Threshold Logic Elements Used as a Probability Transformer

C. L. SHENG
University of Ottawa, Ottawa, Canada

Abstract. Threshold logic elements, although essentially used as gates in deterministic combinatorial circuits, can be applied to switching circuits with discrete random inputs to generate discrete random variables of any probability distribution, i.e. to serve as a discrete probability transformer. In this paper some properties of threshold functions related to the realization of a probability transformer are studied. The principle of using threshold logic elements for probability transformation is presented and the techniques for realization are developed. Examples are given for illustration.

1. *Introduction*

A discrete probability transformer, as described by Gill [1, 2], is a system which converts a discrete stochastic source (either natural or artificial) generating the statistically independent symbols $\alpha_1, \alpha_2, \cdots, \alpha_r$ with the probabilities $p_1, p_2, \cdots, p_r$, respectively, into another discrete stochastic source generating the statistically independent symbols $\beta_1, \beta_2, \cdots, \beta_s$ with the probabilities $q_1, q_2, \cdots, q_s$, respectively.

Methods and techniques for generating artificial stochastic, or pseudorandom symbols are well developed and can be found in the literature. They are not the concern of this paper and are not discussed here. We are only concerned with the transformation of probability distribution.

Probability transformation can be realized with a sequential circuit, as was developed by Gill [1]. The idea is to design a sequential machine with n states such that with the given symbols $\alpha_1, \alpha_2, \cdots, \alpha_r$ as the random inputs, eventually the machine will have a probability of q_1/n to be in a set of q_1 states, a probability of q_2/n to be in a set of q_2 states, $\cdots$, and a probability of q_s/n to be in a set of q_s states, where $\sum_{i=1}^{s} q_i = n$. Considering the machine as a Moore model [3], we can associate the s random outputs with the s sets of states, which have the desired probability distribution.

Instead of a general sequential machine with the feedback loop and with n delay units, a definite-event machine with much fewer delay units can serve this purpose of probability transformation as well. In this case the grouping of states can be very simply and conveniently done by using threshold logic elements. Our scheme for the probability transformer is similar to that of Gill. In fact we are studying the design of the "sequence identifier" with threshold logic elements.

First, the input alphabet $\{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ is converted into a binary alphabet through a symbol converter. As pointed out by Gill, this conversion permits the usage of binary elements for the transformation logic. Let this new binary random input variable be x_0, which can be either "1" or "0", with a probability of p and $1 - p$, respectively. It is preferable to split the input alphabet in such a way as to make p and $1 - p$ as close to 0.5 as possible.

This research was supported in part by National Research Council of Canada under Grant No. A-1690.

Journal of the Association for Computing Machinery, Vol. 12, No. 2 (April, 1965), pp. 262–276

x_0 is fed to m delay units in cascade. Let the outputs of the delay units be $x_1, x_2, \cdots, x_m$, respectively. Then we have

$$\begin{aligned} x_1(t) &= x_0(t-1) \\ x_2(t) &= x_1(t-1) = x_0(t-2) \\ &\vdots \\ x_m(t) &= x_0(t-m) \end{aligned}$$

Now the m variables, $x_1, x_2, \cdots, x_m$, each having a probability of p for being "1" and a probability of $1 - p$ for being "0", form an m-tuple. There are 2^m m-tuples altogether, each m-tuple having a probability of

$$p' = p^i(1-p)^{m-i}$$

where i is the number of 1's in the m-tuple and $m-i$ is the number of 0's in the m-tuple.

For the special case of $p = 1 - p = \frac{1}{2}$, the probabilities of the 2^m m-tuples are all equal and are

$$p' = (\tfrac{1}{2})^m = 2^{-m}.$$

Now if we divide the set of 2^m m-tuples into s disjoint subsets with $N_1, N_2, \cdots, N_s$ m-tuples in each subset such that

$$\sum_{i=1}^{s} N_i = 2^m$$

and such that

$$\sum_{j=1}^{N_i} p_{ij} = q_i \tag{1}$$

where p_{ij} is the probability of the jth m-tuple in the ith subset, then the ith subset can represent the ith output with the desired probability q_i.

This is obviously possible, because any subset of m-tuples corresponds to a Boolean function, and any Boolean function is realizable with a number of AND and OR gates.

Equation (1) can be approximated to any desired degree of accuracy by increasing m. The determination of error bounds was also developed by Gill [2], and will not be studied in detail in this paper.

Figure 1 shows a schematic diagram of the probability transformer.

To ensure independence of output symbols, it is necessary to use a sampler or a pulse clock with a certain period, as discussed in Section 8.

2. *Properties of Threshold Functions*

Although it is possible to realize any Boolean function with AND and OR gates, such a scheme has the disadvantage of using a large number of gates. Threshold logic elements are found to be particularly convenient and suitable for this purpose of probability transformation, and it is shown later that if threshold logic elements are used the minimum number of elements required is equal to $s - 1$, or one less than the number of output symbols.

