ENGINEERING RESEARCH INSTITUTE
UNIVERSITY OF MICHIGAN
ANN ARBOR

Informal Memorandum 4

LANGUAGE CONVERSION FOR DIGITAL COMPUTERS

Volume II

THE PHYSICAL REALIZATION OF CODE AND FORMAT CONVERSION

ARTHUR W. BURKS

CARL H. POLLMAR

DON W. WARREN

JESSE B. WRIGHT

Note: This material is issued as an informal memorandum because it is tentative, it has not been completely checked, and the exposition is unpolished. Administrative circumstances prevent further work on the material. Consequently, it is necessary to present the results so far achieved in this form rather than in a complete regular report.

Project M828

BURROUGHS CORPORATION
RESEARCH CENTER
PAOLI, PENNSYLVANIA

August 15, 1954

engr
UMR 1022

Table of Contents

Acknowledgments

The organization and semi-final draft of this memorandum were produced by Mr. Pollmar just before he left the project. Final editing, which has been necessarily cursory, has been done by the other authors without benefit of consultation with Mr. Pollmar. Their remarks are inserted in footnotes labeled "Ed." wherever possible.

The seeds of this three volume series are to be found in an unpublished paper, dated 1951, by Hugh Livingston of the Burroughs Corporation Research Center. Unlike Vols. I and III in which Mr. Livingston's ideas performed solely a catalytic service, the present volume incorporates an appreciable portion of his work including a number of only slightly modified circuits for which the authors wish to express their indebtedness. Mr. Livingston did not, however, participate in the preparation of this volume and is to be relieved of any responsibility for it.

THE PHYSICAL REALIZATION OF CODE AND FORMAT CONVERSION*

1. Introduction.

In Vols. I and III we are concerned with presenting a theory to aid in the first stage of design, i.e., the construction of an abstract logical design based on logical operations. Here we are concerned with the relationship between such logical designs and physical circuits.

---

*In the abstract of this series (Vol. I, p. ii) it is stated that in Vol. II would be discussed: (1) the physical realization of logical nets, and (2) circuits for format conversion. However, the considerations that led to the issuance of this volume as an informal memorandum rather than as a regular report preclude an elaboration of theory for format conversion and memory equipment. Consequently, only code conversions and a few code-and-format conversions which are predominantly transliterative are discussed here. Section 4 contains three examples involving some format conversion and for this reason the title of the volume has been left unchanged.

This relationship has many aspects with most of which we will not be concerned. For example, a logical net may be used to study the behavior of a physical circuit, each circuit element being replaced by a small net of logical elements. The converse of this, the construction of circuits having the same behavior as a given logical net (i.e., the "physical realization" of the net), is the problem with which we will be primarily concerned.

Some characteristics of the net which are significant relative to circuits may be described by "indices". Roughly speaking, indices are numbers associated in some manner with nets or with portions of nets. The number of input pairs, the element count, the serial loading coefficient, the rank of an element, the "use" of a wire are all indices. Not all of these need have significance for physical realizations, and the significance of those that do will depend upon the type of circuit element used in the realization. For example, the element input count in the case of a realization of a conjunction by crystal rectifiers counts the number of crystals. It is less significant in the case of a relay realization where both contacts and coils must be considered. In general, the usefulness of these indices might be extended if (1) more use were made of auxiliary elements, e.g., those with associated functions equivalent to tube operators, and (2) if the appropriate indices, e.g., element input count or some other weighted index, were associated with them.

Obviously much work could be done in the study of indices and of their significance for nets and circuits. In this volume, however, only the element count, the element input count, and the parallel and serial loading coefficients will be used. Their significance will be pointed out in the consideration of examples.

As noted above, the present report is confined primarily to a survey of the problems of "realization" in terms of examples and a brief discussion of them. To do this, the concept of realization must be defined explicitly. This requires that (1) a physical interpretation (called the <u>0,1 convention</u>) be associated with the states

0 and 1, and (2) that each logical element be replaced by a physical circuit (whose components, for example, may be relays, crystals, tubes, etc.) with a single output, with the same number of inputs, and with the same "behavior" as the logical element. To say that such a circuit $\overline{X}$ has the same behavior as an element X (or is a <u>realization</u> of X) means that (1) the behavior of the circuit's input and output may be described by a function table in terms of two physical states, and (2) if those states are translated into 0's and 1's by the 0,1 convention, then the resulting function table is that of the logical element. Note that the same physical equipment may realize different logical elements if different 0,1 conventions are employed. The truth (function) tables for the common logical elements are

Conjunction

0	0	0
0	1	0
1	0	0
1	1	1

Disjunction

0	0	0
0	1	1
1	0	1
1	1	1

Negation

0	1
1	0

Conjunction Stroke		
0	0	1
0	1	1
1	0	1
1	1	0

Disjunction Stroke		
0	0	1
0	1	0
1	0	0
1	1	0

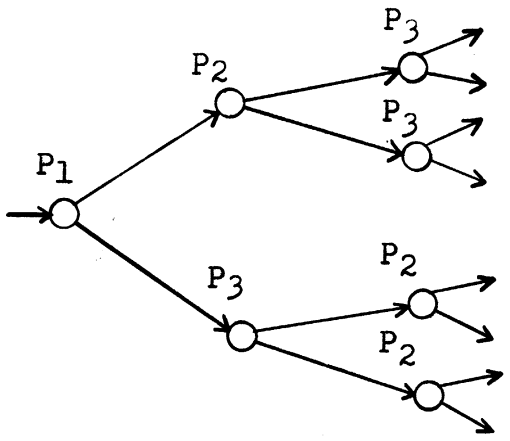
It is possible, as the following example shows, to use different 0,1 conventions at different points of a circuit provided that no contradiction arises. When this is the case, a _multiple 0,1 convention_ is employed. If T is a logical tree with key which is realized by a relay tree (see Fig. 1(D)), then either of the following conventions is possible.

(1)* $1 \longleftrightarrow$ high voltage, $0 \longleftrightarrow$ low voltage, everywhere.

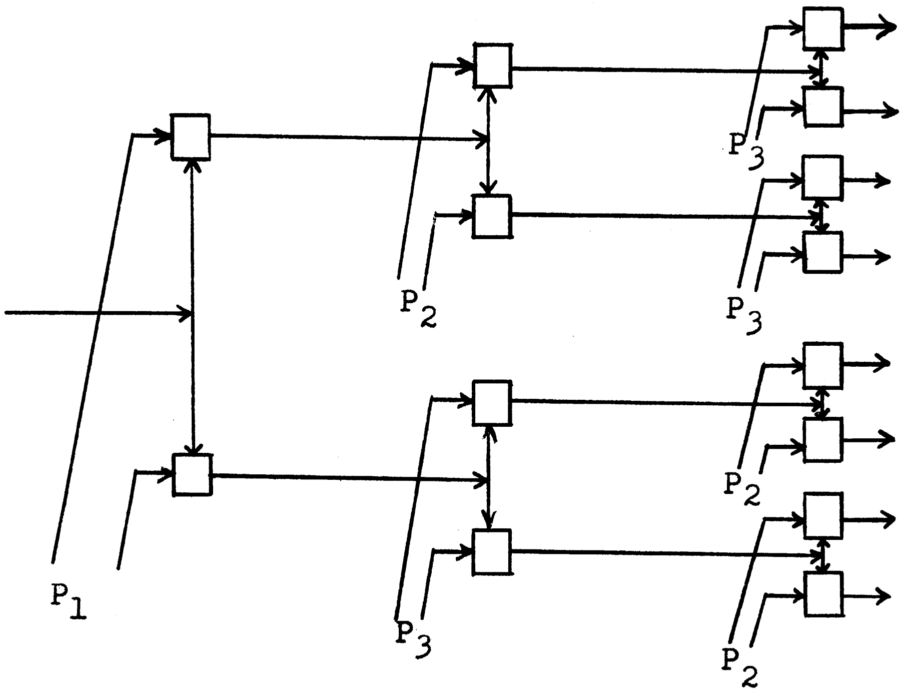
(2) Convention (1) on the input pairs (the inputs corresponding to the coils) and $1 \longleftrightarrow$ pulse, $0 \longleftrightarrow$ absence of a pulse, elsewhere.

The first convention defines a converter operated statically, and the second, a converter with static input and pulse output.

*Here and in several succeeding discussions of relays the convention might more appropriately be expressed as $1 \longleftrightarrow$ closed circuit, $0 \longleftrightarrow$ open circuit.- Ed.

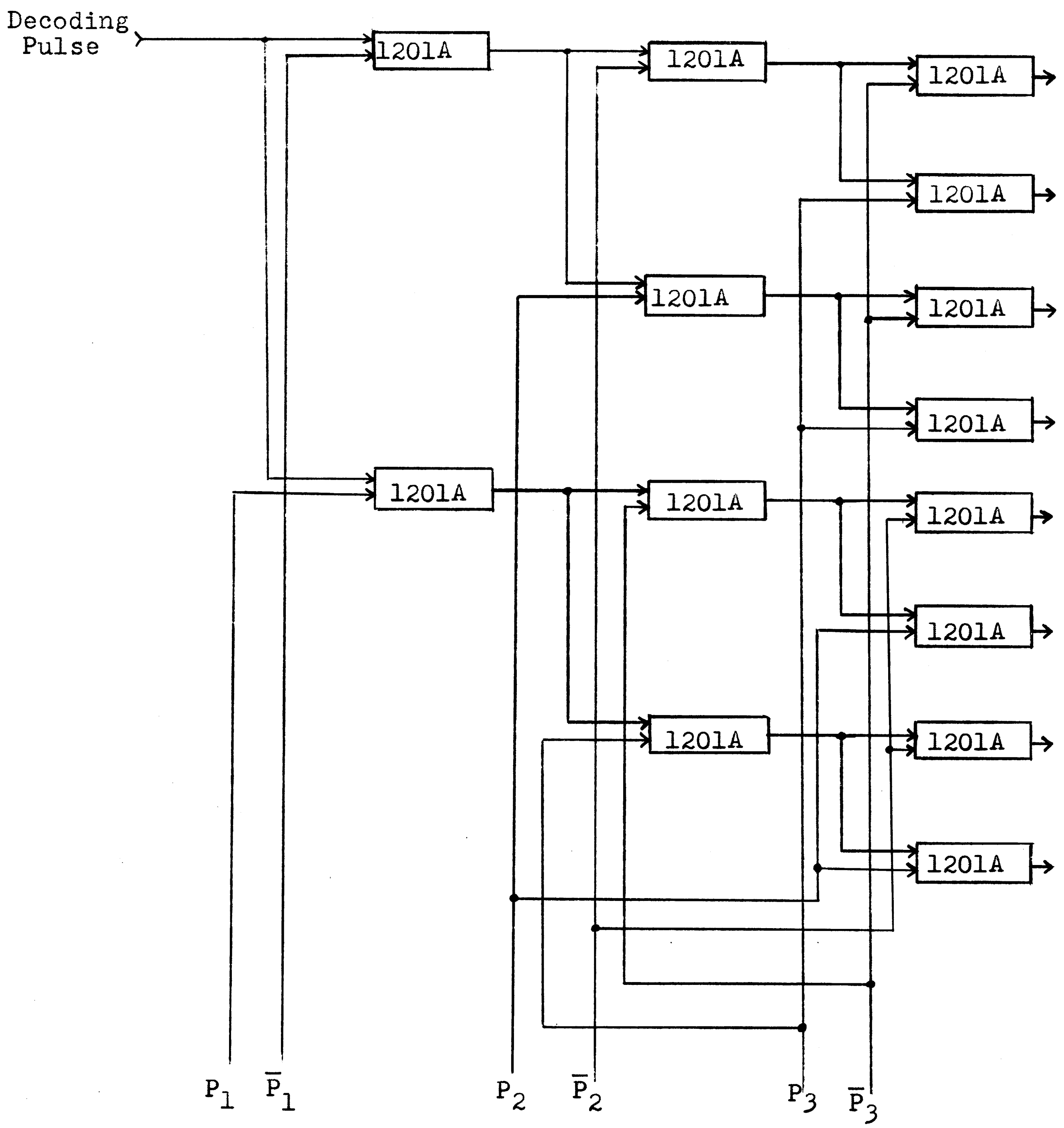


(A)



(B)

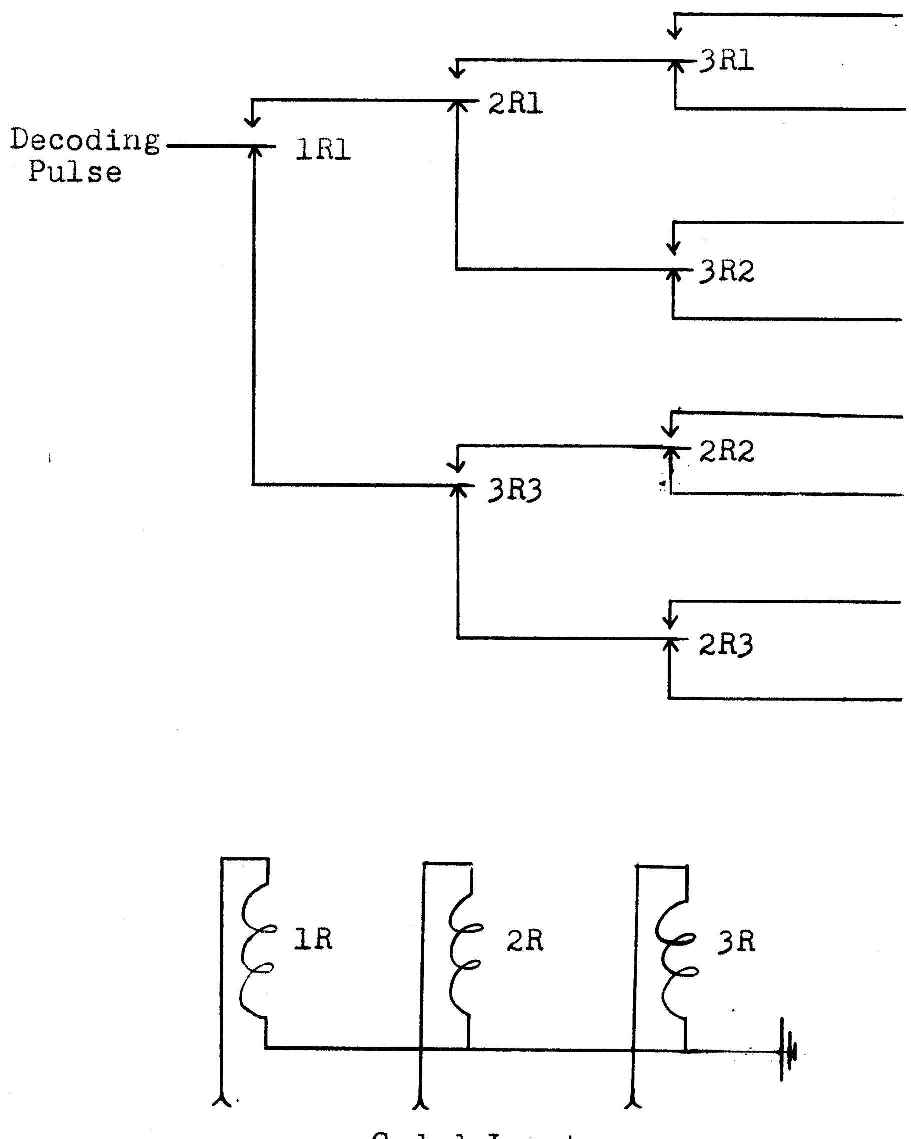
Fig. 1



(C)

Fig. 1

Pulse Control Unit Folded Tree



(D)

Fig. 1

Relay Folded Tree

In Section 2 various ways of realizing primitive and auxiliary logical elements are considered. The 0,1 convention for each case is explicitly stated. Among those considered are the conjunction elements, the negation element, the stroke elements, and the double-and. Section 3 contains examples of different physical circuits that are realizations of simple decoding and encoding nets such as the exponential switch, the tree, and the balanced multiplicative switch (MS) net. It also includes two examples of converters for an arbitrary transliterative function: the first illustrating the decoding-encoding techniques, the second illustrating the "direct" method.

A circuit produced by substituting a physical realization for each logical element in a net is quite often not as simple as it could possibly be. The simplification which may be carried out depends largely on the types of realization (e.g., relay, crystals, tubes, etc.) employed. We wish to emphasize that we are not engineers and have made no attempt, in general, to simplify the circuits. However, in certain obvious cases, some simplification has been made by example. Throughout Section 3 the discussion is directed toward net characteristics and the significance of the various indices for the corresponding physical circuits.

In Section 4 the use of these circuits as components in more complicated code converters, such as those required for conversion involving a shift code, is illustrated by three examples.

2. Realization of Primitive and Auxiliary Elements.

The most important primitive elements for the theory developed in this report are the conjunction ("and") and the disjunction ("or") elements. Subsections 2.1 and 2.2 are devoted to examples of circuits which realize them or logical elements closely related to them. Negation and the stroke element are considered in Subsection 2.3 where also realizations of an MS and of an encoding net are discussed for these can be considered as auxiliary elements from which more extensive nets can be constructed (e.g., MS nets).

2.1. Conjunction ("and") Elements.

A physical unit corresponding directly to the 2-input conjunction is the Burroughs pulse control* Coincidence Detector, Type 1201 A. It is essentially a dynamic gate (ordinarily driven by a flip flop). If the 0,1 convention is

for p and r: $1 \longleftrightarrow$ presence of a pulse,

$0 \longleftrightarrow$ absence of a pulse,

for q: $1 \longleftrightarrow$ 0 volts, $0 \longleftrightarrow$ -23 volts,

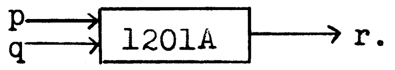
then the operation of a 1201 A is that of a

*See bibliography, item 3, for details of all pulse control units.-Ed.

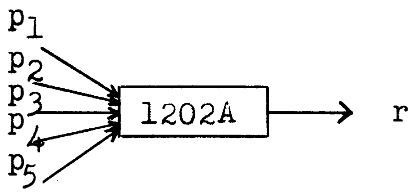
conjunction and is given by

Table I

p	q	r
0	0	0
0	1	0
1	0	0
1	1	1

where 

The Type 1202 A Coincidence Detector



may operate as a conjunction under the following convention: 1 $\longleftrightarrow$ 0 volts, 0 $\longleftrightarrow$ -23 volts. Its behavior table, then, is

Table II

p_1	p_2	p_3	p_4	p_5	r	
0	0	0	0	0	0	
0	0	0	0	1	0	
.	.	.	.	.	.	all
.	.	.	.	.	.	0's
.	.	.	.	.	.	
1	1	1	1	0	0	
1	1	1	1	1	1	

The 1201 A is a 2-input conjunction and the 1202 A, used as above, is an n-input conjunction, $n = 2,\ldots,5$. This difference is of little significance in the logical analysis, for an n-input conjunction can always be realized by a net of 2-input conjunctions.

A conjunction may also be realized by a relay coil with a make-contact. Consider

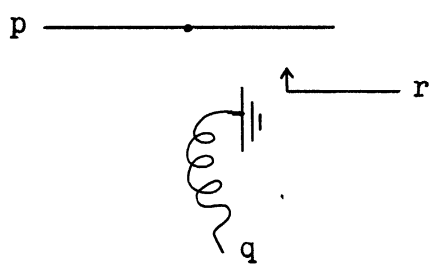
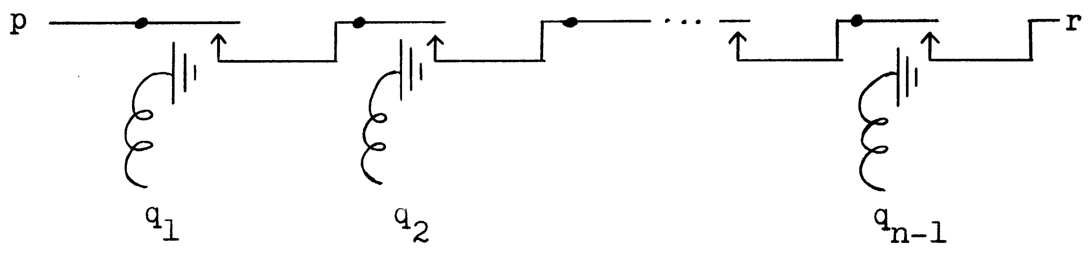


Fig. A

If the 0,1 conventions are: $1 \longleftrightarrow$ positive voltage, $0 \longleftrightarrow$ ground potential (for p the conventions might alternately be: $1 \longleftrightarrow$ pulse, $0 \longleftrightarrow$ absence of pulse), then the behavior of this circuit is described by Table I above. Thus, it is a physical realization of a 2-input conjunction.

