

**NRL Report 6230**

# **Procedure for the Design and Analysis of Arbitrary Length Digital Counters**

**R. J. ORSINO AND H. G. TALMADGE, JR.**

*Data Processing Branch  
Applications Research Division*

May 6, 1965



**U.S. NAVAL RESEARCH LABORATORY  
Washington, D.C.**



## CONTENTS

Abstract	ii
Problem Status	ii
Authorization	ii
INTRODUCTION	1
APPLICATION OF THE RIPPLE COUNTER	1
Count Rate Versus Counter Size	2
Setting Counter to Desired State	4
DESIGN OF A MORE VERSATILE COUNTER	7
Preparation of a Truth Table	7
P Terms	8
Use of Veitch Diagram	9
EXAMPLE: UP-DOWN COUNTER DESIGN	11
Gate Functions for Up Counter	11
Use "Don't Care" Terms to Simplify	13
Gate Functions for Down Counter	14
Composite Counter Formed with U Gate	15
CIRCUIT ANALYSIS	18
Write Trigger Functions	18
Veitch Diagrams Used to Rewrite Functions as P Terms	19
Truth Table Leads to Count Sequence and "Don't Care" Terms	20
CONCLUSIONS	21
REFERENCES	21

## ABSTRACT

A step-by-step procedure is described for designing digital counters having complex, arbitrary sequences. A divide-by-ten up-down counter is then designed to illustrate use of truth tables, "P" terms, Veitch diagrams, and "don't care" terms for simplifying the gate functions used to trigger each counter stage. The value of this procedure in circuit analysis as well as for circuit synthesis is then emphasized and illustrated by example. This procedure makes use of fundamental Boolean algebra and minimization techniques.

## PROBLEM STATUS

This is an interim report; work on the problem is continuing.

## AUTHORIZATION

NRL Problem Y01-01  
Project SF 019-01-03, Task 6168

Manuscript submitted November 25, 1964.

# PROCEDURE FOR THE DESIGN AND ANALYSIS OF ARBITRARY LENGTH DIGITAL COUNTERS

## INTRODUCTION

Many approaches, tricks, and shortcuts are used in the synthesis and analysis of digital counter circuits. The following discourse treats of one method which has been used to advantage in the course of current digital design assignments. Several types of counters, along with possible trouble areas, will be discussed and alternatives given where possible. A detailed design example is given for the synthesis of a forward-backward divide-by-10 counter. Also illustrated is the circuit analysis of a counter for which the count sequence is hypothetically unknown.

While the material contained in this note can be found in many sources on the subject of Boolean algebra and its application, it is believed that the step-by-step design and analysis procedures as presented here form a useful contribution.

## APPLICATION OF THE RIPPLE COUNTER

Digital systems make extensive use of the so-called modulo- $2^n$  counter, where  $n$  is the number of bits or stages of the counter. In general, such a counter can assume  $2^n$  discrete states, including the all-ZERO or reset state, and thus can represent a maximum number ( $2^n - 1$ ) before starting over with a zero count as additional pulses are applied. Thus, a modulo counter cannot distinguish between two numbers which differ by an integral multiple of its modulus,  $2^n$ . Depending upon the application, various counter circuit configurations can be used. For counting pulses of low repetition rates, a straightforward and simple binary "ripple" counter, such as the one shown in Fig. 1a, usually will suffice. The type of flip-flop (labeled FF) considered throughout this report is the well known "complement trigger"; a pulse applied to a complement input will always cause the flip-flop to change from the state it is in to the other state. In addition, each flip-flop has an ac set and an ac reset input. All inputs are assumed to require a positive transition to activate the flip-flop. A negative level, say -6 volts, is here assumed to represent a logical ONE and a more positive level, say, 0 volts, then represents a logical ZERO.

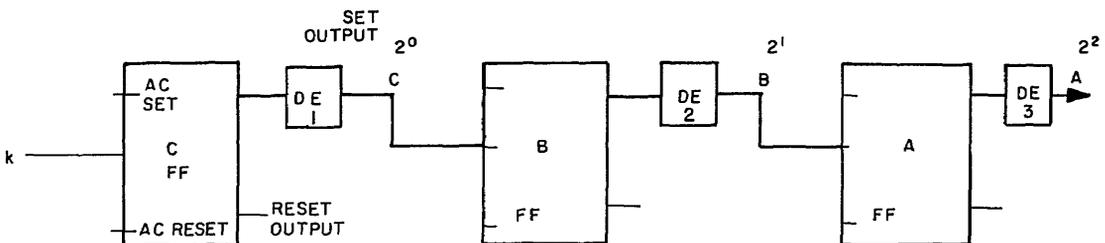


Fig. 1a - Ripple counter, modulo  $2^3$

The design of counters using flip-flops other than those of the type mentioned above will take a slightly different design approach than is to be undertaken in this report. In Fig. 1a, the intrinsic circuit delay of each flip-flop, referred to as propagation delay, is shown for clarity as being attributable to a separate block marked DE. It is evident that if the pulse rate is too rapid, some stages may not yet have reacted to a count pulse before a subsequent pulse has arrived at the first stage. As a result, during part of the pulse interval of the pulse train shown, the state of the counter will not represent the true count. In fact, for a given combination of clock period and delay, the counter states may not follow the desired sequence. This is illustrated by Fig. 1b, which depicts waveforms for a counter starting from an initial reset condition.

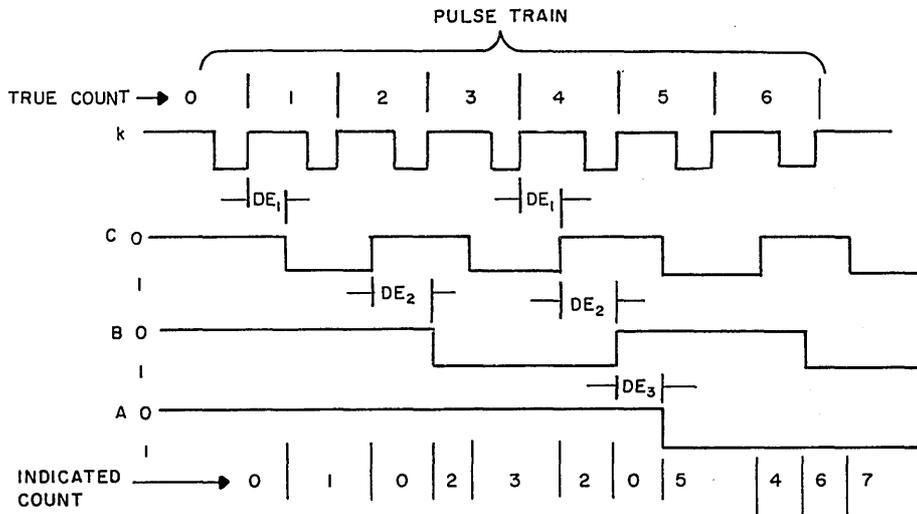
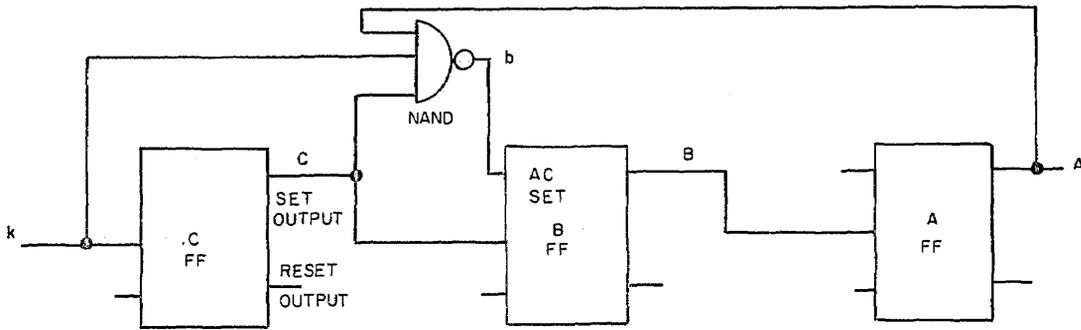


Fig. 1b - Waveforms describing the counter of Fig. 1a. For this combination of clock period and delay, the counter states vary in an undesirable sequence starting from an initial reset condition.

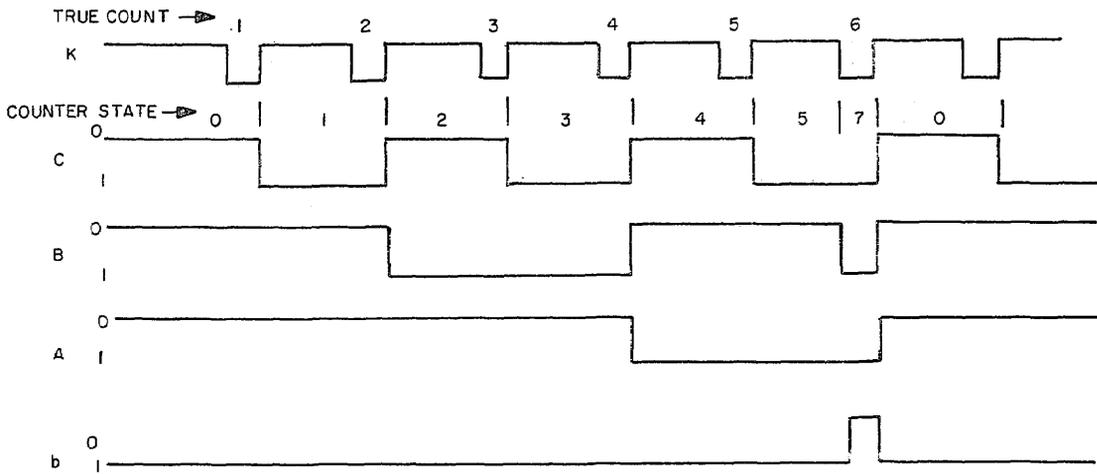
### Count Rate Versus Counter Size

The foregoing discussion makes it clear that when the state of a counter must be read between count pulses, the maximum useful count rate must decrease as the number of flip-flop stages is increased. In selecting a ripple type counter, then, the designer must take into account circuit delay and count rate.

