fects of the first member of the next set of changes. Thus

$$\delta_2 + d_m \geq d_M + D_M + d_M$$

or

$$\delta_2 \geq D_M + 2d_M - d_m = 3\delta_1 + 44. \quad (10)$$

If inertial delay elements are *also* used in the output leads, then the Moore form is not required and **(10)** can be relaxed to allow error signals of duration less than D_m at the logic outputs so

$$\delta_2 \geq D_M - D_m + 2d_M - d_m = 2\delta_1 + 36. \quad (11)$$

C. *Unrestricted Input Change*

I have treated this topic at some length in an earlier paper [3]. Here I shall briefly summarize, supplement and correct the previous results.

One cause of unsatisfactory operation is the possibility of a y_i-change taking place and not being perceived at the Y, terminal before y, changes as a result of some later input changes. As noted earlier, this cannot occur if

$$D_m \geq 2d_M - d_m. \quad (12)$$

The other basic problem is that different y's might change as a result of each seeing a different input state. Where there are k input variables and where **r** is the recovery time of the inertial delay elements, this possibility can be avoided by

$$D_m \geq (k-1)(r + d_M - d_M) + d_M - d_m \quad (13)$$

and by not allowing any one-input variable to change twice during any interval Δ, where

$$\Delta = k(r + d_M - d_m). \quad (14)$$

Note that this last item contradicts the concept of "unrestricted input change." It can be satisfied without such a violation by simply inserting in each input lead an inertial delay element with a magnitude corresponding to (14).

It should be pointed out that, since **(13)** does not necessarily dominate (12), D_m must be chosen to exceed the *maximum* of these expressions. To avoid the need for more than one delay element size in cases where delays are to be inserted in the input leads, it may be desirable to use the maximum of **(12)** and **(14)** as the uniform delay element size. Expressions (9) and **(10)** define δ_1 and δ_2. The latter was incorrectly specified in my earlier paper [3].

In that earlier paper the constraints that had to be satisfied by a USTT state assignment for UIC operation were shown to be somewhat complex. I have since realized that the Tan–Liu assignment [4], [6], in which a y-variable is associated with each destination set is always satisfactory. Although far from minimal in terms of the number of state variables required, these assignments are reasonably efficient with respect to logic requirements.

V. Conclusions

SS circuits can be used to realize functions with asynchronous SIC or MIC circuits. In cases where Huffman type sequential circuits are costly in terms of design time and circuit complexity, due to the need for complex state assignments necessary to avoid criticial races, the SS circuits may be significantly more economical. It should be noted that this is by no means always the case, and in fact where simple state assignments suffice, the Huffman circuit is likely to be simpler since it does not require either FF's or the special pulse producing circuits.

The *CD* type circuit will often be simpler than the RV type because of the additional unspecified states for the JK circuits. However, this effect may be more than cancelled out if the pulse generating circuit, which has no inherent unspecified entries, turns out to be complex.

Neither of the SS forms discussed here appear to be adaptable for UIC operation, which *can* be handled by appropriately designed Huffman circuits.

The Huffman type is in general the fastest (in terms of the rate at which input changes can be accepted), particularly for SIC operation, and even more so in the case where delay-free realizations are feasible. But this speed advantage is not great for MIC operation as compared with the RV type. The CD circuits are slower by a factor of about 2.

It should be noted that synchronous circuits based on edge triggered or MSFF's are similar to circuits with two-phase clocks and are comparable in speed to the circuits discussed above.

References

[1] H. Y. H. Chuang and S. Das, "Synthesis of multiple-input change asynchronous machines using controlled excitation and flip-flops," IEEE *Trans. Comput.*, vol. C-22, pp. 1103–1109, Dec. 1953.

[2] C. A. Rey and J. Vaucher, "Self-synchronized asynchronous sequential machines," IEEE *Trans. Comput.*, vol. C-23, pp. 1306–1311, Dec. 1974.

[3] S. H. Unger, "Asynchronous sequential switching circuits with unrestricted input changes," IEEE *Trans. Cornput.*, vol. C-20, pp. 1437–1444, Dec. 1971.

[4] ——, *Asynchronous Se'quential Switching Circuits.* New Uork: Wiley-Interscience, 1969.

[5] A. D. Friedman and P. R. Menon, "Synthesis of asynchronous sequential circuits with multiple-input changes." IEEE *Trans. Electron Comput.*, vol. C-17, pp. 559–566, June 1968.

[6] C. J. Tan, "State assignments for asynchronous sequential machines," *IEEE Trans. Comput.*, vol. C-20, pp. 382–391, Apr. 1951.

[7] R. M. Keller, "Towards a theory of universal speed-independent modules," IEEE *Trans. Comput*, vol. C-23, pp. 21–33, Jan. 1954.

Design of Ternary COS/MOS Memory and Sequential Circuits

H. T. MOUFTAH AND I. B. JORDAN

Abstract—Ternary storage elements are realized using ternary operators and fundamental circuits, designed with the COS/MOS integrated circuits. Several ternary flip-flops (tri-flops) are constructed and described in detail: the PZN (set positive, set zero, and set negative), the clocked PZN, the D-type, and the T-type. Ternary shift registers and ring counter are formed by means of these tri-flops. A master–slave T-type tri-flop is used for the construction of a ternary up counter able to count from 0 to 3^n using the normal ternary code or from $-(3^n - 1)/2$ to $+(3^n - 1)/2$ when the signed-ternary code is employed. With a small modification, a ternary down counter is also constructed. A divide-by-M ternary counter which can be programmed is described. A memory cell is designed for the construction of a ternary random-access-memory array (TRAM). A ternary decoder and encoder are presented to be the elements of a complete ternary read-only memory (TROM). A modified ternary inverter (MTI) is taken as a unit cell of the ternary memory matrix.

***Index Terms*—COS/MOS integrated circuits, fundamental ternary circuits, multiple-valued logic, programmable divide-by-N counters, ternary counters, ternary flip-flops, ternary logic implementation, ternary memory cells, ternary sequential logic.**

Introduction

With the advancing electronic technology, interest in multivalued logic, and especially in the ternary one, has grown rapidly. Much work has been done on ternary combinational logic circuits [1]–[4], but few studies have been centered on the design of ternary sequential circuits. This may be attributed to the lack of suitable ternary memory elements. The use of integrated circuits in designing ternary memory elements may be a good solution to this problem.

