NASA TECHNICAL NOTE

NASA TN D-3273

C. 1

NASA TN D-3273

TECH LIBRARY KAFB, NM

0079797

AN ALGORITHM FOR SYNTHESIS OF ASYNCHRONOUS SEQUENTIAL CIRCUITS

by John S. Tripp

Langley Research Center

Langley Station, Hampton, Va.

RECEIVED
18 NOV 1966

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • NOVEMBER 1966

TECH LIBRARY KAFB, NM

0079797

NASA TN D-3273

AN ALGORITHM FOR SYNTHESIS OF ASYNCHRONOUS SEQUENTIAL CIRCUITS

By John S. Tripp

Langley Research Center
Langley Station, Hampton, Va.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – Price $1.00

AN ALGORITHM FOR SYNTHESIS OF ASYNCHRONOUS SEQUENTIAL CIRCUITS*

By John S. Tripp
Langley Research Center

SUMMARY

An algorithm for synthesis of asynchronous sequential circuits is developed; this development is an extension of the well-known Boolean methods for synthesis of synchronous circuits. A matrix system of function representation is introduced and utilized in the execution of the algorithm. As an example, a six-count counter is synthesized with synchronous and asynchronous designs. Both designs are presented for comparison.

INTRODUCTION

A sequential switching circuit is a black box having inputs, outputs, and a number of possible internal states. The values of the inputs, outputs, and the internal state are considered at periodic intervals of time, known as bit times. At a given bit time, the internal state is determined by the inputs and internal state at the previous bit time; the outputs are determined by the inputs and internal state at the present bit time (ref. 1, p. 144). For purposes of the present paper, a sequential switching circuit is defined as an interconnection of logical and memory elements, where the internal state is the set-reset pattern of the memory elements.

Sequential circuit designs may be classed as being either synchronous or asynchronous. The class to which a circuit belongs is determined by the source of the switching commands to the memory elements. In a synchronous sequential circuit, those memory elements which are to be switched during each internal-state change do so simultaneously under command of a central pulse source or clock. In the asynchronous sequential circuit, however, the switching pulse is derived from the switching action or transition of another memory element. During each internal-state change the memory elements are switched in a chain reaction, which can result in significant delays and may lead to timing

*The information presented herein is largely based on a paper entitled "Design of Asynchronous Sequential Circuits Using Boolean Algebra" by John S. Tripp published in Proceedings of the Central States Simulation Council Meeting on Analog Computation and Allied Digital Problems, Special Rept. No. 37, Kansas State Univ. Bull., vol. 47, no. 9, Sept. 1963.

difficulties. However, an asynchronous design has the advantage that the number of logic elements required is usually less, sometimes considerably less, than that for an equivalent synchronous design.

Formal procedures for the synthesis of synchronous sequential circuits using Boolean algebra are presented in detail in the literature. Synthesis of asynchronous circuits has been limited to nonanalytic procedures and the designer's ingenuity. This paper extends the synchronous synthesis methods to include asynchronous circuits and develops a matrix system of function representation which aids in the performance of the synthesis procedure.

SYMBOLS

General:

A,B,C	flip-flop stages
D	arbitrary Boolean function
E	flip-flop enable input
f,g	general Boolean functions
f_k	kth function value of f
H	Boolean constraining function
I_p	general input of memory element Q_p
$K\left(\Lambda'\left(I_p\right)\right)$	cardinality of the set $\Lambda'\left(I_p\right)$
m_k	kth minterm
p	flip-flop pulse input
Q_k^n	internal state of kth memory element at time n
R_p	reset input of flip-flop A_p
S_p	set input of flip-flop A_p
R_{pE}	reset enable input for flip-flop A_p
S_{pE}	set enable input for flip-flop A_p
X	"don't care" condition

x_m	mth system input
α, β	transition variables
Δ_p	the set of transition functions of order less than p, and the corresponding matrix
δ_{1k}	α transition function from δQ_k transition equation
δ_{2k}	β transition function from δQ_k transition equation
δQ_k	transition equation for memory element Q_k
$\Lambda(I_p)$	the set of input gating functions for input I_p, and the corresponding matrix
$\Lambda'(I_p)$	a subset of $\Lambda(I_p)$
$\lambda_{ij}(I_p)$	input gating function from transition function δ_{ij} for input I_p
$\lambda'_{ij}(I_p)$	an element of $\Lambda'(I_p)$
τ_{I_p}	pulse input for input I_p

Subscripts and superscripts:

a,b,i,j,k,m,p,y,z	integral indices
n	time
r	number of memory elements in a sequential circuit
s	number of variables in a function

Operators and relations:

$-$	complement
$+$	"or"
$\cdot$	"and"; intersection
$\bar{\cdot}$	nand
$\odot$	special operator
Σ	disjunction
ϵ	is an element of

$\subseteq$	set inclusion
$<$	is less than
$\leq$	is less than or equal to

TYPICAL LOGIC CIRCUITS

Logic circuits suitable for use with asynchronous designs are illustrated in figures 1 and 2. Figure 1 shows a typical resistor-coupled transistor inverting gate. A typical ac coupled set-reset flip-flop circuit which is well suited for both synchronous and asynchronous designs is shown in figure 2. The flip-flop is switched only by satisfying certain conditions on a pulse-enable (P-E) input pair. The pulse (P) input receives a switching pulse, which may either be enabled or blocked by the condition existing on the enable (E) input of the pair. To provide "or" gating, several pairs of pulse and enable inputs may be provided on both the set and reset sides of each flip-flop. This type of flip-flop is available commercially from a large number of manufacturers. For the particular circuit in figure 2, if the enable (E) input is held at ground, then a voltage transition from the positive level to ground on the pulse input will result in a switching command. A positive voltage on the enable input will block the effect of transitions on the pulse input. A voltage transition from ground to the positive level will be ignored regardless of the condition at the level input.

REVIEW OF SYNCHRONOUS SYNTHESIS METHODS

Details of synchronous synthesis procedures are well presented in reference 1. Any logical memory element may have its operation represented by a Boolean equation in which its internal state, after being subject to a switching command, is expressed as a function of the inputs and the previous internal state. This equation is known as the characteristic equation of the element. Let the internal state at time n be represented by Q^n, and the state at time $n + 1$ after receiving a clock pulse be Q^{n+1}. For an R-S flip-flop the characteristic equation is

$$Q^{n+1} = S + Q^n\bar{R} \tag{1}$$

with the constraint that $RS = 0$. The following general characteristic equation is a Boolean difference equation:

$$Q^{n+1} = f\left(Q^n, x_1, \ldots, x_m\right) \tag{2}$$

with the constraint that $H\left(Q^n, x_1, \ldots, x_m\right) = 0$, where $x_1, \ldots, x_m$ are inputs.

