

[54] LOGICAL FUNCTION GENERATOR
[72] Inventor: Wilford E. Silvertson, Jr., Yorktown, V.I.
[73] Assignee: The United States of America as represented by the National Aeronautics and Space Administration
[22] Filed: Dec. 16, 1970
[21] Appl. No.: 98,798
[52] U.S. Cl. 235/152, 307/207, 307/215
[51] Int. Cl. G06f 7/38, G06f 15/34
[58] Field of Search 235/152; 307/215, 207; 340/172.5

3,579,119	5/1971	Yau et al.	307/207
3,596,075	7/1971	Van Weelden	307/207 X
3,538,443	11/1970	Tague	307/215 X
3,584,207	6/1971	Avsan et al.	307/215 X
3,458,240	7/1969	Hanson	307/207 X
3,576,984	5/1971	Gregg	307/207 X
3,341,823	9/1967	Connolly	340/172.5
3,262,101	7/1966	Halpern	340/172.5

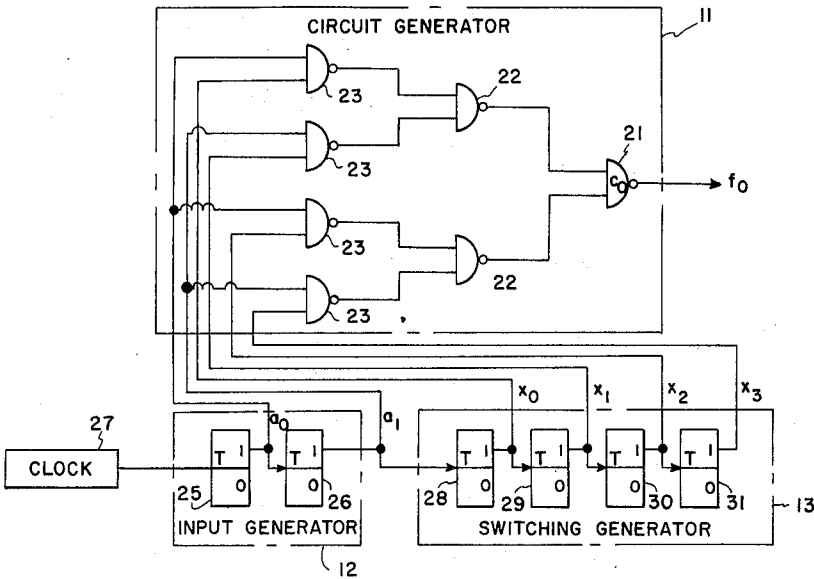
Primary Examiner—Malcolm A. Morrison
Assistant Examiner—James F. Gottman
Attorney—Howard J. Osborn, William H. King and John R. Manning

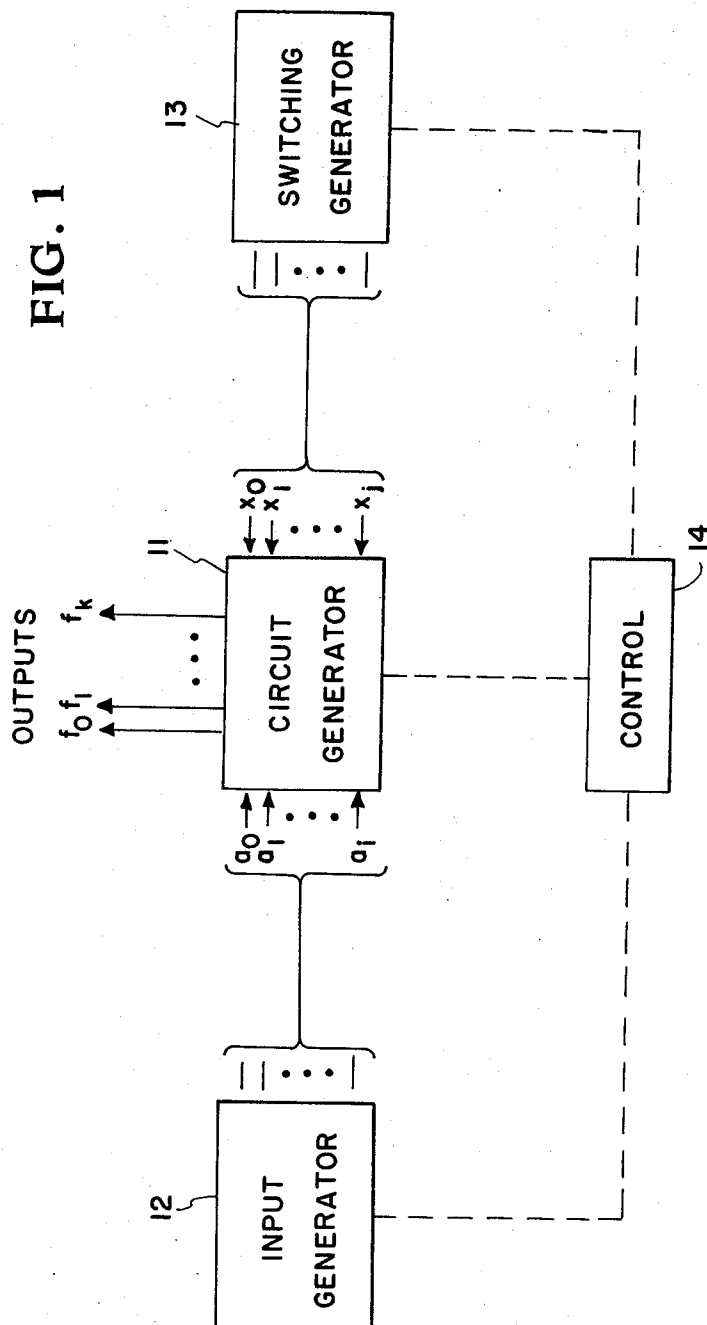
[56] References Cited
UNITED STATES PATENTS

3,566,359	2/1971	Connolly	340/172.5
3,381,117	4/1968	Forslund et al.	307/207 X

[57] ABSTRACT
Apparatus and technique for generating logical functions and circuits, and for defining the circuit connections required to generate these functions thereby providing an aid in designing and constructing hardware to generate logical circuits.