Due to some properties of the Complementary-Symmetry Metal Oxide Semiconductors (COS/MOS), one may use them in the design of ternary

Manuscript received January 10, 1975; revised July 15, 1975 and April 20, 1976.

This work was supported by the National Research Council of Canada.

H. T. Mouftah is with the Department of Electrical Engineering, University of Toronto, Toronto, Ont., Canada.

I. B. Jordan is with the Department of Electrical Engineering, Laval University, Quebec City, P.Q., Canada.

logic circuits--of course, in a way which differs somewhat from designing normal binary logic circuits. In this paper, ternary memory elements are designed with the COS/MOS integrated circuits. This is done by means of the ternary operators and fundamental circuits previously described in [5]–[8]. Several ternary sequential circuits are then constructed using these memory elements.

All circuits presented in this paper have been realized using the CD4007AE and the Quad Bilateral Switch CD4016AE RCA COS/MOS integrated circuits. The first chip is composed of three n-channel and three p-channel enhancement-type MOS transistors. The second one contains four independent bilateral signal switches, each consisting of an n-channel and a p-channel device.

TERNARY OPERATORS AND FUNDAMENTAL CIRCUITS

The three voltage levels, low, intermediate, and high, are presented here by 0, 1, and 2, respectively. The simple ternary inverter (STI), the positive ternary inverter (PTI), and the negative ternary inverter (NTI) are the three unary operators considered (Table I). The ternary OR (TOR) and ternary AND (TAND) represent two multiple entry operators. These operators can be defined by (1)–(4), respectively:

$$\text{STI: } \overline{X^1} = 2 - X \tag{1}$$

$$\text{PTI, NTI: } \overline{X^i} = \begin{cases} i & \text{if } X \neq i \\ 2 - i & \text{if } X = i \end{cases} \tag{2}$$

where i takes the value of 2 for the PTI and 0 for the NTI operator.

$$\text{TOR: } X \vee Y = \max(X, Y) \tag{3}$$

$$\text{TAND: } X \wedge Y = \min(X, Y). \tag{4}$$

The minus sign in (1) and (2) represents arithmetic subtraction.

General theorems and relationships governing these ternary operators and their proofs are presented in [7] and [8]. Here only those which will be used in the design described in this report are mentioned.

DeMorgan's law:

$$\overline{(X \wedge Y)^i} = \overline{X^i} \vee \overline{Y^i} \tag{5}$$

$$\overline{(X \vee Y)^i} = \overline{X^i} \wedge \overline{Y^i} \tag{6}$$

where i = 0, 1, or 2.

Law of involution:

$$\overline{\overline{X^1}} = X. \tag{7}$$

If i takes the value of 2, the ternary NAND (TNAND) function expressed by $\overline{(X \wedge Y)^i}$ in (5) will be called the positive TNAND, if i is 1 the simple TNAND, and the negative TNAND for i equals 0. Similar notations are assumed for the TNOR function stated in (6). The truth table of the TNAND and TNOR functions is illustrated by Table II. In this presentation, the discussion is restricted to the use of simple TNAND and simple TNOR gates called, for abbreviation reasons, TNAND and TNOR gates. Circuits of inverters, TNAND, and TNOR gates are presented in Figs. 1–3. These three circuits could be combined in one universal circuit [6]. Fig. 4 shows symbols used in the representation of the above ternary operators. It is clear that a TAND (TOR) gate can be easily realized by inserting an STI circuit at the output of a TNAND (TNOR) gate.

TERNARY STORAGE ELEMENTS

Ternary storage elements can be realized using the above ternary operators, designed with COS/MOS integrated circuits. In this section, several kinds of ternary flip-flops (tri-flops) are described in detail.

The PZN Tri-Flop

Fig. 5 shows the block diagram of the *PZN* tri-flop. It is constructed by cross-coupling two TNOR gates. The first TNOR has three inputs: P for setting the tri-flop to the high state "2" (positive), Z to set it to zero voltage (intermediate state "1"), and the third input coming from the output (Q) of the other TNOR to ensure the regenerative property. The second TNOR gate also has three inputs: N for setting the tri-flop to the low state "0" (negative), Z to set it to the intermediate one, and the third input coming from the output ($\overline{Q^1}$) of the first TNOR. The operation of the tri-flop will depend essentially on the law of involution (7).

To set the tri-flop to state "2," we put a low to high pulse on the P input. The output of the first TNOR will be at the low level; therefore, the three inputs of the second TNOR will be at the low level and its output will be in state "2," which will force the output of the first TNOR to stay in state "0" and so on, keeping the tri-flop in a stable state. Under these conditions, the tri-flop is said to be in the high state.

A low to intermediate pulse on the Z input will set the tri-flop to state "1." To set it to the state "0," a low to high pulse is needed on the N input.

TABLE I
Truth Table of the Three Ternary Inverters

x	$\overline{x^2}$	$\overline{x^1}$	$\overline{x^0}$
2	0	0	0
1	2	1	0
0	2	2	2

The Clocked PZN Tri-Flop

To use the same input signals as in the above tri-flop, the clocked *PZN* tri-flop is constructed by cross-coupling two TNAND gates and controlling the three inputs, P, Z, and N by means of three TNAND's as shown in Fig. 6. The tri-flop will change states as described above only when a low to high pulse is present at the input CL.

The D-Type Tri-Flop

The D-type tri-flop, similarly to the corresponding binary one, is constructed by cross-coupling two TNAND gates and inverting the input of one of them by an STI circuit. This is shown in Fig. 7(a). The output Q will follow the input D whatever it is. The D-type tri-flop can be controlled by a clock pulse by means of two TNAND gates inserted in the input of each TNAND gate forming the tri-flop [Fig. 7(b)]. This tri-flop will hold its state indefinitely if no clock pulses are applied.