Operation of any sequential switching circuit consisting of k memory elements may be represented by k Boolean difference equations. These expressions are known as application equations and each expresses the state Q^{n+1} of its corresponding memory device at time n. A general application equation is

$$Q_i^{n+1} = g_i\left(Q_1^n, \ldots, Q_r^n, x_1, \ldots, x_m\right) \tag{3}$$

where $Q_1, \ldots, Q_r$ are the r memory-element stages, and $x_1, \ldots, x_m$ are the m system inputs.

By solving the application equation (eq. (3)) of flip-flop Q simultaneously with the characteristic equation (eq. (1)), a set of input equations is obtained; each equation represents a required gating function to be provided to a memory element input. For ac coupled flip-flops the gating function determines the required gating structures into each enable input. A clock feeds all pulse inputs.

ASYNCHRONOUS SYNTHESIS - BASIC CONSIDERATIONS

The asynchronous synthesis technique presented herein is useful only for implementation of sequential circuits whose switching sequences have been specified and for which input equations have been determined as for a synchronous design. Circuits having pulse feedback are not easily considered.

Let flip-flop circuits having pulse and enable inputs, as in figure 2, be used. An association of voltage levels and logic values will be made by letting the enabling voltage level be associated with logical "1."

Logical transition variables similar to those in reference 2 are now defined. Consider a Boolean variable Q_k. If it is true that, during the time transition from n to n + 1, Q_k goes from "0" to "1", then an α transition has occurred, and the α transition function δ_{1k} equals 1; otherwise δ_{1k} equals 0. Correspondingly, if Q_k goes from "1" to "0", a β transition has occurred and the β transition function δ_{2k} equals 1; otherwise δ_{2k} equals 0. It is seen that δ_{1k} and δ_{2k} are mutually exclusive.

As shown in reference 1, the general application equation is

$$Q_k^{n+1} = g_{1k}Q_k^n + g_{2k}\overline{Q_k^n} \tag{4}$$

where g_{1k} and g_{2k} are independent of Q_k^n. Truth equations for δ_{1k} and δ_{2k} are now derived from the application equation by means of the following truth table:

g_{1k}	g_{2k}	Q_k^n	Q_k^{n+1}	δ_{1k}	δ_{2k}
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	1	0
0	1	1	0	0	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	1	0	0

It is seen that $\delta_{1k} = g_{2k}\overline{Q_k^n}$ and that $\delta_{2k} = \overline{g_{1k}}Q_k^n$. The transition equation δQ_k for Q_k is defined as

$$\delta Q_k = g_{2k}\overline{Q}_k\alpha + \bar{g}_{1k}Q_k\beta = \delta_{1k}\alpha + \delta_{2k}\beta \tag{5}$$

Since δ_{1k} and δ_{2k} are mutually exclusive, δQ_k will equal either α, β, or 0, indicating the occurrence of an α transition when $\delta Q_k = \alpha$, or the occurrence of a β transition when $\delta Q_k = \beta$. The transition equation δQ_k thus is nonzero for those states in the switching sequence at which Q_k undergoes α or β transitions. The transition function δ_{1k} (δ_{2k}) is an algebraic representation of the set of states at which α (β) transitions occur. Any implicant (ref. 1) of δ_{1k} (δ_{2k}) may be selected by intersecting δ_{1k} (δ_{2k}) with an appropriate function. A graphic method for determining a function to accomplish such a selection process is presented in reference 2. An algebraic method will be developed later in this paper.

In an asynchronous circuit a given flip-flop input will be connected in the following way: A pulse input will receive switching commands from the output of another flip-flop, and the corresponding enable input will receive enabling levels from gating structures. Given a flip-flop input equation and the set of transition equations of the sequential machine, the problem is to determine which flip-flop outputs should be connected to the pulse inputs of the given flip-flop and what gating function should feed each enabling input. Hence, the requirement of the synthesis procedure is to imply each input equation completely with a disjunction of implicants of appropriate transition functions. The group of transition functions from which implicants are selected will determine which flip-flop outputs must feed the pulse inputs, and the particular implicants chosen from each transition function will determine the gating function feeding each associated enable input. For example, if implicants of transition functions from flip-flop stages A and B are necessary to imply reset input function R_k, A and B will each feed a pulse input on R_k, and gating levels selecting the desired implicants from the transition functions of A and B will feed the associated enable inputs on R_k.

The flip-flops of the sequential circuit must be ordered to determine the propagation sequence of switching commands. Determination of the best ordering may be difficult and is affected by the following considerations: Generally the flip-flops undergoing transitions most frequently in the switching sequence should precede those undergoing transitions less frequently. One or more of the flip-flops undergoing transitions most frequently will be chosen to receive clock pulses, so that the totality of their transition functions contain complete implication for input functions of following flip-flops. For circuits with several flip-flops, propagation delays in switching transitions may result in an enabling level changing before the arrival of its associated switching transition. Hence, the ordering must be such that all enabling functions which change with their associated switching transitions have propagation delays at least as great as those of the transition delay. A method for determining an ordering will be presented later.

Assume that flip-flop pulse inputs respond to 0 to 1, or α, transitions. Then if an implicant of the α transition function, δ_{1k}, of flip-flop Q_k is an implicant for the input function I_p, the asserted or "true" output of Q_k will be connected to the pulse input of I_p. If an implicant of the β transition function, δ_{2k}, of Q_k is a desired implicant for I_p, the negated or "false" output of Q_k will feed the pulse input of I_p.

When several transition sources are required to imply input I_p, redundant implication of some minterms (ref. 1) of I_p may result. If the delay between redundant input transitions is greater than the recovery time of the flip-flop and if multiple activation of that input cannot be allowed, as with "T" inputs, the selection of implicants must be nonredundant to avoid unwanted switching. If no implicants for certain minterms of I_p exist in the set of appropriate transition equations, clock pulses may be used.

FORMAL SYNTHESIS DEVELOPMENT

The basis for the formal synthesis procedure is now developed. Assume that all design steps of the synchronous procedure have been carried out through the determination of input equations and that a transition equation has been obtained from each application equation. Let a general input equation be

$$I_p = f_p\left(Q_1, \ . \ . \ ., \ Q_r, \ x_1, \ . \ . \ ., \ x_m\right) \tag{6}$$

and let the set of lower order transition equations for implication of I_p be

$$\begin{matrix} \delta Q_1 = g_{21}\bar{Q}_1\alpha + \overline{g_{11}}Q_1\beta = \delta_{11}\alpha + \delta_{21}\beta \\ \vdots \\ \delta Q_r = g_{2r}\bar{Q}_r\alpha + \overline{g_{1r}}Q_r\beta = \delta_{1r}\alpha + \delta_{2r}\beta \end{matrix} \tag{7}$$

If δ_{ij} is intersected with I_p, the common portion will remain. If I_p is intersected with each element of the set of all δ_{ij}'s, a comparison of the set of intersections with I_p will show which implicants of the δ_{ij}'s are implicants of I_p and whether I_p can be completely implied by implicants of the δ_{ij}'s. Given the necessary implicants from the δ_{ij}'s to imply I_p completely, for each selected δ_{ij} a function $\lambda_{ij}(I_p)$ must be determined which, when intersected with δ_{ij}, will yield the desired implicant from δ_{ij}. Comparison of all $\delta_{ij} \cdot I_p$ with I_p, selection of a complete set of implicants for I_p, determination of all $\lambda_{ij}(I_p)$, and determination of the simplest solution must be accomplished by the synthesis procedure to be developed. For simple cases this procedure may be accomplished by inspection by using mapping techniques. Otherwise a tabular method involving minterm checks similar to the Quine reduction method (ref. 1), or the algorithm to be developed in this paper, must be used.