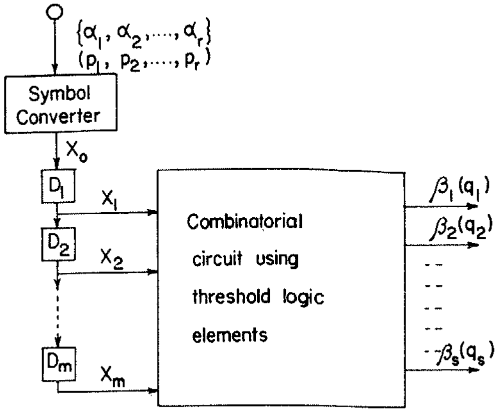

FIG. 1. Schematic diagram of a probability transformer using threshold logic elements

In order to apply threshold logic elements to probability transformation, let us study some of the properties of threshold functions, which will be found useful later.

Definition. Two threshold functions are *isobaric* [4] to each other iff they can be realized with the same set of m input variables of the same weight vector $(a_1, a_2, \cdots, a_m)$ but with two different threshold values.

Definition. n threshold functions are isobaric iff they can be realized with the same set of m input variables of the same weight vector $(a_1, a_2, \cdots, a_m)$ but with n different threshold values.

For example, consider some threshold functions of three variables x_1, x_2 and x_3 with weights $a_1 = 3$, $a_2 = 2$ and $a_3 = 1$, respectively.

$$\text{For} \quad T = 2, \qquad F_1 = x_1 + x_2,$$

$$T = 3, \qquad F_2 = x_1 + x_2x_3,$$

$$T = 4, \qquad F_3 = x_1(x_2 + x_3).$$

Thus, F_1, F_2 and F_3 are isobaric.

THEOREM 1. *n threshold functions are isobaric iff any two of the functions are isobaric to each other.*

PROOF. (1) First we prove the necessity. If two of the n functions, F_1 and F_2, are not isobaric to each other, then any assignment of weights of the variables to realize the n functions that can realize F_1 with a certain threshold value T_1, can never realize F_2; and vice versa. Thus no assignment can realize all the n functions, and the necessity is proved.

(2) Next we prove the sufficiency. Each function has a requirement on the primary ordering of weights. This requirement may be strict for the ordering between certain weights, and may be not strict for the ordering between others. For instance, $F = x_1 + x_2x_3$ requires that $a_1 > a_2$ and $a_1 > a_3$, but a_2 can be greater than, equal to, or less than a_3. Thus any assignment satisfying the conditions that $a_1 > a_2$ and $a_1 > a_3$ can realize the function F. Now suppose that out of the n functions, F_1 requires that $a_1 > a_2$. Then no other function can have a requirement that $a_1 \leqq a_2$, because if a function F_2 requires that $a_1 \leqq a_2$, then F_1 and F_2 cannot be realized with variables of the same weight vector, and therefore are not

isobaric to each other. This contradicts the assumption. Thus if the functions are pairwise isobaric, the relation $a_1 > a_2$ must exist for all the n functions.

Similarly there exists no contradiction in the relations among all the weights. Thus there exists a certain primary ordering which satisfies all the functions.

If there exists a secondary ordering among the incremental weights [5], the same reasoning for primary ordering of weights can be extended to secondary ordering. For if a function requires a certain relation between any two incremental weights, say $\Delta a_1 > \Delta a_3$, no other function can have a contradictory requirement. Thus there must exist a certain secondary ordering which satisfies all the functions.

The same reasoning can be extended to higher orderings of weights, if they ever exist. Since the assignment of weights is completely determined by the orderings of weights, there must exist an assignment which satisfies all the requirements of all the n functions. This completes the proof of Theorem 1.

THEOREM 2. *Two threshold functions isobaric to each other are comparable, and the function having the lower threshold value contains the function having the higher threshold value.*

PROOF. Suppose F_1 and F_2 are two threshold functions of m variables and are isobaric to each other. Let

$$T_1 > T_2 .$$

Now

$$F_1 = 1 \quad \text{iff} \quad \sum_{i=1}^{m} a_i x_i \geqq T_1 .$$

But

$$\sum_{i=1}^{m} a_i x_i > T_2 \quad \text{if} \quad \sum_{i=1}^{m} a_i x_i \geqq T_1 .$$

Therefore

$$F_2 = 1, \quad \text{if} \quad F_1 = 1.$$

This means that $F_1 \rightarrow F_2$, or $F_2 \supset F_1$. This completes the proof.

COROLLARY 2-1. *If $F_1, F_2, \cdots, F_n$ are n threshold functions which are isobaric and if*

$$T_1 > T_2 > \cdots > T_n ,$$

then

$$F_n \supset F_{n-1} \supset \cdots \supset F_1 .$$

PROOF. The proof is obvious from Theorem 2 and is omitted.

THEOREM 3. *The maximum number of isobaric threshold functions that can be formed from m variables for a fixed assignment of weights is equal to $2^m + 1$, including the trivial functions* 1 *and* 0.