6 Claims, 8 Drawing Figures





INVENTOR.
WILFORD E. SIVERTSON, JR.

BY *Howard J. Osborn*
William H. King
ATTORNEYS

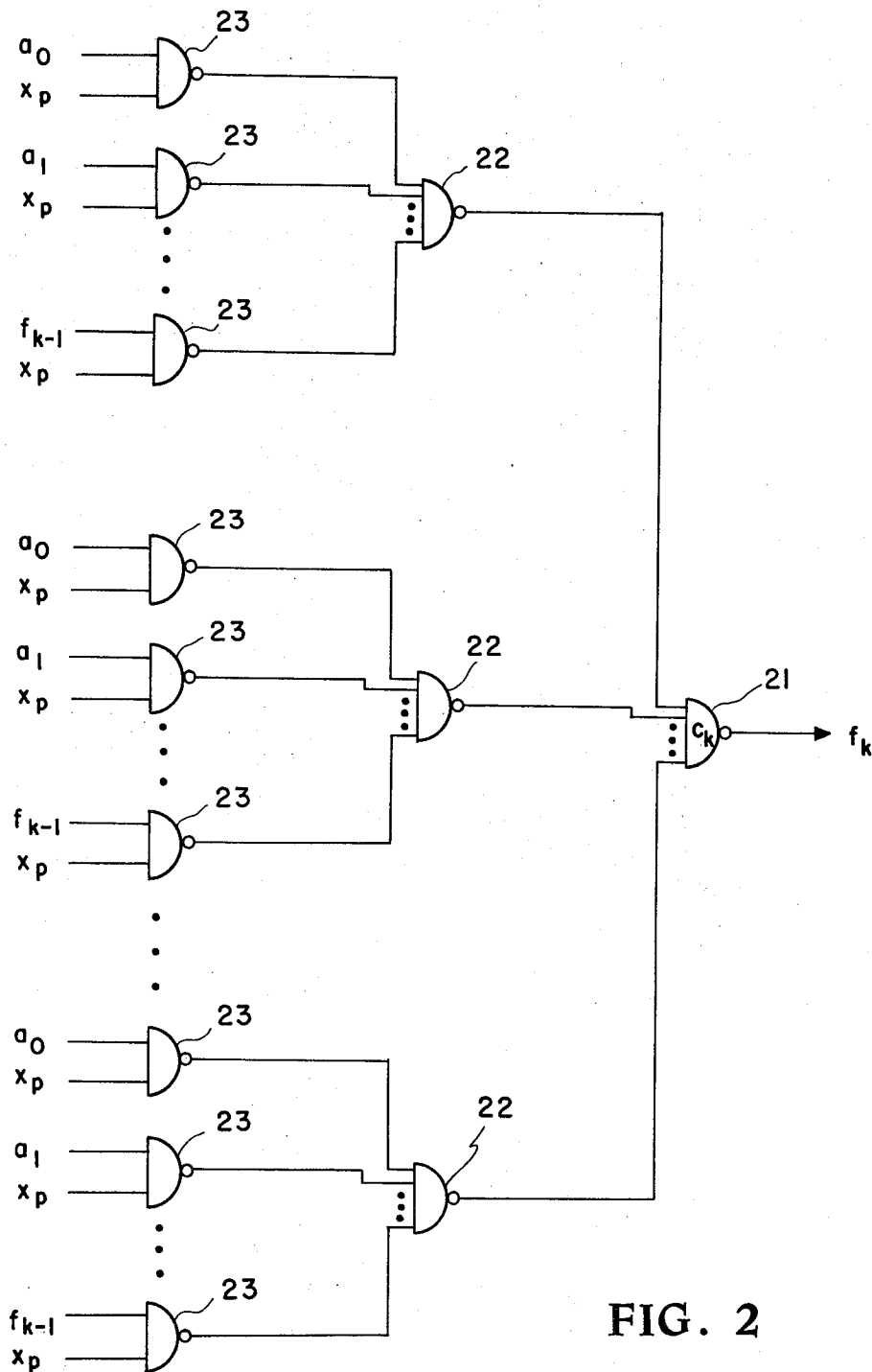


FIG. 2

INVENTOR.
WILFORD E. SIVERTSON, JR.