The existence of the function $\lambda_{ij}(I_p)$ and its simplest form are established by the following theorems:

Theorem 1.-

For any Boolean functions δ_{ij} and I_p there exists a function, $\lambda_{ij}(I_p)$, where

$$\lambda_{ij}(I_p) \cdot \delta_{ij} = I_p \cdot \delta_{ij} \tag{8}$$

such that

(a) $\lambda_{ij}(I_p)$ is equal to I_p or may have fewer literals than I_p

(b) $\lambda_{ij}(I_p) \cdot \delta_{ij}$ is an implicant of I_p

Proof:

Assume that $\lambda_{ij}(I_p) = \overline{\delta_{ij}}D + \delta_{ij}I_p$

Then

$$\lambda_{ij}\left(I_p\right) \cdot \delta_{ij} = \left(\overline{\delta_{ij}}D + \delta_{ij}I_p\right) \cdot \delta_{ij} = \delta_{ij}I_p$$

and $\overline{\delta_{ij}}D + \delta_{ij}I_p$ is such a function; hence, the existence of $\lambda_{ij}\left(I_p\right)$.

(a) Choose $D = I_p$. Then $\lambda_{ij}\left(I_p\right) = \overline{\delta_{ij}}I_p + \delta_{ij}I_p = I_p$.

Thus $\lambda_{ij}\left(I_p\right)$ may always be made equal to I_p.

Suppose $\delta_{ij} \subseteq I_p$. Then $\delta_{ij}I_p = \delta_{ij}$.

Choose $D = 1$. Then $\lambda_{ij}\left(I_p\right) = \overline{\delta_{ij}} + \delta_{ij} = 1$.

Hence for some cases $\lambda_{ij}\left(I_p\right)$ may be as simple as "1." In any case $\lambda_{ij}\left(I_p\right)$ will not have more literals than I_p.

(b) $\lambda_{ij}\left(I_p\right) \cdot \delta_{ij} = I_p \cdot \delta_{kj} \subseteq I_p$

Since $\lambda_{ij}\left(I_p\right) \cdot \delta_{ij}$ is wholly contained in I_p, it is an implicant of I_p.

Theorem 2.-

The function $\lambda_{ij}\left(I_p\right)$, satisfying the condition of theorem 1, which has the fewest literals will be obtained from

$$\lambda_{ij}\left(I_p\right) = \overline{\delta_{ij}}D + \delta_{ij}I_p \tag{9}$$

where D may be any arbitrary function of the variables involved.

Proof:

$\lambda_{ij}\left(I_p\right)$ is a function only of I_p and δ_{ij}. Choosing $\lambda_{ij}\left(I_p\right)$ to satisfy $\lambda_{ij}\left(I_p\right) \cdot \delta_{ij} = I_p \cdot \delta_{ij}$, gives by exhaustion

δ_{ij}	I_p	$I_p \cdot \delta_{ij}$	$\lambda_{ij}(I_p)$
0	0	0	D
0	1	0	D
1	0	0	0
1	1	1	1

Hence $\lambda_{ij}(I_p) = \delta_{ij}D + \delta_{ij}I_p$ and can be made to have the minimum possible number of literals since a maximum number of "don't cares" can be included in a map of $\lambda_{ij}(I_p)$ in the variables involved.

MATRIX PROCEDURE

A matrix procedure is now developed which readily permits determination of $\lambda_{ij}(I_p)$, easily considers redundancies, and allows determination of realizability and the simplest solution. The technique is suitable for programing on a digital computer.

Consider a Boolean function f which is represented in canonical form as a disjunction of minterms in s variables. This form is

$$f = \sum_{k=0}^{2^s-1} f_k m_k \tag{10}$$

where f_k is the kth function value and m_k is the kth minterm. See reference 1 for detailed definitions. Let f be represented in column matrix form as follows:

$$f = \begin{bmatrix} f_0 \\ f_1 \\ \cdot \\ \cdot \\ \cdot \\ f_{2^s-1} \end{bmatrix} \tag{11}$$

It can easily be shown that for functions f and g:

$$f + g = \begin{bmatrix} f_0 + g_0 \\ f_1 + g_1 \\ \cdot \\ \cdot \\ \cdot \\ f_{2^s-1} + g_{2^s-1} \end{bmatrix} \tag{12}$$

$$f \cdot g = \begin{bmatrix} f_0 \cdot g_0 \\ f_1 \cdot g_1 \\ \cdot \\ \cdot \\ \cdot \\ f_{2^s-1} \cdot g_{2^s-1} \end{bmatrix} \tag{13}$$

Order the set of memory element transition equations into a set $\{\delta Q_1, \delta Q_2, \ldots, \delta Q_r\}$ such that for all $i < j$ the number of "0's" in δQ_i is less than or equal to the number of "0's" in δQ_j.

I_p represents an input to memory element Q_p. Then let Δ_p represent the set of δ_{ij} transition functions associated with the set of memory elements of lower order than Q_p.

$$\Delta_p = \{\delta_{11}, \delta_{21}, \delta_{12}, \delta_{22}, \ldots, \delta_{1(p-1)}, \delta_{2(p-1)}\} \tag{14}$$

Using column matrix form for the δ_{ij} functions, Δ_p can be represented as a 2^s row by $2(p - 1)$ column matrix, where row i corresponds to the ith minterm, and each column to a particular δ_{ij} function. The matrix form for Δ_p is

$$\Delta_p = \begin{bmatrix} \delta_{11,0} & \delta_{21,0} & \cdots & \delta_{1(p-1),0} & \delta_{2(p-1),0} \\ \delta_{11,1} & \delta_{21,1} & \cdots & \delta_{1(p-1),1} & \delta_{2(p-1),1} \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ \delta_{11,2^s-1} & \delta_{21,2^s-1} & \cdots & \delta_{1(p-1),2^s-1} & \delta_{2(p-1),2^s-1} \end{bmatrix} \tag{15}$$

This matrix displays as "1" entries the set of switching sequence states at which memory elements Q_1 through Q_{p-1} undergo α or β transitions.