An n-input conjunction, $r \equiv p \cdot q_1 \cdot q_2 \cdot \ldots \cdot q_{n-1}$, may be realized by putting (n-1) relays in series thus:*



If p is connected to a positive voltage source (or a clock pulse source), then r may be considered as the conjunction of $q_1 \ldots q_{n-1}$ where the 0,1 convention for the q_i is as for Fig. A.

If a break-contact is used, then the relay unit is a realization of a net consisting of a conjunction with the q-input negated. Consider the relay,

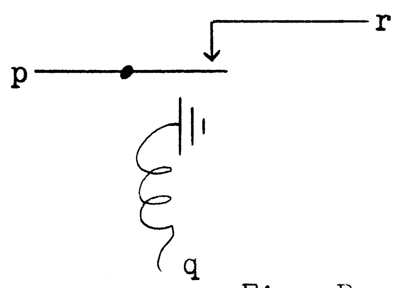


Fig. B

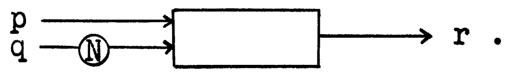
*Note that this relay realization of an n-input conjunction could also be considered as a compound circuit built up from the relay circuits (Fig. A) realizing a 2-input conjunction.-Ed.

Using the same 0,1 convention as before, this relay unit has for its associated behavior table

Table III

p	q	r
0	0	0
0	1	0
1	0	1
1	1	0

and this is the table for $p \cdot \sim q$ which in terms of logical elements (see Vol. I) can be represented by



Finally, consider a transfer-contact relay (Fig. C) which is composed of a make-contact, a break-contact, and the coil.

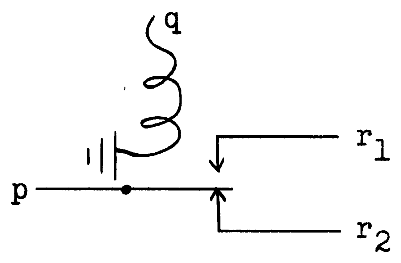


Fig. C

If the 0,1 conventions are

for p, r_1, r_2: $1 \longleftrightarrow$ pulse or positive voltage,

$0 \longleftrightarrow$ absence of a pulse or ground potential,

for q: $1 \longleftrightarrow$ positive voltage,

$0 \longleftrightarrow$ ground potential,

then the behavior table is

Table IV

p	q	r_1	r_2
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

which is the same as that for the net below, a double-and with polarizer (see Vol. I).

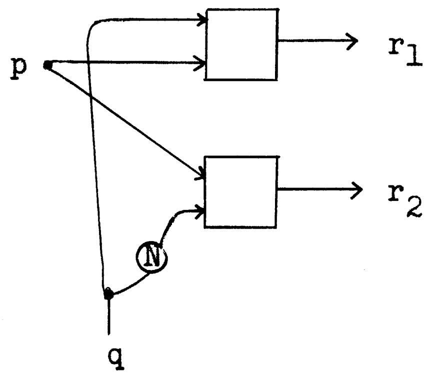


Fig. D

It should be noted that a relay circuit realizing a conjunction under a given set of 0,1 conventions may realize it under one or more other 0,1 conventions as well because of the physical nature of the relay and particularly because of the coil. For example, some of the conventions for which the circuits of Fig. A will realize a conjunction are:

$$\begin{cases} p: & 1 \longleftrightarrow \text{pulse}, \quad 0 \longleftrightarrow \text{absence of a pulse}, \\ q: & 1 \longleftrightarrow + \text{voltage}, \quad 0 \longleftrightarrow 0 \text{ voltage}; \end{cases}$$

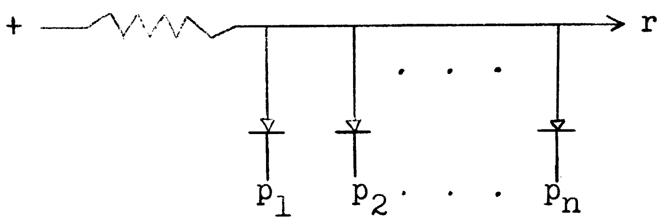
$$\begin{cases} p: & 1 \longleftrightarrow \text{pulse}, \quad 0 \longleftrightarrow \text{absence of a pulse}, \\ q: & 1 \longleftrightarrow - \text{voltage}, \quad 0 \longleftrightarrow 0 \text{ voltage}; \end{cases}$$

$$\begin{cases} p: & 1 \longleftrightarrow + \text{voltage}, \quad 0 \longleftrightarrow 0 \text{ voltage}, \\ q: & 1 \longleftrightarrow + \text{voltage}, \quad 0 \longleftrightarrow 0 \text{ voltage}; \end{cases}$$

$$\begin{cases} p: & 1 \longleftrightarrow - \text{voltage}, \quad 0 \longleftrightarrow 0 \text{ voltage}, \\ q: & 1 \longleftrightarrow - \text{voltage}, \quad 0 \longleftrightarrow 0 \text{ voltage}. \end{cases}$$

This is usually true of realizations using other components. However, the change of the 0,1 convention will in many cases alter the logical function which the circuit realizes.

Conjunctions may also be realized by crystal rectifiers [1]. If the 0,1 convention is $1 \longleftrightarrow$ high voltage, $0 \longleftrightarrow$ low voltage, then the behavior of the following circuit is that of an n-input conjunction. For $n = 5$, its function is given by Table II above.



The output, r, has high voltage if and only if $p_1 \ldots p_n$ have high voltage.

Conjunctions may, of course, be realized by tube circuits. Because the operation of tubes as used in computers seems to be most naturally described by the stroke function, and because the stroke function is a "logically-sufficient primitive", the conjunction may be realized by a small tube circuit. Because of the relationship of tubes and stroke functions, the discussion of the realization of conjunctions by tubes is included with the discussion of the stroke functions in Subsection 2.3.

2.2. Disjunction ("or") Elements.

In some cases a disjunction may be realized simply by joining the two or more wires which form its input set. If this is possible, it is the cheapest form of realization. Often, however, it is impossible to realize disjunction without using diodes, for example, to prevent "sneak circuits" and some realizations of this kind are discussed later in this subsection.

Disjunctions, like conjunctions, may be realized directly by Burroughs pulse control units. For example, the Type 1202 A Coincidence Detector which, with the 0,1 convention specified in the last section, represents a conjunction can with the opposite convention, $1 \longleftrightarrow -23$ volts, $0 \longleftrightarrow 0$ volts, represent a disjunction. Its behavior table, then, is

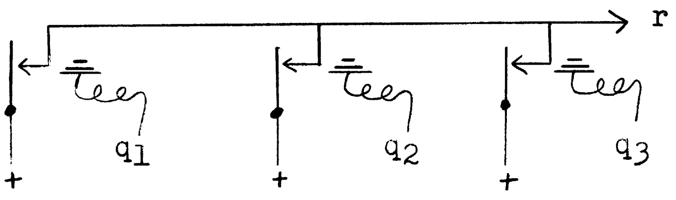
Table V

p_1	p_2	p_3	p_4	p_5	r	
0	0	0	0	0	0	
1	0	0	0	0	1	
0	1	0	0	0	1	
1	1	0	0	0	1	
0	0	1	0	0	1	all
.	.	.	.	.	.	1's
.	.	.	.	.	.	
.	.	.	.	.	.	
1	1	1	1	1	1	

This is the table for a 5-input disjunction.

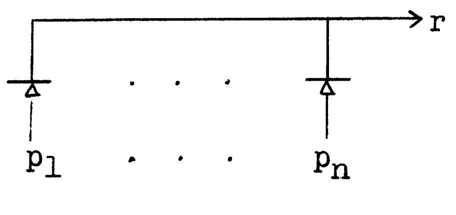
The Type 1601 A and Type 1602 A Mixers are also physical realizations of disjunction for if $1 \longleftrightarrow$ presence of a pulse, $0 \longleftrightarrow$ absence of a pulse, then their behavior is represented by Table V (for the 4-input 1601 A delete column p_5 and the bottom sixteen rows).

Disjunctions may be realized by relay circuits. Consider the following with the convention, $1 \longleftrightarrow +$ voltage, $0 \longleftrightarrow 0$ voltage, defined in the previous subsection for the relay of Fig. A.



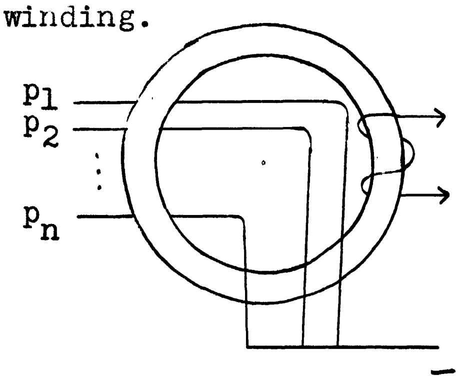
The behavior of this relay circuit may be described by columns p_1, p_2, p_3, and r in the first eight rows of Table V and clearly is disjunctive.

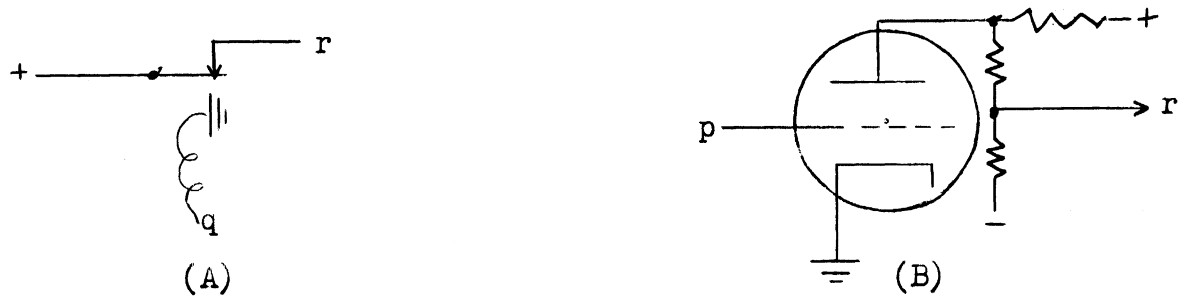
Crystal rectifiers may also be used to realize disjunctions. Consider



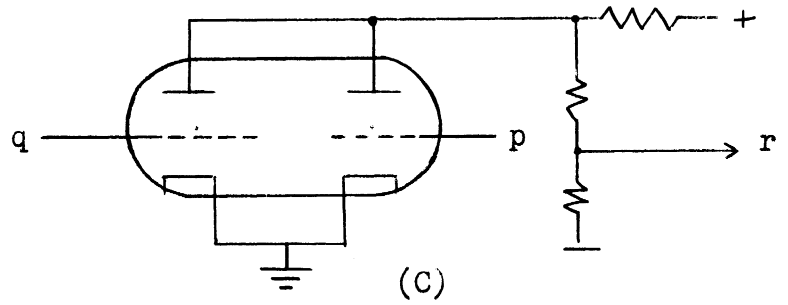
If at least one of the p_i is at high voltage, then r will be at high voltage (because the rectifying properties of crystal prevents the flow to any p_j which might be at low voltage), otherwise, r will be at low voltage. Thus, if $1 \longleftrightarrow$ high voltage, $0 \longleftrightarrow$ low voltage, the behavior of this net can be described, for $n = 5$, by Table V.

Another realization of a disjunction -- and a very economical one -- is by means of a Bell A.M. A. ring-transformer such as is represented below. The ring is the core of the transformer, the input wires p_i which are led through the ring represent primary windings, and the winding r on the right is the output or secondary winding.

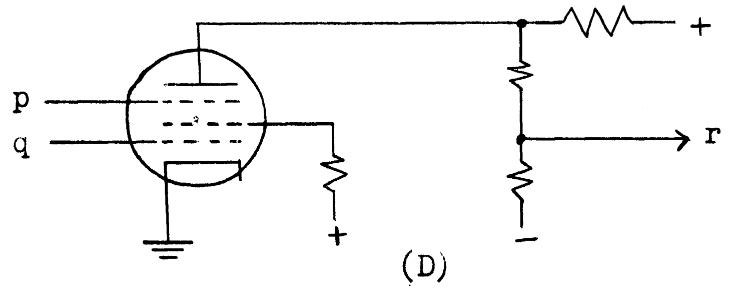




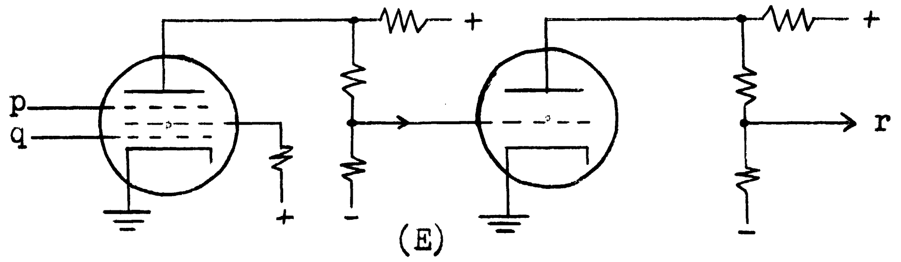
Relay and Tube Realization of Negation



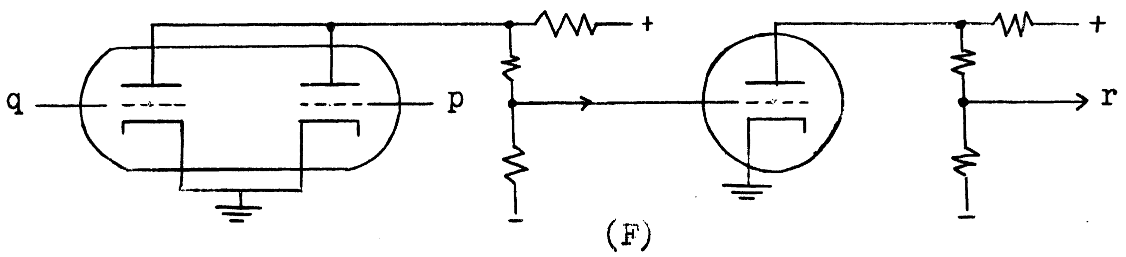
Tube Realization of Disjunctive Stroke



Tube Realization of Conjunctive Stroke



Tube Realization of a Conjunction



Tube Realization of a Disjunction

Fig. 2*

*See bibliography, item 2, for further details.-Ed.

If a high voltage is applied to one or more of the p_i, current will flow in that primary and a potential will be induced in r, otherwise not. Using the customary conventions, the behavior of this disjunction is again represented, for $n = 5$, by Table V.

As in the case of conjunction, disjunction may be realized by tubes. A discussion of this kind of realization is postponed until Subsection 2.3.

2.3. Other Elements.

Conjunction and disjunction are the two logical elements most widely employed in the networks of this report. Several others, however, should be briefly considered.

Negation, which is required for polarizing nets, may be realized, for example, by the relay circuit of Fig. 2(A) (cf. also Fig. B, p. 14) or the triode circuit of Fig. 2(B). The output voltage of the triode will be high if the voltage applied to the control grid is low and vice versa. If then the 0,1 convention is $1 \longleftrightarrow$ high voltage, $0 \longleftrightarrow$ low voltage, the behavior table of this circuit is that of negation,

p	r
0	1
1	0

. In practice negations are often unnecessary because the input is in polar pair form.

The stroke functions may be realized easily by tubes. The circuit of Fig. 2(C) is a realization of the disjunctive stroke. The twin triode with a common plate resistor will have a high output voltage if and only if both control grids (p and q) are low, i.e., its behavior table is

p	q	r
0	0	1
0	1	0
1	0	0
1	1	0

.

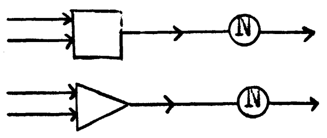
Clearly, this is the table of $\sim(p \vee q)$, the disjunctive stroke.

Fig. 2(D) is a pentode realizing the conjunctive stroke. Its output will be high at all times except when the input voltages applied to the grids, p and q, are both high. Thus, it has the following table.

p	q	r
0	0	1
0	1	1
1	0	1
1	1	0

This is the table for $\sim(p \cdot q)$, the conjunctive stroke.

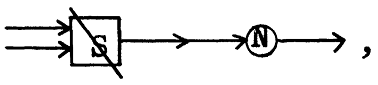
Since the two stroke elements are represented by the following logical nets,



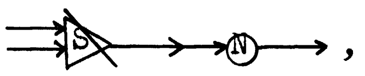
(conjunctive stroke),

(disjunctive stroke),

and since conjunctions, disjunctions, and negations are realizable in relay nets, it is clear that these stroke elements can be realized in relay circuits (i.e., substitute the relay circuit for the corresponding logical element). Conversely, the following logical nets (see Vol. I, Subsection 3.2),



conjunction



disjunction

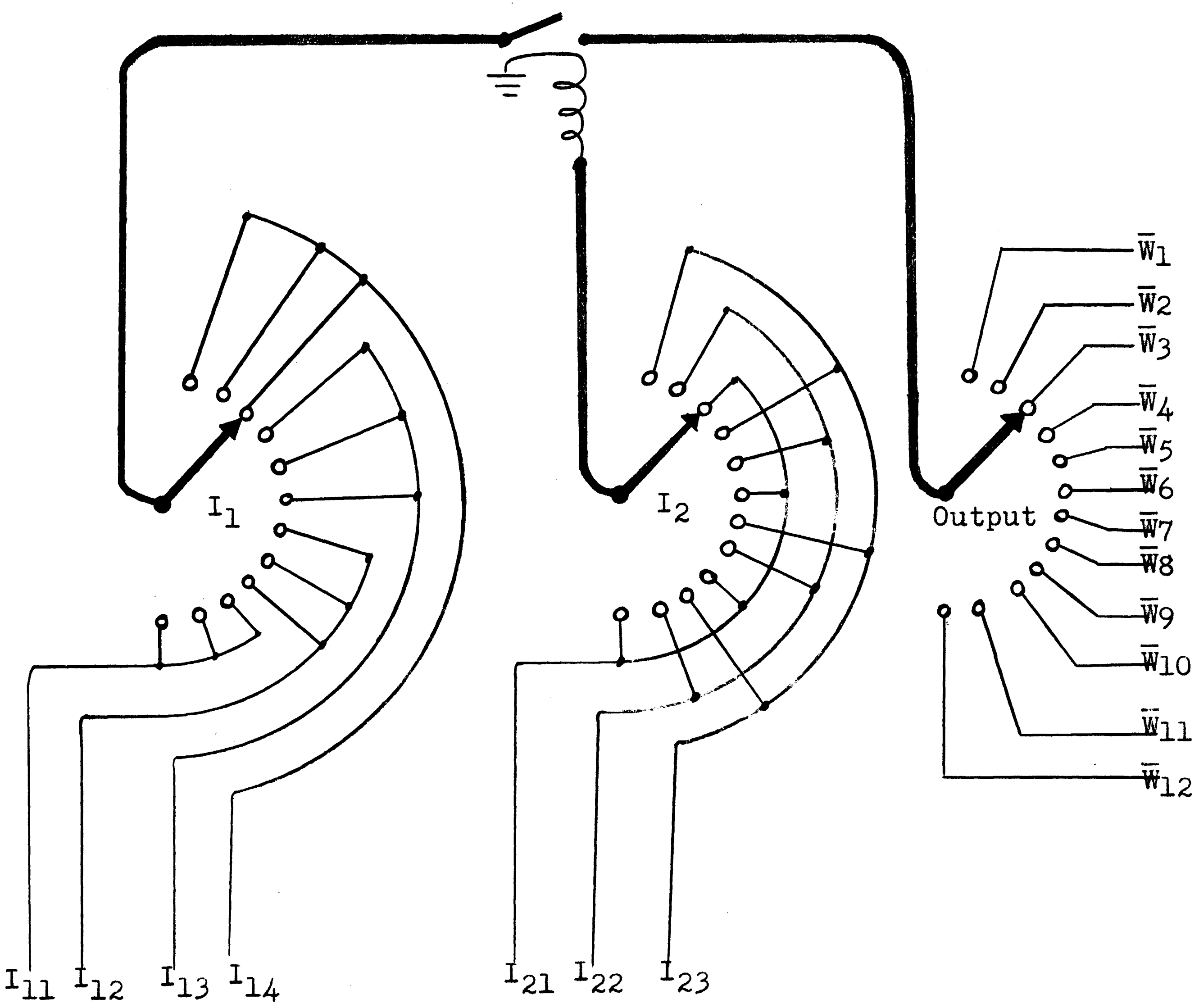
are realized as in Fig. 2(E) and Fig. 2(F) by substituting tube stroke and negation circuits for the corresponding logical elements.