BY

Howard J. O'Brien
William H. King
ATTORNEYS

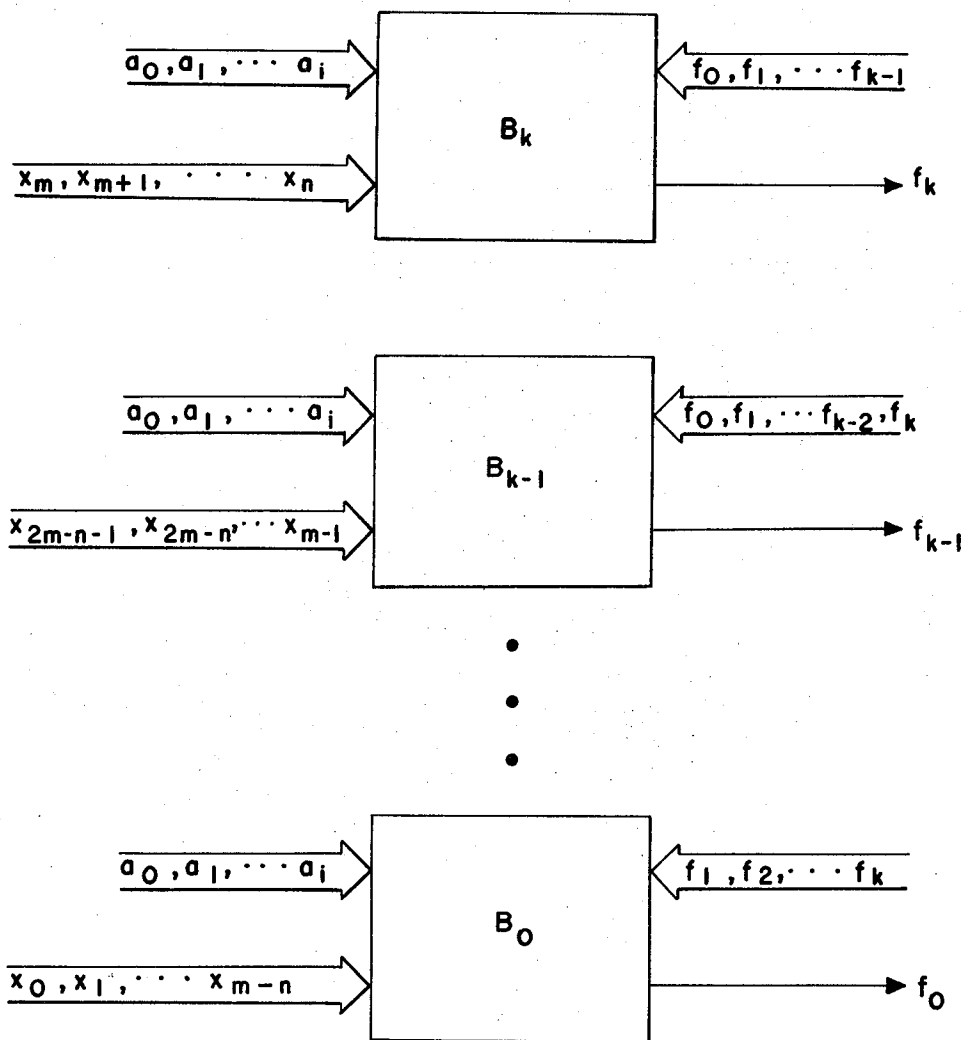


FIG. 3

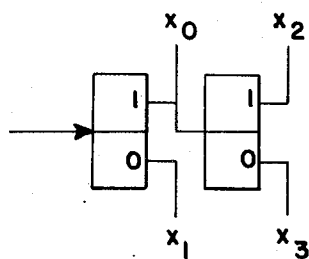
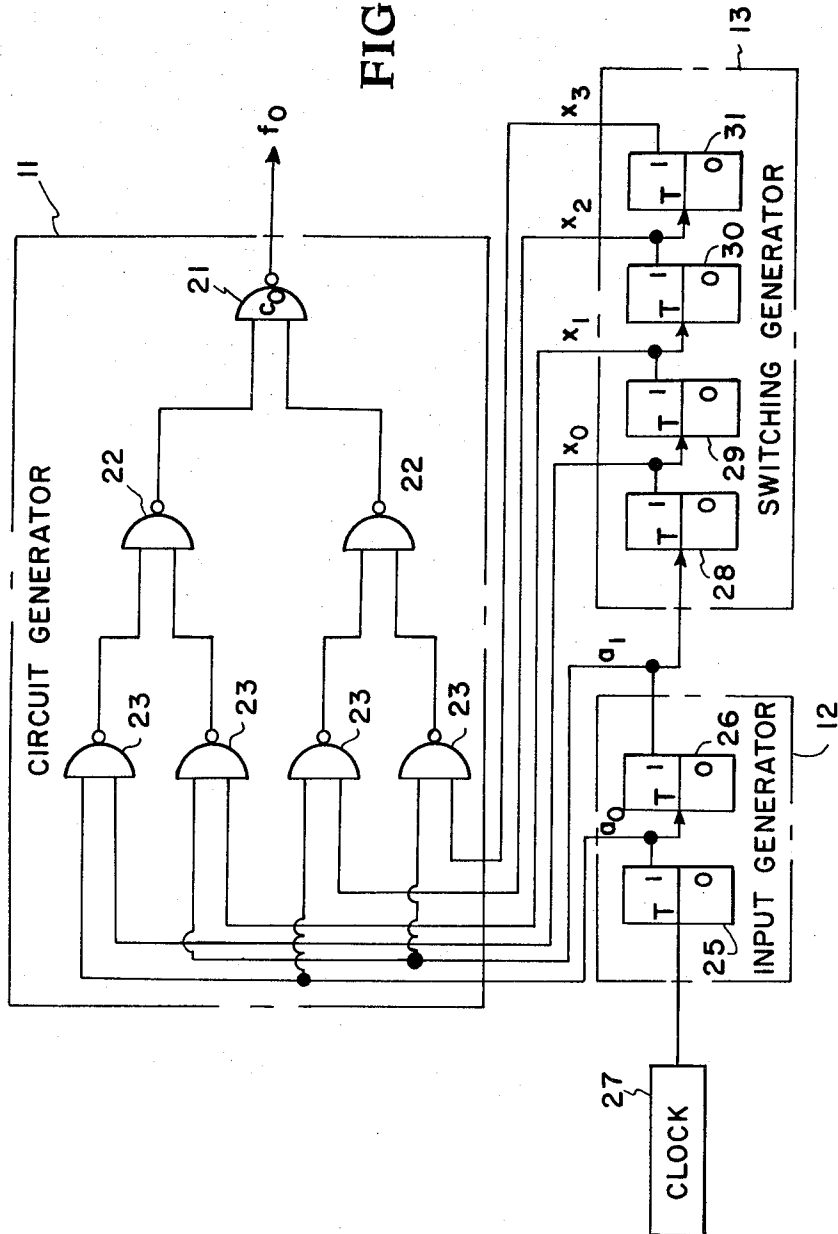


FIG. 5

INVENTOR.
WILFORD E. SIVERTSON, JR.
BY *Howard J. Osborn*
William H. King
ATTORNEYS

FIG. 4



INVENTOR.
WILFORD E. SIVERTSON, JR.

