AN ALGORITHM FOR SYNTHESIS OF ASYNCHRONOUS SEQUENTIAL CIRCUITS

by John S. Tripp

Langley Research Center
Langley Station, Hampton, Va.
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

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 $Q_p$

$K(\Lambda'(I_p))$ 
cardinality of the set $\Lambda'(I_p)$

$m_k$ 
kth minterm

$p$ 
flip-flop pulse input

$Q_{kn}$ 
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
\( x_m \)  
\( m \)th system input

\( \alpha, \beta \)  
transition variables

\( \Delta_p \)  
the set of transition functions of order less than \( p \), and the corresponding matrix

\( \delta_{1k} \)  
\( \alpha \) transition function from \( \delta Q_k \) transition equation

\( \delta_{2k} \)  
\( \beta \) transition function from \( \delta Q_k \) transition equation

\( \delta Q_k \)  
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 \( \delta_{ij} \) for input \( I_p \)

\( \lambda'_{ij}(I_p) \)  
an element of \( \Lambda'(I_p) \)

\( \tau_{I_p} \)  
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"

. "and"; intersection

: nand

\( \bigcirc \)  
special operator

\( \Sigma \)  
disjunction

\( \epsilon \)  
is an element of
set inclusion

≤

is less than

<

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

\[ Q^{n+1} = S + Q^n R \tag{1} \]

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

\[ Q^{n+1} = f(Q^n, x_1, \ldots, x_m) \tag{2} \]
with the constraint that \( H(q^n, x_1, \ldots, x_m) = 0 \), 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

\[
q_i^{n+1} = g_i(q_1^n, \ldots, q_r^n, x_1, \ldots, x_m)
\]  

(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 \( \alpha \) transition has occurred, and the \( \alpha \) transition function \( \delta_{1k} \) equals 1; otherwise \( \delta_{1k} \) equals 0. Correspondingly, if \( Q_k \) goes from "1" to "0", a \( \beta \) transition has occurred and the \( \beta \) transition function \( \delta_{2k} \) equals 1; otherwise \( \delta_{2k} \) equals 0. It is seen that \( \delta_{1k} \) and \( \delta_{2k} \) are mutually exclusive.

As shown in reference 1, the general application equation is

\[
Q_k^{n+1} = g_{1k} Q_k^n + g_{2k} Q_k^n
\]  

(4)
where $g_{1k}$ and $g_{2k}$ are independent of $Q_k^n$. Truth equations for $g_{1k}$ and $g_{2k}$ are now derived from the application equation by means of the following truth table:

<table>
<thead>
<tr>
<th>$g_{1k}$</th>
<th>$g_{2k}$</th>
<th>$Q_k^n$</th>
<th>$Q_k^{n+1}$</th>
<th>$g_{1k}$</th>
<th>$g_{2k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

$$\delta Q_k = g_{2k}Q_k^{n} + \overline{g_{1k}}Q_k^{n} = \delta_{1k}^{\alpha} + \delta_{2k}^{\beta}$$

(5)

Since $\delta_{1k}$ and $\delta_{2k}$ are mutually exclusive, $\delta Q_k$ will equal either $\alpha$, $\beta$, or 0, indicating the occurrence of an $\alpha$ transition when $\delta Q_k = \alpha$, or the occurrence of a $\beta$ transition when $\delta Q_k = \beta$. The transition equation $\delta Q_k$ thus is nonzero for those states in the switching sequence at which $Q_k$ undergoes $\alpha$ or $\beta$ transitions. The transition function $\delta_{1k}$ ($\delta_{2k}$) is an algebraic representation of the set of states at which $\alpha$ ($\beta$) transitions occur. Any implicant (ref. 1) of $\delta_{1k}$ ($\delta_{2k}$) may be selected by intersecting $\delta_{1k}$ ($\delta_{2k}$) 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 a, transitions. Then if an implicant of the a 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

\[
I_p = f_p(Q_1, \ldots, Q_r, x_1, \ldots, x_m)
\]  