PROOF. Let F_i and F_{i+1} be two functions isobaric to each other and with $T_i > T_{i+1}$. Then from Theorem 2, $F_{i+1} \supset F_i$.

F_{i+1} must contain at least one more m-tuple than F_i, for otherwise either F_{i+1} and F_i would be identical or F_i would contain F_{i+1}. Now for m variables, there are 2^m m-tuples. Let F_1 contain 0 m-tuple, which is the trivial function 0, let F_2 contain one m-tuple, let F_3 contain two m-tuples, $\cdots$, etc. Then F_{2^m+1} contains 2^m m-tuples and is the trivial function 1. Thus there are at most $2^m + 1$ functions. This completes the proof.

THEOREM 4. *The minimal assignment of integral weights to m variables to obtain $2^m + 1$ isobaric functions is* $1, 2, 4, \cdots, 2^{m-1}$.

PROOF. To obtain $2^m + 1$ functions from m variables, we must have $2^m + 1$ different threshold values, and in decreasing the threshold value from T_i to T_{i+1}, we must increase one and only one m-tuple to the function at a time. So every m-tuple must have a weight different from that of every other m-tuple, for if two m-tuples have the same weight they will be both contained or both not contained in the function. In other words, the function containing only one of them can not be obtained from this assignment of weights, and the total number of functions will be less than $2^m + 1$. Now each m-tuple has a weight equal to the sum of some i of the weights, where $i = 0, 1, \cdots, m$. The minimum integral weight is 1. Let $a_m = 1$. Then a_{m-1} can not be 1 too, for otherwise the weights of some m-tuples, for instance, $x_1x_2 \cdots \bar{x}_{m-1}x_m$ and $x_1x_2 \cdots x_{m-1}\bar{x}_m$, would be the same. So the minimum weight of a_{m-1} is 2. This extends to the higher weights, and the ith weight must be greater than the sum of all the previous $i - 1$ weights. Thus we have

$$\begin{aligned}
a_m &= 1 = 2^0 \\
a_{m-1} &= a_m + 1 = 2 = 2^1 \\
a_{m-2} &= a_m + a_{m-1} + 1 = 1 + 2 + 1 = 4 = 2^2 \\
\vdots \quad & \quad \vdots \\
a_1 &= 2^{m-1}
\end{aligned}$$

This completes the proof of Theorem 4.

COROLLARY 4-1. *For the particular assignment of weights of Theorem 4, the following relation holds:*

$$N + T = 2^m \quad \text{or} \quad T = 2^m - N$$

where N = number of m-tuples contained in the function, T = the threshold value.

PROOF. When $T = 0$, $F = 1$. In other words, the function contains all the 2^m m-tuples. Therefore Corollary 4-1 holds for $T = 0$. Now for this particular assignment of weights, the m-tuples have all the different integral weights ranging from 1 to $2^m - 1$, and to form all the $2^m + 1$ functions, T is increased by 1 at a time and therefore has all the $2^m + 1$ different values from 0 to 2^m. When T is increased by 1, N is decreased by 1. Therefore we have $T + N = 2^m$ for $N = 1, 2, \cdots, 2^m$. This completes the proof.

3. *Realization of Probability Transformer*

For the sake of simplicity of presentation, we first assume that for $x_1, x_2, \cdots, x_m$, $p = 1 - p = \frac{1}{2}$. Then each m-tuple has a probability of $\frac{1}{2}^m$. It is required to obtain an output which is 1 with a probability of q_1. Let us express q_1 as a fraction $N_1/2^m$, or let $N_1 = [2^m q_1]$ where $[2^m q_1]$ = the integer closest to $2^m q_1$.

Then N_1 represents the number of m-tuples contained in the function which has a probability equal or closest to q_1. Since each m-tuple is a random variable, we can choose any N_1 m-tuples to form the function. Here we purposely choose those N_1 m-tuples such that they form a threshold function.

Let the assignment of weights be

$$a_1 = 2^{m-1}, \quad a_2 = 2^{m-2}, \cdots, a_{m-1} = 2, \quad a_m = 1$$

By Corollary 4-1,

$$T_1 = 2^m - N_1 .$$

T_1 can be expressed as the sum of powers of 2 as

$$T_1 = c_1 2^{m-1} + \cdots + c_{m-1}2^1 + c_m 2^0$$

where $c_1, c_2, \cdots, c_m$ are the coefficients of the powers of 2 equal to 1 or 0.

Then the value of c_i to be 1 or 0 represents the presence or absence of x_i in the lowest vertex [6] (or the m-tuple with the minimum weight) in the function. For instance, in Example 1 given below, for F_1, the lowest vertex is $x_1x_3x_5x_6x_7$. According to the assignment of weights of Theorem 4, $a_i > \sum_{j=i+1}^{m} a_j$. So as $x_1x_3x_5x_6x_7$ is contained in the function, x_1x_2 and $x_1x_3x_4$ must also be contained. Thus the function F_1 is easily found.