For each δ_{ij} contained in Δ_p, the set of common implicants of I_p and and δ_{ij} is represented by $I_p \cdot \delta_{ij}$. By theorem 1, $I_p \cdot \delta_{ij} = \lambda_{ij}(I_p) \cdot \delta_{ij}$, where $\lambda_{ij}(I_p)$ is given by equation (9). Equation (9) shows that $\lambda_{ij}(I_p)$ consists of a disjunction of $I_p \cdot \delta_{ij}$ and a collection of "don't cares." Therefore, each "1" element of the column matrix for $\lambda_{ij}(I_p)$ represents a common implicant of δ_{ij} and I_p; all other elements of $\lambda_{ij}(I_p)$ will be "0" or "X."

Let I_p be applied by means of equation (9) to each δ_{ij} function contained in Δ_p, yielding the set of functions

$$\Lambda(I_p) = \left\{\lambda_{11}(I_p), \lambda_{21}(I_p), \ldots, \lambda_{1(p-1)}(I_p), \lambda_{2(p-1)}(I_p)\right\} \tag{16}$$

where

$$\lambda_{ij}(I_p) = \overline{\delta_{ij}}D + \delta_{ij}I_p \tag{17}$$

Using column matrix form for the functions $\lambda_{ij}(I_p)$, $\Lambda(I_p)$ can be represented as a 2^s row by $2(p - 1)$ column matrix as follows:

$$\Lambda(I_p) = \begin{bmatrix} \lambda_{11,0}(I_p) & \lambda_{21,0}(I_p) & \cdots & \lambda_{1(p-1),0}(I_p) & \lambda_{2(p-1),0}(I_p) \\ \lambda_{11,1}(I_p) & \lambda_{21,1}(I_p) & \cdots & \lambda_{1(p-1),1}(I_p) & \lambda_{2(p-1),1}(I_p) \\ \vdots & \vdots & & \vdots & \vdots \\ \lambda_{11,2^s-1}(I_p) & \lambda_{21,2^s-1}(I_p) & \cdots & \lambda_{1(p-1),2^s-1}(I_p) & \lambda_{2(p-1),2^s-1}(I_p) \end{bmatrix} \tag{18}$$

Using equations (10) through (18) it is seen that each element of the $\Lambda(I_p)$ matrix is determined from corresponding elements of Δ_p and I_p by the following equation:

$$\lambda_{ij,k}(I_p) = \overline{\delta_{ij,k}} \cdot X + \delta_{ij,k} \cdot I_{p,k} \quad \begin{pmatrix} 1 \leq i \leq 2 \\ 1 \leq j \leq p-1 \\ 1 \leq k \leq 2^s-1 \end{pmatrix} \tag{19}$$

where k is the row number. Equation (19) generalized into matrix form is

$$\Lambda(I_p) = \overline{\Delta_p} \cdot D + \Delta_p \cdot I_p = I_p \odot \Delta_p \tag{20}$$

The "$\odot$" symbol is introduced to effect notational compactness as demonstrated in the example which follows.

It can now be determined from $\Lambda(I_p)$ whether I_p is equivalent to some disjunction of implicants of the set Δ_p. Since each "1" element of $\Lambda(I_p)$ represents a common implicant of some δ_{ij} and I_p, a necessary condition for the realizability of an asynchronous implementation of the switching circuit under consideration is that for each row of I_p containing a "1" there exists a "1" in some column of the corresponding row of $\Lambda(I_p)$. If this condition is satisfied, the procedure continues by selecting from functions contained in

$\Lambda(I_p)$ a minimal subset $\Lambda'(I_p)$ which also satisfies the realizability condition. The subset is then

$$\Lambda'(I_p) = \{\lambda_{ab}'(I_p), \ldots, \lambda_{yz}'(I_p)\} \tag{21}$$

The set union of the implicants represented by "1's" in the matrix form of $\Lambda'(I_p)$ is the complete set of implicants of I_p. Therefore it is seen that

$$I_p = \sum_{\lambda_{ij}'(I_p) \in \Lambda'(I_p)} \lambda_{ij}'(I_p) \cdot \delta_{ij} \tag{22}$$

Now δ_{ij} is associated with an output of memory element Q_j which is asserted or negated according to the value of i. The function $\lambda_{ij}(I_p)$ represents the output of a gating network. The "." operation is accomplished by a P-E input pair of a flip-flop. Thus, to implement input I_p, $K(\Lambda'(I_p))$ P-E input pairs are necessary, where $K(\Lambda'(I_p))$ is the cardinality of the set $\Lambda'(I_p)$. For each P-E pair, an output of Q_j (asserted if $i = 1$ and negated if $i = 2$) will feed the P input and the gating network implementing $\lambda_{ij}(I_p)$ will feed the E input. By theorem 2, $\lambda_{ij}(I_p)$, by proper choice of values for its "X" entries, is the simplest function satisfying equation (8) and hence will give the most economical implementation.

If redundant implication of I_p cannot be allowed because of the type of flip-flop circuit in use, the process of forming the subset $\Lambda'(I_p)$ must be modified so that only a single "1" exists in each row where a "1" is required. This can be readily accomplished by setting redundant "1's" to "0." The selection of a minimal subset $\Lambda'(I_p)$ and the determination of which redundant "1's" in $\Lambda'(I_p)$ to set to "0", such that the most economical implication of I_p is realized, may be lengthy for cases having large numbers of variables; it would be best carried out by a digital computer.

THE ALGORITHM

A step-by-step procedure for synthesis is now given and is followed by an example for clarification.

(1) Determine the switching sequence of the sequential machine.

(2) Make state assignments to the memory elements.

(3) For each memory element Q determine the column matrix of the transition equation δQ. Order the memory element transition equations into a set

$\{\delta Q_1, \delta Q_2, \ldots, \delta Q_r\}$ such that for all $i < j$ the number of "0's" in $\delta Q_i \leq$ the number of "0's" in δQ_j.

(4) For each transition equation δQ_p determine the α and β transition functions, δ_{1p} and δ_{2p}, and the input equations I_p, all in column matrix form.

(5) For each input I_p form a transition matrix Δ_p whose column space consists of all δ_{ij} transition functions of order less than p.

(6) Compute the matrix $\Lambda(I_p)$ by applying equation (20), $\Lambda(I_p) = I_p \odot \Delta_p$, to Δ_p and I_p.

(7) Compare I_p with $\Lambda(I_p)$. An asynchronous implementation is realizable only if for each row of I_p that contains a "1" there exists at least one "1" in the same row of Λ_p.

(8) Select a minimal subset, $\Lambda'(I_p)$, of column matrices from $\Lambda(I_p)$ such that for each row of I_p containing a "1", a "1" exists in at least one column of the corresponding row of $\Lambda'(I_p)$.

(9) Change undesired "1's" in $\Lambda'(I_p)$ to zero. If nonredundant implication of I_p is necessary, then for each "1" in I_p only one of the selected column matrices may contain a corresponding "1." Note that the preceding restriction will not affect "don't cares."

