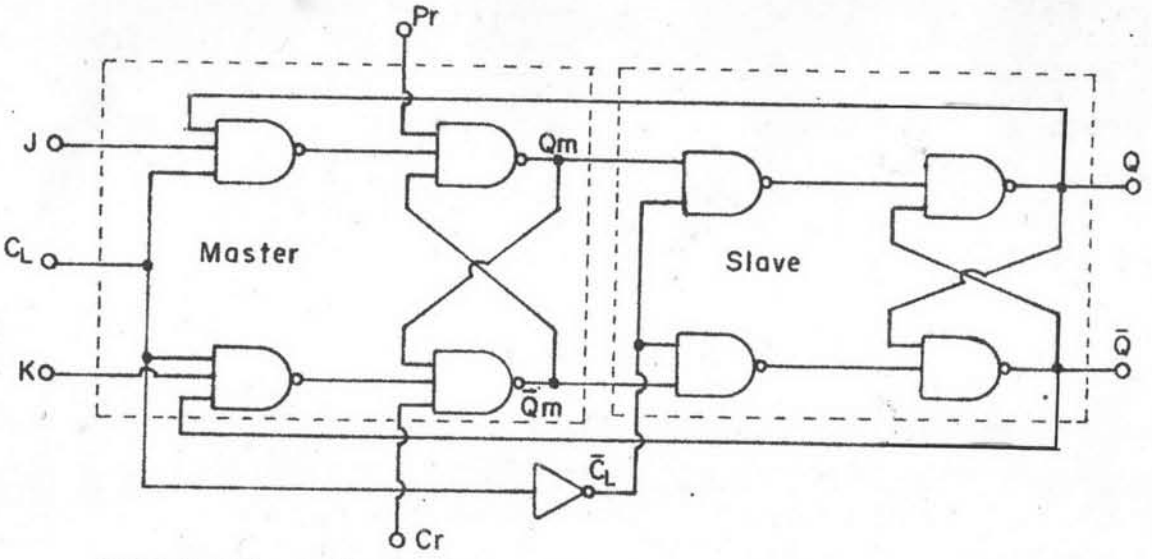


FIG.A.23 J-K Master - Slave Flip-flop.

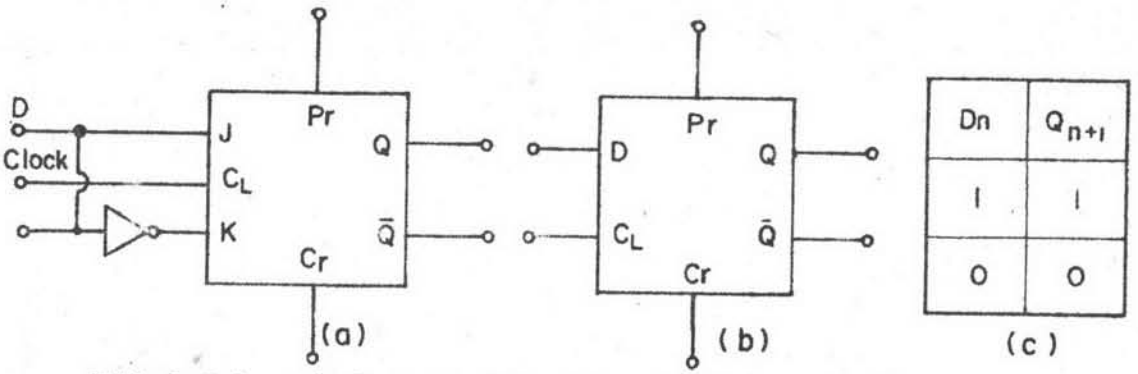


FIG.A.24 a) A J-K Flip-flop is Connected into a D-Type Latch.
b) Logic Symbol. c) Truth Table.

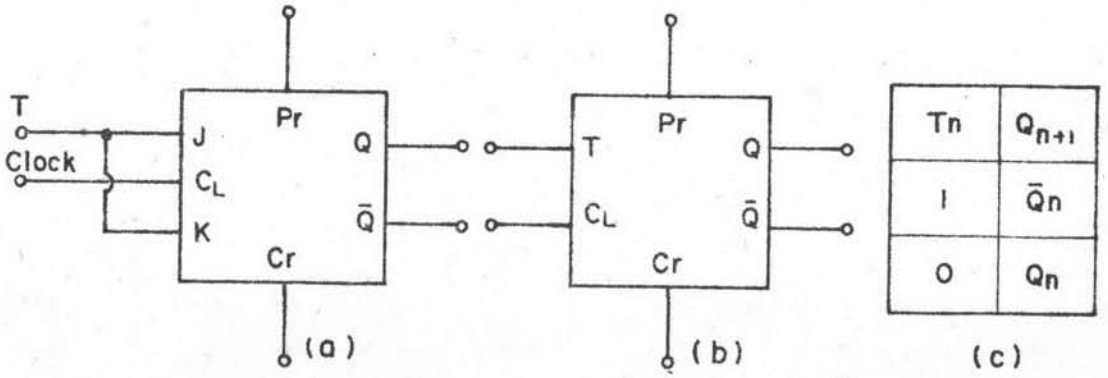


FIG.A.25 a) A J-K Flip- flop is Connected into a T-Type Flip-flop.
b) Logic Symbol. c) Truth Table.

the slave. When $C_L = 1$ only the master changes state, the slave is inhibited by $C_L = 0$. After C_L changes from 1 to 0 the master is inhibited and the data Q_m ($\bar{Q}_m$) is transferred to the slave. Hence, the race-around condition is circumvented.