The remainder of this section is devoted to two circuits (for the MS and for an encoding net) which might appear more closely related to the code converters of Section 3 than to the primitive elements of Section 2. These two realizations are exceptional, however, in consisting basically of a rotary selector switch which, though it does not correspond directly to any of the primitive logical elements, does clearly have the primitive attributes of being a relatively indivisible, "atomic" piece of equipment and of having an internal structure which does not seem to be amenable to any obvious analysis in terms of the standard logical primitives.* Thus, inclusion of these two circuits here confines to Section 2 the introducing of all the types of equipment to be considered and allows Section 3 to focus more sharply on the techniques for combining these types of equipment to effect code conversions.

---

*This point might warrant further study.-Ed.

Fig. 3 shows a rotary selector circuit which realizes an MS. One arc of contacts is assigned to each input set I_j (these sets are defined by the MS) and one additional arc is assigned to the output. Each arc has as many contacts as the net has outputs which, since an MS effects a complete decoding, must be equal to $\prod_{1}^{J} N(I_j)$ (in the example, 4x3 = 12). In operation the wipers, being rigidly joined together, will sweep from one row of contacts (one contact from each arc) to the next physically adjacent row. Each (kth) row is connected through a relay buffer to a unique output w_k and the jth input contact of this row is connected to that wire of I_j which must be activated in order to activate w_k. The circuit joining the wipers is a relay conjunction: its inputs being connected successively, as the wiper sweeps, to each of the possible input combinations (elements of the Cartesian Product of the I_j) and its output, at each step, being connected to the corresponding circuit output. If we take $1 \longleftrightarrow +$ voltage, $0 \longleftrightarrow 0$ voltage, as the 0,1 convention, the operation of the circuit is



Rotary Selector Switch as an MS

For $I_1, I_2, \ldots, I_J$:
number of arcs = J;
number of contacts per arc = $\prod_{1}^{J} N(I_j)$.

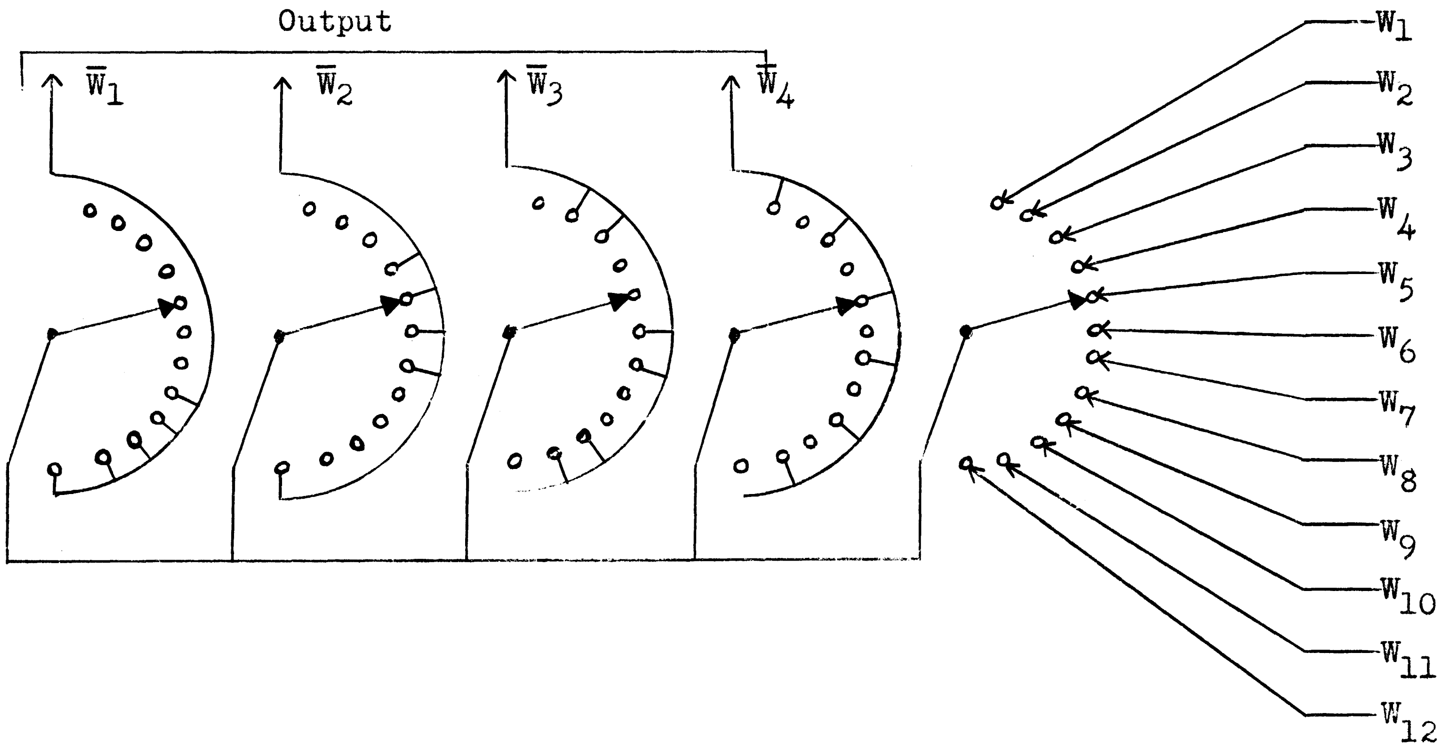
Fig. 3

described by the following table (where the blank spaces represent zeros).

I_{11}	I_{12}	I_{13}	I_{14}	I_{21}	I_{22}	I_{23}	$\overline{W}_1$	$\overline{W}_2$	$\overline{W}_3$	$\overline{W}_4$	$\overline{W}_5$	$\overline{W}_6$	$\overline{W}_7$	$\overline{W}_8$	$\overline{W}_9$	$\overline{W}_{10}$	$\overline{W}_{11}$	$\overline{W}_{12}$
			1			1	1											
			1		1			1										
			1	1					1									
		1				1				1								
		1			1						1							
		1		1								1						
	1					1							1					
	1				1									1				
	1			1											1			
1						1										1		
1					1												1	
1				1														1

Since this is a table for an MS with input sets I_1 and I_2, it is clear that this circuit (under appropriate operation) is a realization of an MS.

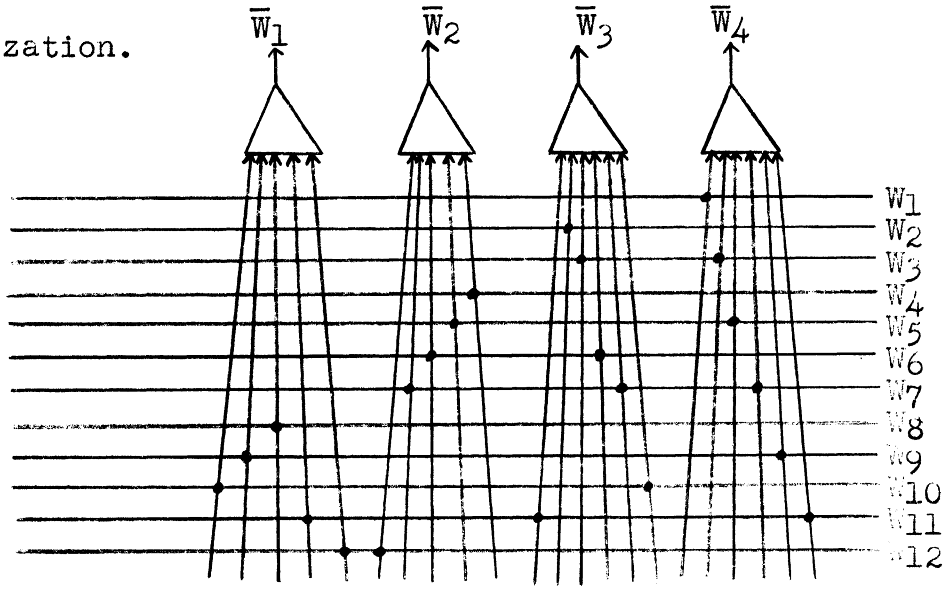
Fig. 4 is a circuit realizing an encoding switch. $M+1$ arcs are needed, one for each of the M bit positions in the range code (i.e., the code into which we are translating) and one for the input. Some of the $N(I)$ contacts of an arc are connected to the appropriate output in such a way as to effect a disjunction. Under operation as an encoding circuit, exactly one of the W_i will



Rotary Selector Switch as an Encoding Circuit

Fig. 4

carry positive voltage, the others will be at ground potential. If W_i is positive and the wipers are in contact with the ith contact of each arc, then the $\overline{W}_j$ which are connected to the ith contact of the jth arc will be positive. Using the 0,1 convention, $1 \longleftrightarrow +$ voltage, $0 \longleftrightarrow 0$ voltage, the operation of this circuit can easily be seen to be equivalent to that of the following logical net of which it is a realization.



The principal objection to the use of the rotary selector is its low speed of operation. It is, nevertheless, an interesting device.

3. Code Converters for Transliterative Functions.

In this section circuits are constructed for logical nets which realize (for definition, see Vol. 1, p. 26) transliterative functions. Transliterative functions are, roughly speaking, single-valued functions which map a set of sequences of 0's and 1's, all of the same length, onto another set of sequences of 0's and 1's, also all of the same length.

For our purposes transliterative functions may be divided into three classes: decoding functions (Subsection 3.1), encoding functions (Subsection 3.2), and arbitrary transliterative functions (Subsection 3.3).

The technique of circuit construction which will be used (and which is assumed to yield a circuit with the desired behavior) consists, first, of drawing a logical net to effect the desired transliteration, and, second, of replacing each logical element by a realization of that element. Different physical circuits may be obtained by replacing a given logical element of the net by different physical realizations of that element.

The realizations of the logical elements must be "compatible"(e.g., the 0,1 conventions must not conflict, no significant timing discontinuities may arise).

In many cases the resulting circuit may be simplified; usually this possibility arises because of the special properties of the physical elements involved. A few examples of such simplifications are given though no generally applicable theory has been formulated and some of the circuits presented undoubtedly could have been further simplified.

The circuit designer is interested primarily, of course, in constructing a circuit which will perform some given function, i.e., will have a certain specified "behavior" under certain specified operating or input conditions. Within the limitations imposed by behavior requirements, he will be further guided by such considerations as: minimality, reducing the number of component parts and, hence, the cost of the circuit; parallel loading, the number of "driven" units to which a "driver" can be connected in parallel (e.g., tubes per flipflop, contacts per relay coil); and <u>serial</u> <u>loading</u>, the number of units which can be safely connected in series (e.g., crystal rectifiers which attenuate the signal).

The network theory of Vol. I deals very directly with the behavior problem and, by means of indices, also sheds some light on such subsidiary problems as those mentioned above. There are, of course, many design factors which cannot be taken into account at the abstract level, and those that are can be treated in only a very general way as a means of rather rough comparison of broad classes of realizations.

In presenting circuit realizations of logical nets, therefore, we will also attempt to translate the indices associated with these nets into measures of physical characteristics of the circuit wherever resulting figures appear to have practical significance.

The physical behavior of a circuit can be determined very easily from the state of the corresponding net. To calculate the state of a net, given the state (0 or 1) of all inputs, proceed as follows: (1) let N be the given net and label the inputs of N with 0's and 1's in the desired fashion; (2) repeat as many times as possible the following operations, (a) find an unlabeled wire BC whose origin B is the terminus of a wire AB with label α ($\alpha = 0$ or 1) and assign label α to BC, (b) find a logical element all of whose input wires, $w_1, \ldots, w_n$, are labeled, and assign to its output the label $\beta = \Omega\alpha_1, \ldots, \alpha_n$, where α_i is the label of w_i and Ω is the logical function (e.g., conjunction, disjunction, etc.) associated with the element. The states of the corresponding circuit wires are then determined by the 0,1 convention.

3.1. Decoding Functions.

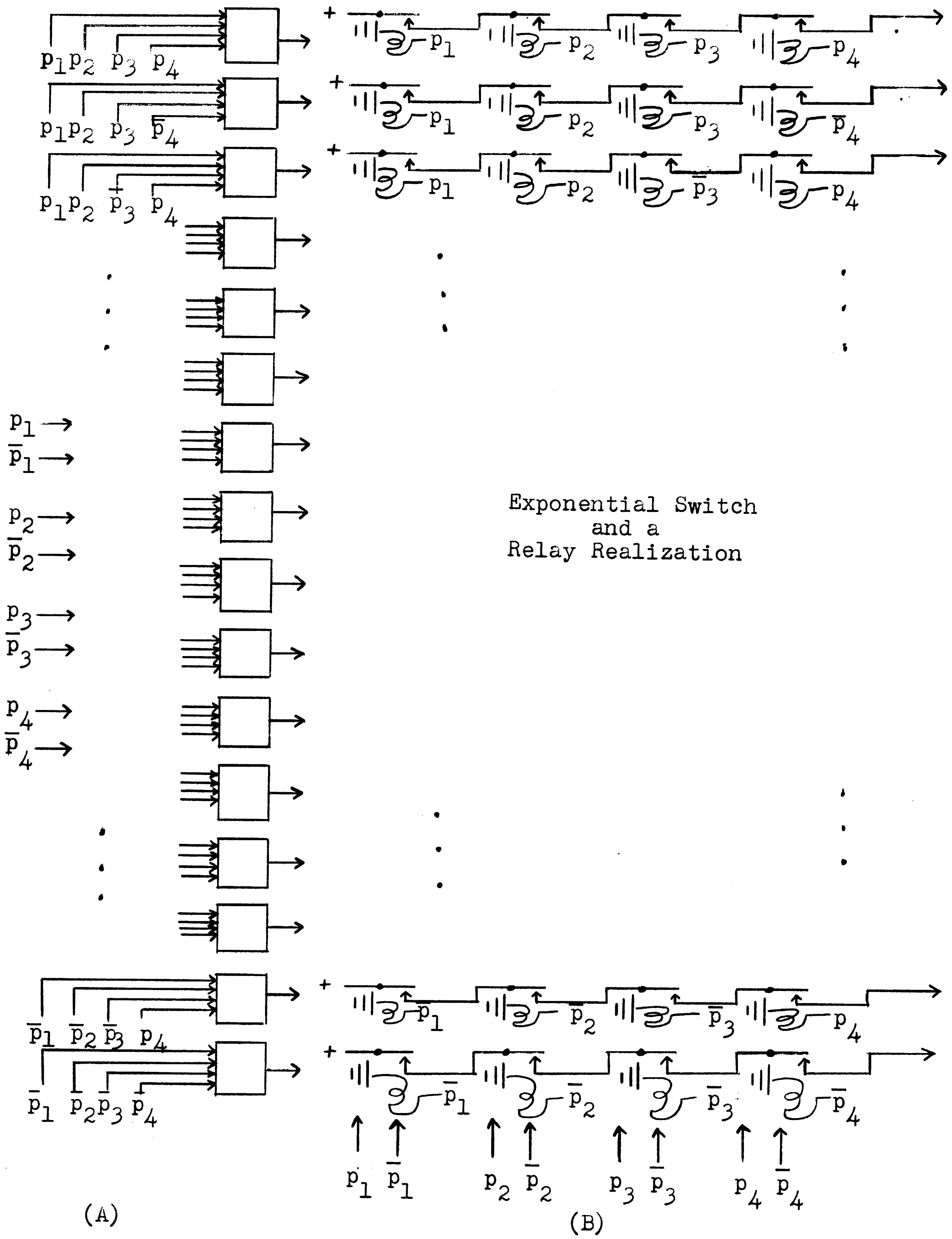
A decoding function is a transliterative function, each range sequence of which contains a single 1. Such functions can be translated into logical nets which are exponential switches, trees, or balanced MS nets. In this subsection many different physical realizations of these nets are constructed and examined. The circuits are drawn so as to emphasize the logical structure, sometimes at a sacrifice of some of the conventional engineering details. For example, in Fig. 5 only enough wires and labels are presented to establish the pattern of interconnection; also, the "remote-connection" device is used, i.e., two wires with the same label are to be considered connected.

A relay realization of a 4-input-pair exponential switch is shown in Fig. 5. Fig. 5(A) is the logical net and Fig. 5(B) is its realization.* The 0,1 convention is $1 \longleftrightarrow +$ voltage, $0 \longleftrightarrow$ open circuit.** The element input count, which is 48 in this case, is the number of relay make-contacts (also, in the diagram, the number of coils). The parallel loading of P_i, which is 8, is the number of contacts operated by P_i. The serial load clearly has little significance here.

One obvious way that this circuit can be simplified is, for each P_i, to replace all eight relays that it drives by a single relay unit consisting of a coil connected to P_i and

*Note that this realization is effected by the purely mechanical procedure of replacing each 4-input conjunction element in the net of Fig. 5(A) by its relay realization as defined in Subsection 2.1. - Ed.

**Other conventions are possible. - Ed.



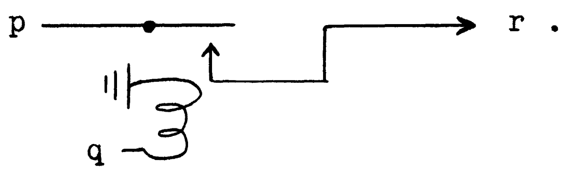
$p_1 p_2 p_3 p_4$
$p_1 p_2 p_3 \bar{p}_4$
$p_1 p_2 \bar{p}_3 p_4$
p_1
$\bar{p}_1$
p_2
$\bar{p}_2$
p_3
$\bar{p}_3$
p_4
$\bar{p}_4$
$\bar{p}_1 \bar{p}_2 \bar{p}_3 p_4$
$\bar{p}_1 \bar{p}_2 \bar{p}_3 \bar{p}_4$
(A)
+
p_1 p_2 p_3 p_4
p_1 p_2 p_3 $\bar{p}_4$
p_1 p_2 $\bar{p}_3$ p_4
$\bar{p}_1$ $\bar{p}_2$ $\bar{p}_3$ p_4
$\bar{p}_1$ $\bar{p}_2$ $\bar{p}_3$ $\bar{p}_4$
p_1 $\bar{p}_1$ p_2 $\bar{p}_2$ p_3 $\bar{p}_3$ p_4 $\bar{p}_4$
Exponential Switch
and a
Relay Realization
(B)

Fig. 5

eight associated make-contacts.

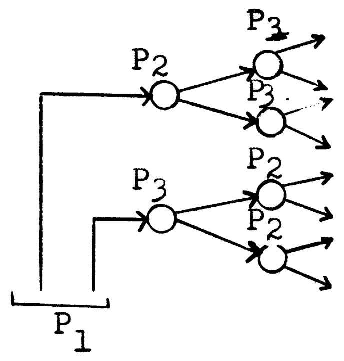
In the original circuit, if the inputs are not polar pairs, a polarizing net is needed. In this case the polarizing net and the conjunctions can easily be combined. All that is necessary is to replace every make-contact operated by $\overline{P}_i$ by a break-contact operated by P_i. In this way only wires P_i will be required.

