PROBABILITY OF ERROR OF THRESHOLD GATE NETWORKS

A Thesis

Presented to

The Faculty of the Department of Electrical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

---

by

James E. Howard

January 1969

ACKNOWLEDGEMENT

I would like to gratefully acknowledge the encouragement and assistance of my advisor, Dr. J. D. Bargainer.

My deepest appreciation goes to my wife, Sara, for her encouragement and understanding.

I would also like to thank Mrs. Norma Turner for her excellent work in typing this thesis.

The research reported here was supported by National Science Foundation Grant NSF GP-1539.

PROBABILITY OF ERROR OF THRESHOLD GATE NETWORKS

An Abstract of a Thesis

Presented to

the Faculty of the Department of Electrical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

---

by

James E. Howard

January 1969

ABSTRACT

A threshold gate is a logic gate in which the output is determined by a weighted sum of the inputs compared to a threshold. If the weighted sum is less than the threshold, the output is a zero. If the weighted sum is greater than the threshold, the output is a one.

In general, the inputs, weights and threshold are correlated random variables. It is possible for the weighted sum and the mean of the weighted sum to lie on opposite sides of the threshold. This will cause an error in the output of the gate.

Techniques are derived for calculating the probability of error of single threshold gates and nonsequential threshold gate networks. It is assumed that the means, variances, and correlations of the inputs and weights are known, and that the probability of occurrence of the network input combinations is known.

Threshold gates with random inputs and weights are then studied by simulation. Each input and weight is replaced by the appropriately generated random number, thereby generating the density function of the weighted sum. The form of this density function is important in the calculation of the probability of error.

The techniques for calculating probability of error are implemented by digital computer programs. These are used as subroutines for an adaptive search technique to minimize the probability of error by adjusting the mean value of the weights. The dependence of probability of error on the variance and correlation of inputs and weights is examined for both optimal and non-optimal realizations.

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER I

THRESHOLD LOGIC

A logic gate is a system in which the output is related to the input by some logic function. Such a device need not be limited to realizing only the simplest logic functions, the AND and OR functions, although these are the easiest to implement. In fact, it is highly desirable that a single gate be capable of realizing more complicated logical functions. In this way, the number of gates needed in a logic circuit may be reduced significantly. It will be shown later that a threshold gate has this property.

Definition of a Threshold Gate

A threshold gate is a logic gate with binary inputs and a binary output. These binary variables can take on values of 0 or 1. Associated with each input, x_1, x_2,...,x_n, there is a weight, w_1, w_2, w_3,..., w_n. The output y of a threshold gate for any combination of n inputs can be expressed as

$$\begin{aligned} y = 1 \quad & \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq T \\ y = 0 \quad & \text{if} \quad \sum_{i=1}^{n} w_i x_i < T \end{aligned} \tag{1.1}$$

where T is a real number which is called the threshold. The notation

$$y = \left\langle \sum_{i=1}^{n} w_i x_i \right\rangle_T \tag{1.2}$$

is used to represent Eq. 1.1.

By subtracting T from both sides of the inequality, Eq. 1.1 becomes

$$\begin{aligned} y = 1 \quad &\text{if} \quad \sum_{i=1}^{n} w_i x_i - T \geq 0 \\ y = 0 \quad &\text{if} \quad \sum_{i=1}^{n} w_i x_i - T < 0 \quad . \end{aligned}$$

If $x_{n+1} = 1$ and $w_{n+1} = -T$ the inequalities become

$$\begin{aligned} y = 1 \quad &\text{if} \quad \sum_{i=1}^{n+1} w_i x_i \geq 0 \\ y = 0 \quad &\text{if} \quad \sum_{i=1}^{n+1} w_i x_i < 0 \quad . \end{aligned} \tag{1.3}$$

Using matrix notation this becomes

$$\sum_{i=1}^{n+1} w_i x_i = \underline{x}^T \underline{w} \quad .$$

where $\underline{x}^T = \{x_1, x_2, \ldots, x_n, 1\}$, $\underline{w}^T = \{w_1, w_2, \ldots, w_n, -T\}$.