BY *Howard J. Osborn*
William H. King
ATTORNEYS

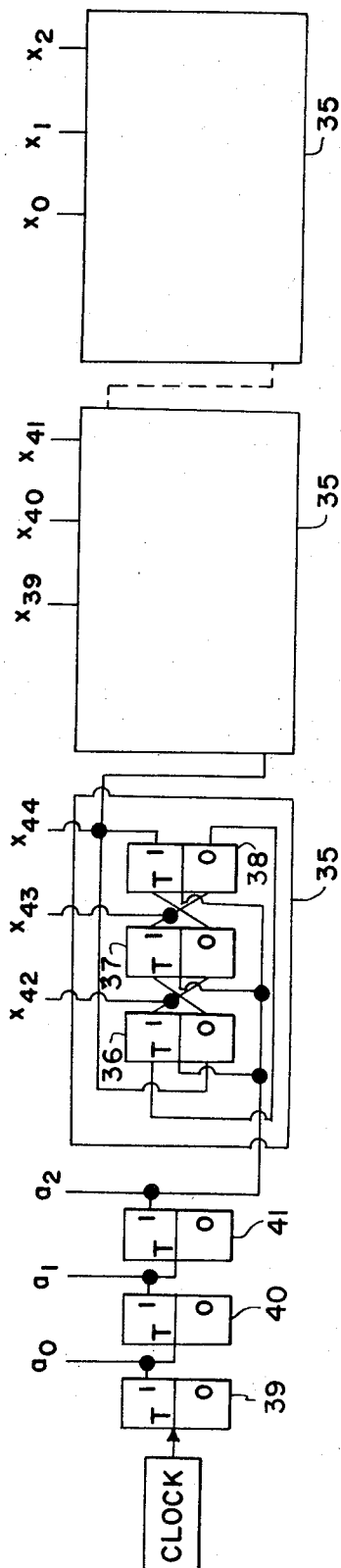


FIG. 6

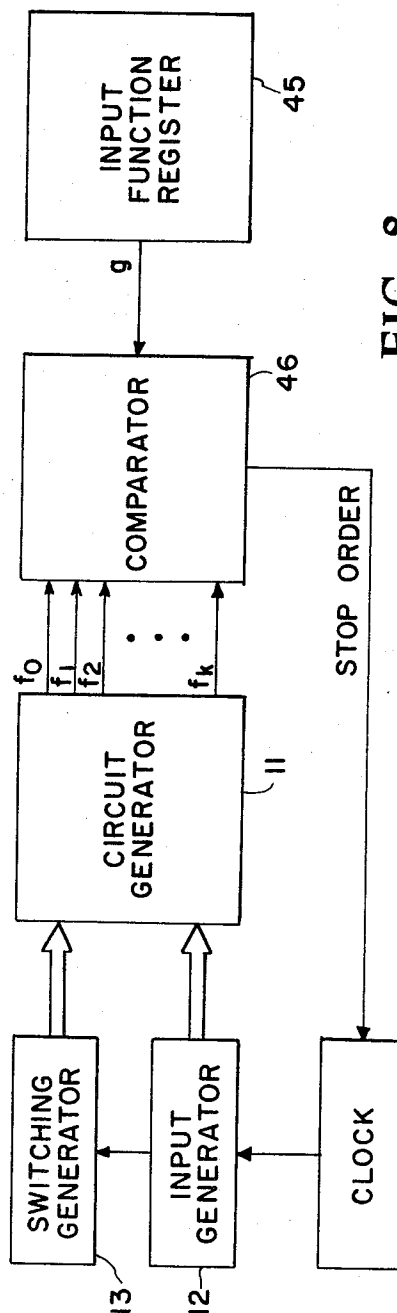


FIG. 8

INVENTOR.
WILFORD E. SIVERTSON, JR.