(6)
and let the set of lower order transition equations for implication of \( I_p \) be

\[
\delta Q_1 = \delta_{+1} + \delta_{-1} \delta_1 = \delta_1 + \delta_2 \\
\vdots \\
\delta Q_r = \delta_{+r} \delta_r + \delta_{-r} \delta_r = \delta_1 + \delta_2
\]

If \( \delta_{ij} \) is intersected with \( I_p \), the common portion will remain. If \( I_p \) is intersected with each element of the set of all \( \delta_{ij} \)'s, a comparison of the set of intersections with \( I_p \) will show which implicants of the \( \delta_{ij} \)'s are implicants of \( I_p \) and whether \( I_p \) can be completely implied by implicants of the \( \delta_{ij} \)'s. Given the necessary implicants from the \( \delta_{ij} \)'s to imply \( I_p \) completely, for each selected \( \delta_{ij} \), a function \( \lambda_{ij}(I_p) \) must be determined which, when intersected with \( \delta_{ij} \), will yield the desired implicant from \( \delta_{ij} \). Comparison of all \( \delta_{ij} \) . \( I_p \) 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 \( \delta_{ij} \) and \( I_p \), there exists a function, \( \lambda_{ij}(I_p) \), where

\[
\lambda_{ij}(I_p) \cdot \delta_{ij} = I_p \cdot \delta_{ij}
\]

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 \( \lambda_{ij}(I_p) = \overline{\delta_{ij}} + \delta_{ij}I_p \)
Then

\[ \lambda_{ij}(I_p) \cdot \delta_{ij} = (\overline{\delta_{ij}D} + \delta_{ij}I_p) \cdot \delta_{ij} = \delta_{ij}I_p \]

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

(a) Choose \( D = I_p \), then

\[ \lambda_{ij}(I_p) = \overline{\delta_{ij}I_p} + \delta_{ij}I_p = I_p. \]

Thus \( \lambda_{ij}(I_p) \) 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

\[ \lambda_{ij}(I_p) = \overline{\delta_{ij}} + \delta_{ij} = 1. \]

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

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

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

Theorem 2.

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

\[ \lambda_{ij}(I_p) = \overline{\delta_{ij}D} + \delta_{ij}I_p \]

(9)

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

Proof:

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

<table>
<thead>
<tr>
<th>( \delta_{ij} )</th>
<th>( I_p )</th>
<th>( I_p \cdot \delta_{ij} )</th>
<th>( \lambda_{ij}(I_p) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Hence $\lambda_{ij}(l_p) = \delta_{ij}^D + \delta_{ij}l_p$ 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 $\lambda_{ij}(l_p)$ in the variables involved.

**MATRIX PROCEDURE**

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

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

(10)

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

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

(11)

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

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

(12)
Order the set of memory element transition equations into a set 
\[ \{Q_1, Q_2, \ldots, Q_r \} \] such that for all \( i < j \) the number of "0's" in \( \delta Q_i \) is less than or equal to the number of "0's" in \( \delta Q_j \).

Ip represents an input to memory element \( Q_p \). Then let \( \Delta_p \) represent the set of \( \delta_{ij} \) transition functions associated with the set of memory elements of lower order than \( Q_p \).

\[ \Delta_p = \{ \delta_{11}, \delta_{21}, \delta_{12}, \delta_{22}, \ldots, \delta_{1(p-1)}, \delta_{2(p-1)} \} \]  

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

\[
\Delta_p = \begin{bmatrix}
\delta_{11,0} & \delta_{21,0} & \cdots & \delta_{1(p-1),0} & \delta_{2(p-1),0} \\
\delta_{11,1} & \delta_{21,1} & \cdots & \delta_{1(p-1),1} & \delta_{2(p-1),1} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\delta_{11,2^s-1} & \delta_{21,2^s-1} & \cdots & \delta_{1(p-1),2^s-1} & \delta_{2(p-1),2^s-1}
\end{bmatrix}
\]