In Fig. 6(A) a 3-input-pair folded tree without key is given. Fig. 6(B) is a realization of the conjunction net in terms of relay circuits introduced in Subsection 2.1 of the type

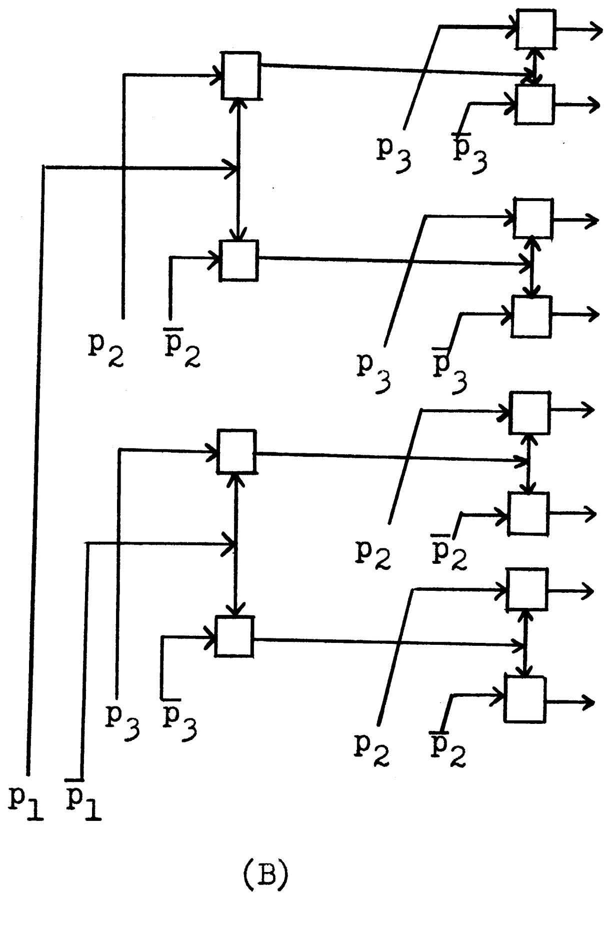


The type of relay conjunction used in the exponential switch example may, of course, be used here.* A more efficient realization could also be produced by substituting relay double-ands (Subsection 2.1) for the vertices of Fig. 6(A). The 0,1 convention

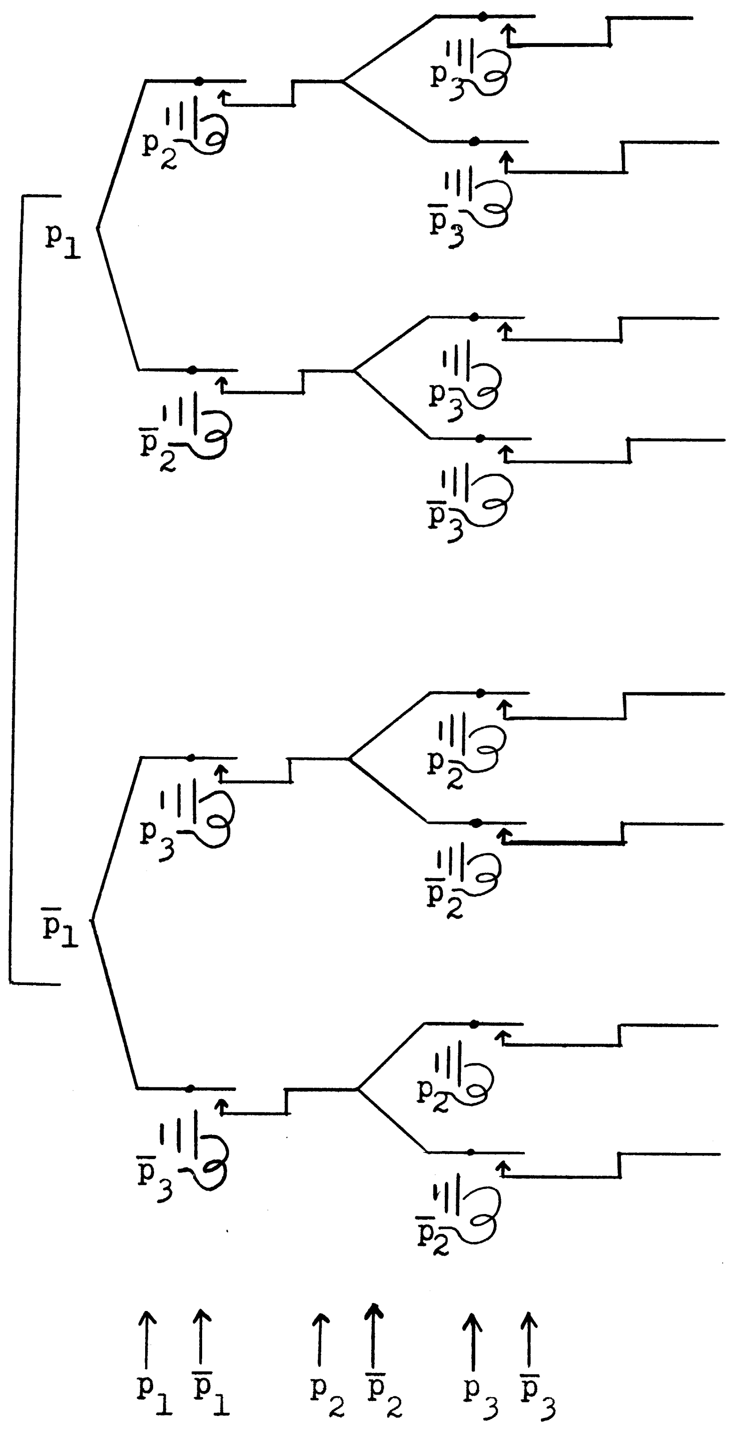
*That is, impress a constant voltage on the wire P and supply a second coil for the P-input. This realization is obviously wasteful of equipment.-Ed.


p_1
p_2
p_3
p_3
p_3
p_2
p_2

(A)


p_1 $\overline{p}_1$
p_2 $\overline{p}_2$
p_3 $\overline{p}_3$
p_3 $\overline{p}_3$
p_3 $\overline{p}_3$
p_2 $\overline{p}_2$
p_2 $\overline{p}_2$

(B)


p_1
$\overline{p}_1$
p_2
$\overline{p}_2$
p_3
$\overline{p}_3$
p_3
$\overline{p}_3$
p_3
$\overline{p}_3$
p_2
$\overline{p}_2$
p_2
$\overline{p}_2$
p_1 $\overline{p}_1$ p_2 $\overline{p}_2$ p_3 $\overline{p}_3$

(C)

Fig. 6

used here is $1 \longleftrightarrow +$ voltage, $0 \longleftrightarrow 0$ voltage.

The parallel load of p_i, $i = 2, 3$, is the number of make-contacts controlled by p_i. A similar fact holds for $\bar{p}_i$. The serial load of p_1 and $\bar{p}_1$ is the number of stages (bays) through which the signal must pass. Finally, the element input count is twice the number of contacts. Clearly, the practical interpretation of these indices will vary with the form of the realization.

If the inputs are p_1, p_2, and p_3 (i.e., are not in polarized form), then replace every make-contact controlled by a relay connected to $\bar{p}_i$ by a break-contact and coil operated by p_i. Then the set of contacts operated by relays controlled by p_i may be replaced by a unit consisting of a single coil and a set of relay transfer contacts. A break-contact operated by a relay coil controlled by p_1 (with its contact connected to a positive voltage) is needed to give $\bar{p}_1$. If, in addition, p_1 is given by a make-contact (although this is not necessary unless the tree is to be operated dynamically), the resulting network is the usual relay tree. See Fig. 1(D).

In Fig. 1(C) the logical tree of Fig. 1(B) is realized by pulse control units. The 0,1 convention is

for p_i: $1 \longleftrightarrow -23$ volts, $0 \longleftrightarrow 0$ volts,

for the others: $1 \longleftrightarrow$ pulse, $0 \longleftrightarrow$ no pulse.

Here the parallel load may be interpreted to mean the number of 1201-A's driven by a single flipflop, and this, according to the specification, has a maximum of 6. This means that, if we do not wish to provide extra flipflops, we are limited to trees (folded) with a maximum of four input pairs. For this case the best possible load distribution (LD) is $1P_1 + 4P_2 + 5P_3 + 5P_4$. If five input pairs are required, the best LD is $1P_1 + 7P_2 + 7P_3 + 8P_4 + 8P_5$,* and additional flipflops are required. What the best possible LD's are when additional flipflops or relay coils are needed has not been investigated.

Trees may be realized by crystal rectifiers or tubes by substituting the desired circuit for the logical conjunctions and defining the appropriate

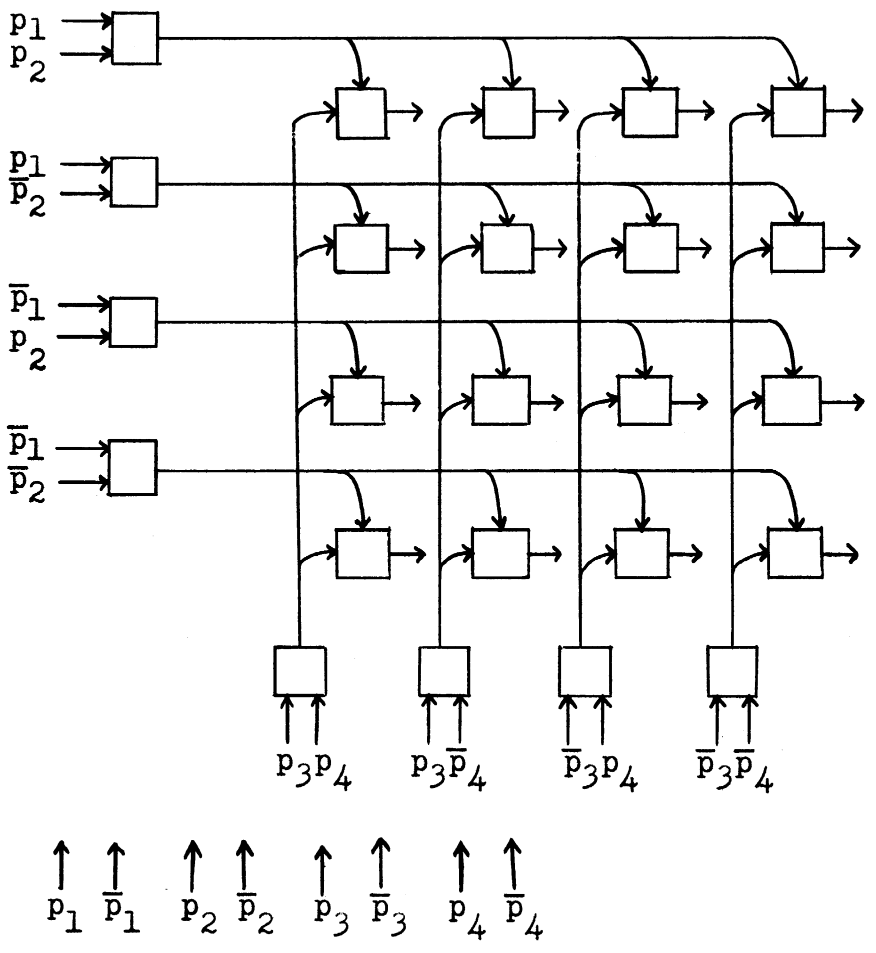
*This is the "best balanced" LD. However, $1P_1 + 5P_2 + 5P_3 + 10P_4 + 10P_5$ might be better for this purpose since it requires only one flipflop for each of P_1, P_2, P_3 and two each for P_4, P_5, and all flipflops except that for P_1 can be loaded equally.-Ed.

0,1 conventions.

Fig. 7(A) shows a balanced MS net with four input pairs, and Fig. 7(B) is its crystal realization. Here, 1 $\longleftrightarrow$ high voltage, 0 $\longleftrightarrow$ low voltage. The element input count is the number of crystals required and so is a direct measure of the cost. The parallel loading coefficients of the input wires do not have much significance for balanced MS nets since they have a maximum value of 4.

Although the parallel loading coefficient is an adequate loading index for trees and exponential switches, it is not for the balanced MS net, for in these nets the parallel loading coefficient of conjunctions (i.e., the number of logical elements to which the output is connected) in the interior of the net may be large.

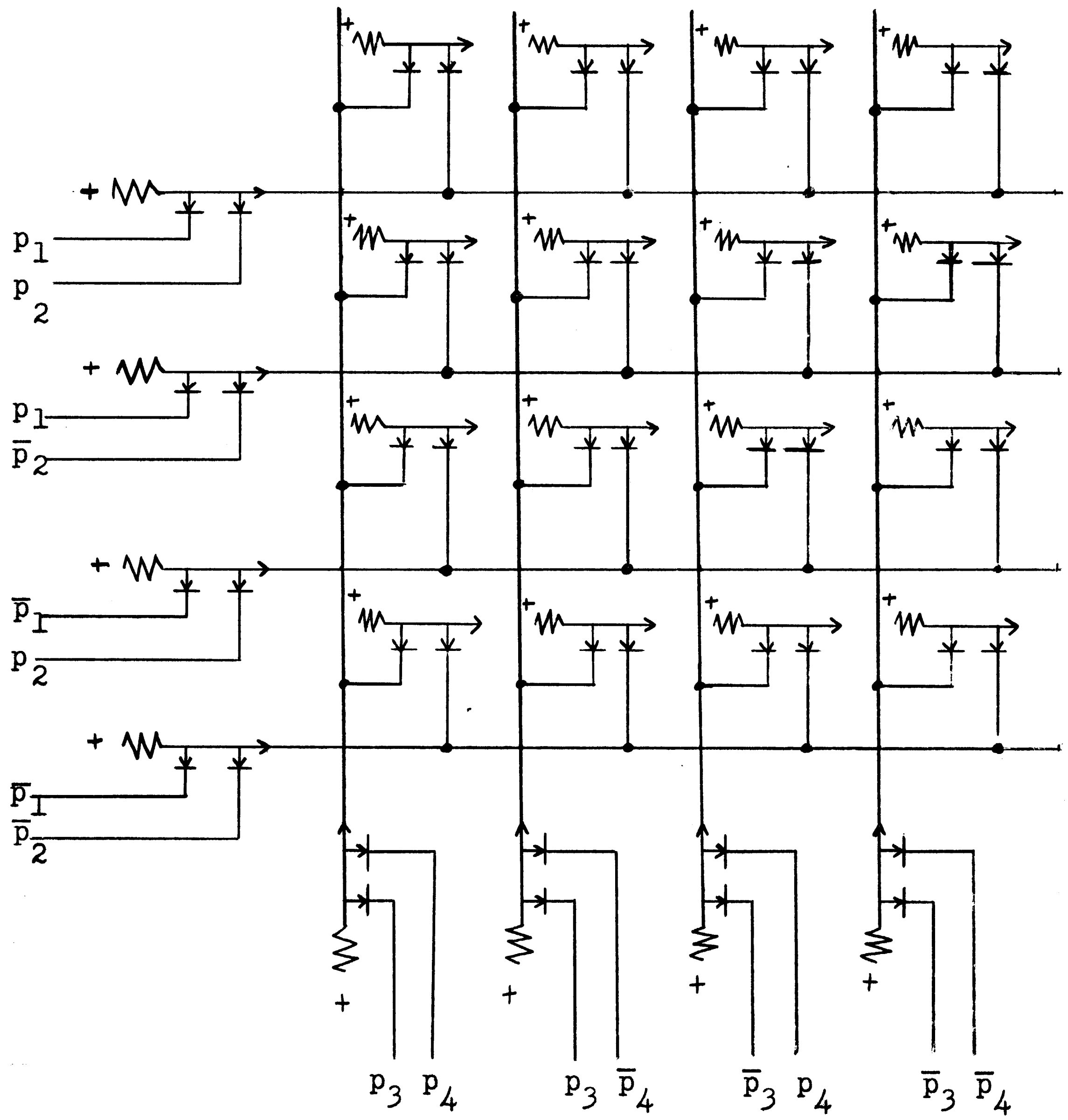
For example, in the diagram below (p. 48) which represents a balanced MS net (each M_i being an MS) with five input pairs, the conjunctions of M_3 have a parallel load of 8. This load is



(A)

4-Input-Pair Balanced MS Net

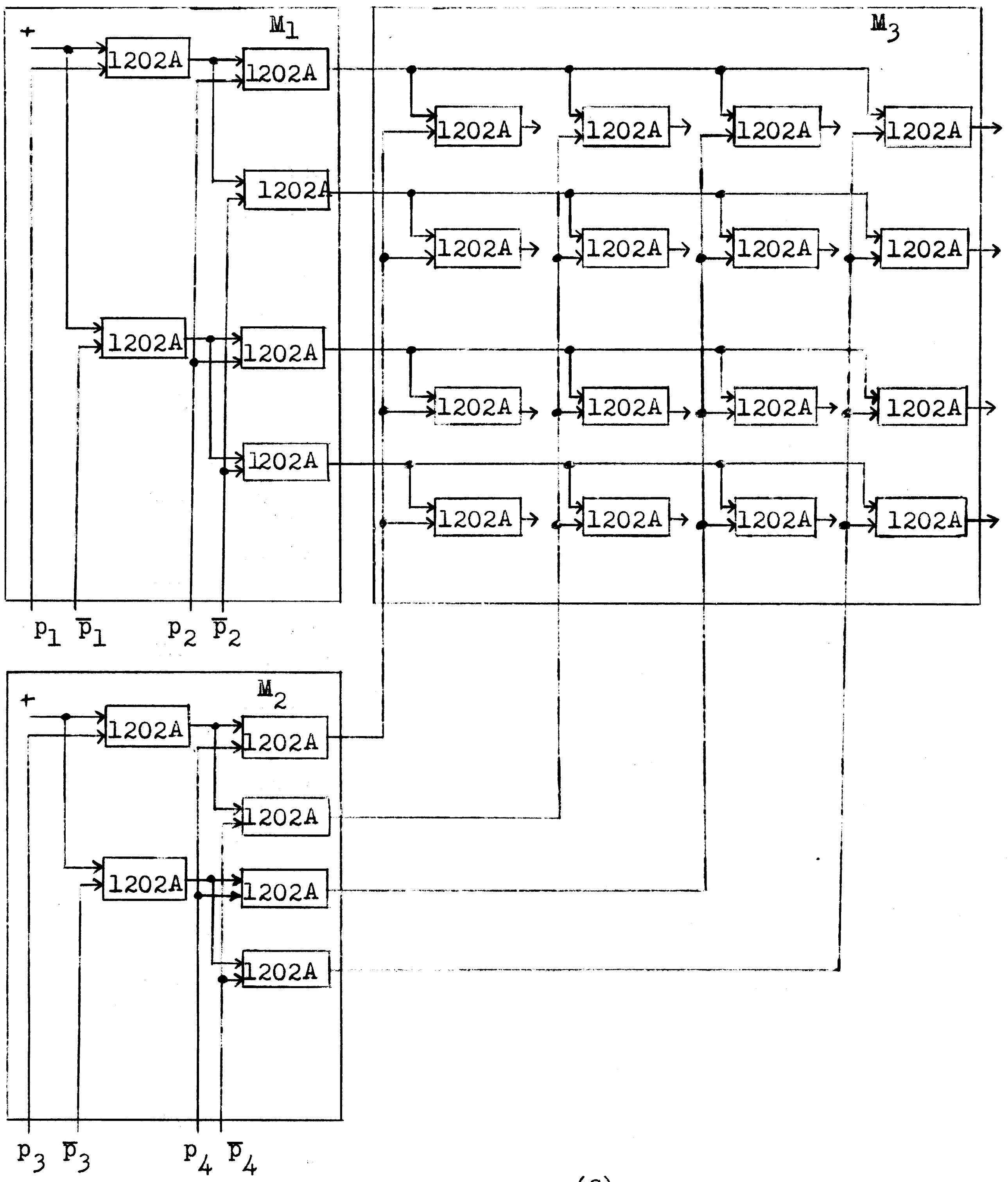
Fig. 7



Crystal Rectifier Realization of Logical Net of Fig. 7(A)

(B)