It will be observed that when an  $n$ -stage "ripple" counter is used as a divide-by- $2^n$  counter, the output of each stage triggers the subsequent flip-flop and the pulses to be counted are applied only to the first stage. For a count-down requirement which is not an integral power of 2, the next highest power determines the number of flip-flops needed. In this case, however, some extra gating is required in order to terminate the counter prior to its normal modulus. One approach is to gate all flip-flops whose outputs are logical ONES at the desired last counter state, together with the clock pulse, and apply the output of this gate to those flip-flops whose outputs are logical ZERO at this last counter state. This action puts all stages of the counter into the ONE state for a portion of the clock or pulse period. The next pulse will then change the state of the counter to all ZEROS, resulting in the desired restart state. Figure 2 illustrates a circuit diagram for a divide-by-6 counter and the associated voltage waveforms. (For a further discussion on this



(a) Logic block diagram

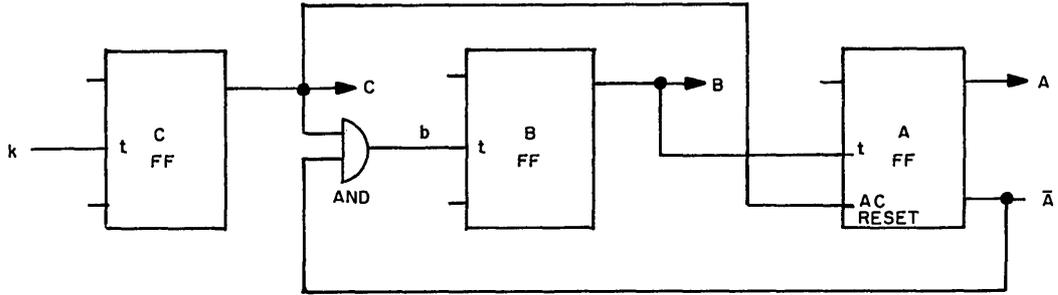


(b) Voltage waveforms.

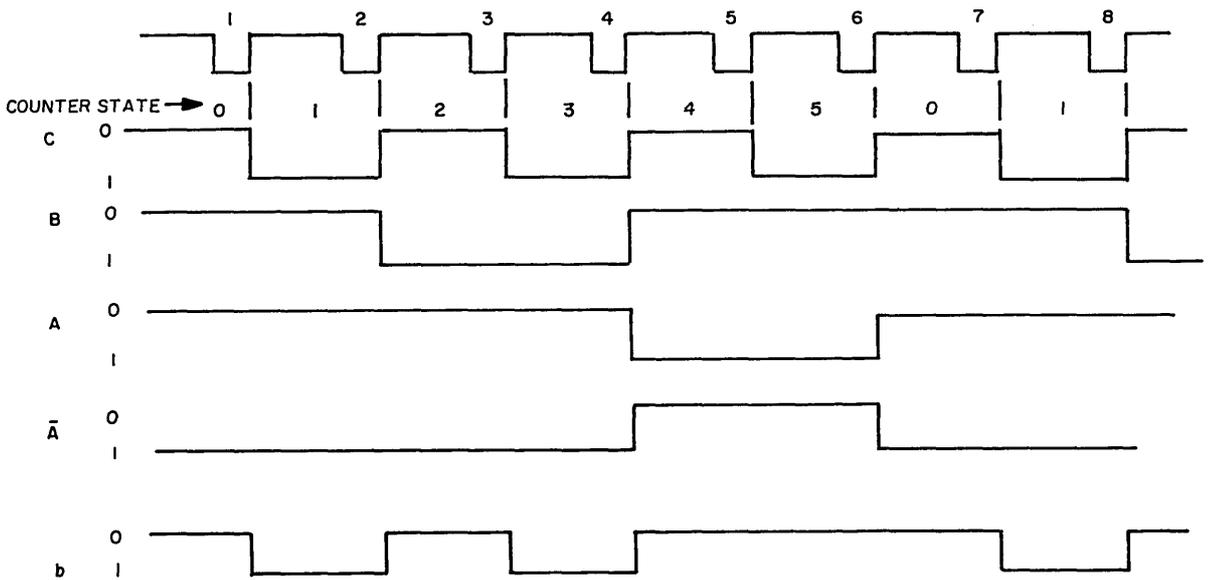
Fig. 2 - Divide-by-6 ripple counter in which all ONEs appear prior to the all ZERO restart state

approach, see Ref. 1.) If only one output pulse is desired for a certain number of input pulses, e.g., one pulse each time the counter passes from state 5 to 0, the foregoing circuit will suffice. However, in the event that all flip-flop output lines are used for parallel readout, the sequence of all ONEs (equivalent to a binary 7, as shown in Fig. 2b) prior to all ZEROS could lead to undesirable results. The designer must determine the effect of this in his application.

If, in fact, the effect of this is undesirable, alternative approaches can be taken. One approach would require the sensing of the flip-flop outputs only during intervals which do not include this invalid count, such as during the negation of the clock or count pulse shown in the figures. Another approach calls for gating appropriate outputs so as to block the triggering of certain flip-flops and permit the triggering of others out of the normal sequence. This technique is exemplified by another divide-by-6 circuit shown in Fig. 3. When the count reaches 5, flip-flops A and C are at the ONE level. The next count pulse complements C,



(a) Logic block diagram



(b) Voltage waveforms. The effect of stage delay is neglected.

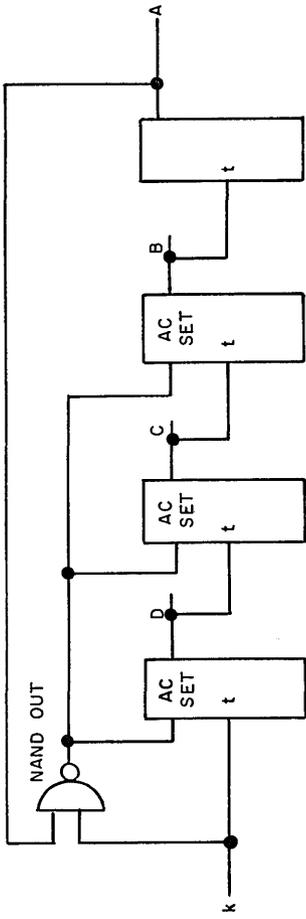
Fig. 3 - Divide-by-6 ripple counter in which the all ONE state is avoided in returning to the all ZERO state

but the resulting transition in C is unable to complement B since  $\bar{A}$ , being at the ZERO level, is AND gated with C to inhibit complementing of B at this time. The positive transition obtained at C is then used to reset flip-flop A, thus returning the counter to the all ZEROS state, ready for the next sequence of pulses.

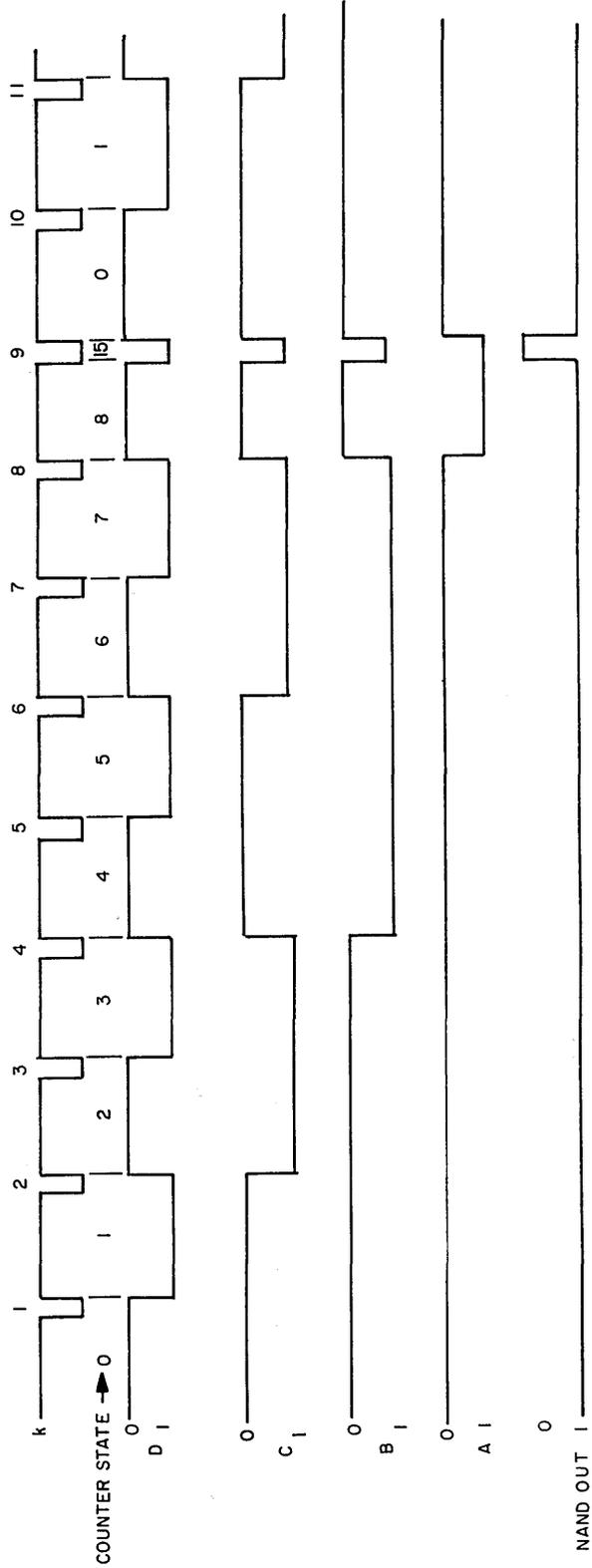
Other count-down sequences can be obtained through the design of counters using either of the methods outlined above. A divide-by-9 counter is shown along with associated waveforms in Figs. 4 and 5 for the two methods discussed.