Next we are to find the function F_2 which has an output with a probability of q_2. If we give the same assignment of weight of F_2 as to either F_1, then F_1 and F_2 are isobaric to each other, and therefore either $F_2 \supset F_1$ or $F_1 \supset F_2$.

Since q_1 and q_2 correspond to two mutually exclusive events, or two disjoint sets of m-tuples, it seems to be impossible to use the same assignment of weights. However, if we use a different assignment for F_2, it is still difficult to find a function completely disjoint from F_1 and yet containing the required number of m-tuples. Here we resort to the property of inhibition to threshold logic elements so that we may use the same assignment of weights. In other words we shall find a function F_2 isobaric to F_1 and having an output with a probability of $q_1 + q_2$, and then inhibit it with F_1 so that when F_1 has an output of "1", F_2 will have an output of "0". Then the output of F_2 will have a probability q_2 which represents a mutually exclusive event from that of F_1.

The weight of inhibition should be such that the output of F_2 is "0" whenever the output of F_1 is "1". Let a_{12} denote the weight of inhibition of F_1 to F_2. Then

$$a_{12} < T_2 - \sum_{i=1}^{m} a_i$$

where T_2 = the threshold value of threshold logic element realizing F_2, and
a_i = weight of input x_i to threshold logic element realizing F_2.

The same reasoning and techniques apply to $F_3, F_4, \cdots, F_m$. Thus F_3 is realized to have a probability of $q_1 + q_2 + q_3$. With inhibitions from F_1 and F_2, it has a probability of q_3. Similarly $F_4, \cdots, F_m$ can be realized.

Let us consider an example.

Example 1.

$$p = 1 - p = \tfrac{1}{2}$$

$$q_1 = 0.32$$

$$q_2 = 0.27$$
$$q_3 = 0.23$$
$$q_4 = 0.18$$
$$m = 7$$

(1) Determination of threshold function F_1 with a probability of $q_1 = 0.32$.

$$N_1 = [2^7 \times 0.32] = [128 \times 0.32] = [40.96]$$
$$= 41$$
$$T_1 = 128 - 41 = 87 = 64 + 16 + 4 + 2 + 1$$
$$= 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

The lowest vertex is $x_1x_3x_5x_6x_7$.

$$F_1 = x_1[x_2 + x_3(x_4 + x_5x_6x_7)].$$

(2) Determination of threshold function F_2 with a probability of $q_1 + q_2$.

$$q_1 + q_2 = 0.32 + 0.27 = 0.59$$
$$N_2 = [128 \times 0.59] = [75.52] = 76$$
$$T_2 = 128 - 76 = 52 = 32 + 16 + 4$$
$$= 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

The lowest vertex is $x_2x_3x_5$.

$$F_2 = x_1 + x_2x_3(x_4 + x_5).$$

The function with inhibition from F_1 is $F_2' = F_2\bar{F}_1$.

(3) Determination of threshold function F_3 with a probability of $q_1 + q_2 + q_3$.

$$q_1 + q_2 + q_3 = 0.32 + 0.27 + 0.23 = 0.82$$
$$N_3 = [128 \times 0.82] = [104.96] = 105$$
$$T_3 = 128 - 105 = 23 = 16 + 4 + 2 + 1$$
$$= 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

The lowest vertex is $x_3x_5x_6x_7$.

$$F_3 = x_1 + x_2 + x_3(x_4 + x_5x_6x_7).$$

The function with inhibition from F_1 and F_2' is $F_3' = F_3\bar{F}_1\bar{F}_2'$.

(4) Determination of threshold function F_4 with a probability of $q_1 + q_2 + q_3 + q_4$.

$$q_1 + q_2 + q_3 + q_4 = 0.32 + 0.27 + 0.23 + 0.18 = 1.00$$
$$F_4 = 1$$

The function with inhibitions from F_1, F_2' and F_3' is

$$F_4' = F_4\bar{F}_1\bar{F}_2'\bar{F}_3' = \bar{F}_1\bar{F}_2'\bar{F}_3' = \overline{(F_1 + F_2' + F_3')}$$

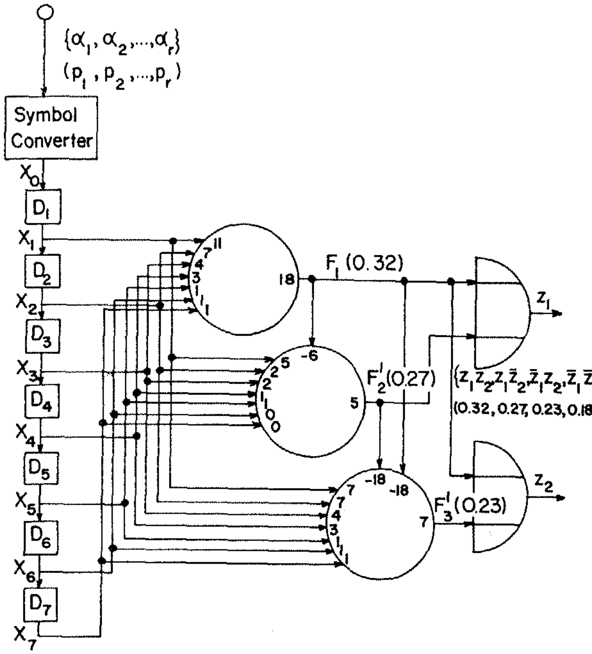

FIG. 2. Realization of Example 1

It is simply an AND gate with inputs which are the complements of F_1, F_2' and F_3', or is the complement of an OR gate with inputs F_1, F_2' and F_3'.