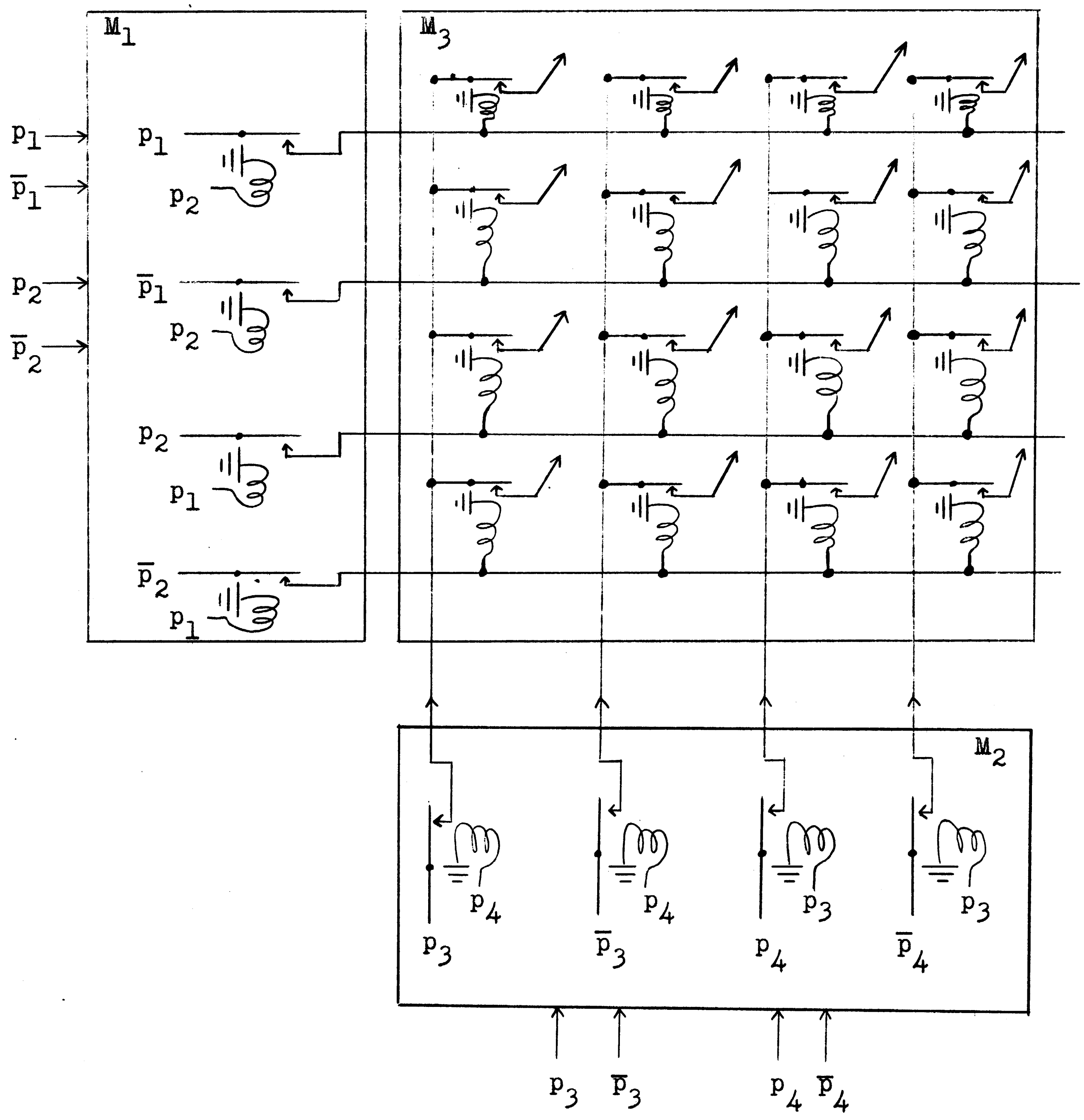
Fig. 7



(C)

Fig. 7

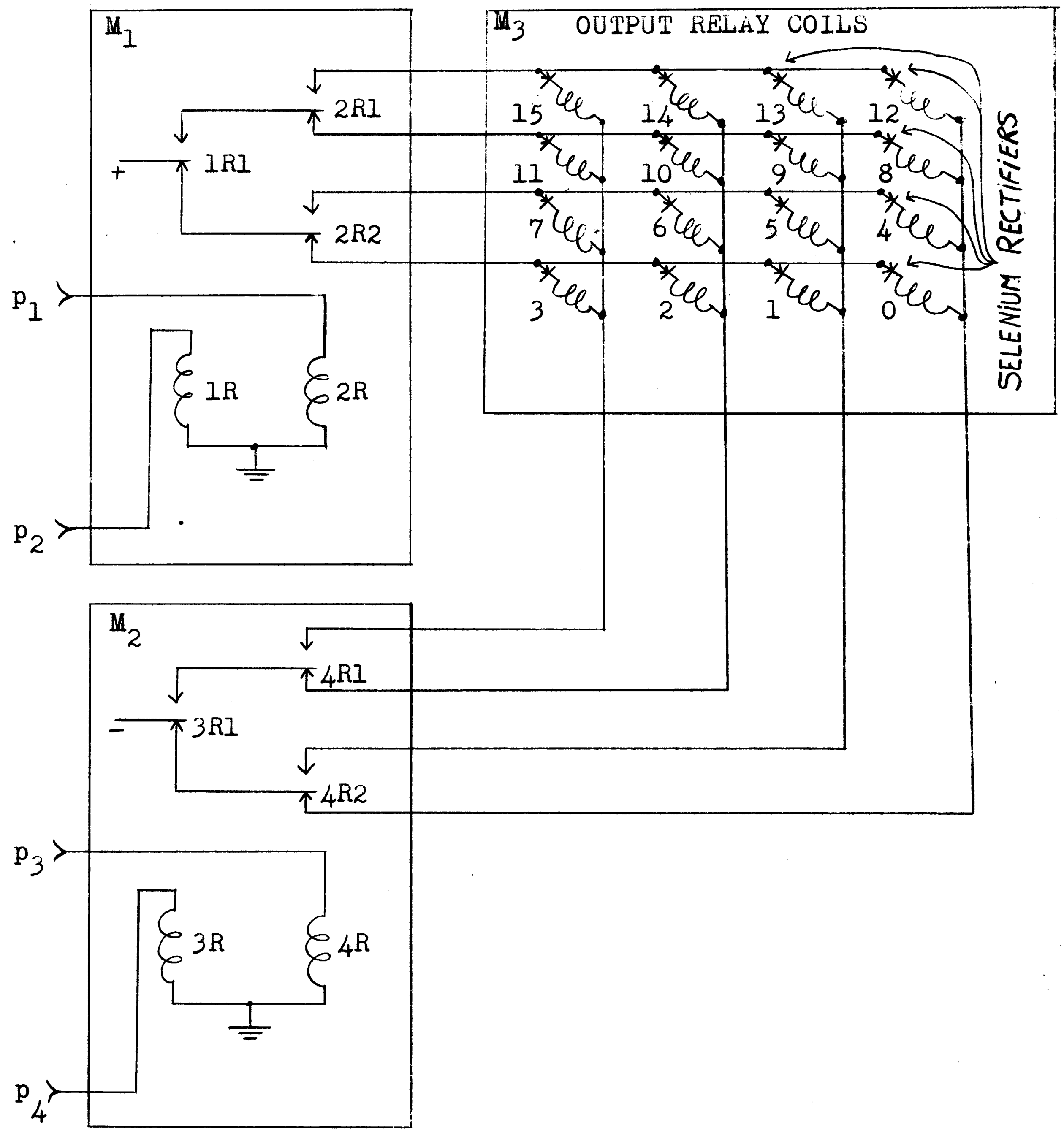
Pulse Control Unit Realization of Logical Net of Fig.7(A)



(D)

Fig. 7

Relay Realization of Logical Net of Fig. 7(A)



Relay Multi-tree Decoder
Realizing the Logical Net of Fig. 7(A)
(E)

Fig. 7

the largest in the net and is the same as the maximum coefficient of the "best" LD for the corresponding folded tree. However, as the number of input pairs increases, the parallel loading characteristic as compared with those of the folded tree improves.

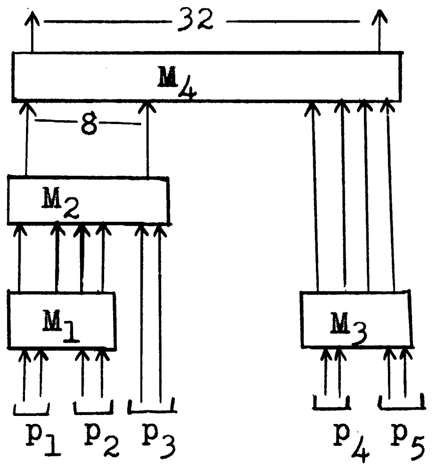


Fig. 7(C) is the pulse control unit realization. Here the element input count is twice the number of pulse control units required. The serial loading coefficient is the number of pulse control units driven in series.

If a static circuit is desired, 1202-A's are used as indicated in the figure, and the 0,1 convention is $1 \longleftrightarrow 0$ volts, $0 \longleftrightarrow -23$ volts.

A pulse output may be achieved by replacing the 1202-A's in M_1 and M_3 by 1201-A's and by connecting the input labeled + in M_1 to a pulse source. This, of course, corresponds to a multiple 0,1 convention where for certain wires the 0,1 convention indicated above holds and where for other wires the following convention holds: $1 \longleftrightarrow$ presence of a pulse, $0 \longleftrightarrow$ absence of a pulse.

Two different types of relay realization are shown in Fig. 7(D) and Fig. 7(E). They suggest the variety of forms which relay realizations can take and illustrate some of the characteristics of indices.

The 0,1 convention adopted for Fig. 7(D) is $1 \longleftrightarrow +$ voltage, $0 \longleftrightarrow 0$ voltage, because the outputs of the relays of M_1 must operate the coils of M_3. The circuit presented here can obviously be simplified. In M_1 and M_2 only two coils each with two make-contacts are needed and in M_3 four coils each with four contacts are required. Clearly here the element input count of 24 is the number of contacts, and the parallel load of an input to an MS is either (1) the number of contacts to which it is connected, or (2) the number of coils in that MS. Thus, there is a sort of "dual" significance to the parallel load coefficient here. This "duality" holds generally for all MS with two input nets where the conjunctions are realized as in this example.

It is of some interest to compare relay trees and relay balanced MS nets at this point. While the tree is the minimal relay contact net (see Bibliography, item 4, p. 113) producing a complete decoding, the balanced MS net (not a relay contact net) for four or more input pairs has a smaller element input count. In terms of the example, this means that the number of contacts in the circuits of Fig. 7(D) is smaller than that for the corresponding tree. However, it is worth pointing out that the number of coils required for a tree is minimal. Four are required for a 4-input-pair tree as compared to the eight required for the 4-input-pair balanced MS net.

Fig. 7(E) is another type of relay realization, and one employing a multiple 0,1 convention of greater complexity. An adequate multiple 0,1 convention can be roughly summarized as follows:

In M_1

$$\begin{cases} 1 \longleftrightarrow + \text{ voltage,} \\ 0 \longleftrightarrow 0 \text{ voltage.} \end{cases}$$

In M_2

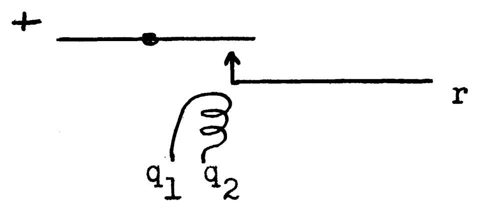
$$\left\{\begin{array}{ll} \text{for coil inputs} & \begin{cases} 1 \longleftrightarrow + \text{ voltage,} \\ 0 \longleftrightarrow 0 \text{ voltage;} \end{cases} \\ \text{for other inputs} & \begin{cases} 1 \longleftrightarrow - \text{ voltage,} \\ 0 \longleftrightarrow 0 \text{ voltage.} \end{cases} \end{array}\right.$$

In M_3

$$\left\{\begin{array}{ll} \text{for coils} & \\ \quad\text{rectifier side} & \begin{cases} 1 \longleftrightarrow + \text{ voltage,} \\ 0 \longleftrightarrow 0 \text{ voltage,} \end{cases} \\ \quad\text{other side} & \begin{cases} 1 \longleftrightarrow - \text{ voltage,} \\ 0 \longleftrightarrow 0 \text{ voltage;} \end{cases} \\ \text{for output contacts (not shown)} & \left\{\begin{array}{l} \begin{cases} 1 \longleftrightarrow + \text{ voltage,} \\ 0 \longleftrightarrow 0 \text{ voltage,} \end{cases} \\ \qquad\text{or} \\ \begin{cases} 1 \longleftrightarrow \text{pulse,} \\ 0 \longleftrightarrow \text{absence of a pulse.} \end{cases} \end{array}\right. \end{array}\right.$$

This circuit differs in other ways from that of Fig. 7(D). Its inputs are not polar pairs, and the consequent modifications of the exponential switch have been made. The make-contacts of the relays of M_3 are not shown. They may be connected to a pulse source or to a positive voltage source to provide either a dynamic or static output.

The use of the relays in M_3 illustrates still another way in which conjunction may be realized by a relay. This is shown in the diagram below where both q_1 and q_2 must be in the 1 state (opposite 0,1 convention applying to these two) if the coil is to operate and r be in the 1 state.



3.2. Encoding Functions.

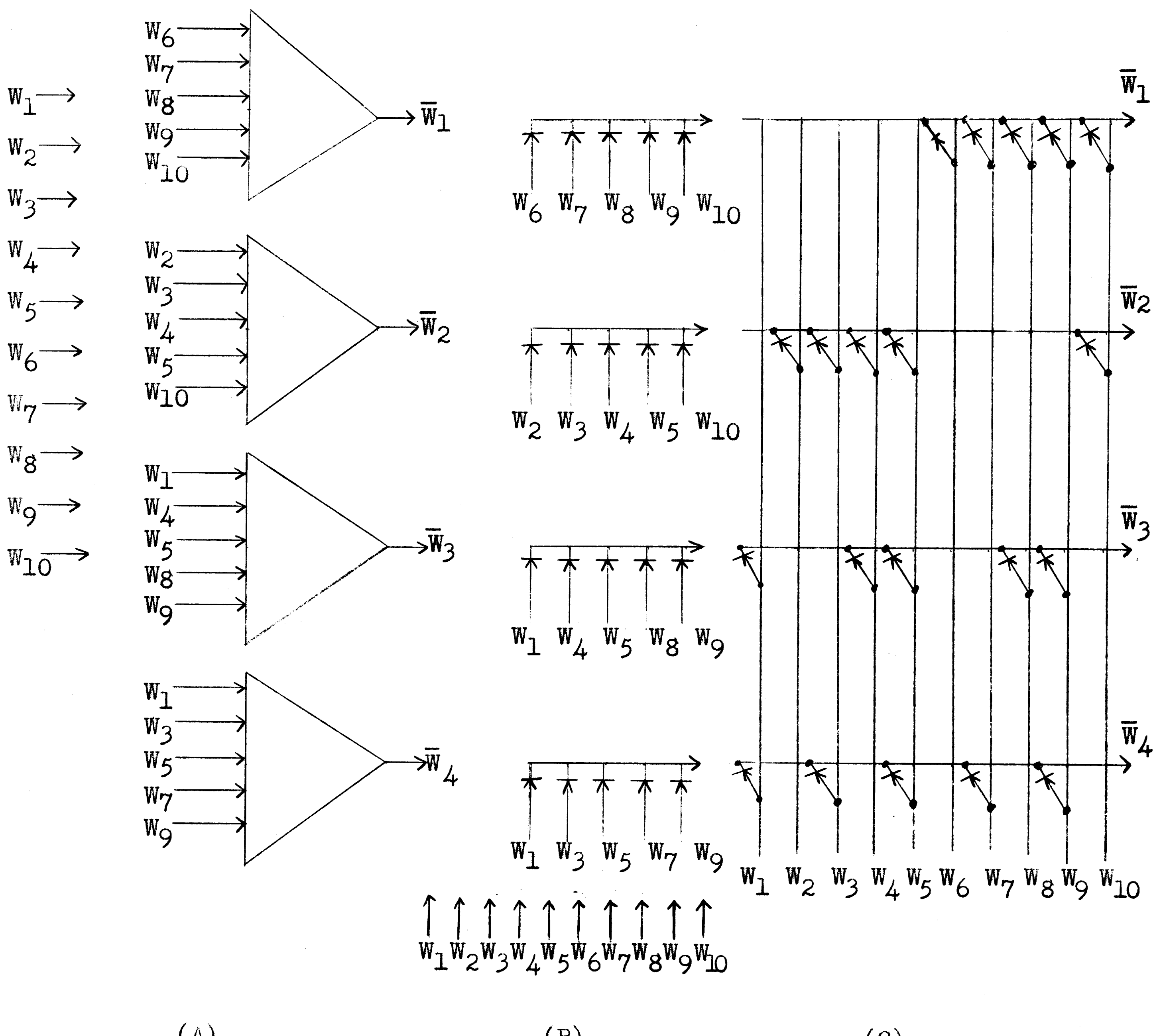
An encoding function is a transliterative function each domain sequence of which contains exactly a single 1. If we choose as our convention that $1 \longleftrightarrow$ positive potential, $0 \longleftrightarrow$ ground potential, then this means that in normal operation only one input wire at a time will be positively charged. On the other hand, if operated dynamically, i.e., if $1 \longleftrightarrow$ presence of a pulse, 0 $\longleftrightarrow$ absence of a pulse, then there will be a pulse on only one input at a time.

The technique given in Vol. I requires a knowledge of the encoding function T_e which is to be realized. For the purposes of our example, T_e is given by the following table (the blank spaces representing zeros) which essentially maps the ten decimal digits onto their excess 3 representation. The corresponding logical net is given in Fig.8 (A). The convention that two wires with the same label are to be considered connected is again used.