#### Setting Counter to Desired State

Since ripple counters depend upon the interaction between flip-flop outputs and inputs, they generally cannot be arbitrarily set to any value. Suppose it is desired to set one of



(a) Logic block diagram



(b) Voltage waveforms

Fig. 4 - Divide-by-9 counter in which all ONEs appear in returning to the all ZERO restart state

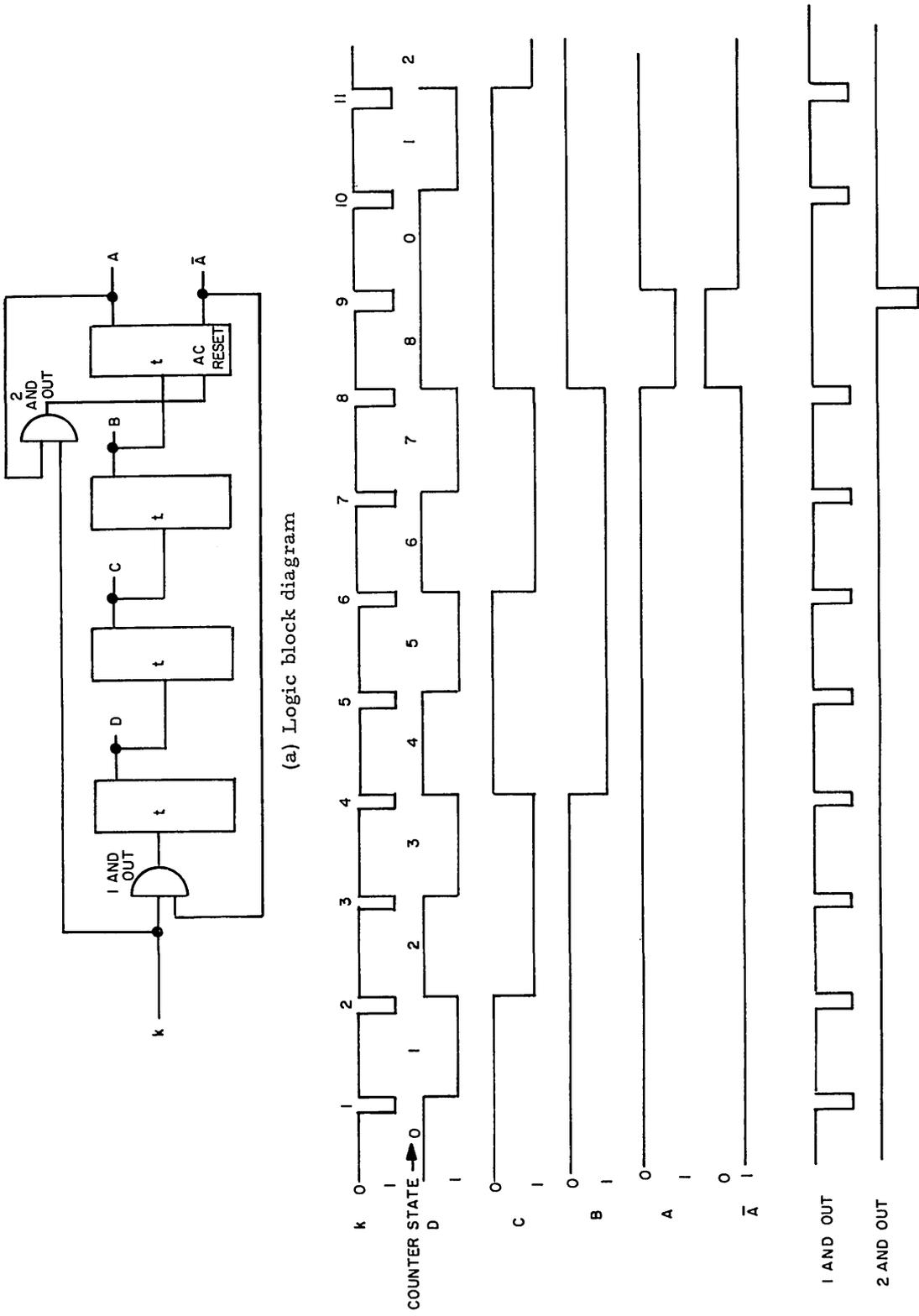


Fig. 5 - Divide-by-9 counter which excludes the all ONE state in returning to the all ZERO state

the flip-flops to a ZERO state; this action causes a positive transition at the output which will then cause the succeeding stage to be complemented, and it in turn may complement the next stage. Obviously, this "chain reaction" is not desirable. The difficulty is eliminated if all stages are simultaneously reset to ZERO prior to setting. Thus the setting of certain stages merely brings the SET outputs down to the ONE level, a transition which will not trigger the subsequent stage. However, in some cases the RESET output of a certain stage is fed back to other stages and, due to some time difference in the presetting action, a positive going pulse can be generated to trigger the wrong flip-flop. This difficulty can be eliminated by the use of flip-flops with DC SET and RESET inputs which respond to pulse levels rather than to a pulse transition.

## DESIGN OF A MORE VERSATILE COUNTER

A procedure will now be described which permits the design of counters for any arbitrary length or any arbitrary sequence, including the ability to count backward. Due to the inherent "instantaneous carry" type of triggering, circuit delays are not cumulative. Therefore, the pulse repetition rates are generally limited only by the bandwidth or transient response capabilities of individual circuit elements and are not a function of the number of counter stages utilized. The counter to be described can also be set to any arbitrary value, since the count pulses which are required to trigger each flip-flop are generally not present during a preset mode or, in fact, can be used to synchronize the presetting. In order to eliminate the "race" problem,\* trailing edge positive going triggering will be used. The previously described convention of negative logic will be retained.

### Preparation of a Truth Table

As a first step, the counting sequences desired are written in tabular form, called a truth table, as illustrated in Table 1. In this table, it is assumed that two count sequences are desired, a count down by 6 in both the forward and the backward directions. Thus, a three-stage counter and one condition bit are required. Corresponding states of stages A, B, and C are shown in their respective columns for each count value. On a given line, the binary number ABC representing the counter state is equivalent to the decimal number shown in the count sequence column. The Trigger Gate columns a, b, and c have binary ONES to indicate when these gates are to enable the passing of a count pulse through to the corresponding flip-flop. These gates are determined as follows: With the counter in any one particular state, we determine which stages must be changed in order to reach the next desired state. We then write binary ONES under the appropriate trigger gate column. For this example, it is evident from the c column that the C or least significant counter stage must be complemented (triggered) by each count pulse. In general, where condition bits are included in the design, the triggering of a given stage of any counter is a function of the state of each condition bit as well as the state of all significant counter bits.

Once the trigger gate functions have been found using the truth table, two approaches can be taken. One approach is a straightforward Boolean algebra manipulation in order to simplify the expressions prior to final circuit design. The other approach is the use of the Veitch diagrams. This latter approach is generally more desirable when the expressions are so long as to make the Boolean algebra manipulation too unwieldy.

---

\*When leading edge triggering is utilized, a flip-flop or other storage device can change state during the trigger pulse. For example, if the change of state is such that an associated AND gate is enabled during the pulse, it will be possible to pass the later part of the pulse through the gate and perhaps complement a flip-flop out of sequence. This action constitutes the "race" problem.

Table 1  
Truth Table for a Count-Down-By-6 Backward-Forward Counter

P Terms	Variables				Trigger Gate			Count Sequence	Pulse
	Counter State			Condition*	a	b	c		
	A	B	C						
$P_1$	0	0	0	1	0	0	1	0	1
$P_3$	0	0	1	1	0	1	1	1	2
$P_5$	0	1	0	1	0	0	1	2	3
$P_7$	0	1	1	1	1	1	1	3	4
$P_9$	1	0	0	1	0	0	1	4	5
$P_{11}$	1	0	1	1	1	0	1	5	6
$P_1$	0	0	0	1				0	7
-	-	-	-	-	-	-	-	⋮	⋮
$P_0$	0	0	0	0	1	0	1	0	$n + 1$
$P_{10}$	1	0	1	0	0	0	1	5	$n + 2$
$P_8$	1	0	0	0	1	1	1	4	$n + 3$
$P_6$	0	1	1	0	0	0	1	3	$n + 4$
$P_4$	0	1	0	0	0	1	1	2	$n + 5$
$P_2$	0	0	1	0	0	0	1	1	$n + 6$
$P_0$	0	0	0	0				0	$n + 7$

\*D = 1, forward; D = 0, backward.