In summary, during a clock pulse the output Q does not change but Q_m follows J-K logic; at the and of the pulse, the value of Q_m is tranferrect to Q.

A.3.3.4 D-type flip-flop

If a J-K flip-flop is modified by the addition of an inverter as shown in Fig.A.24, the unit is called a D (delay) flip-flop. The output Q_{n+1} after the pulse (bit time n + 1) equals the input D_n before the pulse (bit time n) as indicated in the truth table of Fig.A.24c.

A.3.3.5 T-type flip-flop

Tieing J and K inputs together a J-K flip-flop is converted into a T-type flip-flop as shown in Fig.A.25 a. Also the S-R and the D-type latches can be converted into toggle. The truth table and logic symbol is also shown in Fig.A .25c and b

A.3.4 The Counter

In general, a counter may be thought of as any sequential device that uses flip-flop for arithmetic operations for sequential signal generation or, of course, for counting. The basic operational concept of a counter is that for any present state there is a well-defined next state. During the counting operation the counter moves from present state to next state in

accordance with its specified counting sequence. The input signals that a counter receives may be limited in number. Such a counter must exhibit at least one closed path in its state diagram. The length of the path must be 2^N states or less, where N is the number of flip-flops that are present in the memory portion of the counter and there may be more than one closed path. The design of a counter centers around selecting the combinational control logic that interprets the present state and properly enables or disables the control inputs to each flip-flop so that, upon receipt the next clock pulse the counter will tranfer to the proper next state.

The desired counting sequence can be specified by table, by diagram, or by an extension of the Karnaugh map, commonly called transition map:

A.3.4.1 Transition Mapping:

In combinational fuctions Karnaugh map with entries TRUE,FALSE and DON'T CARE was a sufficient tool to define the behavior of the circuit. But in sequential systems employing flip-flops four posible behavioral actions can occur and the behavior of a flip-flop from one clock period to the next can be expressed as follow:

1. If the flip-flop is initially in the 0 state (RESET) and remains in the 0 state after being clocked, its behavior is static and can be represented by the symbol 0 indicating that the flip-flop is RESET in both the present and the next state.

2. If the flip-flop is in the 0 state and change to a 1 state (SET), the behavior is dynamic and canbe represented by the symbol α.

3. If the flip-flop is in the 1 state and changes to the 0 state, the symbol β canbe used, This, of course, is dynamic behavior.

4. If the flip-flop is in the 1 state initially and remains there in following the clock signal, the symbol 1 can be used.

5. The symbol X can be used to represent a present state with an unspecified next state.

A.3.4.2 Excitation Tables

For a particular type of flip-flop the five behavioral action (0,1,α,β,x) will require a certain set of control inputs. Static behavior may require that all control inputs be disabled, while dynamic behavior may require that all inputs be enabled. The following table (tables C.1 to C.4) will describe all the behavior for the four types of flip-flops mentioned carlier.

A.3.4.3 Synchronous Excitation Maping

The basic philosophy of this method is to consider the transition map as containing three types of entries:

1. Must Include those transition symbols which would plot as a 1 on an excitation map and hence must be included in the groupings which give the logical control equations.

2. Must avoid those transition symbols which

would plot as a 0 on an excitation map and hence must be avoided in the combinational groupings to find the control logic

3. Redundancies: all other entries on the transition map.

Table C.5 summarizes the necessary information for the four common flip-flop types.

As an example of applying the transition map, consider an 8421 BCD up-counter of which the state table and behavior listing are given in Fig. A.26.