T_e

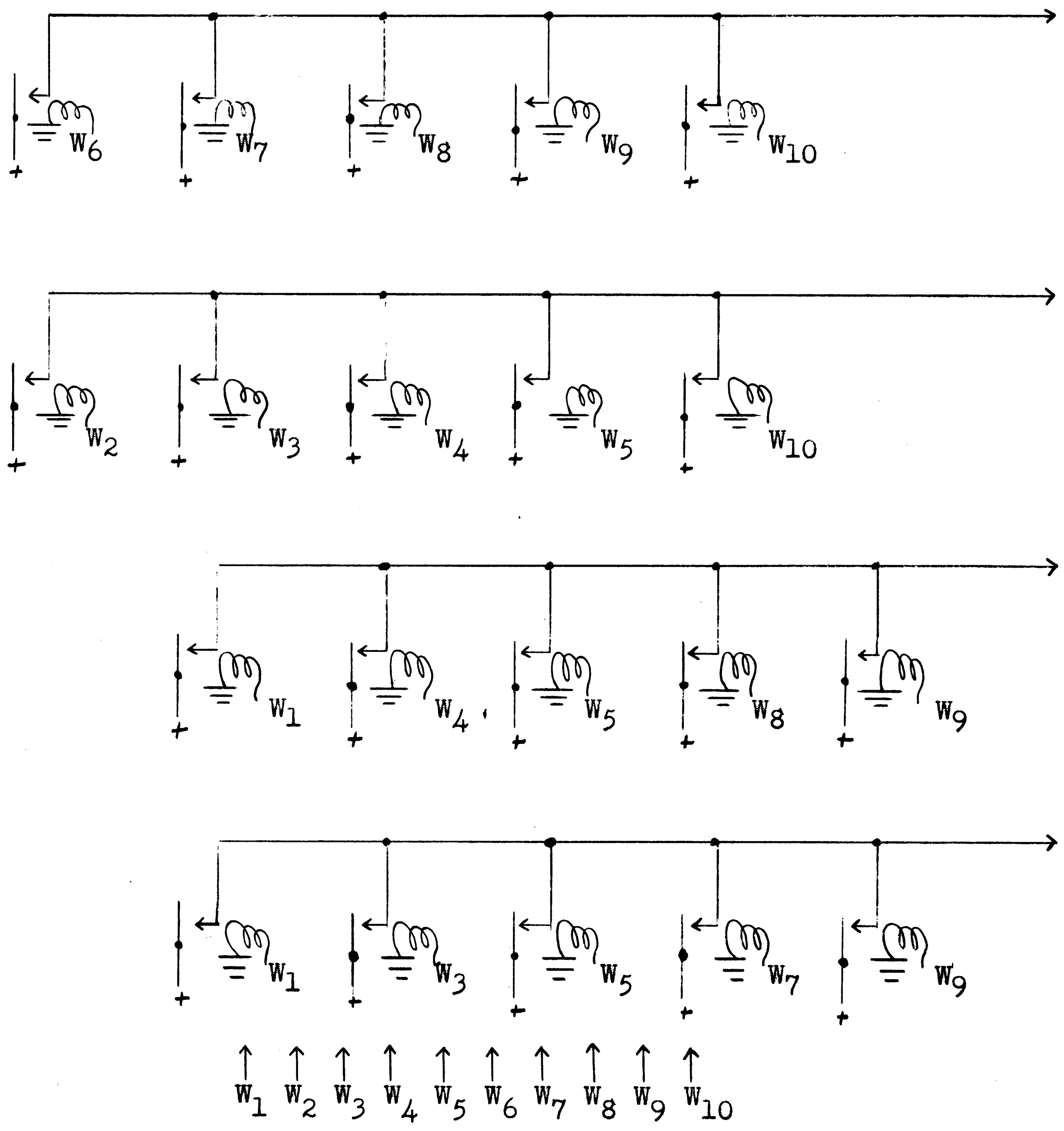
	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	$\overline{W}_1$	$\overline{W}_2$	$\overline{W}_3$	$\overline{W}_4$
0	1												1	1
1		1										1		
2			1									1		1
3				1								1	1	
4					1							1	1	1
5						1					1			
6							1				1			1
7								1			1		1	
8									1		1		1	1
9										1	1	1		

Fig. 8(B) and (C) give the crystal rectifier realization. Fig. 8(B) is drawn so as to emphasize the correspondence between circuit and net, and Fig. 8(C) is the same circuit in more conventional



Encoding Switch and Realizations

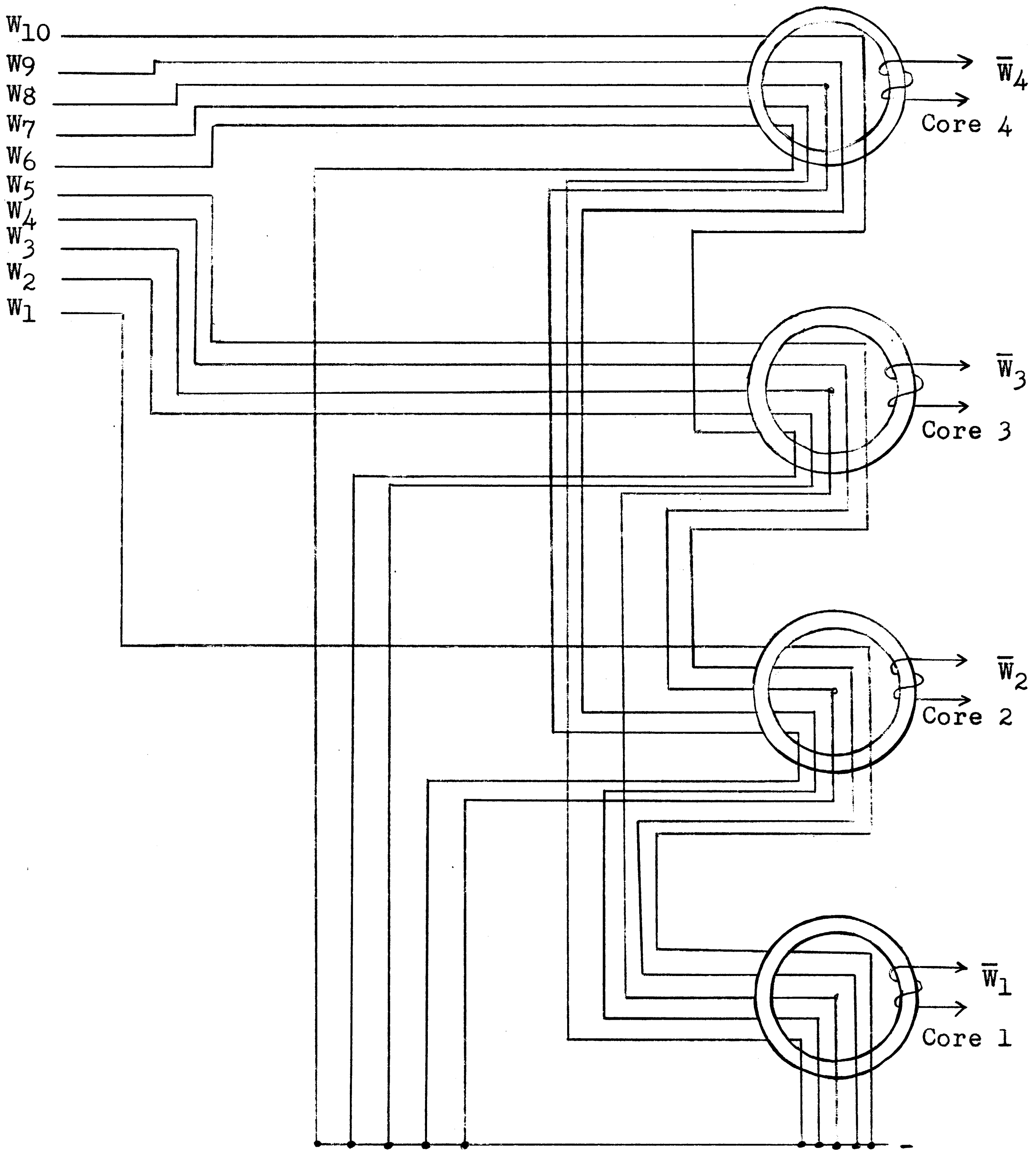
Fig. 8



(D)

Fig. 8

Relay Realization of the Net of Fig. 8(A)



Bell A.M.A. Transformer Realization of Net of Fig.8(A)
(E)

Fig. 8

form. The 0,1 convention used is $1 \longleftrightarrow$ high voltage, $0 \longleftrightarrow$ low voltage.

A relay realization is given in Fig. 8(D). The 0,1 convention used is $1 \longleftrightarrow +$ voltage, $0 \longleftrightarrow 0$ voltage. Here the circuit may be simplified by replacing all coils controlled by W_j by a single coil controlled by W_j and a set of make-contacts.

The element input count for the crystal realization is the number of crystals required; for the relay realization, the number of contacts. The parallel load of input wire W_j is the number of crystals operated in parallel by W_j. In the simplified relay circuit, it is the number of contacts operated by the coil controlled by W_j. The serial loading coefficient is ideal as it is in the case of the exponential switch.

In the relay circuit we have shown each contactor connected to a source of positive voltage. This was done in view of the convention that 1 represented positive voltage and 0 represented ground potential. A pulse output may be achieved by connecting the wires labeled $+$ to a pulse source.

This requires a multiple 0,1 convention.

Fig. 8(E) is a realization of the encoding switch in terms of the Bell A.M.A. transformers.

3.3. Arbitrary Transliterative Functions.

Any transliterative function can be realized according to Technique 3 of Vol. I by first decoding and then encoding. In the section on decoding circuits, only complete decoding circuits were considered. However, any arbitrary decoding function can be realized in physical equipment by a complete decoding circuit with the unnecessary parts deleted. For our purposes then, the circuits described in Subsections 3.1 and 3.2 or the simplification of these circuits may be taken as components from which to construct circuits for any given transliterative function.

On the other hand, it is sometimes possible to construct a single circuit which will realize the given function "directly" and which is, in a sense, simpler than the circuit constructed by first decoding and then encoding. Very little theory has been developed to cover this direct procedure.

In this subsection we will consider two examples of circuits constructed by the decoding-encoding technique and a single example of a circuit realizing a transliterative function directly. In all cases the 0,1 convention will be $1 \longleftrightarrow +$ voltage, $0 \longleftrightarrow 0$ voltage.

In the case of decoding nets, we were not primarily concerned with the actual correspondences produced by a switch.* Now, however, it will be necessary to consider these correspondences in detail because (1) two circuits are to be connected, the second of which, the encoding circuit, requires specification of the function it realizes, and (2) more generally, it will be desirable to eliminate equipment required only for correspondences with which we are not concerned. At the beginning of Section 3 a method was given for determining the state of every wire of the net. This can be directly translated into a description of the states of the corresponding wires of the circuit

*That is, with precisely <u>which</u> output wire is activated for a given input state, so long as the required complete function is produced.-Ed.

by means of the 0,1 convention. This information may also be determined from the circuit by assigning states to the inputs and using electrical properties of the circuit element to determine the states of the other wires.

This data yields the necessary correspondences. Let the outputs of a decoding switch be labeled $\overline{W}_j$ and assign to each input character d that $\overline{W}_j$ which is in the 1 state when d is represented on the inputs. Similarly, let the inputs of an encoding switch be labeled W_i and assign to each W_i that output character which appears in the output when W_i is in the 1 state. For example, consider the stepping switch MS and the stepping switch encoding circuit described in Subsection 2.3 and diagramed in Figs. 3 and 4, respectively. The MS correspondence is given in Table VI below, the encoding correspondence in Table VII.

These tables together with the given transliterative function determine at once the way that the wires of the two switches must be connected to yield the complete circuit.

Table VI

I_{11}	I_{12}	I_{13}	I_{14}	I_{21}	I_{22}	I_{23}	$\overline{W}$'s
			1			1	$\overline{W}_1$
			1		1		$\overline{W}_2$
			1	1			$\overline{W}_3$
		1				1	$\overline{W}_4$
		1			1		$\overline{W}_5$
		1		1			$\overline{W}_6$
	1					1	$\overline{W}_7$
	1				1		$\overline{W}_8$
	1			1			$\overline{W}_9$
1						1	$\overline{W}_{10}$
1					1		$\overline{W}_{11}$
1				1			$\overline{W}_{12}$

Table VII

W's	$\overline{W}_1$	$\overline{W}_2$	$\overline{W}_3$	$\overline{W}_4$
W_1				1
W_2			1	
W_3			1	1
W_4		1		
W_5		1		1
W_6		1	1	
W_7		1	1	1
W_8	1			
W_9	1			1
W_{10}	1		1	
W_{11}	1		1	1
W_{12}	1	1		

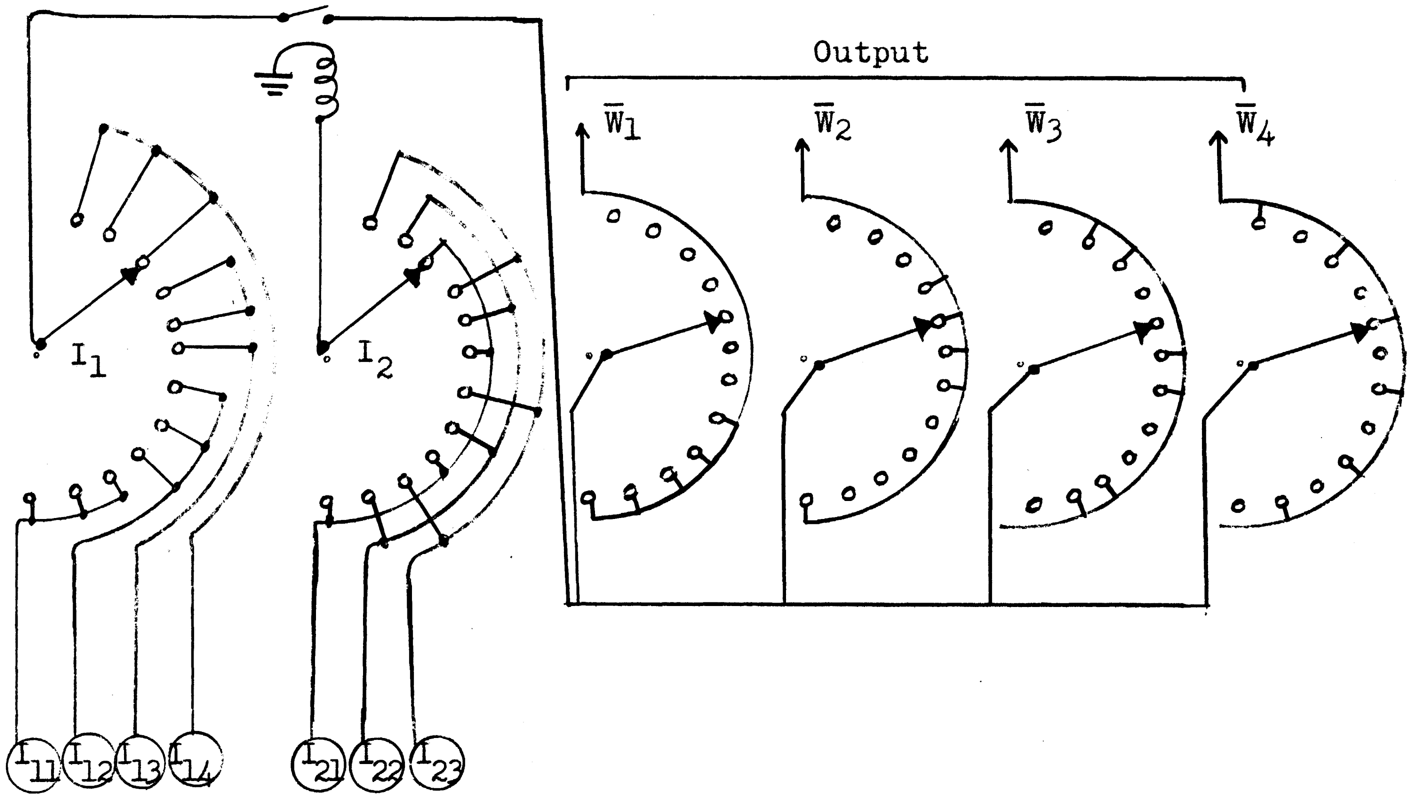
For example, let the transliterative function be given by joining Tables VI and VII and eliminating the $\overline{W}$ and W columns. This transliterative function can be realized by taking the two circuits of Figs. 3 and 4 and connecting $\overline{W}_j$ to W_j. The resulting circuit will realize the given transliterative function. In operation (though the control circuits are not shown) the MS stepping switch will step until it reaches the position determined by the states of the input wires and then the encoding switch will step until it reaches the corresponding position.

This circuit may, of course, be simplified and its operation materially speeded up by (1) using a single stepping switch with $N+M$ arcs (N being the number of input sets and M the number of outputs), and (2) with the contacts so connected that each row of contacts (one from each arc and touched simultaneously by the wiper) can be divided into two parts, one corresponding to an input character and the other to its image under the given transliterative function. This is illustrated in Fig. 9 for this example.

As a second example of the decoding-encoding technique, we construct a circuit realizing the transliterative function in Table VIII.

Table VIII

0000	$\longrightarrow$	0011	0101	$\longrightarrow$	1000
0001	$\longrightarrow$	0100	0110	$\longrightarrow$	1001
0010	$\longrightarrow$	0101	0110	$\longrightarrow$	1010
0011	$\longrightarrow$	0110	1000	$\longrightarrow$	1011
0100	$\longrightarrow$	0111	1001	$\longrightarrow$	1100



Decoding-Encoding by means of a Stepping Switch

Fig. 9

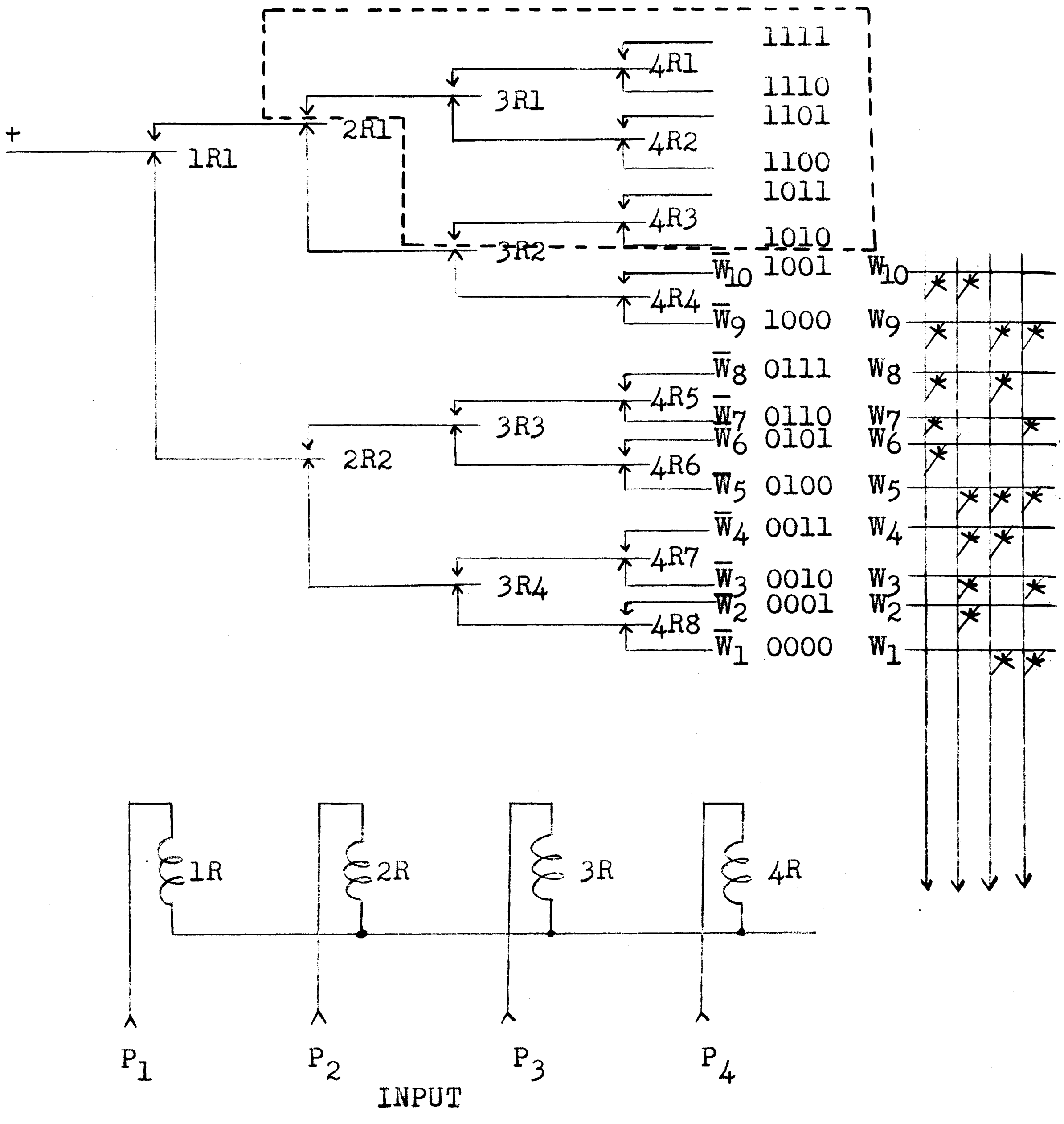
The relay tree of Fig. 10(A) is a realization of a 4-input-pair tree together with polarizer. The binary number labeling its output wires designates the input character for which that output is in the 1 state. Clearly not all the outputs are required and the heavy dotted lines indicate the portions of the circuit which may be eliminated. Further simplification may well be possible.

The encoding switch required is described in Table IX where the W_i are the inputs and where the character corresponding to W_i is the one appearing on the outputs when W_i is in the 1 state.