A master–slave D-type tri-flop is shown in Fig. 7(c). The figure shows a master tri-flop formed of two inverters and two ternary switches that feeds a slave tri-flop which has a similar construction. This configuration is used in the construction of the master–slave D-type binary flip-flop designed with COS/MOS integrated circuits.

When the contfol signal V_c is at a high level 2, the TS_1 ternary switches are closed and the TS_2 switches open. This configuration allows the master tri-flop to sample incoming data, and the slave to hold the data from the previous input and feed them to the output.

When the control signal V_c is low, the TS_1 ternary switches open and the TS_2 switches closed, so that the master holds the data entered and feeds them to the slave.

The T-Type Tri-Flop

As the JK flip-flop is for the binary, the T-type tri-flop is the most important element in the design of ternary sequential circuits. It is constructed by inserting an inverse cycling gate (ICG) in a feedback path between the output Q and the input D of the master–slave D-type tri-flop of Fig. 7(c). The inverse cycling gate is defined here as

$$x'' = (x + 1) \bmod 3 \tag{8}$$

and the cycling gate by

$$x' = (x + 2) \bmod 3. \tag{9}$$

The plus sign here means an arithmetic addition. Block diagrams of a cycling gate and inverse cycling gate are shown in Fig. 8.

Fig. 9 presents a master–slave T-type tri-flop with P, Z, and N capabilities. Inverters are replaced by TNOR gates in order to have several inputs available. The input pulses to P, Z, and N inputs are the same as in the case of the *PZN* tri-flop described above. Its performance is illustrated by Table III. Next-states designated by F are forbidden states and present-states designated by D are DON'T CARE states. From the table it is also seen that when the P, Z, or N inputs are used, the T input can be 2 or 0. If the design is modified to permit utilization of P, Z, or N inputs when the T input is only 0, the master tri-flop could be formed by STI circuits instead of TNOR gates; and when the T input is only 2, the slave tri-flop could be formed by STI circuits instead of TNOR gates. It

TABLE II
Truth Table of TNOR and TNAND Functions

x	y	$x \vee y$	$\overline{(x \vee y)}^0$	$\overline{(x \vee y)}^1$	$\overline{(x \vee y)}^2$	$x \wedge y$	$\overline{(x \wedge y)}^0$	$\overline{(x \wedge y)}^1$	$\overline{(x \wedge y)}^2$
2	2	2	0	0	0	2	0	0	0
2	1	2	0	0	0	1	0	1	2
2	0	2	0	0	0	0	2	2	2
1	2	2	0	0	0	1	0	1	2
1	1	1	0	1	2	1	0	1	2
1	0	1	0	1	2	0	2	2	2
0	2	2	0	0	0	0	2	2	2
0	1	1	0	1	2	0	2	2	2
0	0	0	2	2	2	0	2	2	2

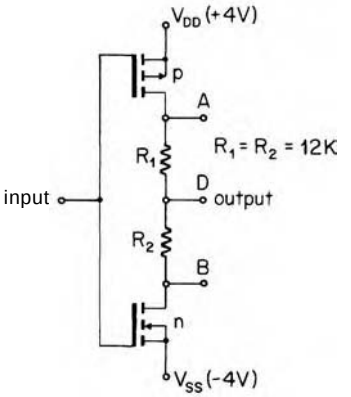


Fig. 1. COS/MOS ternary inverter. *A:* output of PTI. B: output of NTI. D: output of STI.

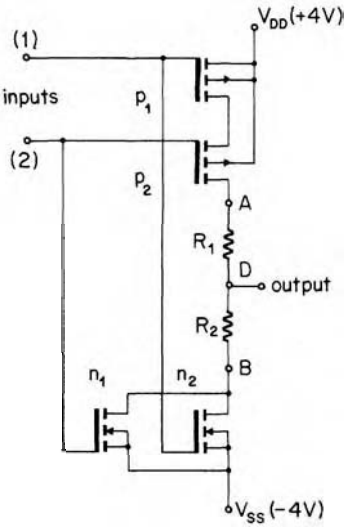


Fig. 2. Ternary NOR circuit. *A:* output of positive TNOR. *B*: output of negative TNOR. *D*: output of TNOR (or simple TNOR).

has to be mentioned that this tri-flop will change state on the leading edge of input pulses.

Ternary Shift Registers

The clocked PZN tri-flop and the D-type tri-flop described above can be considered as one stage of a static ternary shift register.

A dynamic ternary shift register may be constructed as shown in Fig. 10. Each stage consists of two STI circuits and two ternary switches. Each TS is controlled by two complemented clock signals as above, so that when alternate ternary switches are turned on, the others are turned off. The shift register utilizes the input capacitance of the STI for temporary storage. When the first TS in each stage is turned on, it couples the signal from the previous stage to the STI, and causes the signal to be stored on the input capacitance of the STI. When the TS is turned off on the next half cycle of the clock, the signal is stored on the input capacitance of this STI and remains at its output where it is available to the next TS, which is now turned on. Again, this signal is applied to the input of the next STI

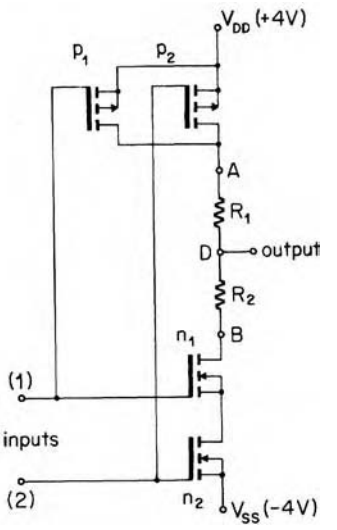


Fig. 3. Ternary NAND circuit. *A*: output of positive TNAND. B: output of negative TNAND. *D*: output of TNAND (or simple TNAND).

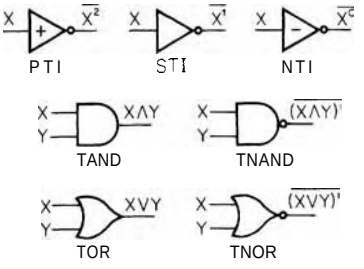


Fig. 4. Symbols representing ternary operators.