The number of state variables are 10, hence the number of flip-flops required to implement the above variables must be 4. Since only 10 different variables are used, the remainder , 6 variables may be considered as don't care conditions in the transition maps. The transition maps for the four flip-flop are then shown in Fig. A.27. Consulting Table C.5 the control equations for each type of flip-flop are as follow:

S-R Flip-flops:

$S_A = BCD.C_L$, $S_B = \bar{B}CD.C_L$, $S_C = \bar{A}\bar{C}D.C_L$, $S_D = \bar{D}.C_L$

$R_A = AD.C_L$, $R_B = BCD.C_L$, $R_C = CD.C_L$, $R_D = D.C_L$

J - K Flip-flops:

$J_A = BCD.C_L$, $J_B = CD.C_L$, $J_C = \bar{A}D.C_L$, $J_D = 1.C_L$

$K_A = D.C_L$, $K_B = CD.C_L$, $K_C = D.C_L$, $K_D = 1.C_L$

T Flip-flops:

$T_A = (A\bar{D} - BCD).C_L$, $T_B = CD.C_L$, $T_C = \bar{A}D.C_L$, $T_D = 1.C_L$

D Flip-flops:

$D_A = (A\bar{D} + BCD).C_L$, $D_B = (\bar{B}CD + B\bar{C} + B\bar{D}).C_L$

Decimal Value	Present State				Next State				Behavior			
	A	B	C	D	A+	B+	C+	D+	A	B	C	D
0	0	0	0	0	0	0	0	1	0	0	0	α
1	0	0	0	1	0	0	1	0	0	0	α	β
2	0	0	1	0	0	0	1	1	0	0	1	α
3	0	0	1	1	0	1	0	0	0	α	β	β
4	0	1	0	0	0	1	0	1	0	1	0	α
5	0	1	0	1	0	1	1	0	0	1	α	β
6	0	1	1	0	0	1	1	1	0	1	1	α
7	0	1	1	1	1	0	0	0	α	β	β	β
8	1	0	0	0	1	0	0	1	1	0	0	α
9	1	0	0	1	0	0	0	0	β	0	0	β

FIG.A.26 State Table and Behavior of an 8-4-2-1 BCD Up-Counter.

(a)

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	α	0
11	X	X	X	X
10	1	β	X	X

(b)

AB \ CD	00	01	11	10
00	0	0	α	0
01	1	1	β	1
11	X	X	X	X
10	0	0	X	X

(c)

AB \ CD	00	01	11	10
00	0	α	β	1
01	0	α	β	1
11	X	X	X	X
10	0	0	X	X

(d)

AB \ CD	00	01	11	10
00	α	β	β	α
01	α	β	β	α
11	X	X	X	X
10	α	β	X	X

FIG.A.27 a) Transition Map of Flip-flop A b) Transition Map of Flip-flo
c) Transition Map of Flip-flop C d) Transition Map of Flip-flo

$D_C = (\bar{A}CD + \bar{C}\bar{D}).\ C_L\ ,\ D_D = \bar{D}.C_L$

A.3.4.4 Asynchronous Transition Mapping

In some counting sequences, particularly those that are closely related to the binary counting sequence, a simplification for the combinational control logic required to implement the counter can be obtained by using transitions from the outputs of one flip-flop, rather than the system clock, as a secondary pulse inputs that initiate changes in one or more of the other flip-flop that make up the counter. At least one of the flip-flops must be activated by the system clock. Thereafter, the state changes of that flip-flop can be used as clocking signals for other flip-flops. The asynchronous design procedure can be summarized as follows:

1. Make a complete synchronous design for the counter.

2. Select the flip-flop that has the simplest synchronous design as the beginning.

3. Using the system clock and the transitions from any flip-flops that are operational, select the control logic for each flip-flop. Use the enabling inputs to strip off transitions occuring in the wrong places.

4. Proceed from one flip-flop to the next; never retrace steps.

Consider the three bit binary up counter shown in Fig. A.28. The transition mapping of Fig. A.28 shown in Fig.A.29 yields the following synchronous J-K flip-flops control equations

Present State			Next State			Behavior		
A	B	C	A+	B+	C+	A	B	C
0	0	0	0	0	1	0	0	α
0	0	1	0	1	0	0	α	β
0	1	0	0	1	1	0	1	α
0	1	1	1	0	0	α	β	β
1	0	0	1	0	1	1	0	α
1	0	1	1	1	0	1	α	β
1	1	0	1	1	1	1	1	α
1	1	1	0	0	0	β	β	β

FIG.A.28 State Table and Behavior of A Binary Up-Counter.

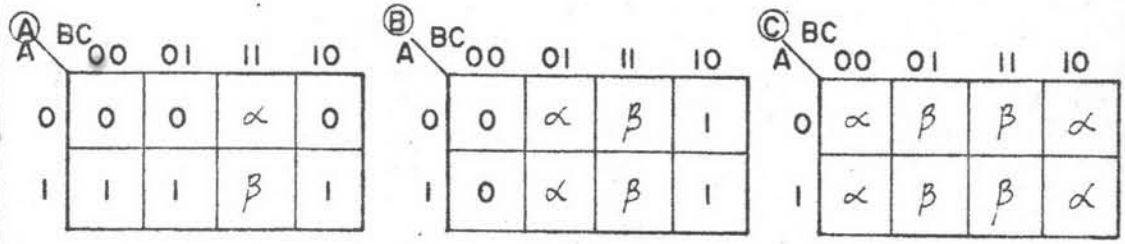


FIG.A.29 Transition Maps for The Three-bit Binary Up-counter.