### P Terms

In order to better utilize these approaches, a clear understanding of P terms and Veitch diagrams is required. While P terms have broad application for simplifying involved Boolean expressions, they are here discussed only with reference to digital counter design. (P terms are called min-terms in some sources.) Each P term corresponds to a definite binary state, and since it must include all the variables involved, it is considered a canonical term. In the example of Table 1, the variables are A, B, C, and D. Once the order of listing these variables is arbitrarily chosen, the subscript  $n$  of the symbol  $P_n$  is written as a decimal representation of the binary number represented by a given state of these variables, including any condition bit such as D of Table 1.

In writing the Boolean algebra representations of P terms, a ONE is represented by an uncomplemented variable, and a ZERO by a complemented variable. For example,  $P_5$  with three variables would simply be  $f(ABC) = A\bar{B}C$ ;  $P_5$  with four variables would be  $f(ABCD) = A\bar{B}CD$ , etc. Furthermore, a situation in which  $P_5$  is  $f(ABC) = A\bar{B}C$  can correspond to a count of 5 for a three-stage counter, or it can correspond to a count of 2 for a two-stage counter with variable C representing a condition bit. Similarly, for a three-stage counter at a count of 5 with a fourth variable D being a condition bit, the P term could be written as  $P_{11} = A\bar{B}CD$ . It is therefore to be noted that the subscript for

a P term does not always correspond to the value of the counter state. This is evident from Table 1, where the P term symbols in the first column are seen to have subscripts differing from the corresponding count value found in the count sequence column.

We can now form the expression for each trigger gate by writing those P terms which correspond to binary ONES under the trigger gate columns. Thus, the Boolean expression for the trigger gate of the B stage is

$$\begin{aligned} b(t) &= P_3 + P_7 + P_8 + P_4 \\ &= \bar{A}\bar{B}CD + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D}. \end{aligned}$$

This is rearranged as

$$b(t) = (B + \bar{B})\bar{A}CD + A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D}$$

and further simplified to become

$$b(t) = \bar{A}CD + A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D}.$$

This equivalent expression for  $b(t)$  is seen to be somewhat more simple to implement than the initial expression. As will be shown later, far more extensive simplifications are possible in some instances.

#### Use of Veitch Diagram

Use of the Veitch diagram will now be described. The reader is assumed to have some prior knowledge of this useful technique (see Refs. 2-4). (In this report Veitch diagrams are used, rather than more improved mapping diagrams, since they are basic to any other modifications. In practice, whichever map seems preferable may be used.) In Fig. 6 is shown possible Veitch diagrams for several sets of variables. It will be noted that for  $m$  variables involved, the Veitch diagram consists of  $2^m$  squares, so that each square can be used to represent a P term. Each variable which brackets a number of squares represents half the diagram and is lettered as shown; the other half is normally not marked and represents the complement of the variable. Thus in the three-variable diagram,  $P_0$ ,  $P_1$ ,  $P_3$ , and  $P_2$  correspond to the complement of A or  $\bar{A}$ .

Any Boolean function which consists of several P terms can be plotted on the appropriate Veitch diagram by marking a ONE in the square assigned to each P term in the function. To minimize circuitry which implements P terms, we begin by observing the Veitch diagram for ONES in groups of two, four, eight, etc., in a row or line or forming a block. The maximum number of ONES in a group provides the greatest amount of minimization. In encircling these groups, it is important to note that the rectangular Veitch diagram may first be rolled until opposite boundaries are joined to form either a vertical or a horizontal cylinder. Familiarization with the possible groupings for each diagram can usually be obtained only through practice and use.

A method which helps systemize the approach and also eliminate some redundancy is to first encircle all ONES which cannot be grouped in sets of two. Then encircle all sets of two ONES which cannot be grouped in sets of four, etc. This assures that no group will be completely covered by another group.

Upon encircling a maximum number of ONES in a group, we determine what minimum number of variables uniquely define that group. For example, a group consisting of ONES in squares  $P_{12}$ ,  $P_{13}$ ,  $P_8$ , and  $P_9$  in the four-variable Veitch diagram is uniquely defined by the simple noncanonical expression  $A\bar{C}$ . It is seen by inspection that these four squares

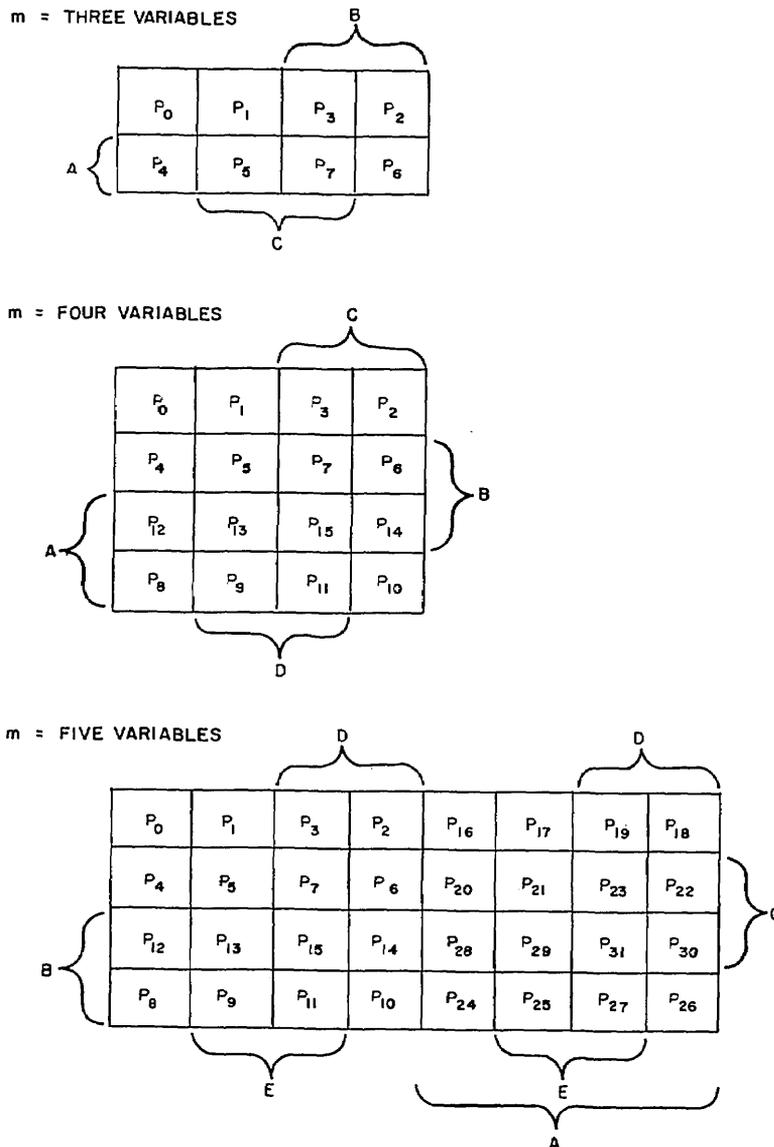


Fig. 6 - Veitch diagrams

are common to both the two A rows and the two  $\bar{C}$  columns. It is true that two of the squares are also common to one B row and two squares are found in one D column, but these facts are superfluous since the intersection of A and  $\bar{C}$  completely cover the block of ONEs.

By the Boolean algebra manipulation approach, one can write

$$\begin{aligned} f(ABCD) &= P_{12} + P_{13} + P_8 + P_9 \\ &= AB\bar{C}\bar{D} + AB\bar{C}D + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D. \end{aligned}$$

This can be successively rearranged and simplified using the theorems of Boolean algebra to secure the following:

$$\begin{aligned} f(ABCD) &= AB\bar{C}(D + \bar{D}) + A\bar{B}\bar{C}(D + \bar{D}) \\ &= (AB\bar{C} + A\bar{B}\bar{C})(D + \bar{D}) \\ &= A\bar{C}(B + \bar{B})(D + \bar{D}) \\ &= A\bar{C}. \end{aligned}$$

Thus, the two approaches are seen to lead to the same result, a dramatic simplification of the original Boolean expression.

It should be noted that groups can also be formed by ONES which do not at first appear to be in a group, such as ONES falling in the outer squares of the diagram. For example, a ONE in  $P_0$ ,  $P_2$ ,  $P_8$ , and  $P_{10}$  in the four-variable diagram actually forms one group of four. This can easily be shown to be true for this example. By inspection of the diagram, one can immediately write,

$$P_0 + P_2 + P_8 + P_{10} = \bar{B}\bar{D}.$$

By a more circuitous route, we begin with the Boolean expressions for each P term:

$$P_0 + P_2 + P_8 + P_{10} = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D}$$

and proceed to simplify this as follows:

$$\begin{aligned} P_0 + P_2 + P_8 + P_{10} &= (\bar{A}\bar{C} + \bar{A}C + A\bar{C} + AC)\bar{B}\bar{D} \\ &= [A(C + \bar{C}) + \bar{A}(C + \bar{C})]\bar{B}\bar{D} \\ &= (A + \bar{A})\bar{B}\bar{D} \\ &= \bar{B}\bar{D}. \end{aligned}$$

It is clear that the Veitch diagram has provided a quick simplification of the functions. The straightforward Boolean reduction techniques lead to the same results but with quite a bit more work, including some trial and error attempts. Since each variable represents an input to a logic element and each four-variable term is a logic block, the minimization of logic elements is obvious. There is a one-to-one correspondence between the number of distinct groups and the number of separate terms (and therefore circuit or logic blocks) in the minimized function.