(10) Each selected $\lambda_{ij}(I_p)$ column matrix will indicate a flip-flop output which will feed a pulse input of I_p. If the chosen column matrix exhibits the same type of transition as is desired, that is, 0 to 1 (α) or 1 to 0 (β), the asserted output of its associated flip-flop will feed the pulse input. If it exhibits the opposite type of transition to what is desired, the negated output will feed the pulse input.

(11) Determine the minimum algebraic form of each selected $\lambda_{ij}(I_p)$ column matrix. This is the set of functions which will feed the enable inputs of I_p.

(12) Repeat steps 5 to 11 for each input I_p.

EXAMPLE OF ASYNCHRONOUS DESIGN

A six-count counter will now be designed for both synchronous and asynchronous implementation.

Step 1. The switching sequence is 0, 1, 2, 3, 4, 5, 0, 1, ...

Step 2. Let flip-flop A_3 have a weight of 4, A_2 a weight of 2, and A_1 a weight of 1. The transition table is as follows:

A_3^n	A_2^n	A_1^n	A_3^{n+1}	A_2^{n+1}	A_1^{n+1}
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	X	X	X
1	1	1	X	X	X

Step 3. The application equations for A_3, A_2, and A_1 in column matrix form are

$$A_3^{n+1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ X \\ X \end{bmatrix} \quad A_2^{n+1} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ X \\ X \end{bmatrix} \quad A_1^{n+1} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ X \\ X \end{bmatrix} \tag{23}$$

and the transition equations are

$$\delta A_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \alpha \\ 0 \\ \beta \\ X \\ X \end{bmatrix} \quad \delta A_2 = \begin{bmatrix} 0 \\ \alpha \\ 0 \\ \beta \\ 0 \\ 0 \\ X \\ X \end{bmatrix} \quad \delta A_1 = \begin{bmatrix} \alpha \\ \beta \\ \alpha \\ \beta \\ \alpha \\ \beta \\ X \\ X \end{bmatrix} \tag{24}$$

Since A_1 has the fewest "0's" and since A_3 has the most "0's", the ordering is $\{\delta A_1, \delta A_2, \delta A_3\}$.

Step 4. The α and β transition functions are

$$\left.\begin{aligned}
\delta_{13} &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \\ 0 \\ 0 \\ X \\ X \end{bmatrix} \qquad \delta_{23} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \\ 0 \\ 1 \\ X \\ X \end{bmatrix} \\
\delta_{12} &= \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \\ 0 \\ 0 \\ X \\ X \end{bmatrix} \qquad \delta_{22} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \\ 0 \\ 0 \\ X \\ X \end{bmatrix} \\
\delta_{11} &= \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \\ 1 \\ 0 \\ X \\ X \end{bmatrix} \qquad \delta_{21} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ \\ 0 \\ 1 \\ X \\ X \end{bmatrix}
\end{aligned}\right\} \tag{25}$$

Assume that R-S flip-flops will be used. Then let R_p and S_p represent the reset and set input equations for flip-flop A_p. Also let τ_{Rp} and τ_{Sp} represent the reset and set pulse inputs, and let R_{pE} and S_{pE} represent the reset and set enable inputs. The input equations in column matrix form are as follows:

$$R_3 = \begin{bmatrix} X \\ X \\ X \\ 0 \\ \\ 0 \\ 1 \\ X \\ X \end{bmatrix} \quad S_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \\ X \\ 0 \\ X \\ X \end{bmatrix} \quad R_2 = \begin{bmatrix} X \\ 0 \\ 0 \\ 1 \\ \\ X \\ X \\ X \\ X \end{bmatrix} \quad S_2 = \begin{bmatrix} 0 \\ 1 \\ X \\ 0 \\ \\ 0 \\ 0 \\ X \\ X \end{bmatrix} \quad R_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ \\ 0 \\ 1 \\ X \\ X \end{bmatrix} \quad S_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \\ 1 \\ 0 \\ X \\ X \end{bmatrix} \tag{26}$$

Note that the last two elements of all input column matrices are "X" and correspond to "forbidden" states of the counter. The other "X's" result from the fact that a flip-flop in a set (reset) state may be set (reset) repeatedly without changing its state. For synchronous design, the input equations are therefore $R_1 = A_1$, $S_1 = \overline{A}_1$, $R_2 = A_2A_1$, $S_2 = \overline{A}_3\overline{A}_2\overline{A}_1$, $R_3 = A_3A_1$, and $S_3 = A_2A_1$. (See fig. 3 for a schematic diagram of the synchronous design.)

Step 5. Since A_1 is of lowest order, its inputs are the same as for the synchronous design. Only A_1 is properly ordered to provide pulse inputs to A_2. The Δ_2 matrix is then

$$\Delta_2 = \left[\delta_{11}\delta_{21}\right] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \\ 1 & 0 \\ 0 & 1 \\ X & X \\ X & X \end{bmatrix} \tag{27}$$

Step 6. Applying equation (20) to R_2 and Δ_2 gives

$$\Lambda(R_2) = R_2 \odot \Delta_2 = \begin{bmatrix} X \\ 0 \\ 0 \\ 1 \\ \\ X \\ X \\ X \\ X \end{bmatrix} \odot \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \\ 1 & 0 \\ 0 & 1 \\ X & X \\ X & X \end{bmatrix} = \begin{bmatrix} X & X \\ X & 0 \\ 0 & X \\ X & 1 \\ \\ X & X \\ X & X \\ X & X \\ X & X \end{bmatrix} = \left[\lambda_{11}(R_2),\ \lambda_{21}(R_2)\right] \tag{28}$$

Step 7. Comparing R_2 with $\Lambda(R_2)$, it is seen that $\lambda_{21}(R_2)$ contains a "1" in the row where R_2 contains a "1", satisfying the realizability requirements.

Step 8. Only $\lambda_{21}(R_2)$ has a "1" corresponding to the "1" in R_2. Consequently only δ_{21} implies R_2, and furthermore it implies R_2 completely.

Step 9. There are no undesired "1's" in $\lambda_{21}(R_2)$, nor is redundant implication a problem.

Step 10. Assume that pulse inputs respond to α transitions. Since δ_{21} exhibits the β transition space of A_1, the negated output of A_1

must be connected to the pulse input of R_2 in order to supply the required α transition. The function $\lambda_{21}(R_2)$ is the enabling level to feed the enable input of R_2.

Step 11. The simplest expression for $\lambda_{21}(R_2)$ is A_2.