Table IX

$W_1 \longrightarrow$	0011	$W_6 \longrightarrow$	1000
$W_2 \longrightarrow$	0100	$W_7 \longrightarrow$	1001
$W_3 \longrightarrow$	0101	$W_8 \longrightarrow$	1010
$W_4 \longrightarrow$	0110	$W_9 \longrightarrow$	1011
$W_5 \longrightarrow$	0111	$W_{10} \longrightarrow$	1100

Fig. 10(B) is a crystal rectifier circuit realizing the required encoding function. The circuit for this given transliterative function is completed by connecting the $\overline{W}_i$ and W_i. The



(A) (B)

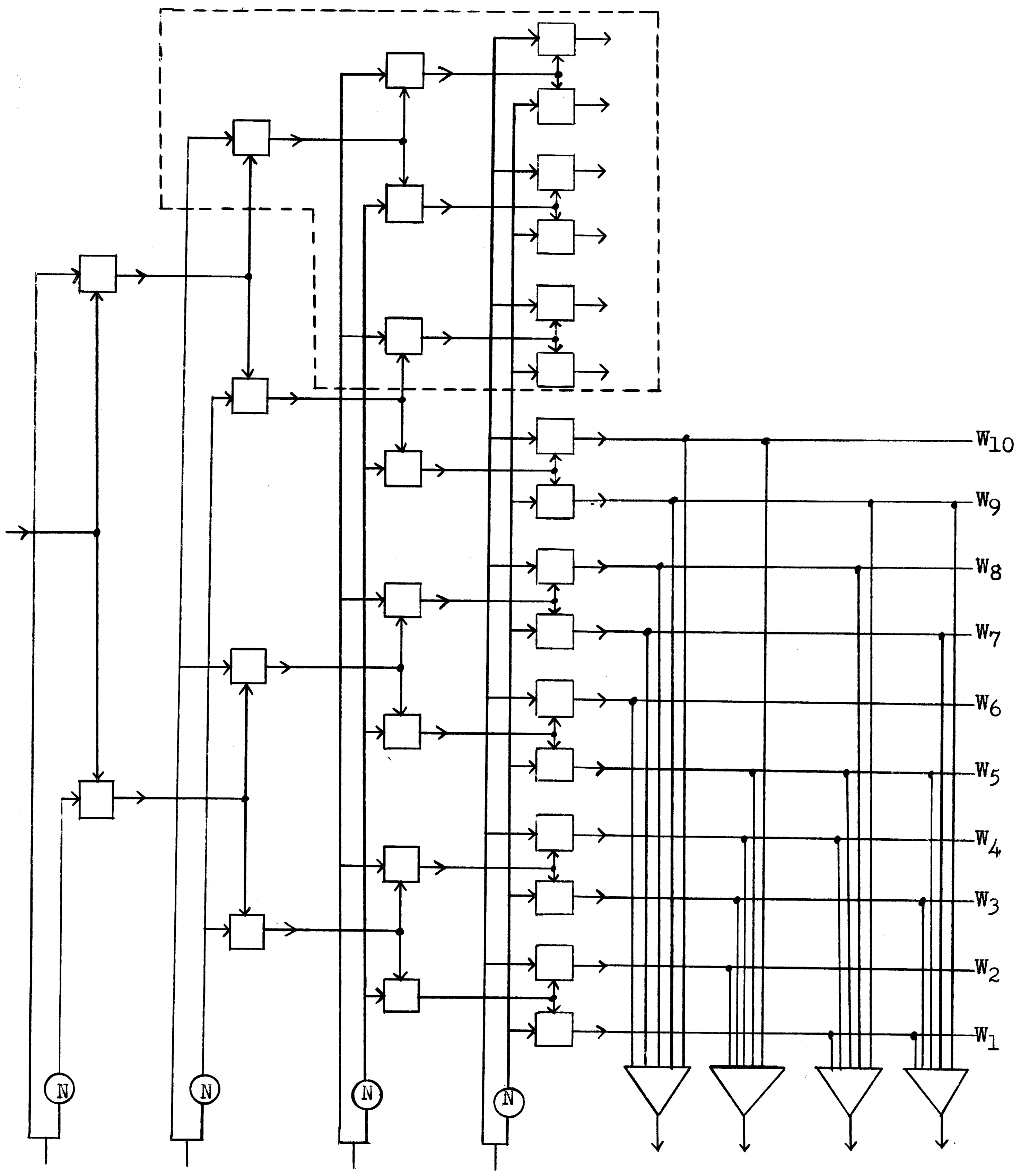
Relay Tree

Fig. 10

logical net for this transliterative function is given in Fig. 11.

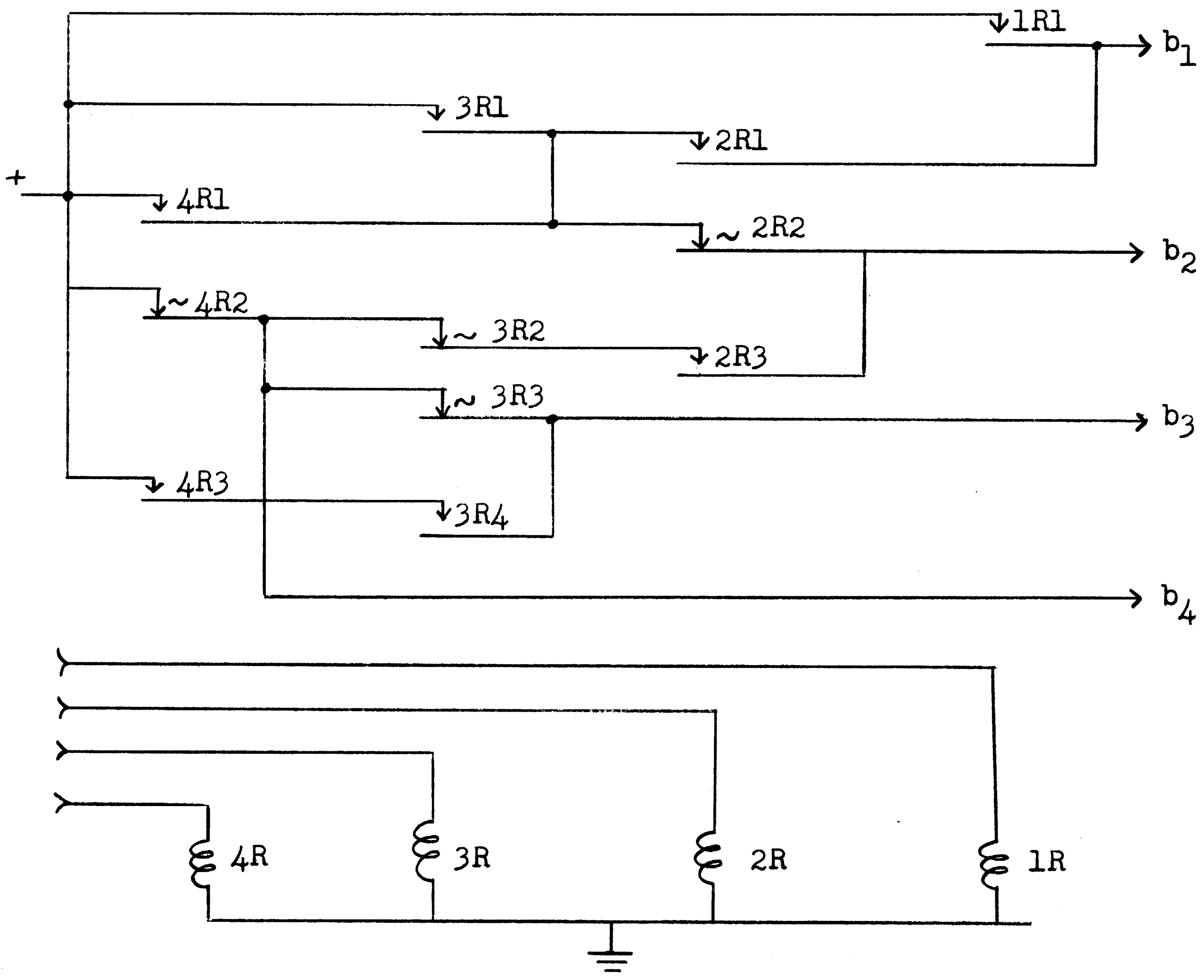
As a final example of circuits realizing transliterative functions, consider the relay circuit of Fig. 12. It realizes the same transliterative function (Table IX) as the circuit of Fig. 10. It does so directly, however, i.e., without first decoding and then encoding. Such direct transliterations usually take advantage of the incompleteness of the decoding and the special characteristics of the function to be realized (i.e., the specific nature of the correspondence it defines). Fig. 13 is the logical net describing the circuit of Fig. 12 (r stands for the reading input). The circuit may be checked by assigning 0's and 1's to the inputs and calculating the output states to see whether it actually realizes the given function. If variables p_i are assigned to the inputs in an appropriate fashion* well-formed formulas may be associated with every wire of the net describing

*If u and v are two input wires whose states are assigned independently, then the associated variables may be denoted p_i and p_j when $i \neq j$. If, on the other hand, u and v are considered a polar pair and p_i is assigned to one of them, then $\sim p_i$ must be assigned to the other.



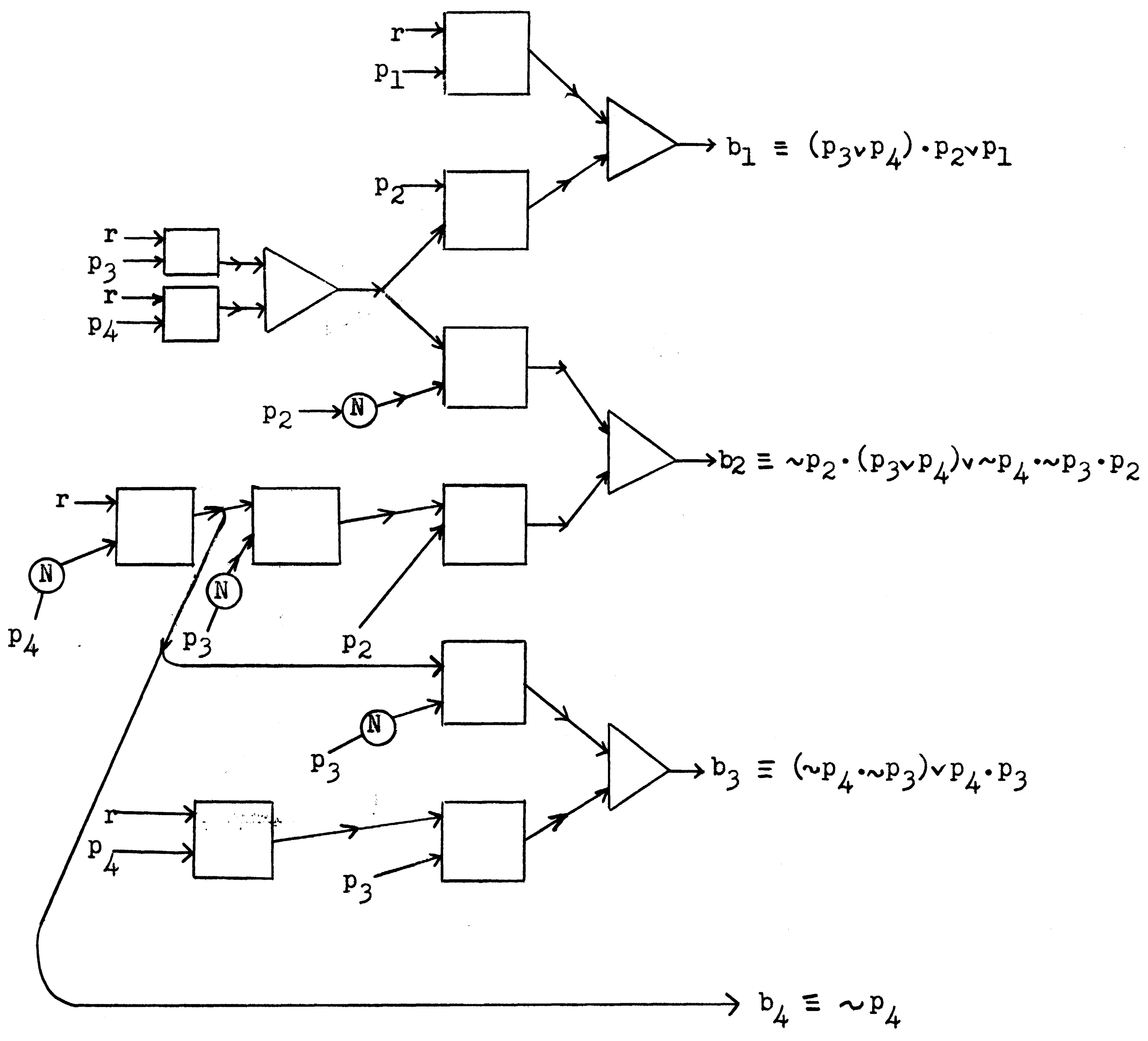
Logical Net for the Transliterative Function of Table VIII

Fig. 11



Direct Relay Realization of the Transliterative Function of Table VIII

Fig. 12


r
p_1
p_2
$b_1 \equiv (p_3 \vee p_4) \cdot p_2 \vee p_1$
r
p_3
r
p_4
p_2
N
$b_2 \equiv \sim p_2 \cdot (p_3 \vee p_4) \vee \sim p_4 \cdot \sim p_3 \cdot p_2$
r
N
p_4
N
p_3
p_2
N
p_3
$b_3 \equiv (\sim p_4 \cdot \sim p_3) \vee p_4 \cdot p_3$
r
p_4
p_3
$b_4 \equiv \sim p_4$

Logical Net for the Circuit of Fig. 12

Fig. 13

that wire's behavior in terms of logical functions and the input variables. The logical formulas associated with the b_i in Fig. 13 are the well-formed formulas describing the behavior of the outputs.

It is interesting to compare the two nets (Figs. 11 and 13) in terms of the various indices. Let S_1 denote the decoding-encoding net and S_2 the direct net. A comparison in terms of the element count and the element input count is given in the following table.

Switch	Element Count	Element Count by Types	Element Input Count
S_1	28	$N^1_4 K^2_{20} D^5_4$	64
S_2*	19	$N^1_4 K^2_{11} D^2_4$	37

The theory of Vols. I and III is concerned largely with nets for complete decoding, and these are constructed primarily of conjunctions. Because of this, an element count would be of some significance.

*These counts can be reduced by better handling of the r-input. In fact, one might question the necessity for including <u>any</u> read-circuits in this comparison.-Ed.

In nets composed of different types of logical elements, as are S_1 and S_2, a breakdown by types is more helpful and this is indicated in the third column of the preceding table. The letters with superscripts denote the type of logical element (N stands for negation, K for conjunction, and D for disjunction), and the superscript indicates the number of inputs. The subscript indicates the number of that type of element appearing in the net.

This yields the expression in the third column which is, in a sense, analogous to chemical formulas. The element input count is given in the first column. It too would be more significant if broken down in an analogous way. For example, in S_1 the element input count for the disjunction is that of the encoding switch and is equal to the number of crystal rectifiers. However, in S_2 the element input count for the disjunction lacks significance because it is possible to realize them in this case by merely joining wires.

This illustrates one significant way of

breaking the net up into parts; in the case of S_1, the decomposition into element types corresponds to the decomposition of the net into polarizer, decoding net, and encoding net. Many other significant ways of breaking up the net undoubtedly exist, and, for each, values of the different indices may be determined.

4. Code Converters Requiring Memory.

In this concluding section three examples of code conversion requiring memory are described. They illustrate several ways in which circuits for transliterative functions may be combined with other circuits (e.g., memory, sensing, and control circuits) to realize more complicated conversions. As presented, they embody an element of format conversion for the input is in parallel order and the output in serial order.

The discussion for each example includes a table defining the function to be realized, a schematic diagram of an appropriate code converter, and a behavior table succinctly describing its behavior. Little detail is included for here we are interested not so much in the circuits as in the broad organization of the components. Nevertheless, a schematic circuit diagram corresponding to a part of the block diagram for Example I is included as an illustration.

It should be mentioned that the only portions of the control which are considered here are those needed for the conversion of one character of the input code into the corresponding character of the output code.

The code conversion to be considered first is that of a teletype[*] code into a 4-8 code. This is not a transliterative function for two reasons: (1) the mapping is not single valued, and (2) the image sequences are not all of equal length. This conversion is fully defined in the following function table.

Table X

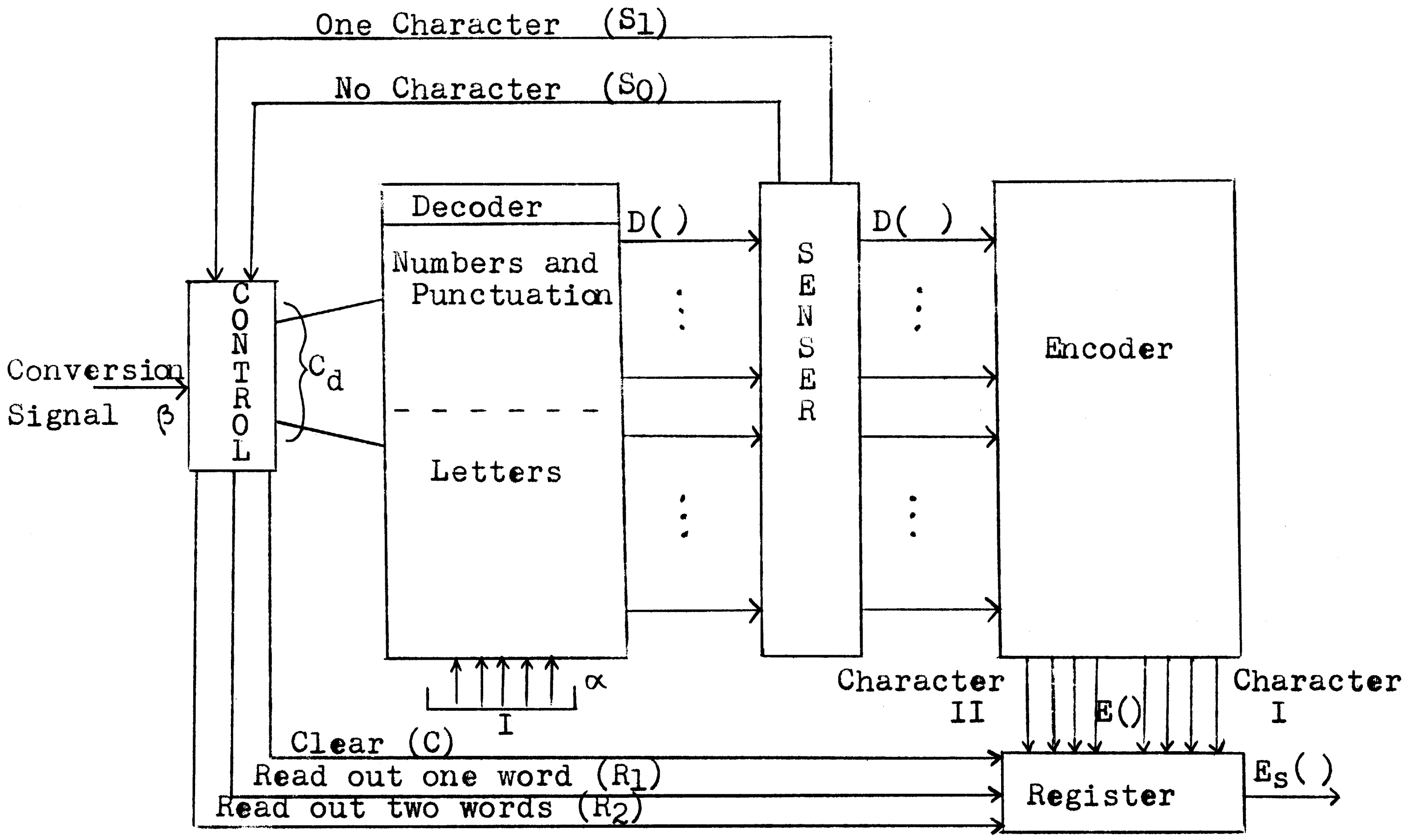
	Teletype	4 - 8		Teletype	4 - 8		Teletype	4 - 8
A	11000	0000 0000	S	01010	1101 0000	/	? 11001	1110 1111
B	10101	0000 0001	T	00001	1101 0001	.	11000	1110 0010
C	10110	0000 0010	U	11010	1101 0010	@	? 11001	1110 1110
D	01100	0000 1101	V	01111	1101 1101	%	10010	1111 1110
E	01000	0000 1110	W	10011	1101 1110	*	01111	1111 1101
F	01110	0000 1111	X	11101	1101 1111	:	01011	1111 1111
G	00111	0001 0000	Y	11001	1110 0000	,	01010	1110 1101
H	01001	0001 0001	Z	10001	1110 0001	$	01100	1111 0000
I	10010	0001 0010	Ø	10110	0011	space	00110	1111 0001
J	11100	0001 1101	1	10111	0100	-	00101	1111 0010
K	11110	0001 1110	2	10011	0101	letters	11111	
L	00011	0001 1111	3	00100	0110	figs.	11011	
M	01101	0010 0000	4	01001	0111			
N	00110	0010 0001	5	10000	1000			
O	00101	0010 0010	6	10101	1001			
P	01011	0010 1101	7	00111	1010			
Q	10111	0010 1110	8	00011	1011			
R	10100	0010 1111	9	11000	1100			

[*]The code referred to here as a 'teletype code' is a five channel shift code with the structure of a teletype code but differing to some extent in the specific assignment of characters.-Ed.