In fact this last AND gate is not necessary. Suppose there are s output symbols $\beta_1, \beta_2, \cdots, \beta_s$, with probabilities $q_1, q_2, \cdots, q_s$, respectively. Then we need $[\log_2 s]$ binary output variables to represent them, where $[\log_2 s]$ is the smallest integer greater than or equal to $\log_2 s$. Since the output symbols are all disjoint, we can use OR gates to obtain the output variables. For instance, in Example 1, there are four output symbols. So we need two OR gates to get two output variables x_1 and x_2. As shown in Figure 2, z_1z_2 (or 11) represents β_1 with a probability 0.32, $z_1\bar{z}_2$ (or 10) represents β_2 with a probability 0.27, $\bar{z}_1\bar{z}_2$ (or 01) represents β_3 with a probability 0.23, and $\bar{z}_1\bar{z}_2$ (or 00) represents β_4 with a probability 0.18. It is obvious that the AND gate for F_4 is not required.

4. *Readjustment of Weights*

One disadvantage of the particular assignment of weights used above is the large magnitudes of the weights. In the above example, m was taken to be 7, and the largest weight, a_1, is equal to $2^6 = 64$. For $m = 10$, a_1 will be $2^9 = 512$. Such a large weight seems to be formidable and is certainly impractical because of errors in the threshold logic elements.

However, this disadvantage is in fact not so serious as it first appears to be. As is well known, any threshold function can be realized with many different assignments of weights. The particular assignment of weights used is minimal for the generation of $2^m + 1$ isobaric functions, but is not minimal for any particular function. The purpose of using this assignment of weights in the beginning is to find out the threshold function having the required number of m-tuples and therefore the required probability. Once the function is determined, this particular assignment of weights is no longer required, and it is usually possible to readjust the weights to much smaller magnitudes. For instance, in Example 1, F_1 can be realized with the following minimal assignment of integral weights:

$$a_1 = 11, \quad a_2 = 7, \quad a_3 = 4, \quad a_4 = 3, \quad a_5 = a_6 = a_7 = 1, \ T = 18.$$

For F_2,

$$a_1 = 5, \quad a_2 = a_3 = 2, \quad a_4 = a_5 = 1, \quad a_6 = a_7 = 0, \ T = 5.$$

And for F_3,

$$a_1 = a_2 = 7, \quad a_3 = 4, \quad a_4 = 3, \quad a_5 = a_6 = a_7 = 1, \ T = 7.$$

The realization of Example 1 is shown in Figure 2.

5. *Decomposition of Threshold Logic Elements*

If after readjustment the weights are still considered to be too large for physical realization, each threshold logic element may be decomposed into two or more threshold logic elements with less input variables of smaller weights. The techniques of decomposition of threshold logic elements have been developed by Mattson [7] and Glinski and Yue [8], and are not discussed in this paper. The application of decomposition to our case is quite simple, and can be easily done without the use of any systematic method. Let us consider Example 1 again:

$$\begin{aligned} F_1 &= x_1[x_2 + x_3(x_4 + x_5x_6x_7)] \\ &= x_1(x_2 + x_3x_4) + x_1x_3x_5x_6x_7 \,. \end{aligned}$$

Thus F_1 can be realized with two threshold logic elements, as shown in Figure 3(a).

F_2 of Example 1 is fairly simple and does not need to be decomposed.

F_3 can be decomposed as

$$\begin{aligned} F_3 &= x_1 + x_2 + x_3(x_4 + x_5x_6x_7) \\ &= (x_1 + x_2 + x_3x_4) + x_3x_5x_6x_7 \,. \end{aligned}$$

The realization of F_3 with two threshold logic elements is shown in Figure 3(b).

6. *General Case*

So far we have considered only the special case for $p = 1 - p = \frac{1}{2}$. For the general case, although for m variables there are still 2^m m-tuples, the probability of each m-tuple is no longer the same, but is different depending upon how many 1's it has. So the problem is now to find a threshold function such that the probability of the output to be "1" is equal to a specified probability, say q_1.