Then

$$\tau_{R2} = \bar{A}_1 \quad \text{and} \quad R_{2E} = A_2 \tag{29}$$

Step 12. Steps 5 through 11 are now repeated for input S_2. Applying equation (20) to S_2 and Δ_2 gives

$$\Lambda(S_2) = S_2 \odot \Delta_2 = \begin{bmatrix} 0 \\ 1 \\ X \\ 0 \\ \\ 0 \\ 0 \\ X \\ X \end{bmatrix} \odot \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \\ 1 & 0 \\ 0 & 1 \\ X & X \\ X & X \end{bmatrix} = \begin{bmatrix} 0 & X \\ X & 1 \\ X & X \\ X & 0 \\ \\ 0 & X \\ X & 0 \\ X & X \\ X & X \end{bmatrix} = \left[\lambda_{11}(S_2), \lambda_{21}(S_2)\right] \tag{30}$$

Since only $\lambda_{21}(S_2)$ has a "1" corresponding to the "1" in S_2, only δ_{21} can imply S_2. Then the pulse input τ_{S2} must be connected to the negated output of A_1. The enable input S_{2E} receives the function $\lambda_{21}(S_2)$ which in its simplest form is equal to $\bar{A}_3\bar{A}_2$. Then

$$\tau_{S2} = \bar{A}_1 \quad \text{and} \quad S_{2E} = \bar{A}_3\bar{A}_2 \tag{31}$$

Both A_1 and A_2 are properly ordered to provide pulse inputs to A_3. Hence Δ_3 becomes

$$\Delta_3 = \left[\delta_{11}, \delta_{21}, \delta_{12}, \delta_{22}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ X & X & X & X \\ X & X & X & X \end{bmatrix} \tag{32}$$

Steps 5 through 11 are repeated for input R_3. Applying equation (20) to R_3 and Δ_3 gives

$$\Lambda(R_3) = R_3 \odot \Delta_3 = \begin{bmatrix} X \\ X \\ X \\ 0 \\ 0 \\ 1 \\ X \\ X \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ X & X & X & X \\ X & X & X & X \end{bmatrix} = \begin{bmatrix} X & X & X & X \\ X & X & X & X \\ X & X & X & X \\ X & 0 & X & 0 \\ 0 & X & X & X \\ X & 1 & X & X \\ X & X & X & X \\ X & X & X & X \end{bmatrix}$$

$$= \left[\lambda_{11}(R_3),\ \lambda_{21}(R_3),\ \lambda_{12}(R_3),\ \lambda_{22}(R_3)\right] \qquad (33)$$

Since only $\lambda_{21}(R_3)$ has a "1" corresponding to the "1" in R_3, only δ_{21} can imply R_3. Then the pulse input τ_{R3} must be connected to $\bar{A}_1$. The enable input R_{3E} receives $\lambda_{21}(R_3)$ which in its simplest form can be either A_3 or $\bar{A}_2$. Then

$$\tau_{R3} = \bar{A}_1 \quad \text{and} \quad R_{3E} = A_3 \text{ or } \bar{A}_2 \qquad (34)$$

Finally steps 5 through 11 are executed for input S_3. Applying equation (20) to S_3 and Δ_3 gives

$$\Lambda(S_3) = S_3 \odot \Delta_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ X \\ 0 \\ X \\ X \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ X & X & X & X \\ X & X & X & X \end{bmatrix} = \begin{bmatrix} 0 & X & X & X \\ X & 0 & 0 & X \\ 0 & X & X & X \\ X & 1 & X & 1 \\ X & X & X & X \\ X & 0 & X & X \\ X & X & X & X \\ X & X & X & X \end{bmatrix}$$

$$= \left[\lambda_{11}(S_3),\ \lambda_{21}(S_3),\ \lambda_{12}(S_3),\ \lambda_{22}(S_3)\right] \qquad (35)$$

Both $\lambda_{21}(S_3)$ and $\lambda_{22}(S_3)$ contain a "1" corresponding to the "1" in S_3; hence either δ_{21} or δ_{22} may imply S_3. Then the pulse input τ_{S3} may be connected to either $\bar{A}_1$ or $\bar{A}_2$. The enable input S_{3E} may receive either $\lambda_{21}(S_3)$ or $\lambda_{22}(S_3)$ equaling A_2 and "1", respectively. The simplest solution is then

$$\tau_{S3} = \bar{A}_2 \quad \text{and} \quad S_{3E} = 1 \tag{36}$$

The application of the algorithm is now completed. The asynchronous design is shown in figure 4. Comparing figure 3 with figure 4 shows the increased simplicity achieved by the asynchronous design.

CONCLUDING REMARKS

A formalized algorithm for synthesizing asynchronous sequential circuits has been developed which replaces previous trial and error methods with an organized approach. The algorithm applies to all sequential circuits except those having pulse feedback. It permits determination of the realizability of an asynchronous implementation for a given sequential circuit and, furthermore, permits determination of the simplest solution if several exist. The matrix method of function representation is readily programable on a digital computer. A computer will be desirable for consideration of cases having a large number of variables.

Langley Research Center,
National Aeronautics and Space Administration,
Langley Station, Hampton, Va., September 15, 1965,
125-24-03-05-23.

REFERENCES

1. Phister, Montgomery, Jr.: Logical Design of Digital Computers. John Wiley & Sons, Inc., 1961.

2. [Mergler, H. W.]: Notes on Digital Control Systems Engineering. Vols. 1 and 2, Eng. Div. of Case Inst. Technol.