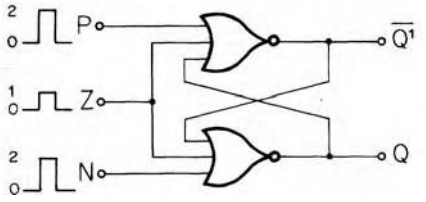


Fig. 5. The PZN tri-flop

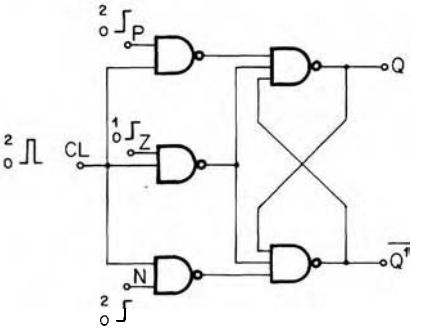


Fig. 6. The clocked *PZN* tri-flop.

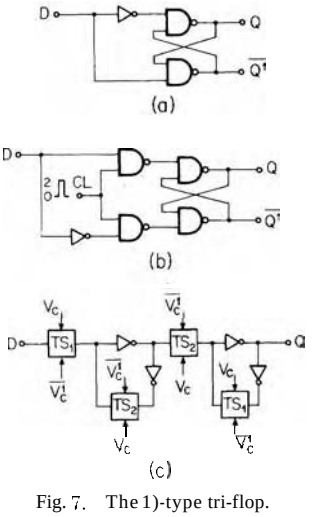


Fig. 7. The 1)-type tri-flop.

where it is stored on its input capacitance, making the signal available at the output of the stage.

This is true whenever the input signal is at the high, the intermediate, or the low level. Thus, a ternary signal progresses to the right by one stage per clock cycle.

It must be noted that there is a minimum frequency at which this dynamic shift register can operate because of its dependence of the stored charge which is subject to slow decay. Reliable operation can be obtained at frequencies as low as 0.1 kHz.

TERNARY COUNTERS

A ring ternary counter is constructed by cascading master–slave D-type tri-flops and feeding back the output Q_n of the last stage, where n is the number of stages, to the input D_1 of the first stage. At each clock pulse (low to high), information stored in each tri-flop, which can be preset via P, Z, or N of each one, will shift one position to the left and the left most significant trit will shift to the first tri-flop.

Fig. 11 shows a ternary counter designed to count from 0 to 27 or from −13 to +13. The counter is formed of three stages; each stage consists of a master–slave T-type tri-flop. In general, the ternary counter can be designed to count from 0 to 3^n with the normal ternary code or from $-(3^n - 1)/2$ to $+(3^n - 1)/2$ with the signed-ternary code, where n is the number of stages of the counter. In the signed-ternary code number, +1 is represented by the high level (positive), 0 the intermediate level (zero potential), and −1 by the low level (negative). Advantages of representation of ternary numbers using the signed-ternary code were mentioned by several authors [2], [9].

An NTI circuit connects the input of each tri-flop to the output of the preceding one. The function of this NTI circuit is to change the state of the tri-flop only when the state of the preceding tri-flop changes from high to low levels. This can be given by the following equation:

$$T_n = \overline{Q_{n-1}^{(1)}} \tag{10}$$

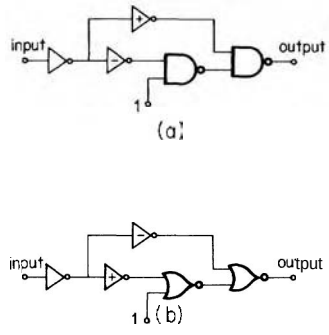


Fig. 8. Cycling and inverse cycling gates.

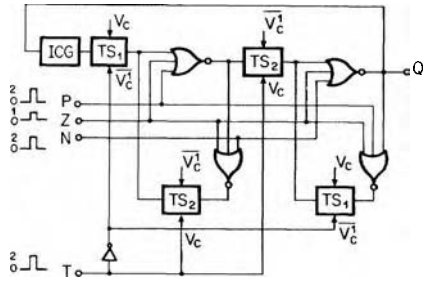


Fig. 9. Master–slave T-type tri-flop with P, Z, and N capabilities.

TABLE III
Next-State Tables for the T-Type Tri-Flop

T	P	Z	N	Q(t)	Q(t+1)
2	0	0	0	0	1
2	0	0	0	1	2
2	0	0	0	2	0
2/0	2	0	0	D	2
2/0	0	1	0	D	1
2/0	0	0	2	D	0
0	0	0	0	D	Q(t)
other combinations				D	F

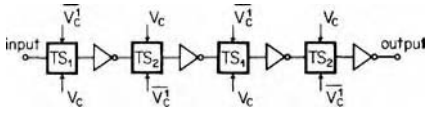


Fig. 10. Two-stage COS/MOS dynamic ternary shift register.

where T_n represents the input of the nth stage and Q_{n-1} the output of the n − 1th stage of the counter. To reset the counter to its initial state, a low to high pulse is sent through the N input if the normal ternary code was used; therefore, all tri-flops will be in the low level to represent the zero decimal number. If the signed-tefnary code was used, a low to intermediate pulse is sent through the Z input; therefore, all tri-flops will be in the intermediate level to represent the zero decimal number. The counter may also be preset to any initial number via the P, Z, or N input of each tri-flop. For every low to high pulse sent through the input CL, the counter will operate and tri-flops will change states.

It has to be mentioned here that all tri-flops of the counter, except the first one, may be reset when their inputs are in the low or high levels. Therefore, referring to the previous section, the master (or the slave) tri-flop of each stage may not be constructed with STI circuits, but with TNOR gates to have excess inputs for the resetting or presetting processes.