Owing to the particular way of assigning the weights, the threshold functions generated are all of the same form. If suitable brackets and parentheses are used, each variable appears only once in the expression, and the variables appear in the order as the ascending order of the subscripts of the variables. For instance, F_1 in

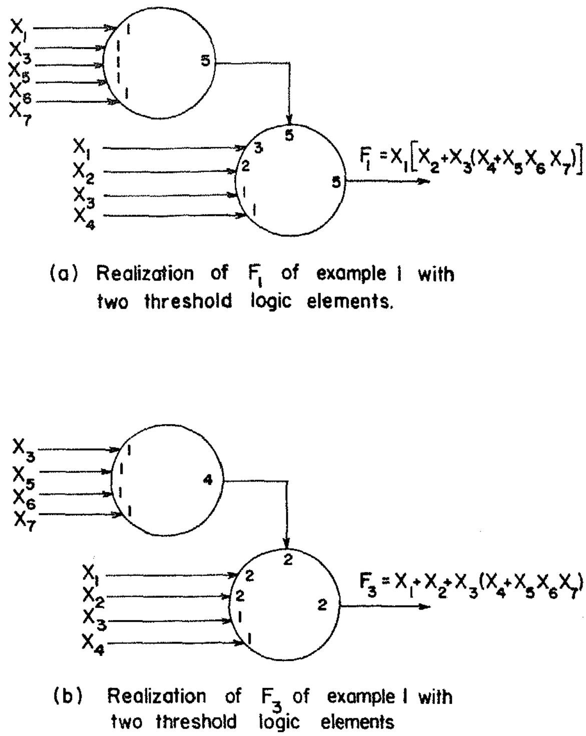

FIG. 3. Examples of decomposition of threshold logic elements

Example 1 is

$$F_1 = x_1[x_2 + x_3(x_4 + x_5x_6x_7)].$$

From this expression, the number of m-tuples contained in it can easily be found. This can be more clearly seen if F_1 is changed from the minimal sum-of-products form to the following form:

$$F_1 = x_1[x_2 + \bar{x}_2x_3(x_4 + \bar{x}_4x_5x_6x_7)]. \tag{2}$$

A term like x_1 contains half of the 2^m m-tuples, a term like x_1x_2 contains $\frac{1}{4}$ of the m-tuples, etc. With the complemented variables introduced, all the terms become disjoint, and the total number of m-tuples of the function is now equal to the sum of the m-tuples contained in each term. So the total number of m-tuples contained in F_1 is equal to 2^m multiplied by the above expression with each variable, complemented or uncomplemented, replaced by $\frac{1}{2}$. Thus

$$\begin{aligned} N_1 &= 128 \times \tfrac{1}{2}[\tfrac{1}{2} + \tfrac{1}{2} \times \tfrac{1}{2}(\tfrac{1}{2} + \tfrac{1}{2} \times \tfrac{1}{2} \times \tfrac{1}{2} \times \tfrac{1}{2})] \\ &= 128 \times \tfrac{41}{128} \\ &= 41 \end{aligned}$$

This checks with Example 1.

For the special case of $p = 1 - p = \frac{1}{2}$, since $q_1 = N_1/2^m$, the probability of F_1 is simply the expression of F_1 in the form of Equation (2) with each variable replaced by $\frac{1}{2}$. For the general case, however, this is not true, and the expression for probability should be slightly modified. Each uncomplemented variable should be replaced by p, and each complemented variable should be replaced by $(1 - p)$. Thus the probability of F_1 becomes

$$q_1 \cong p\{p + (1 - p)p[p + (1 - p)p^3]\}.$$

Now the problem reduces to: Given p, $(1 - p)$ and q_1 (m depends on error tolerance and may be assumed to be given too), to find the threshold function F_1.

The procedure of doing this is as follows:

Step (1). If $q_1 < p$, divide q_1 by p. If the quotient is still less than p, divide it by p again. Repeat this until the quotient is greater than or equal to p. Denote the number of times of division by d_1. Thus q_1 is already divided by p^{d_1} to become greater than or equal to p. Then subtract p from the last quotient, which is greater or equal to p, to obtain the difference D. Go to step (3).

Step (2). If $q_1 \geqq p$, subtract p from q_1 to obtain the difference D. Go to step (3).

Step (3). If the difference in Step (1) or (2) is zero, the procedure is already completed. If the difference D is greater than zero, divide D by $(1 - p)$. Get the quotient q_1'. Then treat q_1' as q_1 and repeat steps (1) or (2). In step (1) the new number of times of division is denoted by d_2.

Step (4). Repeat steps (1) through (3) until the mixed highest power of p and $(1 - p)$ (in the last term) is equal to m.

Step (5). Arrange the p's and $(1 - p)$'s in the following form, which is equal or closest to q_1 for the given m:

$$q_1 \cong p^{d_1}\{p + (1 - p)p^{d_2}[p + (1 - p)p^{d_3}(p + \cdots)]\}. \tag{3}$$

Step (6). Form F_1 according to Equation (3), with each p replaced by an uncomplemented variable and each $(1 - p)$ replaced by a complemented variable, starting from x_1 with each uncomplemented variable appearing once and only once in the ascending order of the subscript and with each complemented variable having the same subscript as the preceding uncomplemented variable.