Finally,

$$y = 1 \quad \text{if} \quad \underline{x}^T\underline{w} \geq 0$$
$$y = 0 \quad \text{if} \quad \underline{x}^T\underline{w} < 0 \tag{1.4}$$

or more compactly

$$y = \left\langle \underline{x}^T\underline{w} \right\rangle_0 .$$

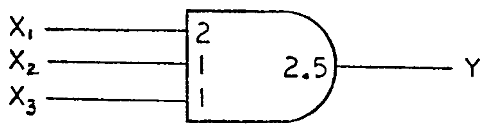


Fig. 1.1. Threshold gate of Example 1.1

Example 1.1. For the threshold gate shown in Fig. 1.1, the output in terms of the separating function is

$$y = \left\langle 2x_1 + x_2 + x_3 \right\rangle_{2.5}$$

The output as a logic function is

$$y = x_1(x_2 + x_3)$$

This is shown in Table 1.1.

Table 1.1. Truth table for $y = \left\langle 2x_1 + x_2 + x_3 \right\rangle_{2.5}$

x_1, x_2, x_3			$2x_1 + x_2 + x_3$	$y = \left\langle 2x_1 + x_2 + x_3 \right\rangle_{2.5}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	2	0
1	0	0	2	0
1	0	1	3	1
1	1	0	3	1
1	1	1	4	1

Geometric Interpretation

Switching space is an n-dimensional Euclidean space (n-space) where each coordinate axis corresponds to an independent binary input of a logic gate or system. Since each input can have only values of 0 or 1, the input combinations are discrete points in the space. Each of the 2^n points corresponding to the 2^n possible input combinations lies on the vertex of a unit n-dimensional hypercube (n-cube) in n-space. Realizing a given logic function of n variables with a single threshold gate corresponds to passing an n-dimensional hyperplane through the n-cube so that the plane separates the points at which the value of the function is equal to 0 from the points at which the value of the function is equal to 1. The equation of the plane is

$$\sum_{i=1}^{n} x_i w_i = T$$

where the domain of each x_i is the i-th coordinate axis.

Functions which can be realized by a single threshold gate are called linearly separable (l.s.). The function

$$f(p) = \sum_{i=1}^{n} x_i w_i$$

is called the separating function, where p is the vertex of the n-cube corresponding to $(x_1, x_2, \ldots, x_n)$. Figure 1.2 shows the n-cube and separating hyperplane for a two-variable function.

Not every partition of the vertices of the n-cube can be separated by a hyperplane as in Fig. 1.2b, hence not every logic function can be realized by a single threshold gate.

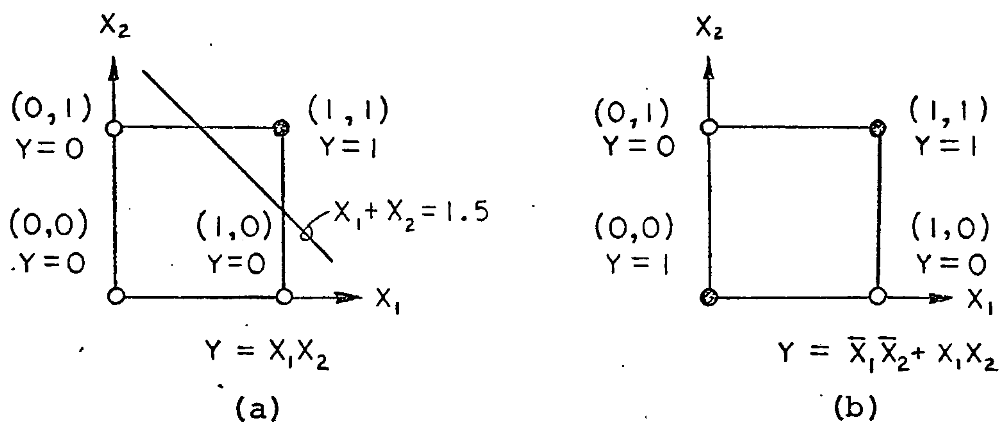


Fig. 1.2. (a) l.s. function and (b) non-l.s. function

Notice that if the equation for the separating plane of a particular function

$$y = \left\langle \sum_{i=1}^{n} w_i x_i \right\rangle_T$$

is multiplied by any positive constant K exactly the same plane results. Therefore