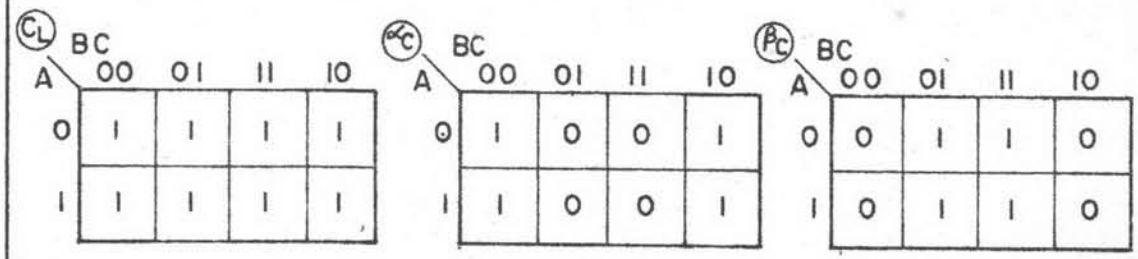


FIG.A.30 The Excitation Maps for C_L, α_C, and βC.

as follow

$$J_A = K_A = BC.C_L$$

$$J_B = K_B = C.C_L$$

$$J_C = K_C = 1.C_L$$

Flip -flop C should be selected as a starting point for the asynchronous design process since it exhibits the simplest logic. Once flip-flop C is selected, two clocking signals (α_C and β_C) are available in addition to the system clock. Hence three clocking signals are given in Fig 7.30

The β_C entries completely cover the α and β entries on the B map while α_C transition do not, hence the asynchronous control inputs for B are.

$$J_B = 1.\beta_B \quad \text{and} \quad K_B = 1.\beta_B$$

Once B flip-flop is operationed, five clocking signals (C_L, α_C, β_C, α_B and β_B) are available. Intermap comparison as previously shown shows that β_B covers the must-include entries for both J and K on the A map giving:

$$J_A = 1.\beta_B$$

$$K_A = 1.\beta_B$$

Since the asynchronous control logic for the three bit binary-counter is less costly than the fully synchronous control logic, asynchronous counting may be chosen.

APPENDIC B

B.1 NON-INVERTING COMPARATOR WITH HYSTERESIS [5]

The circuit shown in Fig. B.1 is a non-inverting comparator with hysteresis which is obtained with only two resistors, R_1 and R_2. The hysteresis is achieved by shifting V_A (the trip voltage) at the positive input as V_0 is also low ($V_0 \approx$ GND). For the output to switch, V_{in} must rise up to $V_{in\ 1}$ where $V_{in\ 1}$ is given by

$$V_{in\ 1} = V_{ref} \frac{(R_1 + R_2)}{R_2}$$

As soon as V_0 switches to $V_{0\ High}$ V_A will drop to a value greater than V_{ref} which is given by:

$$V_A = V_{in\ 1} + \frac{(V_{0\ High} - V_{in\ 1})\ R_1}{R_1 + R_2}$$

To make the comparator switch back to its low state ($V_0 \approx$ GND), V_{in} must go below V_{ref} before V_A will again equal V_{ref}. This lower trip point is now given by:

$$V_{in\ 2} = \frac{V_{ref}\ (R_1 + R_2) - V_{0\ High}\ R_1}{R_2}$$

The hysteresis for this circuit, V_{in} is the difference between $V_{in\ 1}$ and $V_{in\ 2}$ and is given by:

$$\Delta V_{in} = V_{in\ 1} - V_{in\ 2} = V_{0\ High} \frac{R_1}{R_2}$$

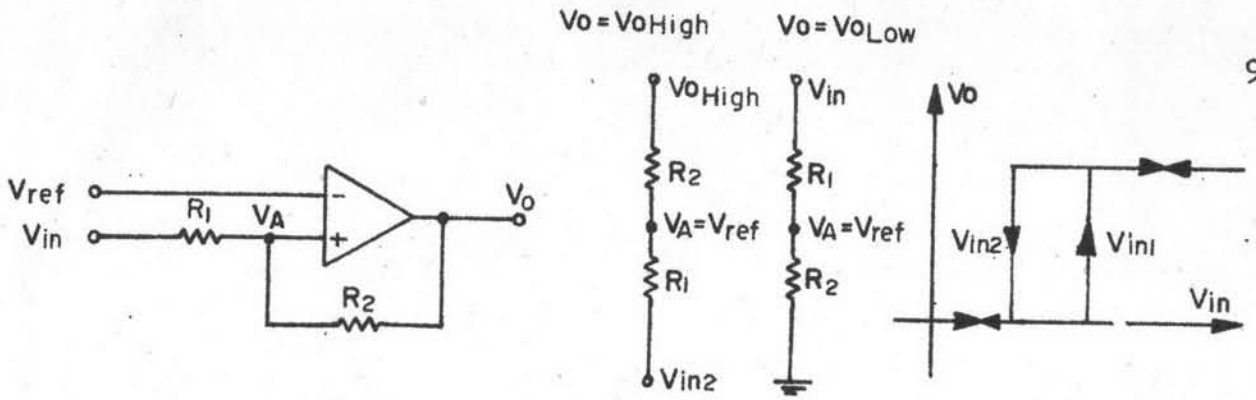


FIG. B.1 Non-inverting comparator with hysteresis.