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

For each \( \delta_{ij} \) contained in \( \Delta_p \), the set of common implicants of \( I_p \) and \( \delta_{ij} \) 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 \( \delta_{ij} \) 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 $\delta_{ij}$ function contained in $\Delta P$, yielding the set of functions

$$\Lambda(I_P) = \{\lambda_{11}(I_P), \lambda_{21}(I_P), \ldots, \lambda_{1(p-1)}(I_P), \lambda_{2(p-1)}(I_P)\}$$

(16)

where

$$\lambda_{ij}(I_P) = \delta_{ij}D + \delta_{ij}I_P$$

(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:

$$\Lambda(I_P) = \begin{bmatrix}
\lambda_{11,0}(I_P) & \lambda_{21,0}(I_P) & \ldots & \lambda_{1(p-1),0}(I_P) & \lambda_{2(p-1),0}(I_P) \\
\lambda_{11,1}(I_P) & \lambda_{21,1}(I_P) & \ldots & \lambda_{1(p-1),1}(I_P) & \lambda_{2(p-1),1}(I_P) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\lambda_{11,2^{s-1}}(I_P) & \lambda_{21,2^{s-1}}(I_P) & \ldots & \lambda_{1(p-1),2^{s-1}}(I_P) & \lambda_{2(p-1),2^{s-1}}(I_P)
\end{bmatrix}$$

(18)

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

$$\lambda_{i,j,k}(I_P) = \delta_{i,j,k} \cdot \chi + \delta_{i,j,k} \cdot I_P, k \quad (1 \leq i \leq 2, 1 \leq j \leq p-1, 1 \leq k \leq 2^s-1)$$

(19)

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

$$\Lambda(I_P) = \overline{\Delta P} \cdot D + \Delta P \cdot I_P = I_P \odot \Delta P$$

(20)

The "⊙" 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 $\Delta P$. Since each "1" element of $\Lambda(I_P)$ represents a common implicant of some $\delta_{ij}$ 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

\[
\Lambda'(I_p) = \{\lambda_{ab}'(I_p), \ldots, \lambda_{yz}'(I_p)\}
\]  

(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

\[
I_p = \sum_{\lambda_{ij}'(I_p) \in \Lambda'(I_p)} \lambda_{ij}'(I_p) \cdot \delta_{ij}
\]  

(22)

Now \( \delta_{ij} \) 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 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 \( \delta Q \). 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 \(\delta_{Q_j}\).

(4) For each transition equation \(\delta_{Q_p}\) determine the \(\alpha\) and \(\beta\) transition functions, \(\delta_{1p}\) and \(\delta_{2p}\), and the input equations \(I_p\), all in column matrix form.

(5) For each input \(I_p\) form a transition matrix \(\Delta_p\) whose column space consists of all \(\delta_{1j}\) 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 \(\Delta_p\) 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 \(\Lambda_p\).

(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 (\(\alpha\)) or 1 to 0 (\(\beta\)), 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:

<table>
<thead>
<tr>
<th>$A_3^n$</th>
<th>$A_2^n$</th>
<th>$A_1^n$</th>
<th>$A_3^{n+1}$</th>
<th>$A_2^{n+1}$</th>
<th>$A_1^{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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

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

(23)

and the transition equations are

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

(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 $\alpha$ and $\beta$ transition functions are

$$\delta_{13} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & X \\ X & X \end{bmatrix} \quad \delta_{23} = \begin{bmatrix} 0 & 0 \\ 0 & 1 & X \\ X \end{bmatrix}$$

$$\delta_{12} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & X \\ X & X \end{bmatrix} \quad \delta_{22} = \begin{bmatrix} 0 & 0 \\ 0 & 1 & X \\ X \end{bmatrix}$$

$$\delta_{11} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ X & X \end{bmatrix} \quad \delta_{21} = \begin{bmatrix} 0 \\ 1 & 0 & 1 \\ X \end{bmatrix}$$