$$y = \left\langle K \sum_{i=1}^{n} w_i x_i \right\rangle_{KT}$$

also realizes the function. Specifically

$$y = \left\langle 2x_1 + 2x_2 \right\rangle_3$$

also realizes the function of Fig. 1.2a. Note that there can be other hyperplanes not necessarily parallel that separate the function such as $x_1 + x_2 = 7/4$ or $x_1 + 2x_2 = 5/2$. Thus the separating plane is not unique, therefore the values of the weights and threshold to realize a particular logic function are not unique.

<u>The Map Interpretation</u>

Every realization of a logic function with a threshold gate specifies a separating function f and a logic function F both of which are defined on the vertices of the n-cube. For each vertex p, there exists an ordered triple (p,f(p), F(p)). The set {(p,f(p),F(p))} of ordered triples for all vertices of the n-cube is called the map of F generated by f.

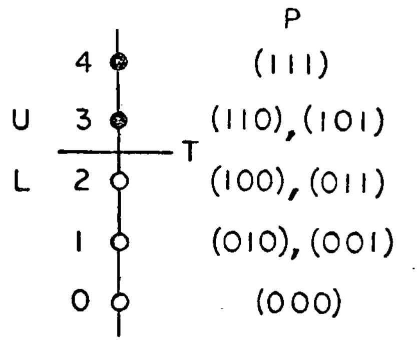


Fig. 1.3. The map of $F(p) = \left\langle 2x_1 + x_2 + x_3 \right\rangle_{2.5}$

For each p, there is a point on the real line at f(p). This point is shown by ○ if F(p) equals 0 or by ● if F(p) equals 1. Figure 1.3 shows the map of Example 1.1.

A map is divided into two disjoint subsets called the zero and unit parts, those for which F(p) = 0 and those for which F(p) = 1. Let U be the smallest f(p) such that F(p) = 1 and let L be the largest f(p) such that F(p) = 0. A map is separated if U > L. It has been proved that a logic function F is l.s. if and only if there exists a function of the form

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i x_i$$

that yields a separated map for F. (Lewis and Coates)

Classes of Threshold Functions

Consider an arbitrary logic function $F(x_1,\ldots,x_k,\ldots,x_n)$. The class of 2^n function obtained by replacing variables in F by their complements is called the complementary symmetry class corresponding to F. Suppose $F_k(x_1,\ldots,\bar{x}_k,\ldots x_n)$ is equivalent to F except that x_k is replaced everywhere by $\bar{x}_k$. If $y = \left\langle w_1x_1 + \ldots + w_kx_k + \ldots + w_nx_n \right\rangle_T$ is a realization of F, then $y = \left\langle w_1x_1 + \ldots + w_k\bar{x}_k + \ldots + w_nx_n \right\rangle_T$ is a realization of F_k. If F is l.s., then all members of the complementing symmetry class are l.s. In addition, there exists a logic function $\phi(x_1,x_2,\ldots,x_n)$ in the same complementing symmetry class as F such that the realization of ϕ has all positive weights when the inputs are x_1, $x_2,\ldots,$ x_n. Also, a realization for F can be obtained from the realization of ϕ by complementing some of the variables and changing the threshold. Therefore, a realization for F with all positive weights can be found.