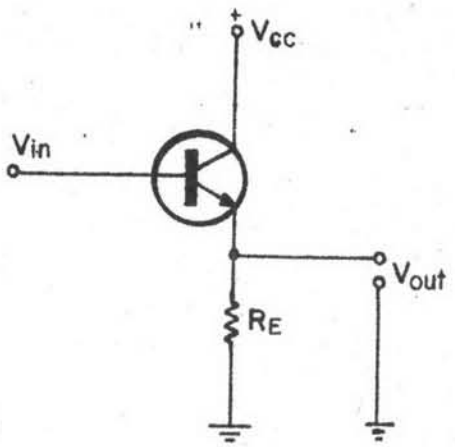


FIG. B.2 Emitter follower configuration.

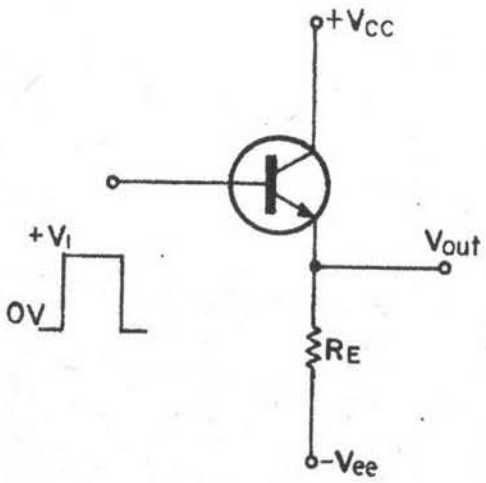


FIG. B.3 Connection of R_E to a negative supply voltage improves turn-on response of emitter-follower stage.

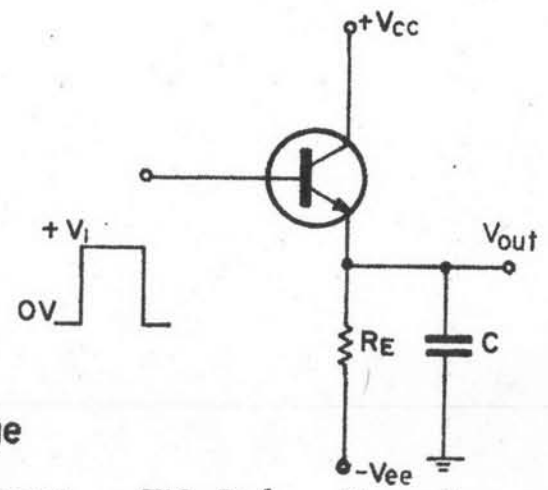


FIG. B.4 Negative supply voltage reduces turnoff time for a capacitive load.

B.2 EMITTER - FOLLOWER:

B.2.1 Voltage and current gain

The type of circuit that a transistor be connected in the common-collector configuration, with input and output signals appearing at base and emitter terminals is often referred to as an "emitter - follower circuit" as shown in Fig. B.2

When the potential difference across base and emitter terminals is slightly above the value of $V_{BE\ (on)}$ base current flows into the transistor. Thus, the output potential is slightly less positive than the input - voltage level and follows closely the voltage excursion of the positive - input signal. Voltage gain of the circuit is given by:

$$A_V = \frac{V_{IN} - V_{BE\ (ON)}}{V_{IN}}$$

WHERE A_V = voltage gain of the emitter - follower circuit

V_{IN} = dc level of input signal.

For $V_{IN} < V_{cc}$; the transistor is in active region, the collector current is then given by: $Ic = h_{FE}\ I_B$

Emitter current is expressed by $I_E = I_B\ (h_{FE} + 1)$

Hence current gain (A_I) is given by $A_I = \frac{I_E}{I_B} = h_{FE} + 1$

B.2.2 Input Impedance

The input impedance of the emitter-follower circuit is affected by current gain of the transistor and by magnitude of the emitter load impedance. The dc input impedance is defined by: $R_{IN} = \frac{V_{IN}}{I_B}$; WHERE $V_{IN} = I_E\ R_E + V_{BE(ON)}$

$$I_B = \frac{I_E}{h_{FE} + 1}$$

Hence

$$R_{IN} = R_E (h_{FE} + 1) + \frac{V_{BE} (ON) (h_{FE} + 1)}{I_E}$$

$$\approx R_E (h_{FE} + 1)$$

For small - signal or ac the input impedance can be expressed by

$$Z_i = \frac{v_i}{i_b} = r_b' + (r_e + Z_E) (h_{fe} + 1)$$

$$\approx Z_E (h_{fe} + 1)$$

where Z_i = ac input impedance

v_i = incremental change in input voltage

i_b = incremental change in base current

Z_E = external impedance at the emitter terminal

B.2.3 Emitter Biasing

Emitter-follower circuits are often biased as shown in Fig. B.3. The emitter-supply voltage is sufficiently negative to forward-bias the base-emitter junction for input signals more positive than $- |V_{ee}| + V_{BE} (ON)$. The magnitude of $- V_{ee}$ is chosen such that the transistor does not cut off for even the most negative level of input signal. The transistor now operates in a region of relatively high f_T and transient response of the circuit is improved.