Let us consider the above example again, but with different p and $(1 - p)$.

Example 2.

$$p = 0.7$$
$$1 - p = 0.3$$
$$q_1 = 0.32$$
$$q_2 = 0.27$$
$$q_3 = 0.23$$
$$q_4 = 0.18$$
$$m = 7$$

(1) Determination of F_1.

$$q_1 = 0.32 < 0.7$$
$$\frac{0.32}{0.7} = 0.457 < 0.7$$

$$\frac{0.457}{0.7} = 0.653 < 0.7$$

$$\frac{0.653}{0.7} = 0.947 > 0.7$$

$$d_1 = 3$$

$$0.947 - 0.7 = 0.247$$

$$\frac{0.247}{0.3} = 0.823 > 0.7$$

$$d_2 = 0$$

$$0.823 - 0.7 = 0.125$$

$$\frac{0.123}{0.3} = 0.41 < 0.7$$

$$\frac{0.41}{0.7} = 0.586 < 0.7$$

$$\frac{0.586}{0.7} = 0.837 > 0.7$$

$$d_3 = 2$$

The mixed highest power is already 7. So we stop here.

$$\begin{aligned} q_1 &= 0.32 \\ &\cong 0.7^3[0.7 + 0.3(0.7 + 0.3 \times 0.7^2)] \\ &= 0.327 \\ F_1 &= x_1x_2x_3[x_4 + \bar{x}_4(x_5 + \bar{x}_5x_6x_7)] \\ &= x_1x_2x_3(x_4 + x_5 + x_6x_7) \end{aligned}$$

The minimal assignment of integral weights is

$$a_1 = a_2 = a_3 = 5, \quad a_4 = a_5 = 2, \quad a_6 = a_7 = 1, \quad T = 17.$$

(2) Determination of F_2.

$$q_1 + q_2 = 0.32 + 0.27 = 0.59 < 0.7$$

$$\frac{0.59}{0.7} = 0.843 > 0.7$$

$$d_1 = 1$$

$$0.843 - 0.7 = 0.143$$

$$\frac{0.143}{0.3} = 0.477 < 0.7$$

$$\frac{0.477}{0.7} = 0.681 < 0.7$$

$$\frac{0.681}{0.7} = 0.973 > 0.7$$

$$d_2 = 2$$

$$0.973 - 0.7 = 0.273$$

$$\frac{0.273}{0.3} = 0.91 > 0.7$$

$$d_3 = 0$$

$$0.91 - 0.7 = 0.21$$

$$\frac{0.21}{0.3} = 0.7 = 0.7$$

$$d_4 = 0$$

$$0.7 - 0.7 = 0$$

The difference is equal to zero. So we stop here.

$$\begin{aligned} q_1 + q_2 &= 0.59 \\ &\cong 0.7\ \{0.7 + 0.3 \times 0.7^2[0.7 + 0.3(0.7 + 0.3 \times 0.7)]\} \\ &= 0.590 \end{aligned}$$

$$\begin{aligned} F_2 &= x_1\{x_2 + \bar{x}_2 x_3 x_4[x_5 + \bar{x}_5(x_6 + \bar{x}_6 x_7)]\} \\ &= x_1[x_2 + x_3 x_4(x_5 + x_6 + x_7)] \end{aligned}$$

$$F_2' = F_2 \bar{F}_1$$

$$a_1 = 10, \quad a_2 = 7, \quad a_3 = a_4 = 3, \quad a_5 = a_6 = a_7 = 1$$

$$T = 17$$

(3) Determination of F_3.

$$q_1 + q_2 + q_3 = 0.32 + 0.27 + 0.23 = 0.82 > 0.7$$

$$d_1 = 0$$

$$0.82 - 0.7 = 0.12$$

$$\frac{0.12}{0.3} = 0.40 < 0.7$$

$$\frac{0.40}{0.7} = 0.571 < 0.7$$

$$\frac{0.571}{0.7} = 0.816 > 0.7$$

$$d_2 = 2$$

$$0.816 - 0.7 = 0.116$$

$$\frac{0.116}{0.3} = 0.387 < 0.7$$

$$\frac{0.387}{0.7} = 0.553 < 0.7$$

$$\frac{0.553}{0.7} = 0.790 > 0.7$$

$$d_3 = 2$$

$$0.790 - 0.7 = 0.090$$

$$\frac{0.090}{0.3} = 0.300 < 0.7$$

The mixed highest power of p and $(1 - p)$ is already 7. So we stop here.