BY *Howard J. Osburn*
William H. King
ATTORNEYS

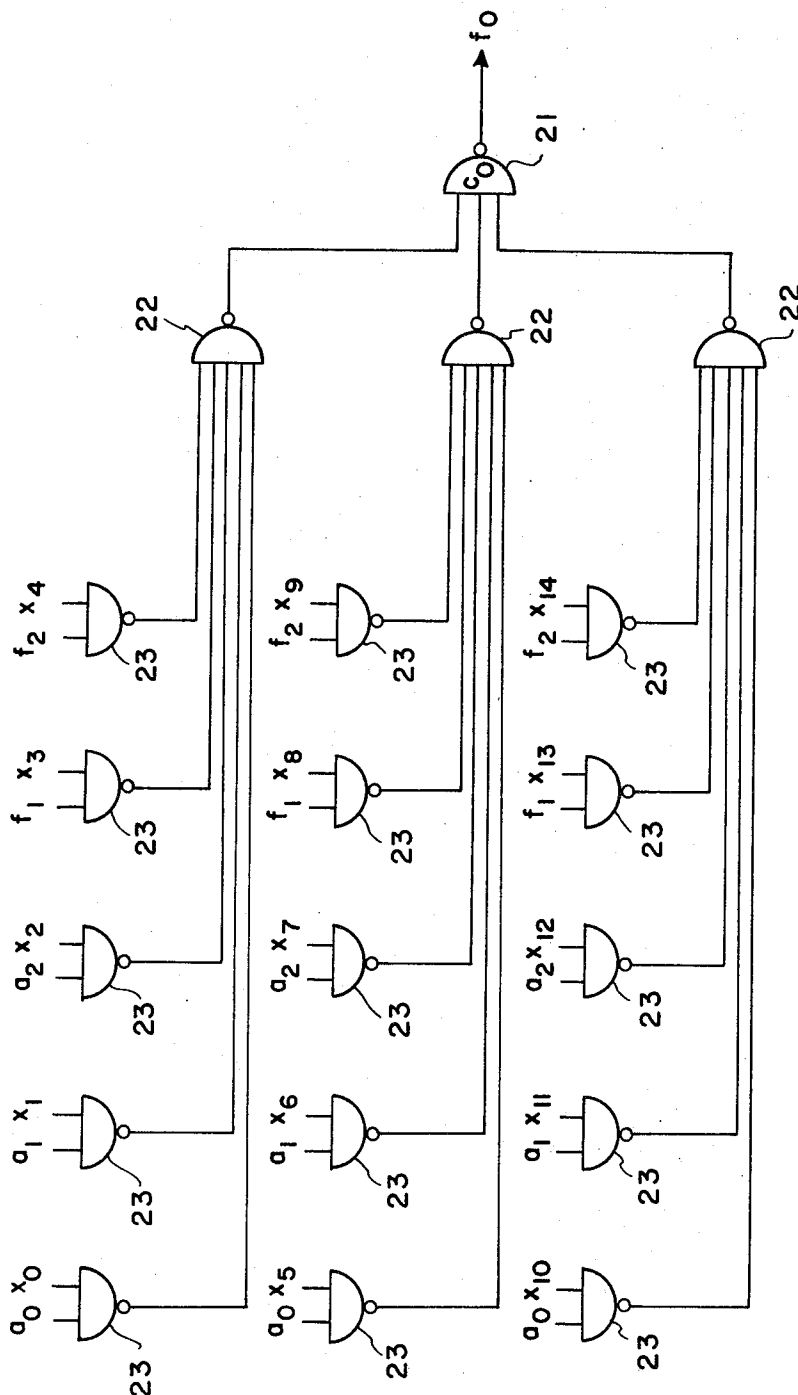


FIG. 7

INVENTOR.
WILFORD E. SIVERTSON, JR.

BY *Howard J. Osborn*
William H. King
ATTORNEYS

LOGICAL FUNCTION GENERATOR

ORIGIN OF THE INVENTION

The invention described herein was made by an employee of the United States Government and may be manufactured and used by or for the Government for governmental purposes without the payment of any royalties thereon or therefor.

BACKGROUND OF THE INVENTION

The invention relates generally to logical circuitry and more specifically concerns an aid in designing and constructing hardware to generate logical circuits.

Previous methods for designing logical circuits have relied on the mathematics of set theory and Boolean algebra; requiring the manipulation of equations until desired results are obtained. This is a time-consuming operation and incases where the simplest form is desired there is no known mathematical method for directly arriving at it. It is a trial and error process and relies on the skill of the individual and a knowledge of what will be accepted as simplest form. Defining what will be accepted as simplest form depends on many factors. The type of gate to be used as well as fan in, fan out, etc., must be considered in arriving at a reasonable definition of simplest form. In most cases experience and intuition provide the best approach to generating this simplification definition and in some cases a well defined one cannot be found. Simplification techniques have been developed and are used. These include the Quine, Harvard, and Nelson methods as well as a number of mapping techniques. These provide methods for special simplifications, but they are not schemes for arriving at the simplest form for all cases. One such case is the case for circuits employing the "NAND" and "NOR" logical connective. In these cases simplification procedures arrive at a form of simplification in terms of "AND" and "OR" and involves considerable manipulation if all functions of i variables and K connective elements are involved. This solution must then be converted to "NAND" or "NOR" form without regard to any alterations to simplest form that may result from this transformation. In a case where $K=7$ and $i=3$ (considering only combinational circuits) there exist approximately 4×10^{12} circuit arrangements. Anyone can be simplest form. To determine the correct circuit a method must be used that exhausts or/in some way accounts all 4×10^{12} circuits that are possible and isolates the one (or ones) appropriate. In the case of "NAND" - "NOR" implementation there exists no known mathematical technique for arriving at simplest form. This forces one to solve for all 4×10^{12} circuits in order to search for a simplest form. This places the problem beyond the time scale of any individual using manual techniques, and present day computers (after extensive programing) cannot provide a practical means for solution. It is therefore the primary purpose of this invention to provide practical means for generating logical functions and circuits.

SUMMARY OF THE INVENTION

This invention embodies the use of standard logical elements in the construction of a special purpose computer for use in automatically designing logic circuits. The Sheffer-stroke primitive and its dual the Pierce are

widely used as fundamental logical elements. For this reason the "NAND" logical element will be used to illustrate the technique with this invention. The use of "NAND" elements is not restrictive and it should be noted that any primitive can be used for constructing automatic logical function and circuit generating machines.