It has to be noted also that a ternary down counter which can count from 3" to 0 or from $+(3^n - 1)/2$ to $-(3^n - 1)/2$ can be constructed easily by replacing the ICG circuit in the tri-flop of each stage by a cycling gate

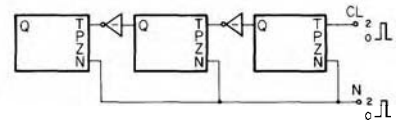


Fig. 11. COS/MOS ternary counter.

and the NTI circuit between stages by a PTI followed by an STI circuit. The function of the PTI and STI circuits is to change the state of the following tri-flop only when the state of the preceding one changes from low to high levels, i.e.,

$$T' = \overline{\overline{Q^{21}_{n-1}}} \tag{11}$$

where T_n is the input of the nth stage and Q_{n-1} is the output of the $n-$ 1th stage. In this case, the counter will be reset to its initial state when a low to high pulse is sent to the P input of each stage.

The divide-by-M ternary counter, which is analogous to the divide-by-N binary one, is given in Fig. 12. It is composed of the ternary counter described above and a feedback circuit formed by a TAND gate with a number of inputs equal to the number of stages of the counter. Each input of the TAND gate comes from a J_K arithmetic circuit mounted on the output of each tri-flop. Then the output of the TAND gate is used to reset the counter to its initial state after a number of pulses depended upon the output (J_0,J_1, or J_2) taken from each of the J_K arithmetic circuits [5]. The J_K arithmetic circuit is defined by

$$J_K(x) = \begin{cases} 2 & \text{if } K = x \\ 0 & \text{if } K \neq x \end{cases} \tag{12}$$

where K = 0, 1, or 2.

As an example of the design of a decade ternary counter, the outputs J_0, J_0, and J_1 will be taken from the J_K arithmetic circuits mounted on the outputs Q_1, Q_2, and Q_3, respectively. The divide-by-M ternary counter can be programmed to count to any number M from 0 to 3^n, where n is the number of stages of the counter. Fig. 13(a)–(d) shows the output waveforms of each stage for two stage up, down, and divide-by-7 ternary counters.

A latch circuit is added in the feedback path, as shown in the figure, to assure reliable clearing of the counter at the Mth pulse, which may not be reliable if the propagation delay from the N input to the tri-flop output varies from stage to stage. This variation may occur if the counter outputs are unevenly loaded. A good operation may be obtained when the width of the clock pulse is in the range of 70–370 μs.

Ternary Random-Access Memory

The basic ternary storage element proposed here consists of two STI circuits cross-coupled to form a tri-flop. A ternary random-access-memory cell may be constructed using the circuit with a ternary switch (TS). The ternary switch can be used as a simple and efficient means of performing the sensing and storage functions associated with storage-cell selection.

Fig. 14 shows a ternary word-organized memory cell which is composed of two STI circuits, one TS, a word line W, and a data line D. Addressing is accomplished by energizing a word line which turns on the T S of the selected tri-flop. The number of transistors composing this ternary memory cell is the same as the corresponding binary one. The main difference between them is the addition of resistances in the ternary memory cell. At the same time, since the information stored in a ternary memory element is higher than the corresponding binary one, the above ternary memory cell may provide greater memory density.

A trit-organized memory cell employing Z-Y selection can be obtained by adding another T S to the circuit of Fig. 14 to form, with the original TS, the X-Y addressing wires. This circuit can be used as a basic element of a large ternary memory array.

Ternary Read-Only Memory

The ternary read-only memory (TROM) consists of ternary decoder followed by a ternary encoder which is the memory matrix. Fig. 15 shows a ternary decoder of N inputs and 3^N outputs. This decoder is composed of NJ_K arithmetic circuits [5], 3^N TNAND gates, and 3^N STI circuits. All the word-lines will be at the low level except the one which was designated at the inputs. They will take the value defined by

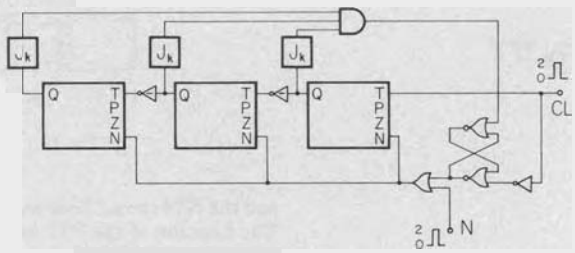


Fig. 12. The divide-by-M ternary counter.

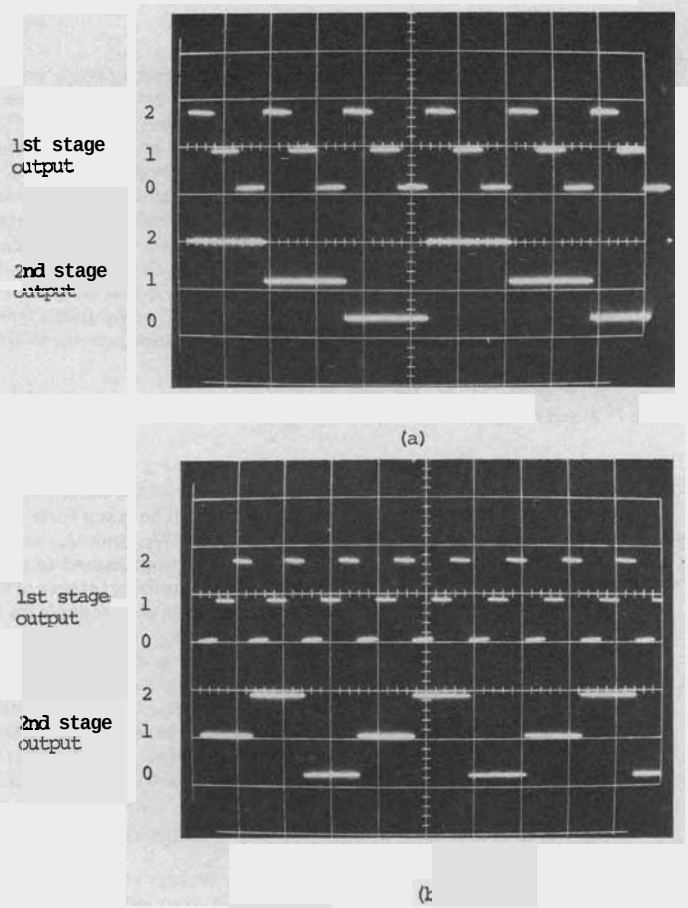


Fig. 13. Output waveforms of each stage for two stages of *A*: up ternary counter, *B*: down ternary counter, C: divide-by-7 ternary counter, *D*: divide-by-7 ternary counter (signed-ternary code).