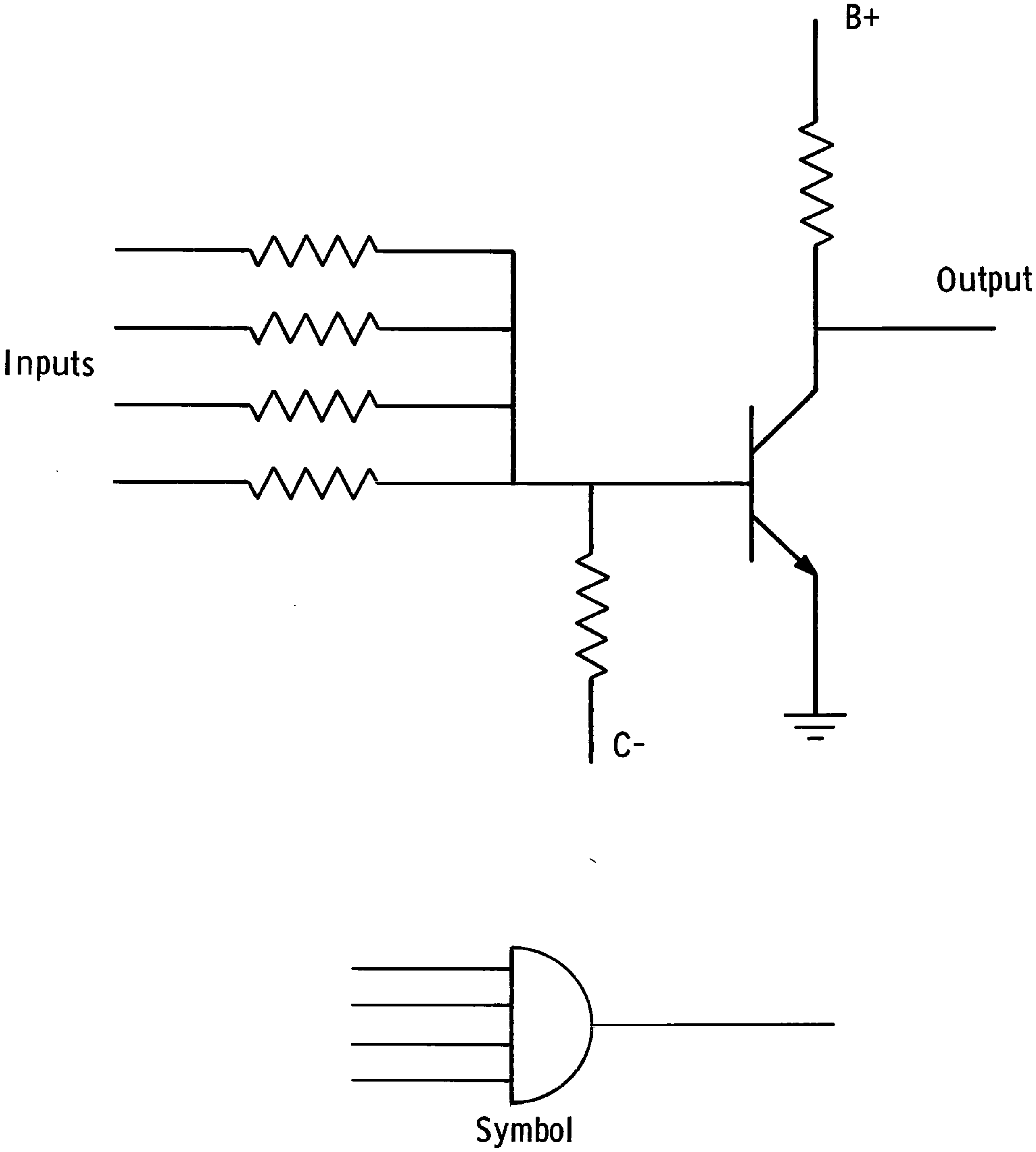

Figure 1.- Typical resistance-coupled inverting gate.

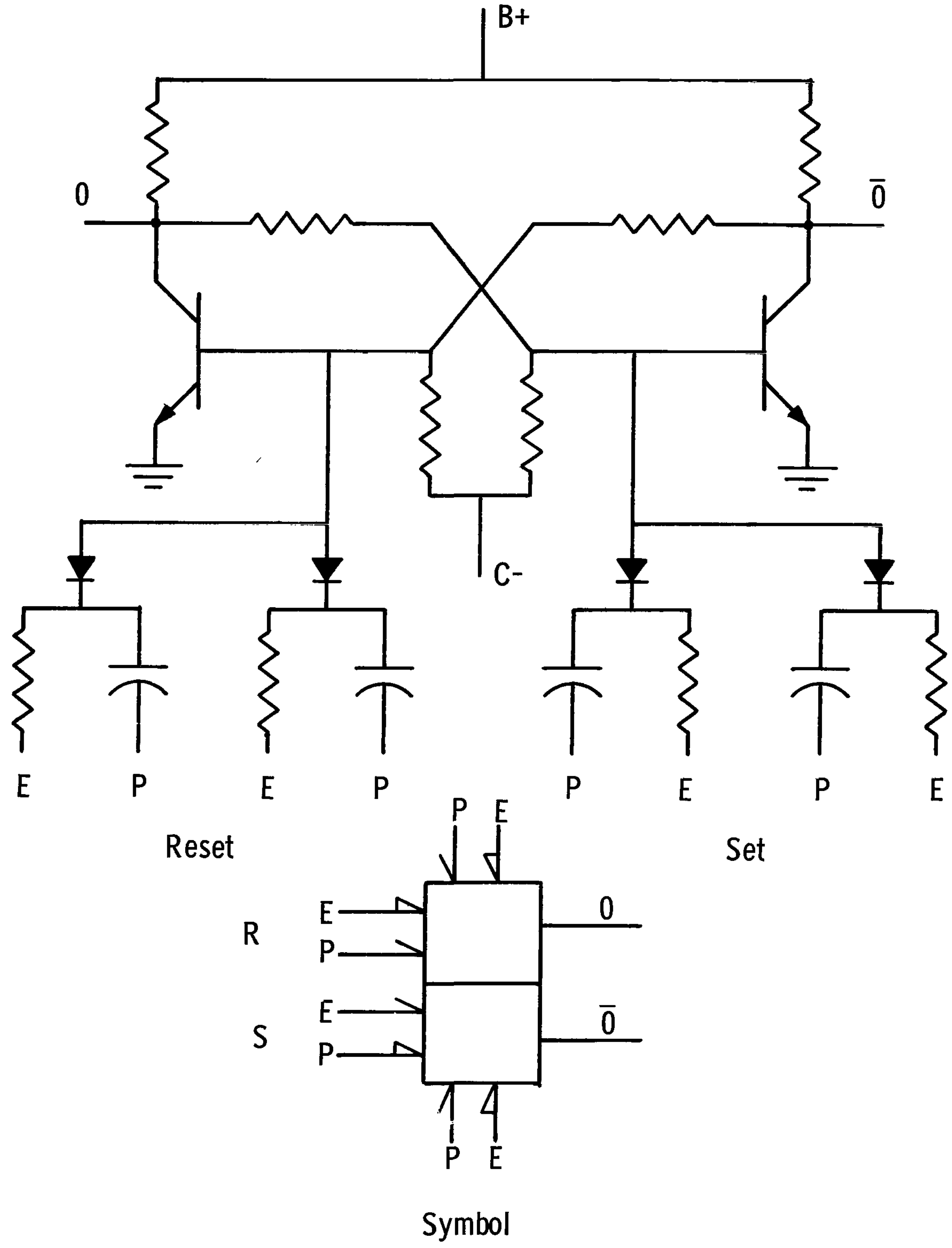

Figure 2.- Typical ac coupled R-S flip-flop circuit.