A logic function is unate if and only if in the minimum sum of products (MSP) form no variables appear both complemented and uncomplemented. It has been proved that if a logic function F is l.s., then it is unate and if

$$f(p) = \sum_{i=1}^{n} w_ix_i$$

is the separating function for F, then for each i, $w_i > 0$

(or $w_i < 0$) if and only if x_i (or $\overline{x}_i$) appears in the MSP form for F. (McNaughton) A function that contains no complemented variables in the MSP form is called a positive unate function. Any other function F in the same complementing symmetry class can be obtained from the positive unate function F_0 by a simple transformation. By replacing variables by their complements in F_0 as needed, any function F in the same complementing symmetry class can be realized. The realization of F has the same realization as F_0 except that some variables are replaced by their complements. This realization of F has all positive weights. In this thesis, only those realizations that have all positive weights will be considered.

For every logic function $F(x_1,x_2,\ldots,x_n)$, there corresponds a dual function $F^d(x_1,x_2,\ldots,x_n)$ which is defined as

$$F^d(x_1,x_2,\ldots,x_n) = F(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_n)$$

A function F is called self dual if $F = F^d$. It has been proved that if $y = \left\langle \underline{x}^T \underline{w} \right\rangle_T$ is a realization of F then $y = \left\langle \underline{x}^T \underline{w} \right\rangle_{\sigma - T}$ is a realization of F^d where

$$\sigma = \sum_{i=1}^{n} w_i$$

and F is l.s. if and only if F^d is l.s. (Lewis and Coates)

Notice that the AND and OR functions of n variables are dual functions.

CHAPTER II

STATISTICAL ANALYSIS OF THRESHOLD GATE CIRCUITS

Threshold gate circuits are subject to input and weight variations. These variations may cause the gate to produce an erroneous output. In this chapter, three threshold gate circuits are examined in order to determine the nature of these variations.

Logic Systems

In the preceding chapter, threshold gates were considered as logic gates whose inputs were 0 or 1. In practice, the inputs are voltage levels which are assigned the logical value of 0 or 1 according to the value of the voltage level. For example, a logical 0 may be a voltage in the range 0.0v to 0.8v and a logical 1 may be a voltage in the range 1.6v to 5.0v. Positive logic is defined as a logic system in which the voltage level that represents a 1 is always greater than the voltage level that represents a 0. Negative logic is defined as a logic system in which the voltage level that represents a 0 is always greater than the voltage level that represents a 1. Now the definition of threshold logic will be extended to such systems.

Consider the positive logic system in which a 0 is represented by a voltage level c_1 and a 1 is represented by a voltage level $c_1 + c_2$, $c_2 > 0$. Let z_i be an input variable in this system. For the logic variable x_i defined on $\{0,1\}$,

$$z_i = c_1 + c_2 x_i \tag{2.1}$$

Let $y = F(x_1, x_2, \ldots, x_n)$ be an arbitrary l.s. logic function of n variables whose threshold gate realization is

$$y = \left\langle f(p) \right\rangle_T = \left\langle \sum_{i=1}^{n} w_i x_i \right\rangle_T$$

Consider $F(z_1, z_2, \ldots, z_n)$. Define a new function

$$f_0(p) = \sum_{i=1}^{n} w_i z_i$$

Substituting $c_1 + c_2 x_i$ for z_i produces

$$\begin{aligned} f_0(p) &= \sum_{i=1}^{n} w_i (c_1 + c_2 x_i) \\ &= c_1 \sum_{i=1}^{n} w_i + c_2 \sum_{i=1}^{n} w_i x_i \qquad (2.2) \\ &= c_1 \sigma + c_2 f(p) \qquad \text{where } \sigma = \sum_{i=1}^{n} w_i \end{aligned}$$

Note that if $F = 1$, $f(p) \geq T$ and $f_0(p) \geq c_1\sigma + c_2 T$, and if

$F = 0$, $f(p) < T$ and $f_0(p) < c_1\sigma + c_2T$. Thus $f_0(p)$ is a separating function of $F(z_1, z_2, \ldots, z_n)$ if the threshold is $c_1\sigma + c_2T$. Therefore, $F(z_1, z_2, \ldots, z_n)$ is l.s. and equivalent to $F(x_1, x_2, \ldots, x_n)$. A realization of $y = F(z_1, z_2, \ldots, z_n)$ is given by

$$y = \left\langle \sum_{i=1}^{n} w_i z_i \right\rangle_{c_1\sigma + c_2T} \tag{2.3}$$

Thus any l.s. function F with positive logic input z_i can be realized with a single threshold gate and this realization can be obtained from the realization for inputs x_i defined on $\{0,1\}$ by changing the threshold.