#### EXAMPLE: UP-DOWN COUNTER DESIGN

Several examples will be given to illustrate the aforementioned approaches. Assume that a modulo-10 counter is desired; that is, the counter repeats its sequence every ten count pulses. Gating functions will be derived separately for the up and down sequences, and these will then be suitably combined to be implemented as one composite counter.

#### Gate Functions for Up Counter

A truth table for the forward sequence is prepared as in Table 1 and appears as in Table 2.

Table 2  
Truth Table for the Forward Sequence

P Terms	Variables (Counter State)				Trigger Gate				Count Sequence	Pulses
	A	B	C	D	a	b	c	d		
P <sub>0</sub>	0	0	0	0	0	0	0	1	0	1
P <sub>1</sub>	0	0	0	1	0	0	1	1	1	2
P <sub>2</sub>	0	0	1	0	0	0	0	1	2	3
P <sub>3</sub>	0	0	1	1	0	1	1	1	3	4
P <sub>4</sub>	0	1	0	0	0	0	0	1	4	5
P <sub>5</sub>	0	1	0	1	0	0	1	1	5	6
P <sub>6</sub>	0	1	1	0	0	0	0	1	6	7
P <sub>7</sub>	0	1	1	1	1	1	1	1	7	8
P <sub>8</sub>	1	0	0	0	0	0	0	1	8	9
P <sub>9</sub>	1	0	0	1	1	0	0	1	9	10
P <sub>0</sub>	0	0	0	0	-	-	-	-	0	-

From the Trigger Gate columns we can write the trigger equations:

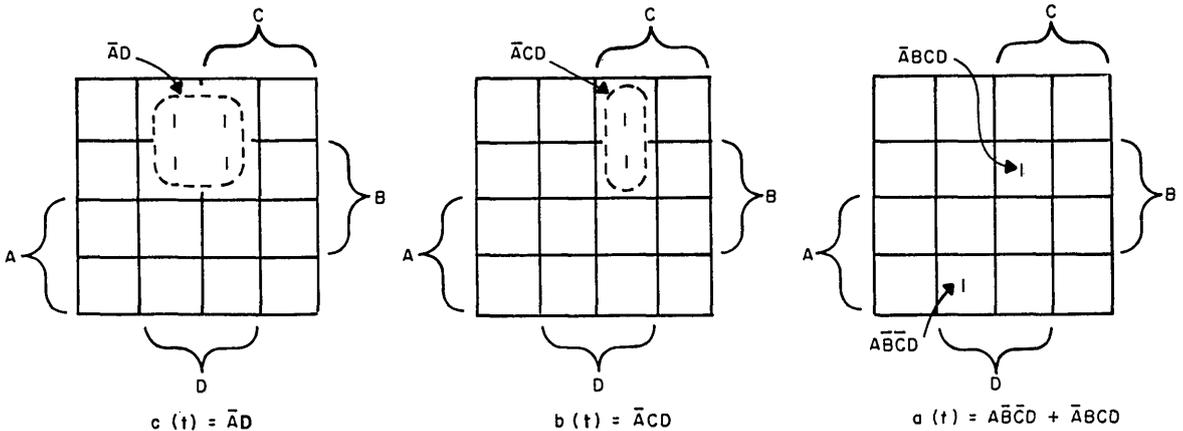
$$d(t) = 1$$

$$c(t) = P_1 + P_3 + P_5 + P_7 = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BCD$$

$$b(t) = P_3 + P_7 = \bar{A}\bar{B}CD + \bar{A}BCD$$

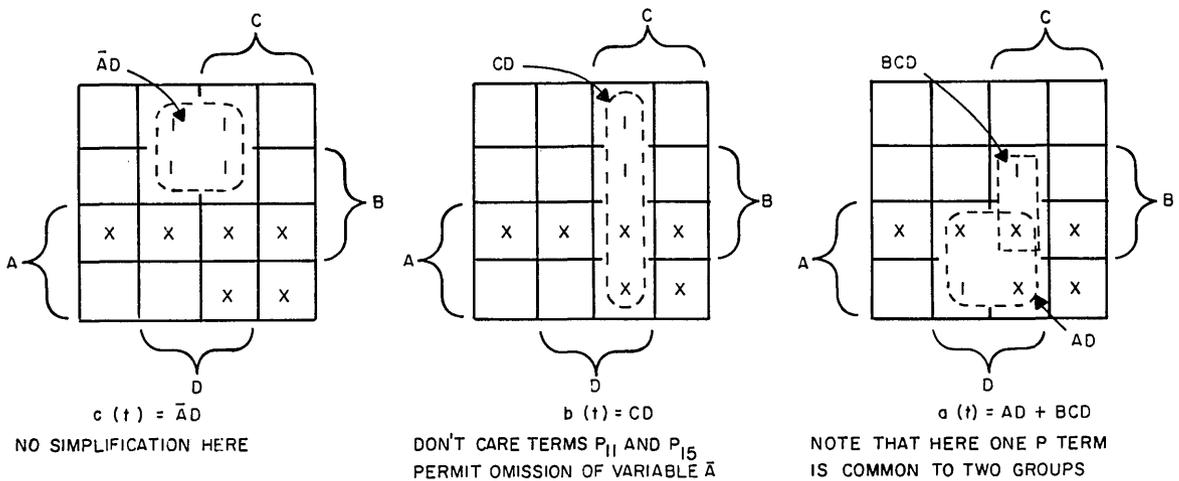
$$a(t) = P_7 + P_9 = \bar{A}BCD + A\bar{B}\bar{C}D.$$

Now insert ONEs in appropriate squares on the four-variable Veitch diagram and obtain groups where possible as shown below:



Use "Don't Care" Terms to Simplify

A further simplification can usually be accomplished through the use of the so-called "don't care" terms. These terms correspond to those states of the counter which do not occur in the desired sequence and hence do not affect the trigger gate logic. In the present example, the "don't care" terms are  $P_{10}$ ,  $P_{11}$ ,  $P_{12}$ ,  $P_{13}$ ,  $P_{14}$ , and  $P_{15}$ . These are the P terms which do not appear in any of the trigger gate expressions. Now, by inserting an X in those squares which correspond to the "don't care" terms and treating them as if they were ONES in our search for large groups, simplification is obtained as follows:



For clarification, the preceding steps leading to the final simplification are listed below:

$$\left. \begin{aligned}
 d(t) &= 1 \\
 c(t) &= \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BCD \\
 b(t) &= \bar{A}\bar{B}CD + \bar{A}BCD \\
 a(t) &= \bar{A}BCD + A\bar{B}\bar{C}D
 \end{aligned} \right\} \text{Unreduced P terms.}$$

$$\left. \begin{aligned}
 d(t) &= 1 \\
 c(t) &= \bar{A}D \\
 b(t) &= \bar{A}CD \\
 a(t) &= \bar{A}BCD + A\bar{B}\bar{C}D
 \end{aligned} \right\} \text{Reduced gate functions.}$$

$$\left. \begin{aligned}
 d(t) &= 1 \\
 c(t) &= \bar{A}D \\
 b(t) &= CD \\
 a(t) &= AD + BCD
 \end{aligned} \right\} \text{Functions reduced using "don't care" terms.}$$

The reader is cautioned that in using "don't care" terms, certain problems do exist. During turn-on it may be possible for the counter to fall into one of the "don't care" states, leading to the possibility of "hanging up" the counter. Therefore, if the counter cannot correct itself by going into its normal sequence after a few count pulses, a pre-setting to any normal sequence state must be provided.

#### Gate Functions for Down Counter

Let us suppose further that we desire our counter to be able to count backward. It is mentioned again that any arbitrary sequence can just as readily be handled. The truth table for the backward count (Table 3) is prepared as before.

Table 3  
Truth Table for the Backward Count

P Terms	Variables (Counter State)				Trigger Gate				Count Sequence	Pulses
	A	B	C	D	a	b	c	d		
P <sub>9</sub>	1	0	0	1	0	0	0	1	9	1
P <sub>8</sub>	1	0	0	0	1	1	1	1	8	2
P <sub>7</sub>	0	1	1	1	0	0	0	1	7	3
P <sub>6</sub>	0	1	1	0	0	0	1	1	6	4
P <sub>5</sub>	0	1	0	1	0	0	0	1	5	5
P <sub>4</sub>	0	1	0	0	0	1	1	1	4	6
P <sub>3</sub>	0	0	1	1	0	0	0	1	3	7
P <sub>2</sub>	0	0	1	0	0	0	1	1	2	8
P <sub>1</sub>	0	0	0	1	0	0	0	1	1	9
P <sub>0</sub>	0	0	0	0	1	0	0	1	0	10
P <sub>9</sub>	1	0	0	1	-	-	-	-	9	-

From the Trigger Gate columns we can write the trigger gate equations:

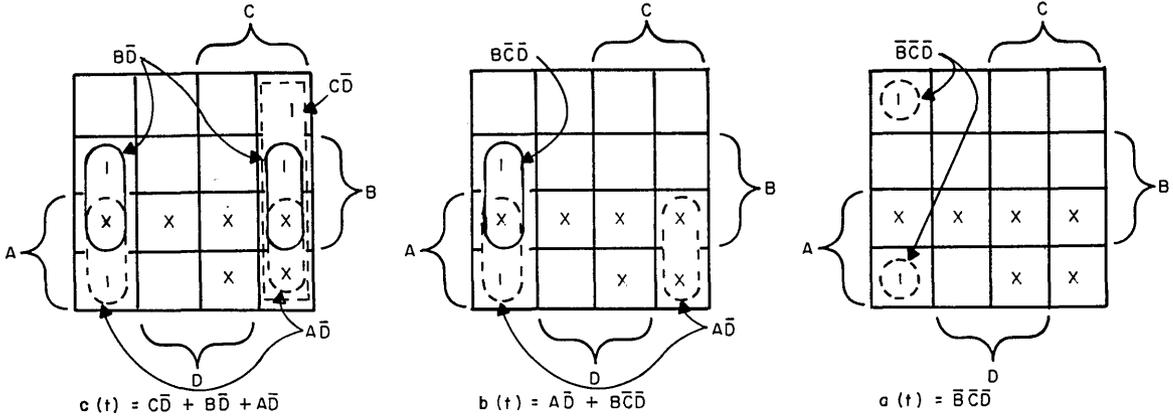
$$d(t) = 1$$

$$c(t) = P_8 + P_6 + P_4 + P_2 = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}D$$

$$b(t) = P_8 + P_4 = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D}$$

$$a(t) = P_8 + P_0 = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D}.$$

We insert ONES in the appropriate squares of the Veitch diagram as before and obtain groups where possible. The "don't care" terms are marked by an X in appropriate squares as previously:



Composite Counter Formed with U Gate

We must naturally distinguish between the two modes of counter operation through an additional gate, since we elected to treat the design as two separate problems. A simple push-pull gate will be applied to each equation as appropriate for the forward-backward modes. We will call this gate U for "up" or forward; hence  $\bar{U}$  will be for backward. The count pulse k is needed to provide the proper operation of the counter and also to provide instantaneous carry type of triggering, thus assuring that all stages change state at the same time. Combining the equations shown above with those from page 13, we have the following for a composite up-down counter:

$$\begin{aligned}
 d(t) &= k \\
 c(t) &= k\bar{A}DU + kB\bar{D}\bar{U} + kA\bar{D}\bar{U} + kC\bar{D}\bar{U} \\
 b(t) &= kCDU + kB\bar{C}\bar{D}\bar{U} + kA\bar{D}\bar{U} \\
 a(t) &= kADU + kB\bar{C}DU + k\bar{B}\bar{C}\bar{D}\bar{U}.
 \end{aligned}$$

These equations can now be implemented directly using AND-OR logic. However, if NAND logic is being used, it is necessary to apply a fundamental principle of Boolean algebra called De Morgan's theorem. This theorem is generally all that is required for this operation. De Morgan's theorem and several variations are given below:

$$\begin{aligned}
 A + B &= \overline{\bar{A}\bar{B}} \\
 \bar{A} + \bar{B} &= \overline{A\bar{B}} \\
 A + 0 &= \overline{\bar{A}}.
 \end{aligned}$$

The final equation is applicable when a signal must be inverted to provide the needed polarity. Applying De Morgan's theorem to our equations, we arrive at the following

set of equations which are now in NAND logic form and ready to be wired:

$$\begin{aligned}d(t) &= k \\c(t) &= \overline{\overline{(k \bar{A} D U)} \overline{\overline{(k B \bar{D} \bar{U})} \overline{\overline{(k C \bar{D} \bar{U})} \overline{\overline{(k A \bar{D} \bar{U})}}}}}} \\b(t) &= \overline{\overline{(k C D U)} \overline{\overline{(k B \bar{D} \bar{C} \bar{U})} \overline{\overline{(k A \bar{D} \bar{U})}}}} \\a(t) &= \overline{\overline{(k A D U)} \overline{\overline{(k B C D U)} \overline{\overline{(k \bar{B} \bar{C} \bar{D} \bar{U})}}}}.\end{aligned}$$

Our circuit configuration can now be drawn in logic block diagram form as shown in Fig. 7. Complementing of stages A, B, and C is seen to be delayed by passage of the count pulse through two levels of logic. Thus, for true instantaneous carry operation, two inverters having like delay are inserted at the D stage input.

Further simplification may be possible at this point through manipulation of the equations using the Boolean algebra techniques. When the Boolean expressions are reduced by straightforward methods using the Veitch diagram, a configuration is obtained which has two levels of logic (assuming of course, that the complements of the input variables are available). Factoring can be used in order to reduce the equations and also the hardware requirements, although some of the speed of the circuit is sacrificed. That is, when a Boolean expression has been factored and is implemented, some signals may have to pass through many levels of logic before appearing at the output. The designer must determine if the alteration, though affording him a savings in circuit elements, will adversely effect the overall circuit operation.

In our present example, the equation for the  $c(t)$  trigger can be further simplified as shown in the following:

$$c(t) = k \bar{A} D U + k B \bar{D} \bar{U} + k C \bar{D} \bar{U} + k A \bar{D} \bar{U}.$$

Factoring this equation, we readily obtain

$$c(t) = k \bar{A} D U + k \bar{D} \bar{U} (B + C + A)$$

and finally

$$c(t) = k \bar{A} D U + k \bar{D} \bar{U} \overline{\overline{(\bar{B} \bar{C} \bar{A})}}$$

since by De Morgan's theorem,

$$B + C + A = \overline{\overline{(\bar{B} \bar{C} \bar{A})}}.$$

Using De Morgan's theorem again, we can put the final equation in NAND logic form as

$$c(t) = \overline{\overline{\overline{\overline{(k \bar{A} D U)} \overline{\overline{(k \bar{D} \bar{U})} \overline{\overline{(\bar{B} \bar{C} \bar{A})}}}}}}}$$

This equation can be implemented using four NAND gates with a total of 13 input diodes whereas prior to simplification, five NAND gates and 20 input diodes were required. However, an extra circuit delay is imposed in the second term because of the  $\overline{\overline{(\bar{B} \bar{C} \bar{A})}}$  logical operation.

To illustrate how this circuit delay may have an adverse effect, consider the sets of hypothetical waveforms and the associated circuitry shown in Fig. 8. In case II since  $\bar{Y}$  does not go to ONE level prior to the count pulse due to added NAND gate delay, flip-flop Z does not get triggered as desired.