The basic scheme provides for switching to permute a number of input literals and recursive inputs into a number of logical connective elements. The outputs from these connective elements provide the set of all functions possible from these connective elements and input literals. In addition, monitoring the switching states provides the set of all logical circuit connections associated with the set of all generated functions with a one-to-many mapping existing between functions and circuits.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of this invention;

FIG. 2 is a schematic diagram of circuitry utilized in the circuit generator of this invention;

FIG. 3 is a block diagram showing how the circuitry in FIG. 2 is used to form the circuit generator;

FIG. 4 is a schematic drawing of a simple embodiment of this invention;

FIG. 5 is a schematic drawing showing how constraints are put on the switching generator in FIG. 4;

FIG. 6 is a block diagram and schematic drawing of an input generator having three outputs and a switching generator having 45 constrained outputs;

FIG. 7 is a schematic drawing showing how a circuit generator can be built to handle all of the outputs from input and switching generators in FIG. 6; and

FIG. 8 is a block diagram showing one way in which this invention can be used.

DETAILED DESCRIPTION OF THE INVENTION

Turning now to the embodiments of the invention selected for illustration in the drawings the number 11 designates a circuit generator. An input generator 12 generates all possible input combinations of the literals $a_0 - a_i$ which are applied to circuit generator 11 on $(i + 1)$ lines. A switching generator 13 generates steering signals $x_0 - x_j$ on $(j + 1)$ lines as inputs to circuit generator 11. These steering signals gate the input literals ($a_0 - a_i$) along with some connective element outputs into $(k + 1)$ logical connective elements (C_k) having outputs $f_0 - f_k$. One set of steering signals is generated for each input cycle. This provides for the generation of a different circuit for each input cycle; this process of steering continues until all circuits have been produced.

The circuit generator 11 supplies the logic required to generate different circuit connections as a function of the $x_0 - x_j$ switching signals. Control 14 provides timing and logical control for the system as required. The input generator 12 must produce and continue to regenerate input cycles. One cycle can be shown as an input matrix as follows:

a_0 0101 ...	0101
a_1 0011 ...	0011
...	...
...	...
a_i (2^i zeros)	(2^i ones)

where: rows represent designation numbers and columns represent minterms.

If designation numbers are visualized as representing literals and minterms as functions of position or time then the input matrix can be visualized as the output from a generator generating all minterms of $(i+1)$ literals as a function of time. Thus a one cycle output of the input generator is a time sequence of $2^{(i+1)}$ minterms. This can be mechanized in many ways. One simple method would use $(i+1)$ bistable elements $e_0 - e_i$ (such as flip flops) connected as a binary counter and driven from a clock. The outputs would be $a_0, a_1, \dots, a_i$ from one side of each element (e_i) respectively, and logical complements could be provided from the remaining sides.

The circuit generator must provide a means for connecting $(i+1)$ input lines to $(k+1)$ logical connective elements for handling combinational circuits, and for handling recursive circuits must also allow outputs from each connective element to feed back as inputs to any others. Thus if there are $(k+1)$ logical connective elements (such as "NAND" gates) and $(i+1)$ input literals there exist 2^b possible circuits that can be generated.

$$\text{Where: } b = i^2(k+1) + k^2(i+1) + k(3i+2) + 2i + 1 \quad (\text{Equation A})$$

It should be noted that all of the possible circuits indicated in Equation A are not desired. This will be covered later under a discussion of constraints.

The circuit generator (without considering constraints) must be capable of generating these circuits by an orderly switching of the output lines ($x_0 - x_i$) from the switching generator. This can be accomplished by generating an x_i switching matrix similar to the a_i input matrix. From this concept it follows that a counting register (driven at $2^{-(i+1)}$ times the a_i matrix rate) of $(j+1)$ bistable elements can be used where:

$$j \leq i^2(k+1) + k^2(i+1) + k(3i+2) + 2i \quad (\text{Equation B})$$

This will provide a switching generator output as a time sequence of $2^{(j+1)}$ minterms. One minterm for each $2^{(j+1)}$ input generator drive pulses. Each x_i matrix minterm represents a particular logical circuit connection producing outputs $f_0 - f_k$ as functions of $(i+1)$ logical input literals. Thus each output ($f_0, f_1, \dots, f_k$) is a time sequence of ones and zeros representing the presence or absence of the input matrix minterms according to the particular x_i switching matrix state existing for any given input cycle. From this it follows that the output lines $f_0, f_1, \dots, f_k$ contain (as a time sequence) information defining designation numbers of $f_0, f_1, \dots, f_k$ in terms of $a_0, a_1, \dots, a_i$. These outputs define the logical functions as designation numbers, while the state of the switching matrix defines the logical circuit producing those functions. From the designation number the function can be immediately written in logical form as a function of input literal, and from the switching matrix states the logic circuit can be immediately drawn and constructed. This readout can be manually obtained or obtained by various automatic means including automatic circuit drawing, wire table printout, and construction.

The circuit generator logic must provide a means for steering $(i+1)$ inputs to the $(k+1)$ logical connective elements under control of the x_i matrix. This requires

that each literal and a connective output be gated with an x_i output and that they be capable of being connected to all connective element input lines. Where the logical sum of product equation form holds there will be $(i+1)$ "OR" gates required for each logical element and $i^2 + i(k+2) + (k+1)$ "AND" gates required for each element. If "NAND" gates are used to construct this gating each logical connective element will require a circuit as shown in FIG. 2.

In FIG. 2 there is one c_k NAND gate 21 which has the outputs from $(i+1)$ NAND gates 22 applied to it. Each NAND gate 22 has the outputs from $i^2 + i(k+2) + (k+1)$ NAND gates 23 applied to it. A switching signal x_p is applied to an input of each of the NAND gates 23 where p successively increases by one from m through n starting at the top and where

$$m = k^2(i+1) + k(i+1)^2 \text{ and}$$

$$n = i^2(k+1) + k^2(i+1) + 2(k+i) + 3ik$$

The signals $a_0, a_1, \dots, a_i, f_0, f_1, \dots, f_{k-1}$ are applied to the other inputs of each group of NAND gates 23 that have their outputs applied to one of the NAND gates 22.