Consider the negative logic system in which a 1 is represented by a voltage level d_1 and a 0 is represented by a voltage level $d_1 + d_2$, $d_2 > 0$. Let w_i be an input variable in the system. For the logic variable x_i defined on $\{0,1\}$,

$$w_i = d_1 + d_2\bar{x}_i \tag{2.4}$$

Hence each w_i can be made to correspond to the complement of a positive logic variable z_i. If $c_1 = d_1$ and $c_2 = d_2$, then

$$\bar{z}_i = c_1 + c_2\bar{x}_i = d_1 + d_2\bar{x}_i = w_i \tag{2.5}$$

Let $y_1 = F_1(z_1,z_2,\ldots,z_n)$ and $y \ = F \ (w_1,w_2,\ldots,w_n)$ be logic functions where y_1 and $z_1, z_2,\ldots, z_n$ are in a positive logic system and y_2 and $w_1, w_2,\ldots, w_n$ are in a negative logic system and the systems are related by $w_i = \overline{z}_i$ and $y_2 = \overline{y}_1$. Converting from a positive logic system to a negative logic system requires complementing each input and the output

$$y_2 = \overline{F}_1(\overline{z}_1,\overline{z}_2,\ldots,\overline{z}_n) = F_2(w_1,w_2,\ldots,w_n) \tag{2.6}$$

Note that $F_2(w) = F_1{}^d(z)$. Thus the conversion results in the realization of the dual function. If F_1 is l.s., then $F_1{}^d$ is l.s. and consequently F_2 is l.s. Therefore, a threshold gate realization of a logic function F in a positive logic system realizes the dual function of F in a negative logic system.

A Resistor-Transistor Threshold Gate

One of the earliest threshold gate circuits is the resistor-transistor gate due to Rowe shown in Fig. 2.1. Each of the inputs, $v_1, v_2,\ldots,v_n$ is a voltage level V_0 or V_1 which represent a logical 0 or a logical 1, respectively. The value of the weight w_i is inversely proportional to the value of the resistor R_i. The threshold is determined by the values of R_t and V_t.