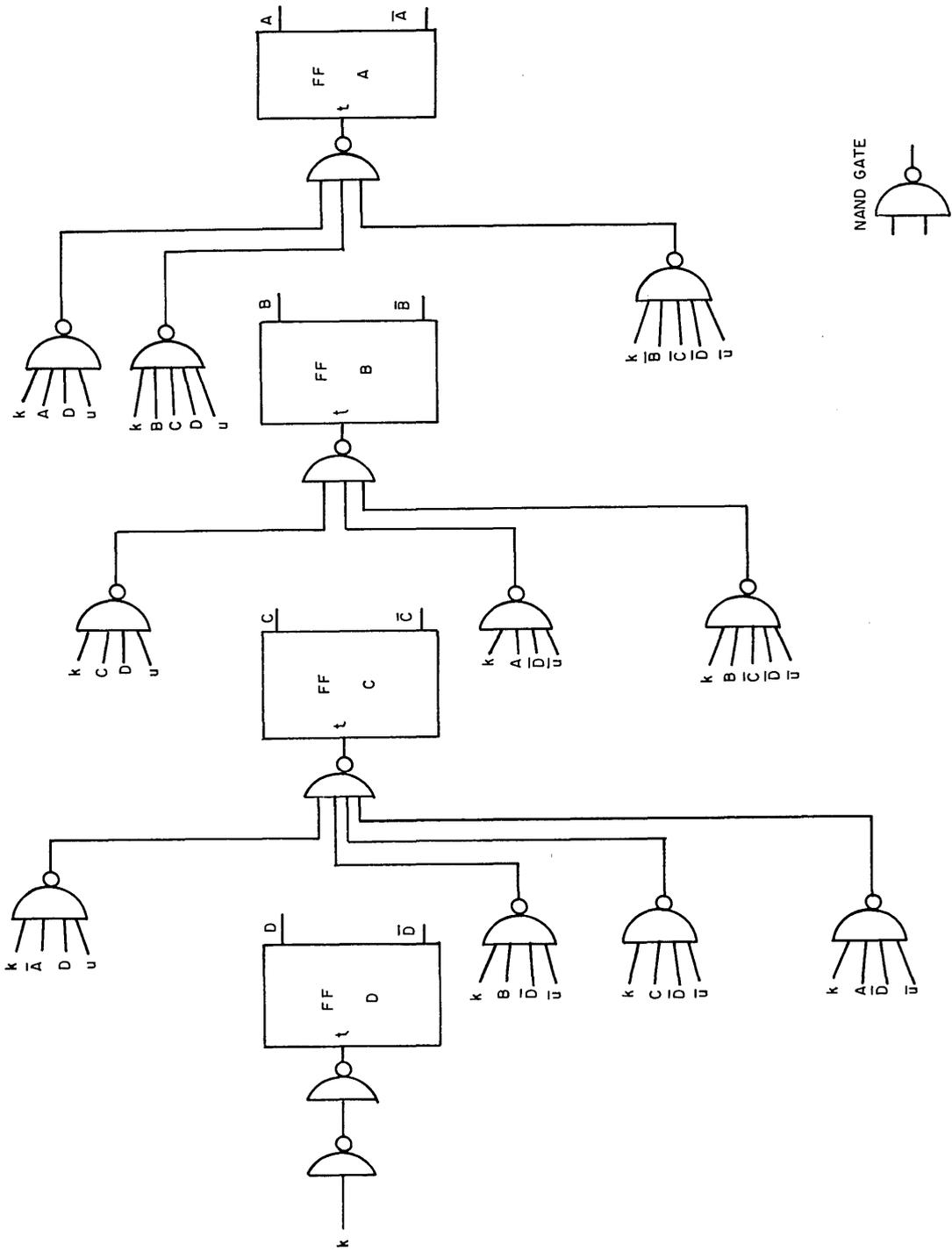


Fig. 7 - Up-down counter for dividing by 10

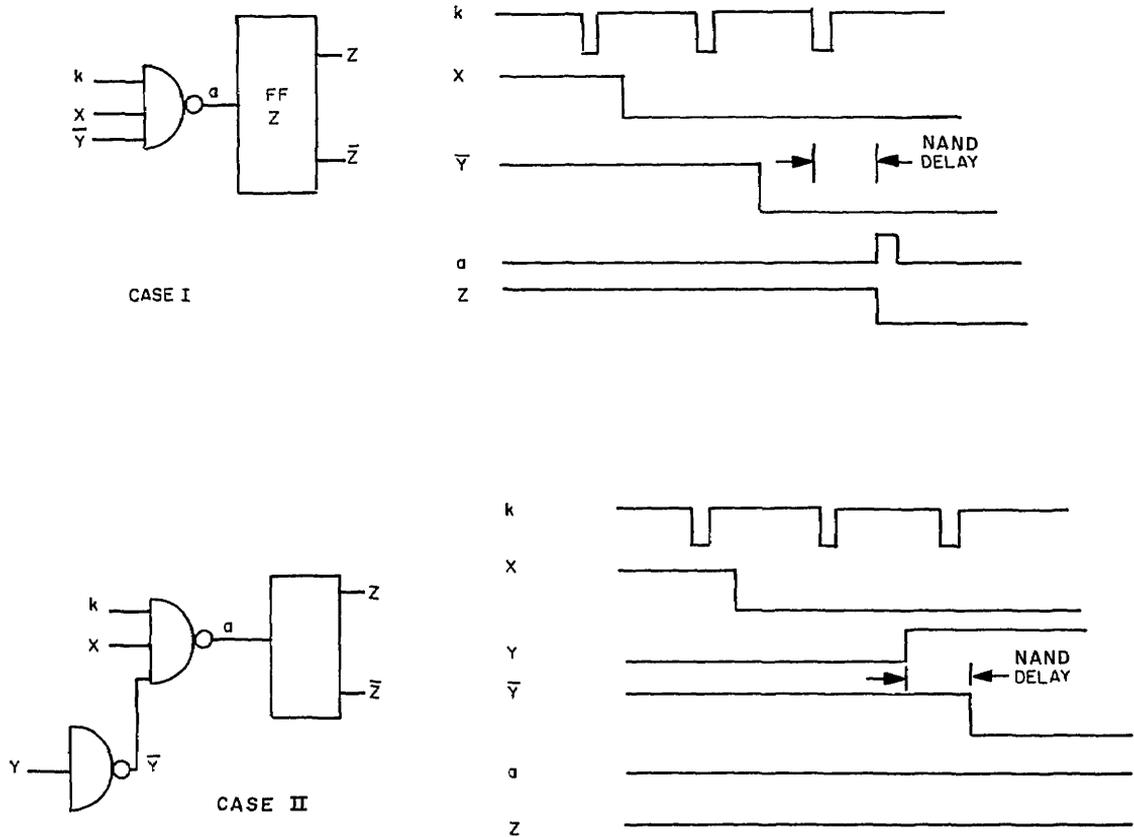


Fig. 8 - Adverse effect of an extra circuit delay

Unless the designer can be absolutely certain that he will not run into the type of problem just illustrated, it would be advisable to maintain only two logic levels throughout the circuit and forego the decrease in components.

CIRCUIT ANALYSIS

As a final treatment, it might be well to consider how the foregoing method for synthesizing a circuit can readily be applied for analysis. The sequence of any complex counter can be ascertained from the logic diagram of the counter by the following procedure.

Write Trigger Functions

Let us analyze a counter (Fig. 9) whose sequence is purely hypothetical and not in the least bit obvious. We first write the Boolean expression for each trigger input:

$$c(t) = \overline{\overline{(k \bar{A} \bar{C})}} \overline{\overline{(k B)}} \overline{\overline{(k A C)}}$$

$$b(t) = \overline{\overline{(k B)}} \overline{\overline{(k C)}}$$

$$a(t) = \overline{\overline{(k B C)}} \overline{\overline{(k A)}} \overline{\overline{(k \bar{B} \bar{C})}}$$

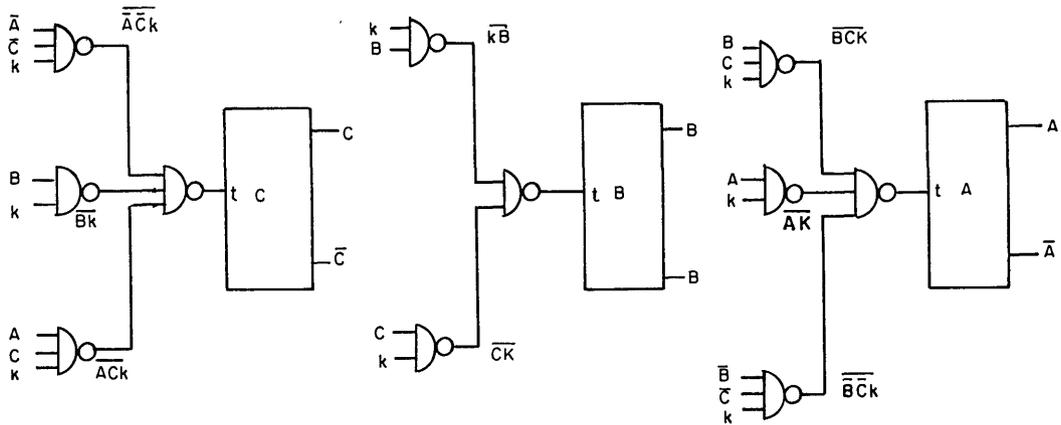


Fig. 9 - A complex counter assumed for analysis

Using De Morgan's theorem we write the equations in AND-OR form.

$$c(t) = k\overline{A}\overline{C} + kB + kAC$$

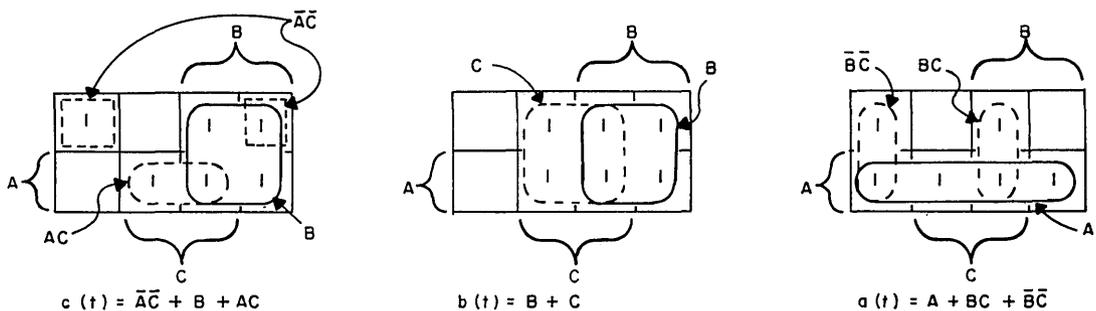
$$b(t) = kB + kC$$

$$a(t) = kB\overline{C} + kA + k\overline{B}\overline{C}.$$

Since the k term is simply the count pulse, it is not required in the analysis of the trigger gates.

Veitch Diagrams Used to Rewrite Functions as P Terms

Now we place ONES in the Veitch diagram squares which correspond to each term of the trigger equation:



Referring to the Veitch diagram for three variables (Fig. 6), we can write the P terms for those squares having ONES in expression form. Thus,

$$c(t) = P_0 + P_2 + P_3 + P_5 + P_6 + P_7$$

$$b(t) = P_1 + P_2 + P_3 + P_5 + P_6 + P_7$$

$$a(t) = P_0 + P_3 + P_4 + P_5 + P_6 + P_7$$

which are the final trigger gate expressions in canonic form. It remains now for us to utilize these functions to determine what sequence they cause the counter to count through.

#### Truth Table Leads to Count Sequence and "Don't Care" Terms

A truth table is set up as was done for synthesis, but with the Counter State (Variables) columns filled in only with the state corresponding to the lowest P term found in the trigger gate expressions. In our example, since  $P_0$  is the lowest term in the final trigger gate expressions in canonic form above, our initial entry in the Variables columns are 0 0 0. As the next step, we place a ONE under each Trigger Gate column which contains  $P_0$  as a term of its expression. In our example, a ONE would be placed under the Trigger Gate columns a and c. This, then, indicates which flip-flops are to be complemented when the counter is in the 0 0 0 state. On the next line of the Variables columns, we place that state of the counter which will occur after we perform the above noted complementing. In our example, since a and c have ONES, A changes from 0 to 1 and C changes from 0 to 1. These few preceding steps are shown tabulated below:

P Terms	Variables			Trigger Gate			Count Sequence
	A	B	C	a	b	c	
$P_0$	0	0	0	1	0	1	0
$P_5$	1	0	1				

Since we now know the next state of the counter, we can write the appropriate P term and once again place ONES under each Trigger Gate column which has that P term in its expression. In our case, we have  $P_5$ ; hence we place a ONE under a, b, and c, since all the gate expressions contain a  $P_5$  term. The appropriate variables are thus complemented, a new P term found, and trigger gates again determined. This procedure is continued until the sequence repeats itself. We then have arrived at the counter sequence, which is what we set out to find.