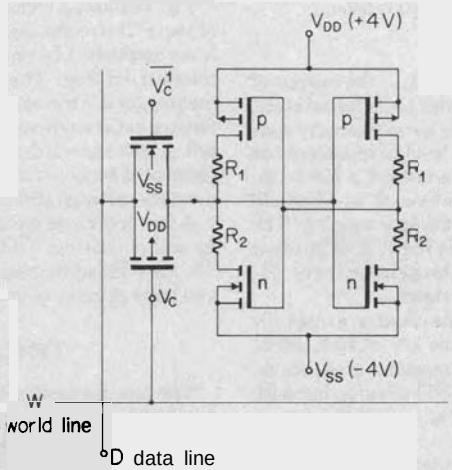


Fig. 14. A word-organized ternary memory cell

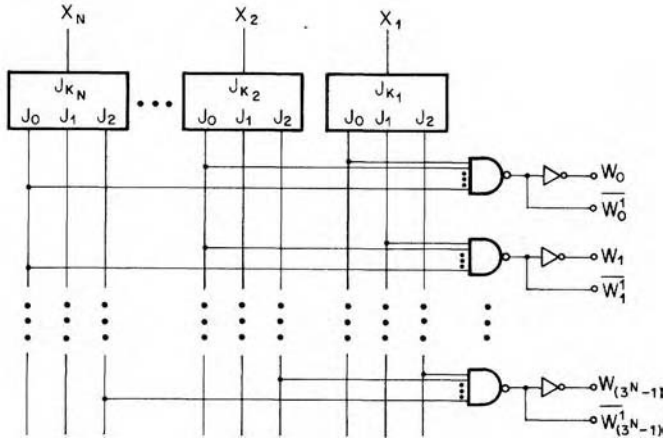


Fig. 15. The ternary decoder

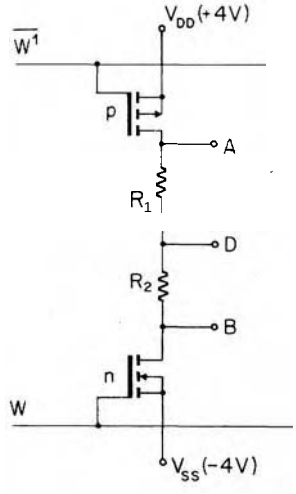


Fig. 16. The modified ternary inverter.

$$W_i = J_{K_1}(X_1) \wedge J_{K_2}(X_2) \wedge \cdots \wedge J_{K_N}(X_N), \quad i = K_1 3^0 + K_2 3^1 + \cdots + K_N 3^{N-1} \tag{13}$$

where i will take the value of $0,1,2,\cdots$, or (3^N-1) when $K_1,K_2,\cdots,K_N$ take the values of 0, 1, or 2, W_i is the word line, and $X_1,X_2,\cdots,X_N$ are the N inputs of the NJ_K arithmetic circuits.

The basic element of the ternary memory matrix is the modified ternary inverter (MTI) shown in Fig. 16. The input of the ternary inverter is divided here into two complemented inputs: one for the p-channel transistor and the other for the n-channel.

If input W, which is an input of an n-channel transistor, is high, the n-channel will be on and the p-channel will be also on because $\bar{W}^1$ will be at the low level. Therefore, the MTI will act exactly as the ternary inverter when its input is at zero potential (state "1"). The output from point A will be in state "2," from B in the low state "0," and from D in the intermediate state "1." But if the input W is low, each of p- and n-channels will be off and the MTI will act as an open circuit.

A TROM encoder of 3^N input and M outputs is shown in Fig. 17. It is composed of $M \times 3^N$ numbers of MTI memory cells. Since only one word line W will be high and all the others low, therefore in each column there will be only one MTI cell on and all the others will be off, i.e., open circuited. Therefore, we can make a wire OR of all the MTI cells of each column without any problem. Each trit line will be 2, 1, or 0, depending upon how the output of the only one energized MTI cell in each column is taken from A, B, or D, respectively. The outputs on the trit lines will have the values of

$$Y_i = \tilde{W}_0 \vee \tilde{W}_1 \vee \cdots \vee \tilde{W}_{(3^N-1)} \tag{14}$$

where i = 1,2,···,M, Y_i are the trit lines, and $\tilde{W}_0,\tilde{W}_1,\cdots,\tilde{W}_{(3^N-1)}$ are the outputs of the MTI cells energized by the word lines $W_0,W_1,\cdots,W_{(3^N-1)}$, respectively.

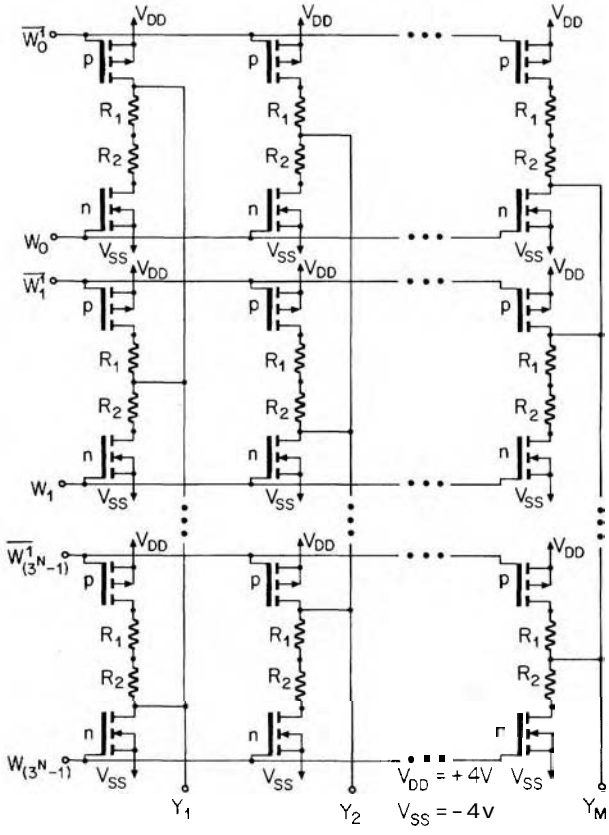


Fig. 17. COS/MOS ternary ROM encoder.