$$\begin{aligned} q_1 + q_2 + q_3 &= 0.82 \\ &\cong 0.7 + 0.3 \times 0.7^2[0.7 + 0.3 \times 0.7^2(0.7 + 0.3)] \\ &= 0.7 + 0.3 \times 0.7^2(0.7 + 0.3 \times 0.7^2) \\ &= 0.825 \end{aligned}$$

$$\begin{aligned} F_3 &= x_1 + \bar{x}_1x_2x_3(x_4 + \bar{x}_4x_5x_6) \\ &= x_1 + x_2x_3(x_4 + x_5x_6) \end{aligned}$$

$$F_3' = F_3\bar{F}_1\bar{F}_2'$$

$$a_1 = 8, \quad a_2 = a_3 = 3, \quad a_4 = 2, \quad a_5 = a_6 = 1, \quad a_7 = 0$$

$$T = 8$$

(4) Determination of F_4.

$$q_1 + q_2 + q_3 + q_4 = 0.32 + 0.27 + 0.23 + 0.18 = 1.00$$

$$F_4 = 1$$

$$F_4' = F_4\bar{F}_1\bar{F}_2'\bar{F}_2' = \bar{F}_1\bar{F}_2'\bar{F}_3'$$

7. *Error Bounds*

The error in using threshold logic elements as a probability transformer depends on m. The larger m, the smaller the error. For $p = 1 - p = \frac{1}{2}$, since N_i is the closest integer to $2^m q_2$, the error bound is half of the probability of the m-tuple, or $\frac{1}{2} \times 1/2^m = 1/2^{m+1}$. For some probabilities, such as q_2 and q_3 in the above example, the probability is the difference between two probabilities each with an error bound of $1/(2^{m+1})$. So the error bound is twice this, or

$$\epsilon = 2 \times \frac{1}{2^{m+1}} = \frac{1}{2^m}.$$

In the case of $m = 7$,

$$\epsilon = \frac{1}{2^7} = \frac{1}{128} = 0.0078 = 0.78\%.$$

For $p \neq 1 - p \neq \frac{1}{2}$, each m-tuple has a different probability. The error bound is more difficult to calculate. It is not attempted to determine the exact error bound here. However, it is roughly estimated that for the same m, the error bound for $p \neq 1 = p \neq \frac{1}{2}$ is a little larger than that for $p = 1 - p = \frac{1}{2}$, as indicated in the above examples.

8. *Sampling or Clock Pulse*

It is assumed that the input symbol is random. Thus the input at time t is independent of the inputs at any previous time. However, in forming the new input symbols, m delay units are used, and the input symbols are therefore dependent upon the past m input symbols. In order to obtain random output variables completely independent of past values, it is necessary to take values of either the input or the output at time intervals of at least m times the time interval of the input. For level inputs and outputs, this can be accomplished by using a sampler at the input or the output synchronized to the input time and with a sampling period $T \geqq m\tau$, where τ is the period of input time interval. In the case of pulse inputs and outputs, clock pulses with a period of $T \geqq m\tau$ can be applied to the input delay units.

9. *Conclusions*

From the above analysis it is seen that probability transformer can be realized with threshold logic elements, to a reasonable degree of accuracy. For s output probabilities, if one threshold logic element is used for each function, only $s - 1$ threshold logic elements are required, together with $[\log_2 s]$ OR gates. So the circuit thus realized is much simpler than a corresponding sequential circuit or that realized with AND and OR gates. High accuracy can be obtained by using larger number of delay units. In that case the weights of the variables tend to be large. However, this can be remedied by the decomposition of each threshold logic element into two or more threshold logic elements. Thus small error tolerance can be achieved at the expense of number of threshold logic elements, and the resulting realization can still be fairly simple. So the use of threshold logic elements for probability transformation appears to be a neat and simple circuit realization.

RECEIVED July, 1964

REFERENCES

1. GILL, A. Synthesis of probability transformers. *J. Franklin Inst. 274* (July, 1962), 1-19.
2. ——. On a weight distribution problem with application to the design of stochastic generators. *J. ACM 10* (Jan. 1965), 110-121.
3. MOORE, E. F. Gedanken-experiments on sequential machines. In *Automata Studies*, ed. by C. E. Shannon and J. McCarthy, Princeton U. Press, 1956, pp. 129-153.
4. ERCOLI, P., AND MERCURIO, L. Threshold logic with one or more than one threshold. Proc. IFIP Congress 62, North-Holland Publishing Co., Amsterdam, 1963, pp. 741-746.
5. SHENG, C. L. A method for testing and realization of threshold functions. Tech. Rep. No. 63-6, Dept. of Electr. Eng., U. of Ottawa Canada, May 1963; also *IEEE Trans. EC-13* (June, 1964), 232-239.
6. GABELMAN, I. J. The synthesis of Boolean functions using a single threshold element. *IEEE Trans. EC-11* (Oct. 1962), 639-642.
7. MATTSON, R. L. The analysis and synthesis of adaptive systems which use networks of threshold elements. Rep. SEL-62-132 (TR No. 1553-1), Stanford Electronics Laboratories, Stanford, Calif., Dec. 1962.
8. GLINSKI, G. S., AND YUE, C. K. Decomposition of n-variable threshold function into p-variable threshold functions, where $p < n$. Tech. Rep. No. 63-10, Dept. Electr. Eng., U. of Ottawa, Canada, June, 1963.