The completed solution is shown in Table 4, with the sequence shown in the right-hand column.

Table 4  
Completed Truth Table for the Counter in Fig. 9

P Terms	Variables			Trigger Gate			Count Sequence
	A	B	C	a	b	c	
$P_0$	0	0	0	1	0	1	0
$P_5$	1	0	1	1	1	1	5
$P_2$	0	1	0	0	1	1	2
$P_1$	0	0	1	0	1	0	1
$P_3$	0	1	1	1	1	1	3
$P_4$	1	0	0	1	0	0	4
$P_0$	0	0	0	1	0	1	0

It will be noted that P terms  $P_6$  and  $P_7$ , although appearing in the expressions for the trigger gates, do not appear in our reconstructed truth table. This is a consequence of the use of "don't care" terms in the original design of the counter in order to minimize circuit elements. In the analysis, the "don't care" terms are not at first apparent, but they show up by their absence in the final reconstruction. In troubleshooting a counter circuit, it may be necessary that we know if "don't care" terms were used in the design so that we may make an intelligent attempt at correcting any malfunctions.

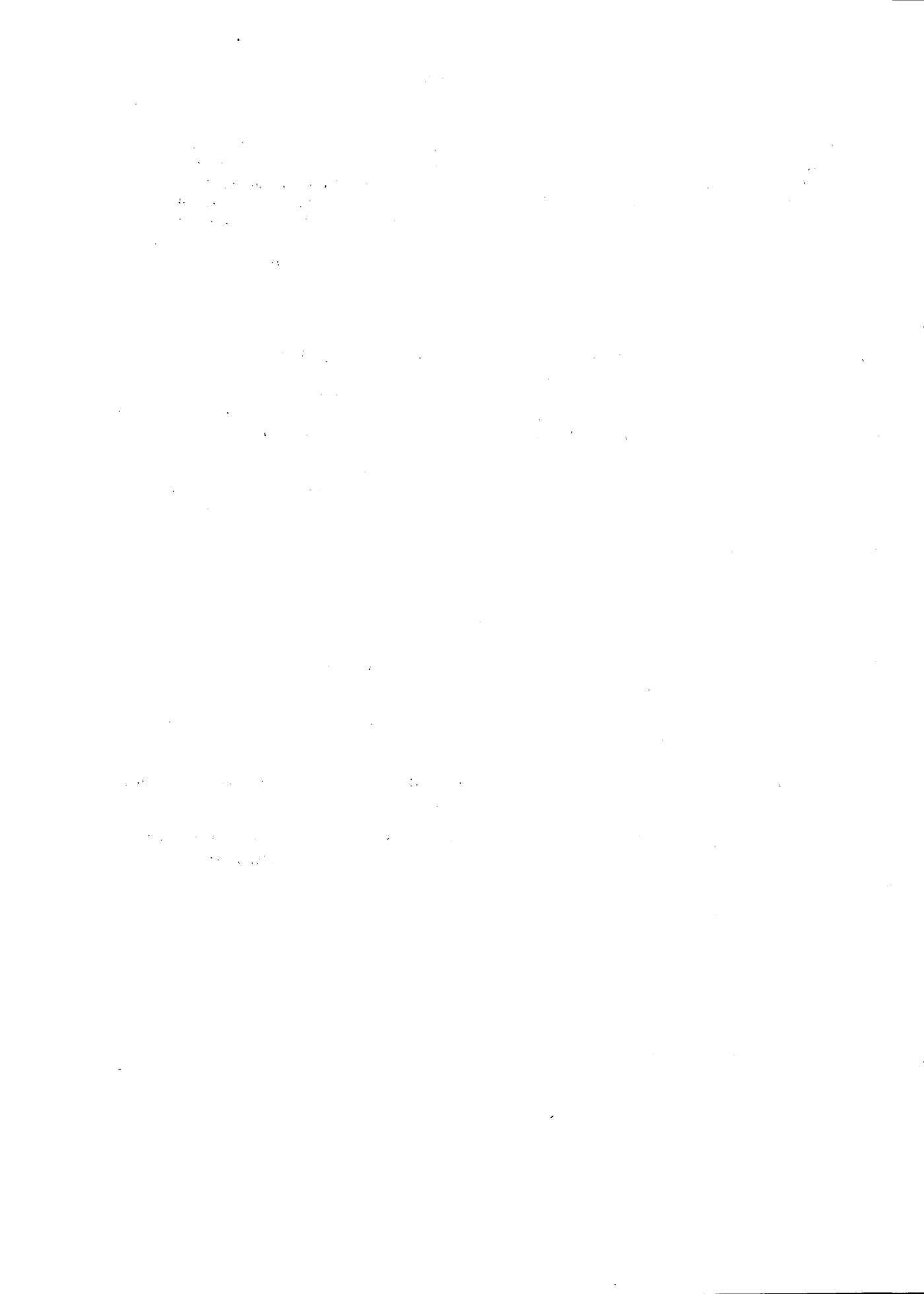
## CONCLUSIONS

It should be quite evident that although steps have been outlined for the design of counters and detailed examples have been given, it still remains for the designer to exercise judgment in determining the approaches to take for each situation. This report has discussed some areas where the designer is faced with a choice of approaches; some choices are clearly defined, while others depend largely upon intuition and experience.

In general, P terms and Veitch diagrams offer a convenient approach to finding a simplified circuit configuration for gating clock pulses to complement each counter stage so as to provide the desired arbitrary sequence. Conversely, the same technique provides a straightforward approach to the far more complex task of analyzing the unknown count sequence of a given counter design.

## REFERENCES

1. Meyer, B.W., "How to Design Arbitrary-Length Binary Counters," *Electronics* 36(No. 52):34 ff (Dec. 27, 1963)
2. Flegg, H.G., "Boolean Algebra and Its Application, Including Boolean Matrix Algebra," New York:Wiley, pp. 68-79, 1964
3. Veitch, E.W., "A Chart Method for Simplifying Truth Functions," *Proc. (Conference) Assoc. Comp. Mach.*, Pittsburgh, May 2-3, 1952, pp. 127-133
4. Tanana, E.J., "The Map Method," ch. 4 in "A Survey of Switching Circuit Theory," E.J. McCluskey, Jr., and T.C. Bartee (editors), New York:McGraw-Hill, 1962



## DOCUMENT CONTROL DATA - R&amp;D

*(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)*

1. ORIGINATING ACTIVITY <i>(Corporate author)</i> U.S. Naval Research Laboratory Washington, D.C. 20390		2a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>	
		2b. GROUP	
3. REPORT TITLE <b>PROCEDURE FOR THE DESIGN AND ANALYSIS OF ARBITRARY LENGTH DIGITAL COUNTERS</b>			
4. DESCRIPTIVE NOTES <i>(Type of report and inclusive dates)</i> <b>An interim report on one phase of the problem.</b>			
5. AUTHOR(S) <i>(Last name, first name, initial)</i> <b>Orsino, R.J., and Talmadge, H.G., Jr.</b>			
6. REPORT DATE <b>May 6, 1965</b>	7a. TOTAL NO. OF PAGES <b>26</b>	7b. NO. OF REFS <b>4</b>	
8a. CONTRACT OR GRANT NO. <b>NRL Problem Y01-01</b>	9a. ORIGINATOR'S REPORT NUMBER(S) <b>NRL Report 6230</b>		
b. PROJECT NO. <b>SF 019-01-03, Task 6168</b>			
c.	9b. OTHER REPORT NO(S) <i>(Any other numbers that may be assigned this report)</i>		
d.			
10. AVAILABILITY/LIMITATION NOTICES <b>Unlimited availability. Available at CFSTI -</b>			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY <b>Dep't of the Navy (Bureau of Ships)</b>	
13. ABSTRACT  <b>A step-by-step procedure is described for designing digital counters having complex, arbitrary sequences. A divide-by-ten up-down counter is then designed to illustrate use of truth tables, "P" terms, Veitch diagrams, and "don't care" terms for simplifying the gate functions used to trigger each counter stage. The value of this procedure in circuit analysis as well as for circuit synthesis is then emphasized and illustrated by example. This procedure makes use of fundamental Boolean algebra and minimization techniques.</b>			

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Digital counters Circuit design Minimization of logic elements Circuit analysis Ripple counter Arbitrary length or sequence counters Boolean algebra application De Morgan's theorem Truth table Veitch diagrams						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.

2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.

8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those

imposed by security classification, using standard statements such as:

- (1) "Qualified requesters may obtain copies of this report from DDC."
- (2) "Foreign announcement and dissemination of this report by DDC is not authorized."
- (3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through \_\_\_\_\_."
- (4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through \_\_\_\_\_."
- (5) "All distribution of this report is controlled. Qualified DDC users shall request through \_\_\_\_\_."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TS), (S), (C), or (U).

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.