The output Y_i could be unstable if the input W, were at zero potential (state 1), but the input W_i is a word line output of the ternary decoder coming from $J_K(x)$ arithmetic circuits and the outputs of these circuits are never in state 1.

CONCLUSION

Ternary memory circuits can be easily designed using COS/MOS integrated circuits. This permits the realization of ternary sequential machines. Ternary RAM and ROM may provide greater memory density than the corresponding binary units. The power dissipation in most presented circuits is relatively high due to the existence of the intermediate stable state in which the two channel transistors are on.

The application of COS/MOS integrated circuits in the construction of ternary memory and sequential circuits may incite designers to wider implementation of three-valued logic in the design of digital machines.

REFERENCES

[1] M. Yoeli and G. Kosenfeld, "Logical design of ternary switching circuits," *IEEE Trans. Electron. Cornput.*, vol. EC-14, pp. 19–29, Feb. 1965.
[2] I. Halpern and M. Yoeli, "Ternary arithmetic unit," *Proc. IEE*, vol. 115, pp. 1385–1388, Oct. 1968.
[3] R. P. Hallworth and F. G. Heath, "Semiconductor circuits for ternary logic," *Proc. IEE*, vol. 109C, pp. 219–225, 1962.
[4] T. Hasegawa, Y. Nagaoka, Y. Tezuka, and Y. Kasahara, "Tunnel-diode circuits for ternary logic," *Elec. Comrnun. Japan*, pp. 88–95, Oct. 1964.
[5] H. T. Mouftah and I. B. Jordan, "Integrated circuits for ternary logic," in *Proc. 1974 Int. Symp, on Multiple-Valued Logic*, May 29–31, 1974, pp. 285–302.
[6] ——, "Implementation of three-valued logic with COS/MOS integrated circuits," *Electron. Lett.*, vol. 10, pp. 441–442, Oct. 1974.
[7] ——, "A design technique for an integrable ternary arithmetic unit," in *Proc. 1975 Int. Symp. Multiple-Valued Logic*, May 1975, pp. 359–372.
[8] H. T. Mouftah. "Three-valued logic and its implementation with COS/MOS integrated circuits," D.Sc. dissertation, Dep. Elec. Eng., Laval University, Quebec City, P. Q., Canada, 1975.
[9] D. I. Porat, "Three-valued digital systems," *Proc. IEE*, vol. 116, pp. 947–954, June 1969.
[10] S. Y. Su and A. A. Sarris, "The relationship between multivalued switching algebra and Boolean algebra under different definitions of complement," *IEEE Trans. Cornput.*, vol. C-21, pp. 479–485, May 1972.
[11] T. A. Irving and H. T. Nagle, "An approach to multivalued sequential logic," in *Conf. Rec., 1973 Int. Symp, on Multiple-Valued Logic*, May 1973, pp. 89–105.
[12] S. G. Shiva and H. T. Nagle, Jr., "On multiple-valued memory elements," in *Proc. 1974 Int. Symp. on Multiple-Valued Logic*, May 29–31, 1974, pp. 209–224.
[13] J. R. Burns, "Switching response of complementary symmetry MOS transistor logic circuits," *RCA Rev.*, vol. XXV, pp. 627–661, Dec. 1964.
[14] L. M. Terman, "MOSFET memory circuits," *Proc. IEEE*, Vol. 59. pp. 1044–1058, July 1971.
[15] Y. Suzuki, K. Odagaaa, and T. Abe, "Clocked CMOS calculator circuitry," *IEEE J. Solid-State Circuits*, vol. SC-8, pp. 462–469, Dec. 1973.
[16] Z. G. Vranesic, K. C. Smith, and A. Druzeta, "Electronic implementation of multi-valued logic networks," in *Proc. 1974 Int. Symp, on Multiple-Valued Logic*, May 1974, pp. 59–77.
[17] A. Kaniel, "Trilogic, A three-level logic system provides greater memory density," *EDN*, pp. 80–83, Apr. 1973.
[18] COS/MOS Integrated Circuits Manual, RCA Tech. Series CMS-271.
[19] COS/MOS Digital Integrated Circuits, Selection Guide Data Application Notes, RCA Solid State Databook Series, SSD-203B, 1974.
[20] G. A. Maley and J. L. Walsh, "Ternary read-only-memory," *IBM Tech. Disclosure Bull.*, vol. 14, pp. 1337–1338, Sept. 1971.
[21] ——, "Ternary latches," *IRM Tech. Disclosure Bull.*, vol. 14, p. 2384, Jan. 1972.

A Triangulation Method for the Sequential Mapping of Points from N-Space to Two-Space

R. C. T. LEE, J. R. SLAGLE, AND H. BLUM

Abstract—A method for the sequential mapping of points in a high-dimensional space onto a plane is presented. Whenever a new point is mapped, its distances to two points previously mapped are exactly preserved. On the resulting map, 2M − 3 of the original distances can be exactly preserved. The mapping is based on the distances of a minimal spanning tree constructed from the points. All of the distances on the minimal spanning tree are exactly preserved.

This method allows the user to select different reference points. This situation is similar to using a microscope. If the user is interested in a particular spot and wants to blow up that part of the picture, he can do it.

Index Terms—Clustering analysis, distance preserving, minimal spanning tree, nonlinear mapping, principal component analysis, reference point approach, second nearest neighbor approach, tree searching.

Manuscript received August 4, 1975; revised April 12. 1976.

R. C. T. Lee is with Institute of Applied Mathematics, National Tsing Hua University, Hsinchu, Taiwan, Republic of China.

J. R. Slagle is with the Computer Science Laboratory, Communications Division, Naval Research Laboratory, Washington, DC 20375.