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 $r_{R_p}$ and $r_{S_p}$ 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:

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

(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 \( R_1 = A_1 \), \( S_1 = \overline{A}_1 \), \( R_2 = A_2 A_1 \), \( S_2 = \overline{A}_3 \overline{A}_2 \overline{A}_1 \), \( R_3 = \overline{A}_2 A_1 \), and \( S_3 = \overline{A}_2 A_1 \). (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 \( \Delta_2 \) matrix is then

\[
\Delta_2 = \begin{bmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
X & X \\
X & X
\end{bmatrix}
\]

Step 6. Applying equation (20) to \( R_2 \) and \( \Delta_2 \) gives

\[
\Lambda(R_2) = R_2 \odot \Delta_2 = \begin{bmatrix}
X \\
0 \\
1 \\
X \\
X \\
X
\end{bmatrix} \odot \begin{bmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
X & 1 \\
X & X \\
X & X
\end{bmatrix} = \begin{bmatrix}
X & X \\
X & 0 \\
0 & X \\
X & X \\
X & X \\
X & X
\end{bmatrix}
\]

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 \( \delta_{21} \) 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 \( \alpha \) transitions. Since \( \delta_{21} \) exhibits the \( \beta \) 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 \( a \) 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

\[
\tau_{R2} = \overline{A}_1 \quad \text{and} \quad R_{2E} = A_2 \quad (29)
\]

Step 12. Steps 5 through 11 are now repeated for input \( S_2 \). Applying equation (20) to \( S_2 \) and \( \Delta_2 \) gives

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

Since only \( \lambda_{21}(S_2) \) has a "1" corresponding to the "1" in \( S_2 \), only \( s_{21} \) can imply \( S_2 \). Then the pulse input \( \tau_{S2} \) 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 \( \overline{A_2A_2} \). Then

\[
\tau_{S2} = \overline{A}_1 \quad \text{and} \quad S_{2E} = \overline{A}_2 \overline{A_2} \quad (31)
\]

Both \( A_1 \) and \( A_2 \) are properly ordered to provide pulse inputs to \( A_3 \). Hence \( \Delta_3 \) becomes

\[
\Delta_3 = [s_{11}, s_{21}, s_{12}, s_{22}] = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 \\
x & x & x & x \\
x & x & x & x
\end{bmatrix} \quad (32)
\]
Steps 5 through 11 are repeated for input $R_3$. Applying equation (20) to $R_3$ and $\Delta_3$ gives

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

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

(33)

Since only $\lambda_{21}(R_3)$ has a "1" corresponding to the "1" in $R_3$, only $\delta_{21}$ can imply $R_3$. Then the pulse input $\tau_{R3}$ must be connected to $\overline{A_1}$. The enable input $R_{3E}$ receives $\lambda_{21}(R_3)$ which in its simplest form can be either $A_3$ or $\overline{A_2}$. Then

$$\tau_{R3} = \overline{A_1} \text{ and } R_{3E} = A_3 \text{ or } \overline{A_2}$$

(34)

Finally steps 5 through 11 are executed for input $S_3$. Applying equation (20) to $S_3$ and $\Delta_3$ gives

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

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

(35)

Both $\lambda_{21}(S_3)$ and $\lambda_{22}(S_3)$ contain a "1" corresponding to the "1" in $S_3$; hence either $\delta_{21}$ or $\delta_{22}$ may imply $S_3$. Then the pulse input $\tau_{S3}$ may be connected to either $\overline{A_1}$ or $\overline{A_2}$. 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
$$r_{s3} = \bar{A}_2 \quad \text{and} \quad S_{3E} = 1 \quad (36)$$

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


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