Consider an emitter follower driving capacitive load as shown in Fig B.4 when V_{IN} approaches V_1 the output voltage is given by

$$V_{out} = V_1 - V_{BE(ON)} + V_{ee}$$

At the instant V_{in} makes an abrupt change to zero, the transistor is OFF and V_{out} is shown to follow

$$V_{out} = - V_{ee} + \left[V_1 - V_{BE(ON)} + V_{ee}\right] e^{-t/R_E C}$$

The time required for V_{out} to drop to $V_{BE(ON)}$ is found to be.

$$T = R_E C \ln \left[\frac{V_1 - V_{BE(ON)} + V_{ee}}{V_{ee} - V_{BE(ON)}} \right]$$

The discharge time can be decreased by increasing the emitter-bias voltage and hence the emitter-bias voltage reduces discharge time due to capacitive loading at the output.

B.3 TRANSISTOR-TRANSISTOR LOGIC SERIES 74

Aboolute maximum ratings over operating free-air temperature range:

Supply Voltage V_{cc} (note 1)	7 V.
Input Voltage V_{IN} (note 1)	5.5 V.
Interemitter Voltage(note 2)	5.5 V.
Resistor Node Voltage, SN 74121 (Note 1)	-5.5V to 7V.
Operating Free-Air Temperature Range:	
Series 74 Circuits	0 C to 70 C
Storage Temperature Range	-65 C to 150C

Notes: 1. Voltage values, except inter-emitter voltage are with respect to network ground terminal.

2. This is the voltage between two emitters of a multiple emitter resistor.

Logic definition:

Series 74 logic is defined in terms of standard Positive LOGIC using the following definitions

HIGH VOLTAGE = LOGICAL 1

LOW VOLTAGE = LOGICAL 0

Input clamping diodes:

Although not shown on all schematic diagrams,all of these SSI circuits incorporate input diodes. Each clamping diode is capable of limiting negative excursion at the input to a maximum of 1.5 volts below ground, even if-12mA of curent is drawn.

Unused inputs of Nand/And gates:

For optimum switching times and minimum noise susceptibility, unused inputs should be maintained at a positive voltage greater than 2.4 V. but not exceed the absolute maximum rating of 5.5V. This eliminates the distributed capacitance associated with the floating input-transistor emitter, bondwire , and package lead, and ensures that no degradation will occur in the propagation times. Some possible ways of handling input emitters are:

1. Connect unused inputs to an independent supply voltage. Preferably, this voltage should be between 2.4V and 3.5V.

2. Connect unused inputs to a used input if maximum fan-out of the driving output will not be exceeded. Each input presents full load in the logical 1 state to the driving output.

3. Connect unused input to Vcc through a 1 K resistor so that if a transient which exceeds the 5.5V maximum rating

should occur, the impedance will be high enough to protect the input. One to 25 unused inputs may be connected to each 1 KΩ resistor.

Input-current requirements

Input-current requirements reflect worst-case conditions on the specified recommended operating free-air temperature and V_{cc} ranges. Each input, of the multiple emitter input transistors which have a 4 KΩ base resistor, requires that no more than -1.6 mA flow out of the input at a logical 0 voltage level, therefore, one load (N = 1) is -1.6 mA maximum. Each input requires current into the input at a logical 1 voltage level. This current is 40 μA maximum for each emitter of input transistors with the 4-K base resistor. Currents into the input terminals are specified as positive values. Arrows on the d-c test circuits indicate the actual direction of current flow.

Fan-out capability

Fan-out reflects the ability of an output to sink current from a number of loads (N) at a logical 0 voltage level and to supply current at a logical 1 voltage level. Each standard output is capable of sinking current or supplying current to 10 loads (N = 10). The buffer gate is capable of sinking current or supplying current to 30 loads (N = 30). Load currents (out of the output terminal) are specified as negative values.