H. Blum is with the Division of Computer Research and Technology, National Institutes of Health, Bethesda, MD 20014.

I. INTRODUCTION

Consider the following problem. We are given a set of data and we want to know the geometrical relationships among the points in the data. If the data are two-dimensional and numerical, this can be done by displaying them. Otherwise, some mapping which preserves the relationships among data points is desirable. In this correspondence, we shall discuss such a mapping method.

A very famous data displaying technique is the principal component analysis technique [3], [8]. While the principal component analysis technique is excellent in many respects, it has the following drawbacks.

1) It involves the finding of eigenvectors of the covariance matrix of the data. If the number of variables is large, it will be very difficult to find these eigenvectors.

2) It cannot handle nonnumerical data.

Recently, a technique called nonlinear mapping [1], [11], has been used to display data. This method is based upon the multidimensional scaling [2], [12], and the distance matrix is the only information necessary for this technique.

The nonlinear mapping technique tries to preserve all of the distances. This might be too ambitious because it usually ends in the case where no distances are exactly preserved. If the purpose of the data analysis is to detect the existence or nonexistence of clusters, this nonlinear mapping is adequate. On the other hand, if we are interested in knowing the detailed structure of the data, it is usually inadequate.

In this correspondence, we shall introduce a data mapping technique which is also based on the distance concept. However, this technique is different from the nonlinear mapping in the following sense.

1) This technique makes no attempt to preserve all of the distances. Instead, a subset of all of the distances are exactly preserved.

2) This technique allows the user to make several different maps of the same data; each map emphasizes a particular viewpoint.

All of the above discussions will become clear later.

Before discussing our method, let us consider the data in Table I. Our data are concerned with amino acids in a protein molecule, cytochrome c, found in the mitochondria of animals and higher plants [4]. Only the positions which vary are recorded in Table I.

Since the data are not numerical data, we cannot use the Euclidean distances. However, we can use Hamming distances. The Hamming distance between two points $P_i = (P_{i1}, \ldots, P_{iN})$ and $P_j = (P_{j1}, \ldots, P_{jN})$ is defined as follows:

$$d(P_i,P_j) = \sum_{k=1}^{N} \delta(P_{ik},P_{jk}),$$

where

$$\delta(P_{ik},P_{jk}) = 1 \quad \text{if } P_{ik} \neq P_{jk}, \text{ and}$$
$$= 0, \quad \text{if otherwise.}$$

II. THE TRIANGULATION METHOD

Consider three high-dimensional points P_i^*, P_j^*, and P_l^*. Suppose on the two-space, P_i and P_j exactly preserve the distance between P_i^* and P_j^*. That is, $d_{ij} = d_{ij}^*$. Then the third point P_k^* can be mapped to a point P_k in the two-space such that the distances among P_i^*, P_j^*, and P_k^* are all exactly preserved. This can be done by drawing two circles with P_i and P_j as centers and d_{ik}^* and d_{jk}^* as radii as shown in Fig. 1. Note that because of the triangular inequality, the circles either intersect in 2 points, or are tangent. Through this basic principle, we can gradually map a set of M high-dimensional points onto a two-dimensional plane. On the resulting map, for every point P_k, there exist two points P_i and P_j such that the distances among P_i, P_j, and P_k are all exactly preserved.

Correction to "Design of Ternary COS/MOS Memory and Sequential Circuits"

H. T. MOUFTAH AND I. B. JORDAN

In the above paper,[1] Fig. 13(c) and (d) were inadvertently omitted. They appear correctly below.

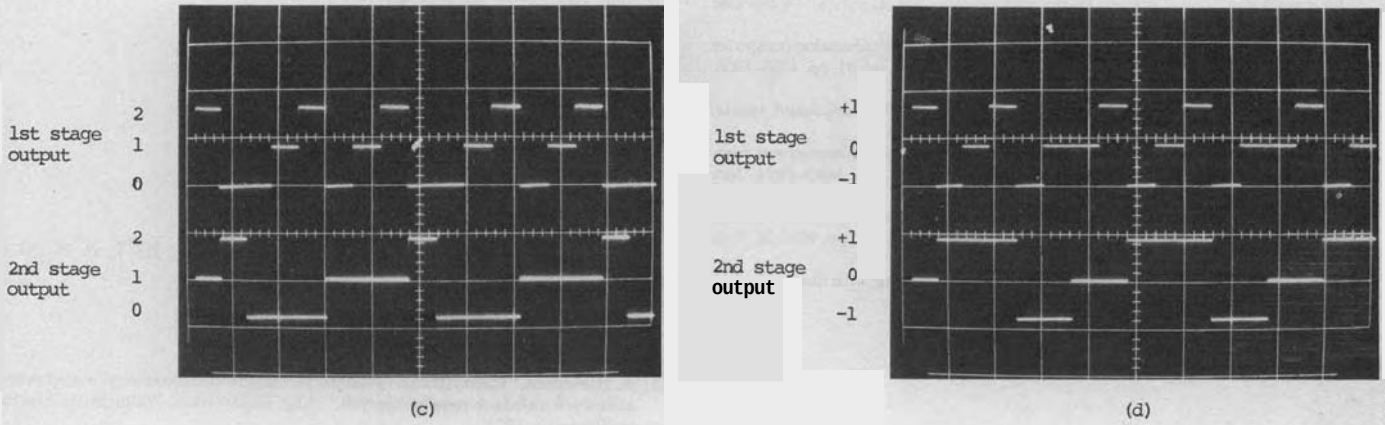


Fig. 13. Output waveforms of each stage for two stages of (c) divide-by-7 ternary counter, (d) divide-by-7 ternary counter (signed-ternary code).

Manuscript received July 19, 1977.

H. T. Mouftah is with Bell-Northern Research, Ottawa, Ont., Canada.

I. B. Jordan is with the Department of Electrical Engineering, Laval University, Quebec City, P.Q., Canada.

[1] H. T. Mouftah and I. B. Jordan, IEEE *Trans. Comput.* (Corresp.), vol. C-26, pp. 281–288, Mar. 1977.

0018-9340/78/0200-0192$00.75 © 1978 IEEE