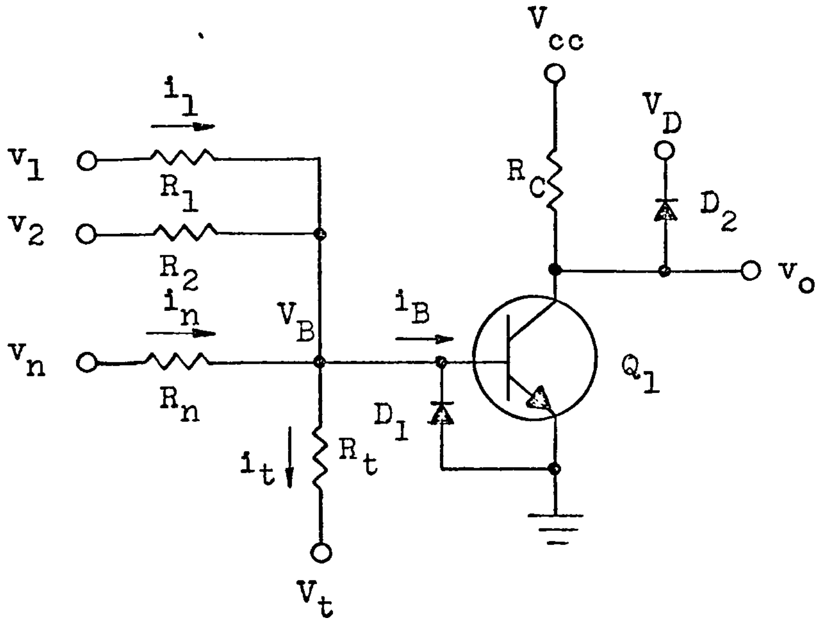


Fig. 2.1 Resistor-transistor Threshold Gate Circuit

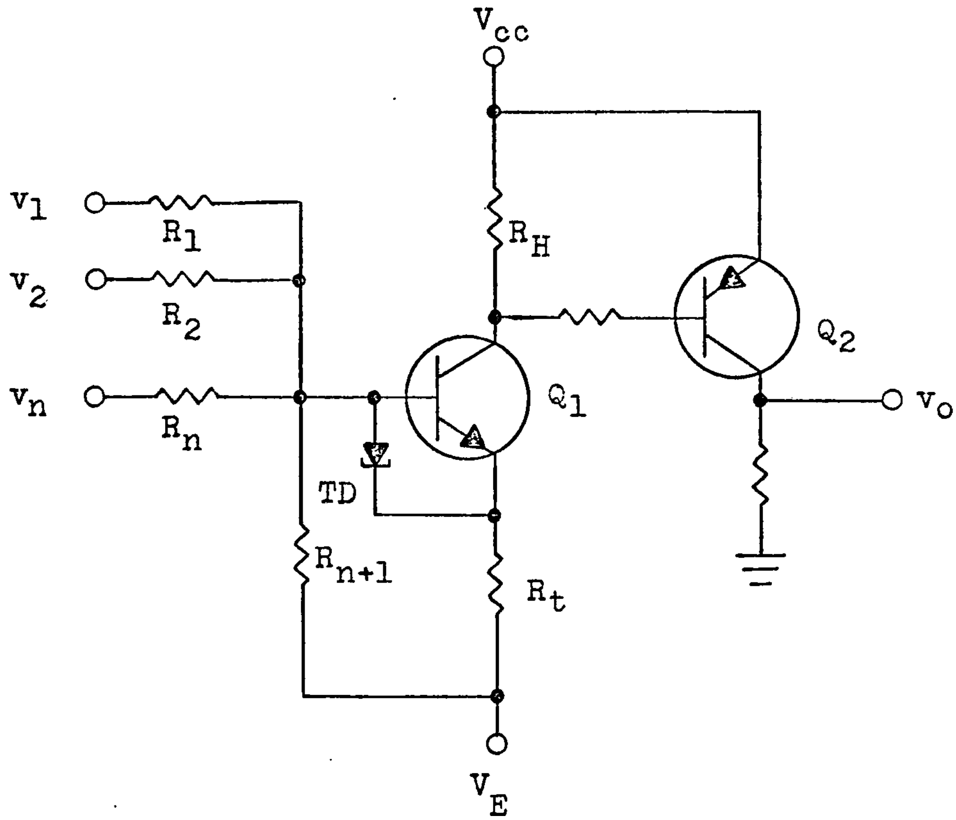


Fig. 2.2 Tunnel Diode-transistor Threshold Gate Circuit

Consider the inputs to be in a positive logic system and let $V_1 > V_0 > 0$. Let V_γ be the base-emitter cutin voltage of the transistor. At cutin, the base current is given by

$$i_B = \sum_{i=1}^{n} i_i - i_t$$

$$= \sum_{i=1}^{n} \frac{v_i - V_\gamma}{R_i} + \frac{V_t - V_\gamma}{R_t} = f(v_1, \ldots, v_n)$$

$$= \sum_{i=1}^{n} w_i v_i + c \qquad \text{where } w_i = \frac{1}{R_i} \text{ and}$$