B.4 COMBINED FAN-OUT AND WIRE-AND CAPABILITIES

The open-collector TTL gate, when supply with a proper

load resistor (R_L), can be paralleled with other similar TTL gates to perform the wire-AND function, and simultaneously, will drive from one to nine series, TTL load. When no other open-collector gates are paralleled, this gate can be used to drive ten TTL loads. For any of these conditions an appropriate load resistor value must be determined for the desired circuit configuration. A maximum resistor value must be determined which will ensure that sufficient load current (to TTL loads) and off current (through paralled outputs) will be available during a logical 1 level at output. A maximum resistor value must be determined which will ensure that current through this resistor and sink current from the TTL loads will not cause the output voltage to rise above the logical 0 level even if one of the paralleled outputs is sinking all the current

In both conditions (logical 0 and logical 1) the value of R_L is the current in amperes.

Logical 1 (Off level) circuit calculations (Fig. B5)

The allowable voltage drop across the load resistor (V_{RL}) is the difference between Vcc applied and the V_{out} (1) level required at the load:

$$V_{RL} = V_{cc} - V_{out}\ (1)\ \text{required}$$

The total current through the load resistor (I_{RL} is the sum of the load currents ($I_{in}(1)$) and the off-level reverse currents ($I_{out}(1)$) through each of the wire-AND connected outputs:

$$I_{RL} = \eta\ I_{out}(1) + N.\ I_{in}(1)\ \text{to TTL loads.}$$

Therefore, calculations for the maximum value of R_L would be:

$$R_{L(Max)} = \frac{V_{cc} - V_{out(1)\ required}}{\eta I_{out(1)} + N.I_{in(1)}}$$

where, η = number of gates wire-AND connected, and

N = number of TTL loads.

As an example consider Fig. 7.35 where:

$$\eta = 4 \ , \ N = 3$$

$$R_{L\ (Max)} = \frac{V_{cc} - V_{out(1)required}}{\eta\ I_{out(1)} + N.I_{in(1)}}$$

$$R_{L(Max)} = \frac{5 - 2.4}{.001 + 0.00012}$$

$$= 2321 \ \Omega$$

Logical 0 (on level) circuit calculations (Fig. B6)

The current through the resistor must be limited to the maximum sink-current capability of one output transistor. Note that if several output transistors are wire-AND connected, the current through R_L may be shared by those paralleled transistors, However, unless it can be absolutely guaranteed that more than one transistor will be ON during logical 0 periods, the current must be limited to 16 mA, the maximum current which will ensure a logical 0 maximum of 0.4 volts.

Also, fan-out must be considered. Part of the 16 mA will be supplied from the inputs which are being driven. This reduces the amount of current allowed through R_L. Therefore, the equation used to determine the minimum value of R_L is:

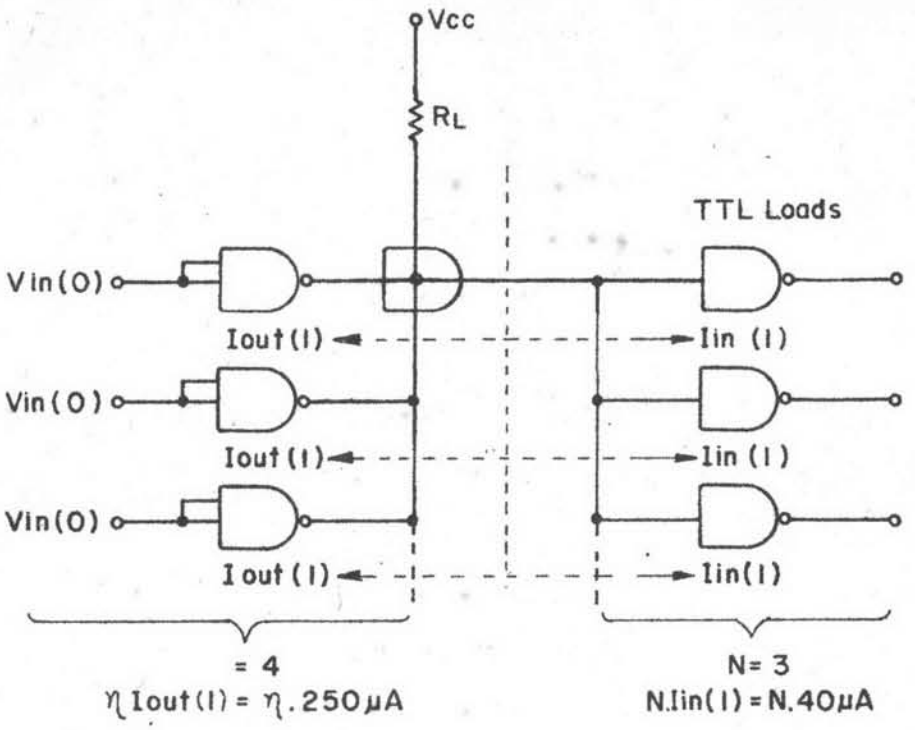


FIG. B.5 Logical (1) circuit condition.