A circuit generator 11 contains $(k+1)$ circuits identical structurally to the one shown in FIG. 2. The only difference is the inputs applied to the NAND gates 23. For the circuit that produced f_{k-1} the subscript p of the switching signals x_p start at $2m - n - 1$ and goes through $m - 1$ and for the circuit that produces f_0 , p starts at 0 and goes through $m - n$. The functions applied to the NAND gates 23 are $f_0, f_1, \dots, f_k$ with the one output function being produced by NAND gate 21 omitted. FIG. 3 shows a circuit generator 11 in block diagram form where each of the blocks contains circuitry identical to that disclosed in FIG. 2. The blocks $B_k, B_{k-1}, \dots, B_0$ produce the functions $f_k, f_{k-1}, \dots, f_0$, respectively, and have the inputs shown applied to the different blocks.

To illustrate the operational use of a machine of this type consider the following elementary example. Let $i = 1$ and $k = 0$. Then following the procedure that has been discussed the system of FIG. 4 is obtained. The input generator 12 consists of flip-flops 25 and 26. A clock 27 produces pulses which are applied to the trigger of flip-flop 25. Hence an input cycle generated by input generator 12 is:

a_0 0101

a_1 0011

The switching generator 13 consists of four flip-flops 28-31 connected as shown. The switching generator produces switching signals x_0, x_1, x_2 and x_3 . The circuit generator 11 consists of four NAND gates 23, two NAND gates 22 and one NAND gate 21 with the $a_0, a_1, x_0, x_1, x_2, x_3$ signals applied to gates 23 as shown. The output from the switching generator 13 is the minterm sequence shown in the following Table I, one minterm for each input cycle with one designation number for f_0 for each input cycle. Table I also indicates the logical functions in terms of the input literals.

TABLE I

Input Cycle	Switching Minterm	Designation No. For f_0	$f_0(a_0, a_1)$
1	M_0	1111	1
2	M_1	1111	1
3	M_2	1111	1

4	M ₃	1111	1
5	M ₄	1111	1
6	M ₅	1010	a ₀
7	M ₆	1110	a ₀ + a ₁
8	M ₇	1010	a ₀
9	M ₈	1111	1
10	M ₉	1110	a ₀ + a ₁
11	M ₁₀	1110	a ₁
12	M ₁₁	1100	a ₁
13	M ₁₂	1111	1
14	M ₁₃	1010	a ₀
15	M ₁₄	1100	a ₁
16	M ₁₅	1000	a ₀ a ₁

Inspection of Table I will show an $f_0(a_0, a_1)$ that would not be obtained in the physical case using a single "NAND" gate 21. This is the function generated from M₁₅. This function is generated by the x_j matrix as a result of not having provided constraints in designing the switching matrix. The x_j matrix has not been constrained to restrict switching only to states generating physically realizable wire connections for the particular logical connective element being used. In the case cited the x_j matrix has simulated the connection of a_0 and a_1 to a single input line of NAND gate 21 effectively creating a hard wire logical "OR" connection. In practice this is not allowed. Also, $f_0(a_0, a_1)$ resulting from M₀, M₁, M₂, M₃, M₄, M₈ and M₁₂ are not useful circuits since they simulate a no-input condition for gate 21. To correct for these undesired circuit and function generations, constraints can be placed on the x_j matrix. In the cases shown if x_0 is forced to be the logical complement of x_1 and if x_2 is forced to be the logical complement of x_3 a proper constraint is obtained, and the results are shown in the following Table II.

TABLE II

Input Cycle	Switching Minterm	Designation No. for f_0	$f_0(a_0, a_1)$
1	M ₃	1010	a ₀
2	M ₆	1110	a ₀ + a ₁
3	M ₉	1110	a ₀ + a ₁
4	M ₁₀	1100	a ₁

The switching generator for generating the functions in Table II is shown in FIG. 5.

The idea of placing constraints on x_j provides a means for eliminating unnecessary circuit setups making useful function-circuit generation faster and more direct. Defining the nature of the constraint becomes a difficult task, and as i and k take on larger values the difficulty increases.

There should be a number of methods for obtaining useful constraints. One simple solution for $i = 1, k = 0$ has been shown where complement x_j terms were used. A solution, useful for any i and k value, can be found by using a combination shift and count register for the generation of the x_j matrix. FIG. 6 indicates one method for mechanizing this for a system with $i = 2$ and $k = 2$. This technique is not restricted by the magnitude of i and k but can be extended to include any combination of values.

As can be seen from Equation B, for $i = 2$ and $k = 2, j = 44$. Hence, the switching generator must generate 45 terms. This is accomplished in FIG. 6 by using fifteen separate circuits 35 connected in tandem as shown. Each circuit 35 consists of flip-flops 36, 37 and 38. a_2 is connected to the trigger of each of these flip-flops in the first 35 circuit. An initial condition for the flip-flops in each 35 circuit (shift register) is selected such that it forces flip-flop 36 to a "logical one" initial state and all

other flip-flops (37, 38) are set initially to the "logical zero" state. The switching signals x_{42}, x_{43} and x_{44} are taken at the "1" output of the flip-flops. Each of these flip flops is the type that when triggered its '1' output follows the logic applied to its '0' input. The input generator in FIG. 6 is the circuit consisting of flip-flops 39, 30 and 41. The B₀ suitable for use with the input and function generators in FIG. 6 is shown in FIG. 7. The B₁ and B₂ required for use with FIG. 6 are identical to the circuitry in FIG. 7 except that different inputs are applied to the NAND gates 23.