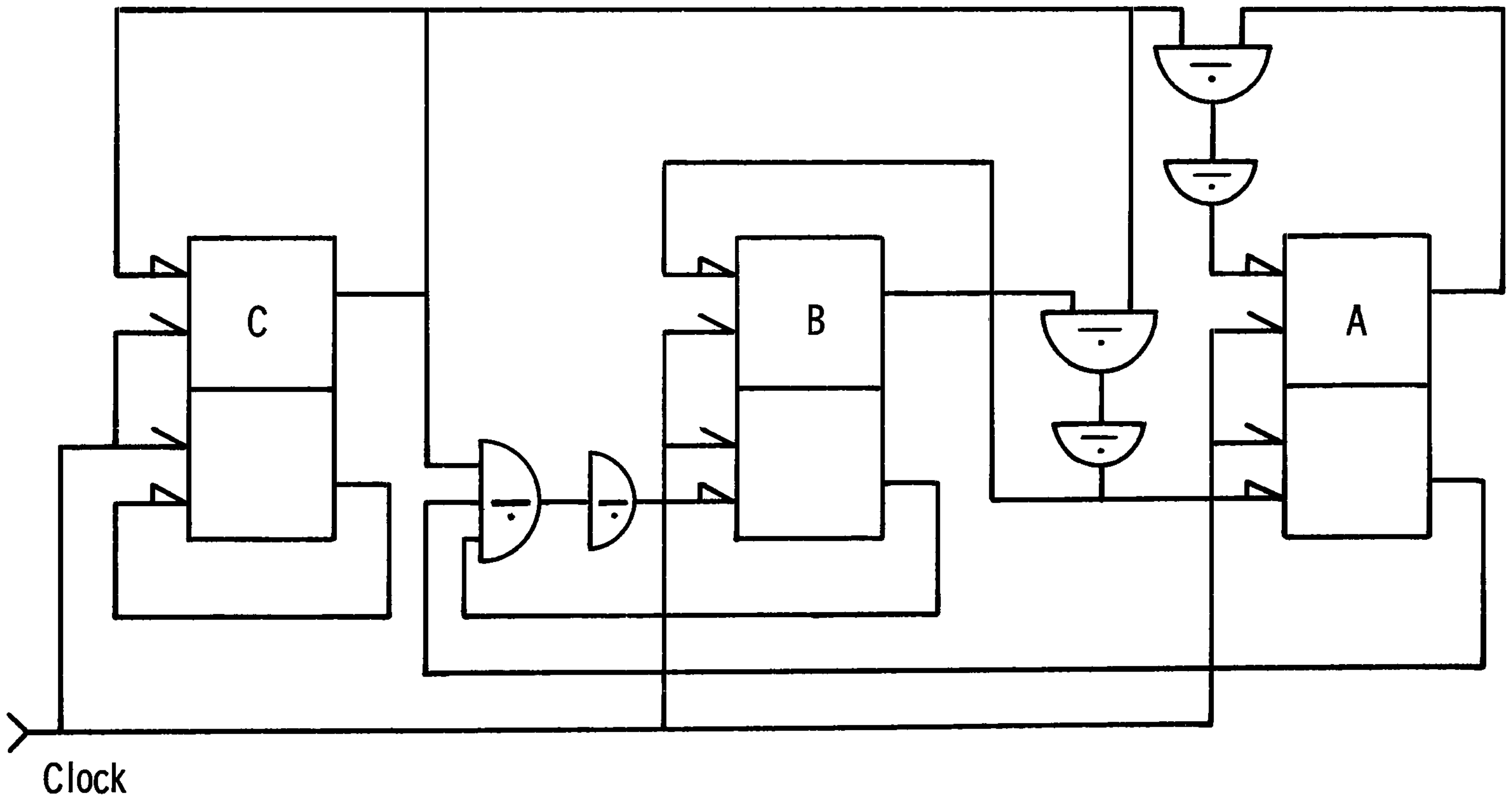

Figure 3.- Synchronous design of six-state counter.

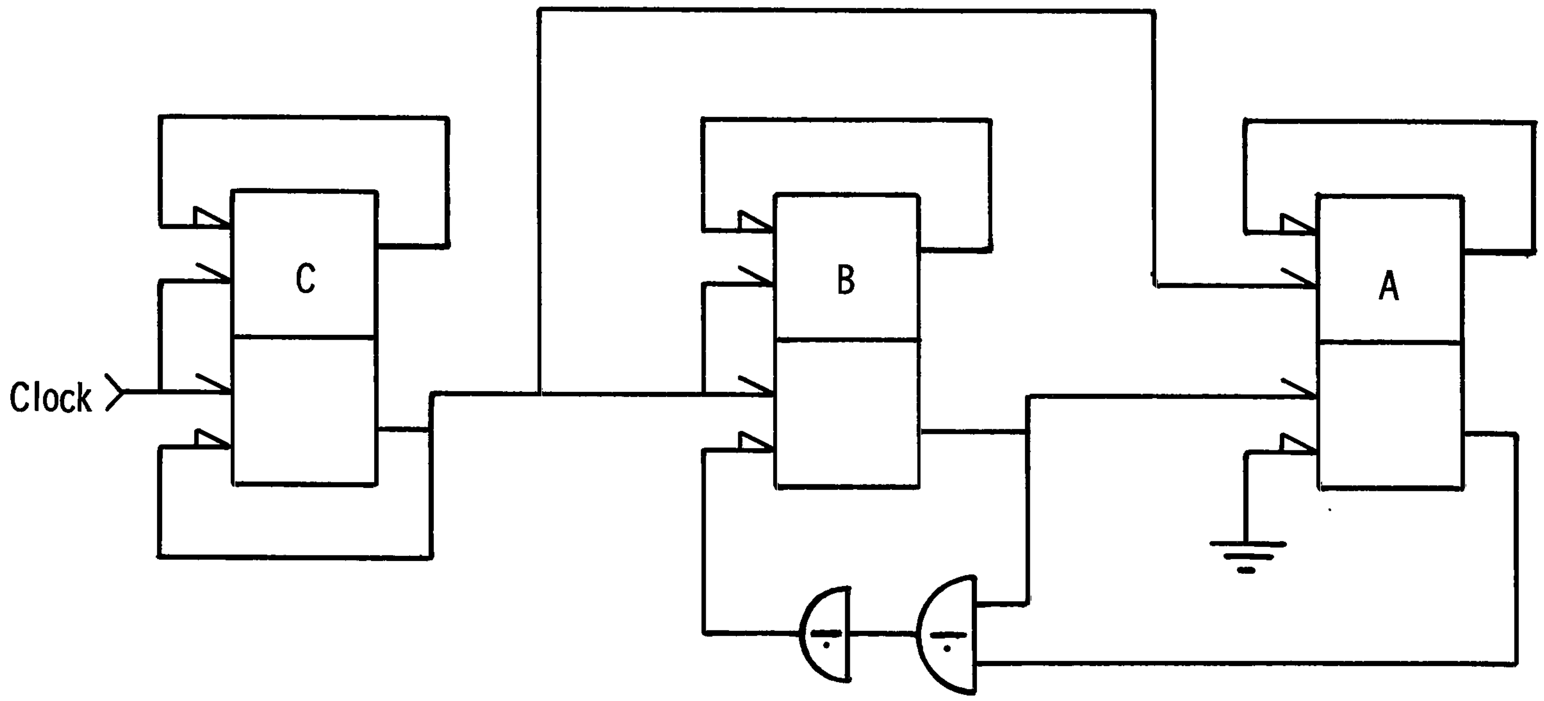

Figure 4.- Asynchronous design of six-state counter.

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Technical information generated in connection with a NASA contract or grant and released under NASA auspices.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

TECHNICAL REPRINTS: Information derived from NASA activities and initially published in the form of journal articles.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities but not necessarily reporting the results of individual NASA-programmed scientific efforts. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION DIVISION

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546