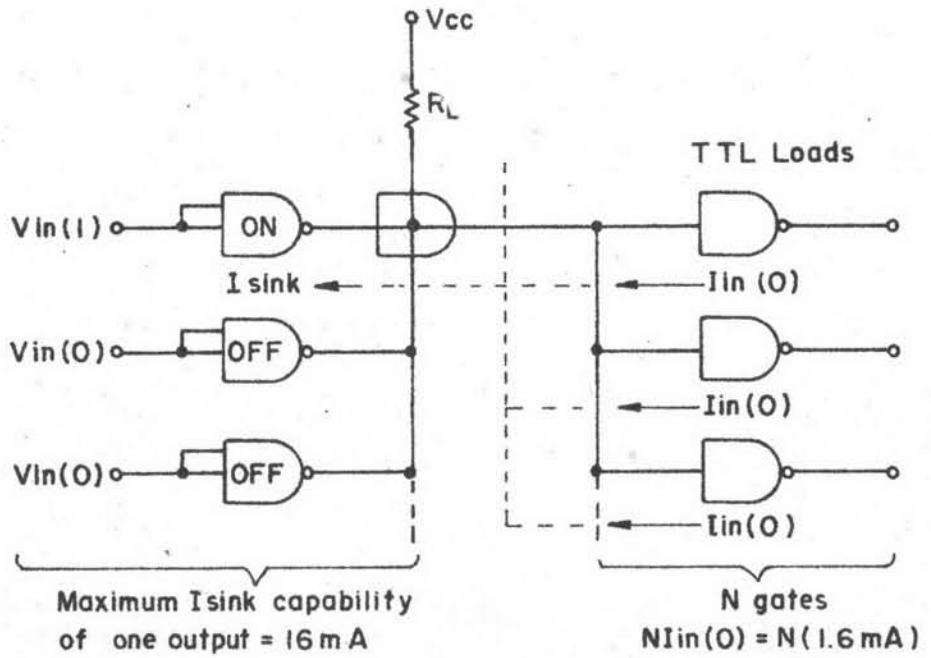


FIG. B.6 Logical (0) circuit condition.

$$R_{L(Min)} = \frac{V_{cc} - V_{out(0)\ required}}{I\ sink\ capability - I\ sink\ from\ TTL\ loads}$$

As an example consider Fig. 7.36 where:

$$R_{L(Min)} = \frac{V_{cc} - V_{out(0)\ required}}{I\ sink\ capability - I\ sink\ from\ TTL\ loads}$$

for $N = 3$

$$R_{L(Min)} = \frac{5 - 0.4}{0.016 - 0.0048} = 410 \quad \Omega$$

Driving TTL loads and combining outputs:

Table 6 provides minimum and maximum resistor values, calculated from equations shown above, for driving one to ten TTL loads and wire-AND connecting two to seven parallel outputs. Each value shown for one wire-AND output is determined by the fan-out plus the cut off current of a single output transistor. Extension beyond seven wire-AND connections is permitted with fan-outs of seven or fewer if a valid minimum and maximum R_L is possible. When fanning-out to ten TTL loads, the calculation for the minimum value of R_L indicates that an infinite resistance should be used ($V_{RL} \div 0 = \infty$); however, the use of a 4 KΩ resistor in this case will satisfy the logical 1 condition and limit the logical 0 level to less than 0.43 V.

TableC1 The Excitation Table for The S-R Flip-flop.

Behavior	S Input	R Input
0	0	X
1	X	0
α	1	0
β	0	1
X	X	X

TableC2 The Excitation Table for The J-K Flip-flop.

Behavior	S Input	K Input
0	0	X
1	X	0
α	1	X
β	X	1
X	X	X

TableC3 The Excitation Table for The D Flip-flop.

Behavior	D Input
0	0
1	1
α	1
β	0
X	X

TableC4 The Excitation Table for The T Flip-flop.

Behavior	T input
0	0
1	0
α	1
β	1
X	X

TableC5 Must-Include / Must Avoid Condition for The Variaus Flip-flop.

Flip-flop Control Input	Must Include	Must Avoid	Redundancies
S	α	0, β	1, X
R	β	1, α	0, X
J	α	0	1, β, X
K	β	1	0, α, X
D	1, α	0, β	X
T	α, β	0, 1	X

Table C6 Min. and Max. Resistance Value for Open Collector Wire-AND Logic.

FAN-OUT TO TTL LOADS	Wire - AND Output							
	1	2	3	4	5	6	7	1 to 7
1	8965	4814	3291	2500	2015	1688	1452	319
2	7878	4482	3132	2407	1954	1645	1420	359
3	7027	4193	2988	2321	1897	1604	1390	410
4	6341	3939	2857	2241	1843	1566	1361	479
5	5777	3714	2736	2166	1793	1529	1333	575
6	5306	3513	2626	2096	1744	1494	1306	718
7	4905	3333	2524	2031	1699	1460	1280	958
8	4561	3170	2429	1969	1656	X	X	1437
9	4262	3223	X	X	X	X	X	2875
10	4000	X	X	X	X	X	X	4000
	MAXIMUM							MIN
	Load resister value in ohms							

Note X - not recommended or not possible.