It can be shown that constraints should exist such that the limits for j are as shown in Equation C.

$$[i^2(k+1) + k^2(i+1) + k(3i+2) + 2i] \geq j \geq [k^2 + k(i+2) + i] \quad (\text{Equation C})$$

The idea of constraints on the x_j matrix has not been exhausted but briefly discussed to illustrate the nature of the technique. By using any of many known logical elements and the method discussed it has been shown that this invention provides a technique for automatically generating logical functions and their circuits.

One manner in which this invention is used can be understood by referring to FIG. 8. Suppose one wants to design a circuit that generates the function g with the least amount of hardware. Then he sets the values g in an input function register 45 and inputs this function to a comparator 46. He then starts the logic function circuit generator (11, 12, and 13). When one output function f is generated such that f_0 or f_1 or $\dots f_k = g$, comparator 46 generates a stop order that will stop further circuit generation. This can be done by having the stop order stop the clock. The output from the switching generator has structured and defines the connections necessary to design a circuit for generating g .

This procedure is repeated until the output of switching generator 13 generates all possible combinations of the switching signals for which circuit generator 11 generates an f that is equal to g . The designer can then select the circuit that requires the least hardware.

Another way in which this invention can be used is as a remote control device such as a control device on a spacecraft. Suppose that circuit generator 11 is on a spacecraft and suppose that it is desired that the generator produce g for controlling some function on the spacecraft. Then g is transmitted to the input register 45 on board the spacecraft. The logic function circuit generator is then started and as soon as f_0 or f_1 or $\dots f_k = g$ comparator 46 produces a stop function g and will continue to do so until it gets further orders.

This invention has several advantages over the prior art. It provides a means for quickly and automatically designing logic and eliminates involved and time-consuming mathematical manipulations. It eliminates the need for the use of expensive and large general purpose computing facilities for logic design. It eliminates lengthy, expensive, and difficult computer programming. It provides a method for directly finding minimal circuits for "NAND" - "NOR" logic although no known direct mathematical method exists. And it provides a means for quickly mapping functions to circuits and provides a tool for use in conducting research on minimal "NAND" - "NOR" form.

Even though only "NAND" elements have been disclosed it should be understood that other logical elements such as "AND," "OR" and "NOR" elements could be used without departing from the spirit and scope of this invention as defined in the following claims.

What is claimed as new and desired to be secured by Letters Patent of the United States is:

1. A logical function generator for sequentially generating binary signals in parallel representing the designation numbers for the functions $f_0, f_1, \dots, f_k$ comprising:

a continuously recycling input generator means for generating binary signals in parallel representing all of the values of the binary number $a_0, a_1, \dots, a_i$ during each cycle;

switching generator means under the control of said input generating means for generating binary signals in parallel representing all of the values of the binary number $x_0, x_1, \dots, x_j$ with the value of $x_0, x_1, \dots, x_j$ changing only at the end of each cycle of said input generator means; and

circuit generator means receiving the signals representing the values of $a_0, a_1, x_0, x_1, \dots, x_j$ for producing $k+1$ sequences of binary signals in parallel with each sequence for each cycle of said input generator means representing the designation number of one of the functions $f_0, f_1, \dots, f_k$ whereby the values $x_0, x_1, \dots, x_j$ define connections for designing logical circuits for generating the logical functions $f_0, f_1, \dots, f_k$;

said circuit generating means includes $k+1$ function generating circuit means with each circuit means receiving the signals representing the values of $a_0, a_1, \dots, a_i, f_0, f_1, \dots, f_{p-1}, f_{p+1}, \dots, f_k$ and selected ones of the values of $x_0, x_1, \dots, x_j$ for generating the function f_p where ($p=0, 1, \dots, k$).

2. A logical function generator according to claim 1

wherein said switching generator means includes means for constraining the switching generator means such that values of the number $x_0, x_1, \dots, x_j$ that do not define useful connections are not generated.

3. A logical function generator according to claim 1 wherein said input and switching generators each includes a series of flip-flops with the least significant flip-flop of the input generator being driven by a source of clock pulses and with the switching generator driven by the most significant flip-flop of the input generator.

4. A logical function generator according to claim 1 wherein each of said $k+1$ function generating circuit means includes three subgroups of logical circuit elements with the first of said subgroups consisting of a single element for producing f_p , with the second of said subgroups of logical circuit elements having their outputs connected to the inputs of said single element, with the third of said subgroups of logical circuit elements having their outputs connected to the inputs of said second subgroups of logical circuit elements and with the $a_0, a_1, \dots, a_i, x_0, x_1, \dots, x_j, f_0, f_1, \dots, f_{p-1}, f_{p+1}, \dots, f_k$ values connected to the inputs of said third subgroups of logical circuit elements.

5. A logical function generator according to claim 4 wherein said third subgroup of logical circuit elements is divided into sub-subgroups with the outputs of each sub-subgroup applied to the inputs of one of the elements in said second subgroup and with the values, $a_0, a_1, \dots, a_i, f_0, f_1, \dots, f_{p-1}, f_{p+1}, \dots, f_k$ and selected ones of the values of $x_0, x_1, \dots, x_j$ applied to the inputs of the elements in each of said sub-subgroups.

6. A logical function generator according to claim 1 including means for storing a binary function g in parallel form and comparing means for comparing the binary function f_p generated by said circuit generator means with g and for producing signal that stops said input and switching generators when any $f_p = g$.

* * * * *

40

45

50

55

60

65