Fig. 14 contains the block diagram of a code converter realizing this function.

The character to be converted, α, is read onto the inputs of a decoder; then a conversion signal is applied to the control which puts a signal on one of the pair of wires to the decoder. The wire chosen depends upon the last shift character X read in. Clearly memory is required at this point. α is decoded and the output $D(X,\alpha)$ goes to the senser and encoder. It is immediately encoded and transmitted to the register.

The senser senses not α alone but (X,α) for α alone is ambiguous. The "no character" output of the senser has a signal if and only if α is a shift character and the "one character" output has a signal if and only if (X,α) defines a a number. If the "no character" output carries a signal, then the register is cleared without an output occurring on the register output. If the "one character" output carries a signal, then one character is read out of the register. Finally, if neither carries a signal, two characters are read out of the register in serial order.



Example I

Schematic Diagram

Fig. 14

The "behavior" of the diagram may be described by means of a table where the rows correspond to inputs and outputs and the columns roughly represent successive points of time. The entries in a column may be considered as occurring at approximately the same time, those to the right later, and those the left earlier. 0 and 1 are used to represent the two possible states of the wires but no physical interpretation is given. The table for Example I is given on the following page.

Fig. 15 shows a schematic circuit which includes the decoder, encoder, senser, and a part of the control circuit.

The second example of code conversion is from 6 pulse to teletype. Here memory is required for the proper insertion of the shift characters. The following incomplete table defines (in part) the code conversion functions to be considered (p. 80).

Behavior Table - Example I

	0	1	2	3
Character Input of Decoder	α			
Conversion Signal Input		β		
Control Output to Decoder (X is first shift character preceding α)		$C_d(X)$		
Decoder to Senser and Encoder		$D(X,\alpha)$		
Encoder to Register		$E(X,\alpha)$		
Case I: $(X,\alpha) \rightarrow$ a letter				
No Character Wire from Senser		$S_0 = 0$		
One Character Wire from Senser		$S_1 = 0$		
Control to Register - Read out two characters			$R_2 = 1$	
Other Control Outputs			$C, R_1 = 0$	
Register Output				$E_s(X,\alpha)$
Case II: $(X,\alpha) \rightarrow$ a figure				
No Character Wire from Senser		$S_0 = 0$		
One Character Wire from Senser		$S_1 = 1$		
Control to Register - Read out one character			$R_1 = 0$	
Other Control Outputs			$C, R_2 = 0$	
Register Output				$E_s(X,\alpha)$
Case III: $(X,\alpha) \rightarrow$ Shift character				
No Character Wire from Senser		$S_0 = 1$		
One Character Wire from Senser		$S_1 = 0$		
Control to Register - clear			$c = 1$	
Other Control Outputs			$R_1, R_2 = 0$	
Register Output				------

Bruned Figure Page

See ERI File Copy

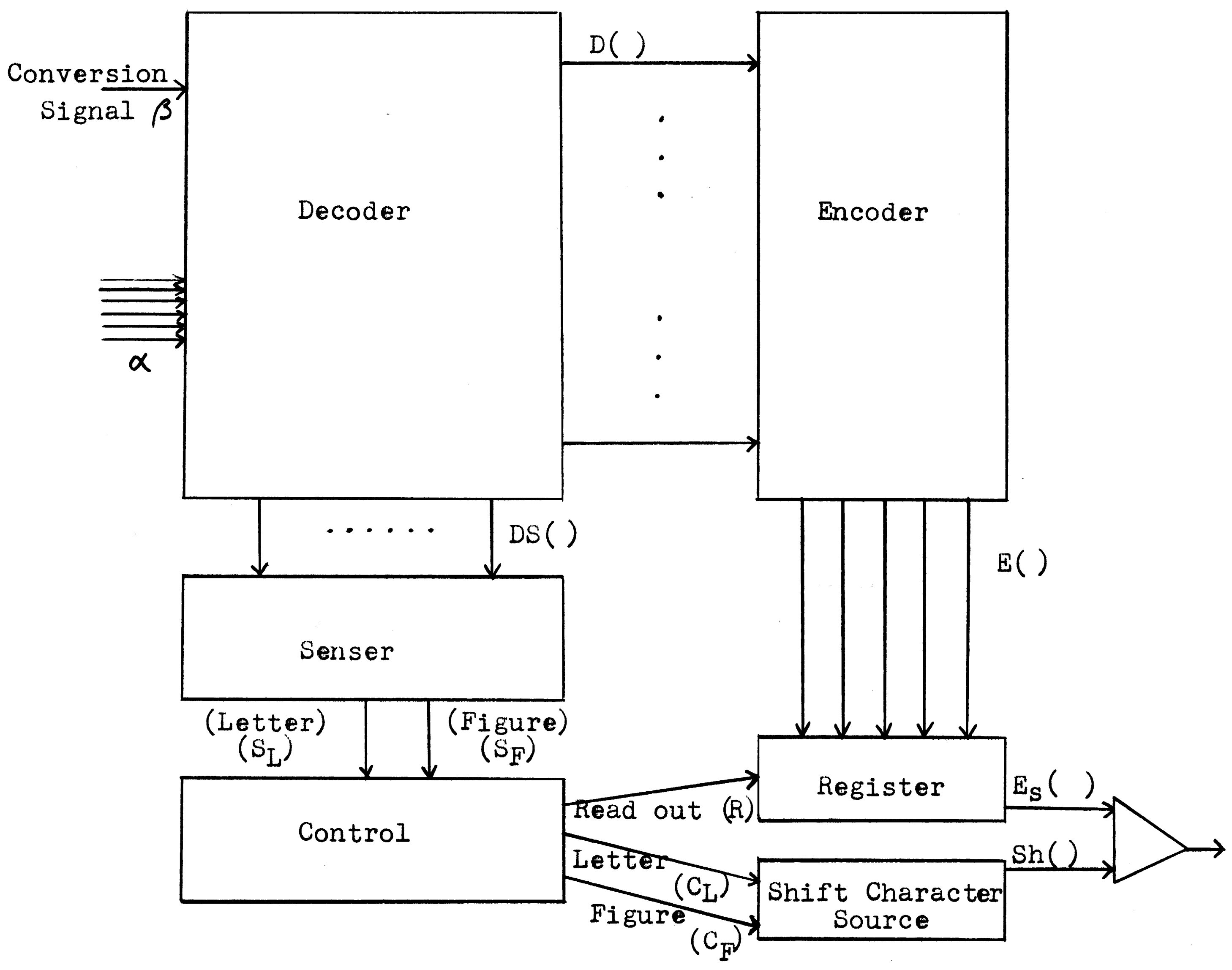
	6 Pulse	Teletype		6 Pulse	Teletype
A	010000	00011	Z	101010	10001
B	010010	11001	Ø	000011	10110
C	010011	01110	⋮		
D	010100	01001	9	001100	11000
E	010101	00001	:	110100	01110
F	010110	01101	⋮		
G	010111	11010	$	101111	01001
.	.	.	/	101110	11101
.	.	.	@	101101	11001
.	.	.	,	101100	01100
X	101000	11101	.	101011	11100
Y	101001	10101			

Fig. 16 is the schematic diagram of a code converter realizing this function.

The character to be converted, α, is read onto the inputs I of the decoder. A conversion signal then decodes α, putting DS(α) on the inputs of the senser and D(α) on those of the encoder. D(α) is then encoded and transmitted to the register.

The letter output of the senser carries a signal if and only if α is a letter, and the figure output carries a signal if and only if α is a figure.

The control must remember on which input the preceding signal occurred. If there is a change,



Example II

Schematic Diagram

Fig. 16

the appropriate shift character must be inserted. This is done by first putting a shift character in the output and then reading out the register.

If there is no change, then the control causes the character to be read out of the register (in serial order) as indicated by the subscript in $E_s(\alpha)$. The behavior table for Example II is given on the following page.

The final example is a conversion from a 4-8 bit code to a 6 bit code. The function which is to be realized is given in the following table.

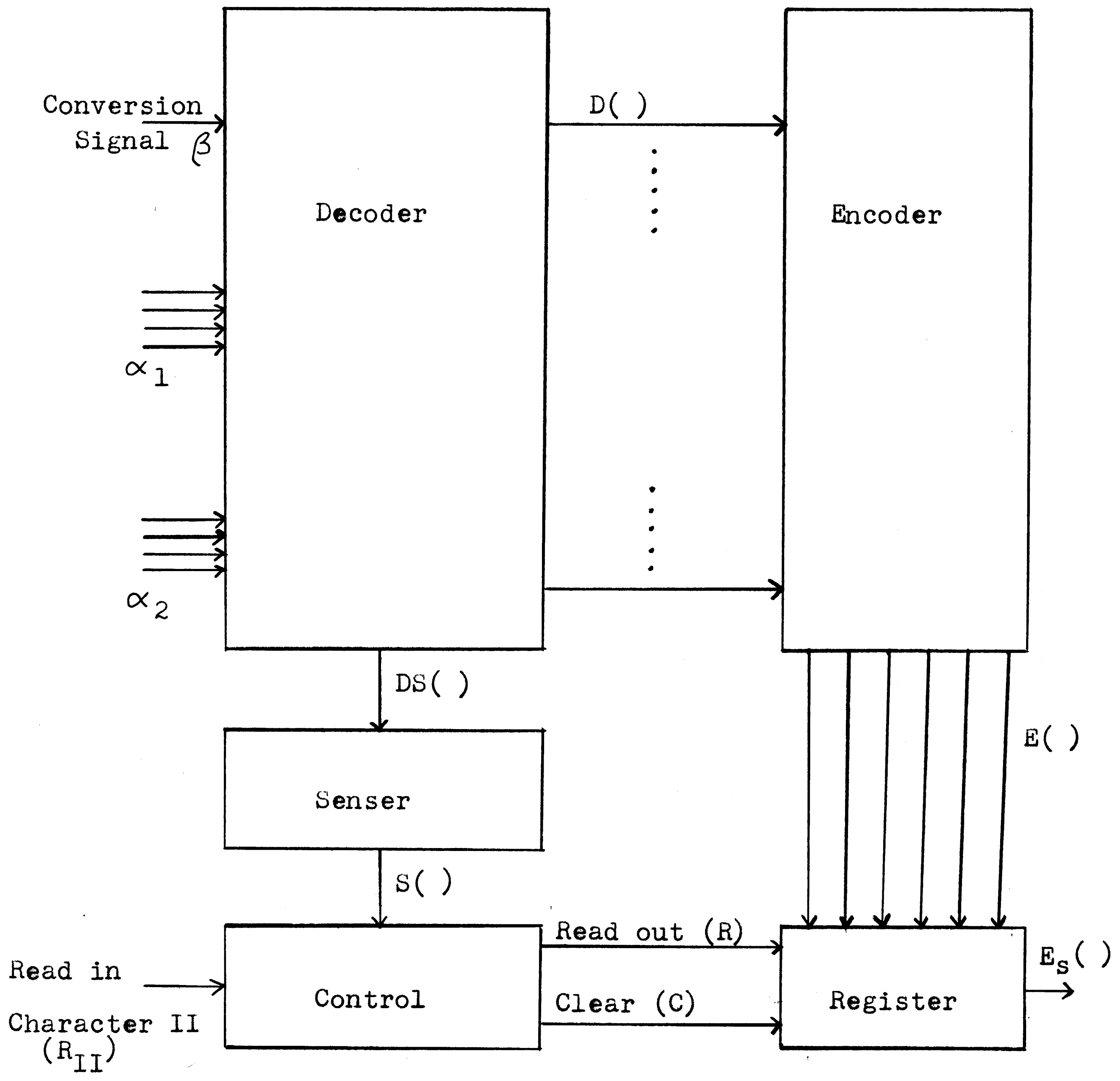
	4 - 8	6 Pulse		4 - 8	6 Pulse		4 - 8	6 Pulse
A	0111 1111	010001	S	1010 1111	100011	.	1001 1101	101011
B	0111 1110	010010	T	1010 1110	100100	,	1001 0010	101100
C	0111 1101	010011	U	1010 1101	100101	@	1001 0001	101101
D	0111 0010	010100	V	1010 0010	100110	/	1001 0000	101110
E	0111 0001	010101	W	1010 0001	100111	$	1000 1111	101111
F	0111 0000	010110	X	1010 0000	101000	Space	1000 1110	110000
G	0110 1111	010111	Y	1001 1111	101001	-	1000 1101	110001
H	0110 1110	011000	Z	1001 1110	101010	*	1000 0010	110010
I	0110 1101	011001	Ø	0100	000011	%	1000 0001	110011
J	0110 0010	011010	1	0011	000100	:	1000 0000	110100
K	0110 0001	011011	2	0010	000101			
L	0110 0000	011100	3	0001	000110			
M	0101 1111	011101	4	0000	000111			
N	0101 1110	011110	5	1111	001000			
O	0101 1101	011111	6	1110	001001			
P	0101 0010	100000	7	1101	001010			
Q	0101 0001	100001	8	1100	001011			
R	0101 0000	100010	9	1011	001100			

The schematic diagram for a converter for realizing this function is given in Fig. 17.

Behavior Table - Example II

	0	1	2	3	4	5
Character Input of Decoder (I)	α					
Conversion Signal Input		β				
Decoder to Encoder		$D(\alpha)$				
Decoder to Senser		$DS(\alpha)$				
Encoder to Register		$E(\alpha)$				
Case I: If α = letter and X* = letter						
Senser Output - letter		$S_L=1$				
Senser Output - figure		$S_F=0$				
Control to Register - read out			$R = 1$			
Other Control Outputs			$C_F, C_L=0$			
Register Output				$E_S(\alpha)$		
Case II: If α = figure and X = figure	Analogous to Case I					
Case III: If α = letter and X = figure						
Senser Output - letter		$S_L=1$				
Senser Output - figure		$S_F=0$				
Control to Shift Source - letter			$C_L=1$			
Other Control Outputs			$R, C_F=0$			
Shift Source Output				Sh(L)		
Control to Register - read out					$R=1$	
Other Control Outputs					$C_F, C_L=0$	
Register Output						$E_S(\alpha)$
Case IV: If α = figure and X = letter	Analogous to Case III					

*X is the preceding character.



Example III

Schematic Diagram

Fig. 17

The input character α_1 is read onto the I_1 input set of the decoder. A conversion pulse then "decodes" α_1 and puts $D(\alpha_1)$ on the inputs of the encoder and $DS(\alpha_1)$ on the input of the senser. $D(\alpha_1)$ is encoded and transmitted to the register. In Case I, if α_1 is a figure, the senser output is unaffected and the control causes the register to read out. In Case II, if α_1 is not a figure, there is an output from the senser which causes the control to clear the register and have α_2 read onto the I_2 input set of the decoder. The next conversion pulse decodes $(\alpha_1 \cup \alpha_2)$ which is then encoded and transmitted to the register. The control causes the character in the register to be read out and the conversion is complete.

Clearly memory is required in this code conversion, (1) for retaining α_1, while it is being sensed and then for combination with α_2 if required, and (2) in the control to interpret the signal from the senser, i.e., the second signal from the senser causes the control to make the register read out whereas the first one caused the register to be cleared.

Behavior Table - Example III

	0	1	2	3	4	5	6
Decoder Input I_1	α_1						
Conversion Signal Input		β_1					
Decoder to Encoder		$D(\alpha_1)$					
Decoder to Senser		$DS(\alpha_1)$					
Encoder to Register		$E(\alpha_1)$					
Case I: If α_1 = fig. Senser to Control		$S(\alpha_1)=0$					
Control to Register -read out			$R=1$				
Other Control Outputs			$R_{II}=0$ $C=0$				
Register Output				$E_S(\alpha)$			
Case II: If $\alpha_1 \neq$ fig. Senser to Control		$S(\alpha_1)=1$					
Control to Register - clear			$C=1$				
Control to Register - read out			$R=0$				
Control Output - Character II in			$R_{III}=1$				
Decoder Input I_2				α_2			
Conversion Signal Input					β_2		
Decoder to Encoder					$D(\alpha_1+\alpha_2)$		
Decoder to Senser					$DS(\alpha_1+\alpha_2)$		
Encoder to Register					$E(\alpha_1+\alpha_2)$		
Senser to Control					$S(\alpha_1+\alpha_2)=1$		
Control to Register - read out						$R=1$	
Control to Register - clear						$C=0$	
Control Output - Character II in						$R_{II}=0$	
Register Output							$E_S(\alpha_1+\alpha_2)$

Bibliography

1. Brown, D.R., and N. Rochester, "Rectifier Networks for Multiposition Switching," Proceedings, I.R.E., Vol. 37, 1949.

2. Staff of the Computation Laboratory, "Synthesis of Electronic Computing and Control Circuits," Annals, Computation Laboratory of Harvard University, Vol. 27, Cambridge, 1951.

3. Specification for Pulse-Control Equipment, April, 1951, Burroughs Corporation Research Center, Paoli, Pennsylvania.

4. Keister, W., A. E. Ritchie, and S. H. Washburn, The Design of Switching Circuits, New York, 1951.

Corrigenda for Volumes I and III, LANGUAGE CONVERSION FOR DIGITAL COMPUTERS

Volume I:

p. 10, last line of Def.1., should read

"see Figure 3a or 5b"

p. 10, the following to be added as a footnote to the Corollary to Def. 1.

"The associative property of Cartesian Conunction was established somewhat inaccurately, as was pointed out by L. C. Robbins, Jr., Burroughs Corporation Research Center. However, both with regard to Cartesian Product and Cartesian Conjunction, a "quasi-associativity" actually obtains, in the sense that an element, (x_1,x_2), x_3, of the Cartesian Product $(S_1xS_2)xS_3$ can be identified with the element (x_1,x_2,x_3) of $(S_1xS_2xS_3)$ or $x_1,(x_2,x_3)$ of $S_1x(S_2xS_3)$. It is recognized that there are other respects in which this discussion is not fully rigorous, but it is felt that such a presentation is adequate for present purposes."

p. 19, the second line of 2)a), following "of an MS" should read

"not belonging to the partially completed net and with input"

p. 21, line 8, following "into two such which differ" should read

"as to the number of P's by at most one. The"

p. 21, second paragraph, line 2 should read

"to build it. An ..."

p. 21, fifth paragraph, third line, following "the Element Input Count is" should read

"$C = 2\sum_{i=0}^{k-1} 2^{(i+2^{k-i})}$."

p. 22, last line of second paragraph, following "upon which" should read

"none of the subsequent theory will depend."

p. 24, line 2 of second paragraph, following "Figure 7e shows", should read

"a Stroke Tree of 2 Bays (3 in-"

p. 25, line 6 of footnote, following "for reasons which," should read

"it is felt, ..."

p. 39, line 2 of sixth paragraph should begin

"$2\sum_{i=0}^{k-1} 2^{(i+2^{k-i})}$, where the number ..."

p. 47, line 2 of fifth paragraph, following "for n - 6," should read

"for $d+F = (64-2)/(6-1) = 12+2/5$;"

p. 49, line 1 of Step Three (A), following "If $a_1 = 2$" should read

"then $b_i = \left\lceil \frac{a_i + \Delta_i}{2} \right\rfloor$ and $c_i = \left\lceil \frac{a_i + 1 - \Delta_i}{2} \right\rfloor$

Volume III:

p. 37, line 6 up from bottom, should begin with the expression

"$(k - 3)a_1 \leq \sum_4^k a_i$; $(k - 3)a_2 \leq \sum_4^k a_i$; ..."

p. 39, line 6 of Proof of Lemma 1, should read

"tion when $b = b_1$ or b_3 ; ..."