$$c = \sum_{i=1}^{n} - \frac{V_\gamma}{R_i} + \frac{V_t - V_\gamma}{R_t}$$

For $f(v_1, \ldots, v_n) \leq 0$, $i_B = 0$ and the transistor is cut off and $v_0 = V_D + V_{D_2}$, where V_{D_2} is the voltage drop across diode D_2. If $f(v_1, \ldots, v_n) > 0$, $i_B > 0$ and the transistor conducts. If i_B is large enough, the transistor will saturate and $v_0 = V_{CE(SAT)}$. Let $V_0 = V_{CE(SAT)}$ and $V_1 = V_D + V_{D_2}$. Thus, the output will be a 0 for those input combinations for which the weighted sum of the input voltages is sufficient to cause the transistor to saturate. For those input combinations which are not sufficient to cause the transistor to conduct, the output will be a 1. Ideally, the gate should be constructed so that no input combination can occur which causes the transistor to remain in the active region, i.e., not saturated or cutoff.

In the foregoing analysis, it is assumed that the input voltage levels were at discrete values V_0 or V_1 and that the weights were constant. In practice, the inputs are not constant voltages but are perturbed by drift of their D.C. level, noise, and power supply variation. Generally, these factors are statistically independent. It is therefore reasonable to assume that each input is a random variable which is normally distributed with mean V_0 and variance σ_0^2 or mean V_1 and variance σ_1^2, the inputs being a 0 or a 1, respectively. In this thesis, it will be assumed that the set of inputs to a threshold gate is a correlated set of jointly normal random variables.

In the resistor-transistor threshold gate, the weights are inversely proportional to resistor values. These resistors vary with temperature, thermal noise, and age. The weights, therefore, are also dependent upon these factors. The threshold is dependent on the values of the resistors $R_1, \ldots, R_n$, the resistor R_t, and the transistor switching characteristics. All these factors may be represented as random variables which will be correlated since they are all dependent upon temperature. It is therefore reasonable to represent the weights as a set of correlated, jointly normal random variables.

Consider the change in resistance with temperature

$$\frac{\Delta R}{\Delta T} = \alpha R_0$$

The resistance of a resistor at temperature $T_0 + \Delta T$ is given by

$$R = R_0 + \Delta R = R_0(1 + \alpha\Delta T)$$

where R_0 is the resistor at the temperature T_0. The change in a weight with temperature is given by

$$w_0 + \Delta w = \frac{1}{R_0 + \Delta R} = \frac{1}{R_0(1 + \alpha\Delta T)} = \frac{1}{R_0} - \left(\frac{\alpha\Delta T}{1 + \alpha\Delta T}\right)\frac{1}{R_0}$$

therefore

$$\Delta w = -\frac{\alpha\Delta T}{1 + \alpha\Delta T}\, w_0$$

Thus the change in the weights is proportional to the new value of the weight. A similar analysis may be done for noise and age variations which are proportional to resistance.

Variation in the inputs and weights causes variation in the separating function and the threshold of a gate. This can cause the transistor to enter the active region or to give a completely erroneous output. That is, the output may be a 1 (or 0) when the logic function is a 0 (or 1). Due to the large hysteresis of the switching characteristics of the resistor-transistor threshold gate (typically as large as 200 mV), functions of more than a few variables may possibly not be realized without a large probability of error.

Transistor-Tunnel Diode Threshold Gate

Another threshold gate circuit is the transistor-tunnel diode gate due to Canion shown in Fig. 2.2. The addition of the tunnel diode in series with the base-emitter junction of the input transistor results in a significant reduction in the switching hysteresis. Other than this, the operation of the circuit is similar to the resistor-transistor threshold gate discussed previously.

Reduction of the switching hysteresis results in more reliable gate operation for randomly varying inputs and weights. It is also possible to reliably realize functions with a greater number of input variables than with the resistor-transistor gate.

Current Switching Threshold Gate

One of the latest threshold gate circuits is that due to Amodei et al. shown in Fig. 2.3. Each input voltage V_{IN_i} is compared to a reference voltage V_{REF} by a differential amplifier. A current I_i, given by

$$I_i = \frac{V_{REF} - V_{BE_i}}{R_i}$$

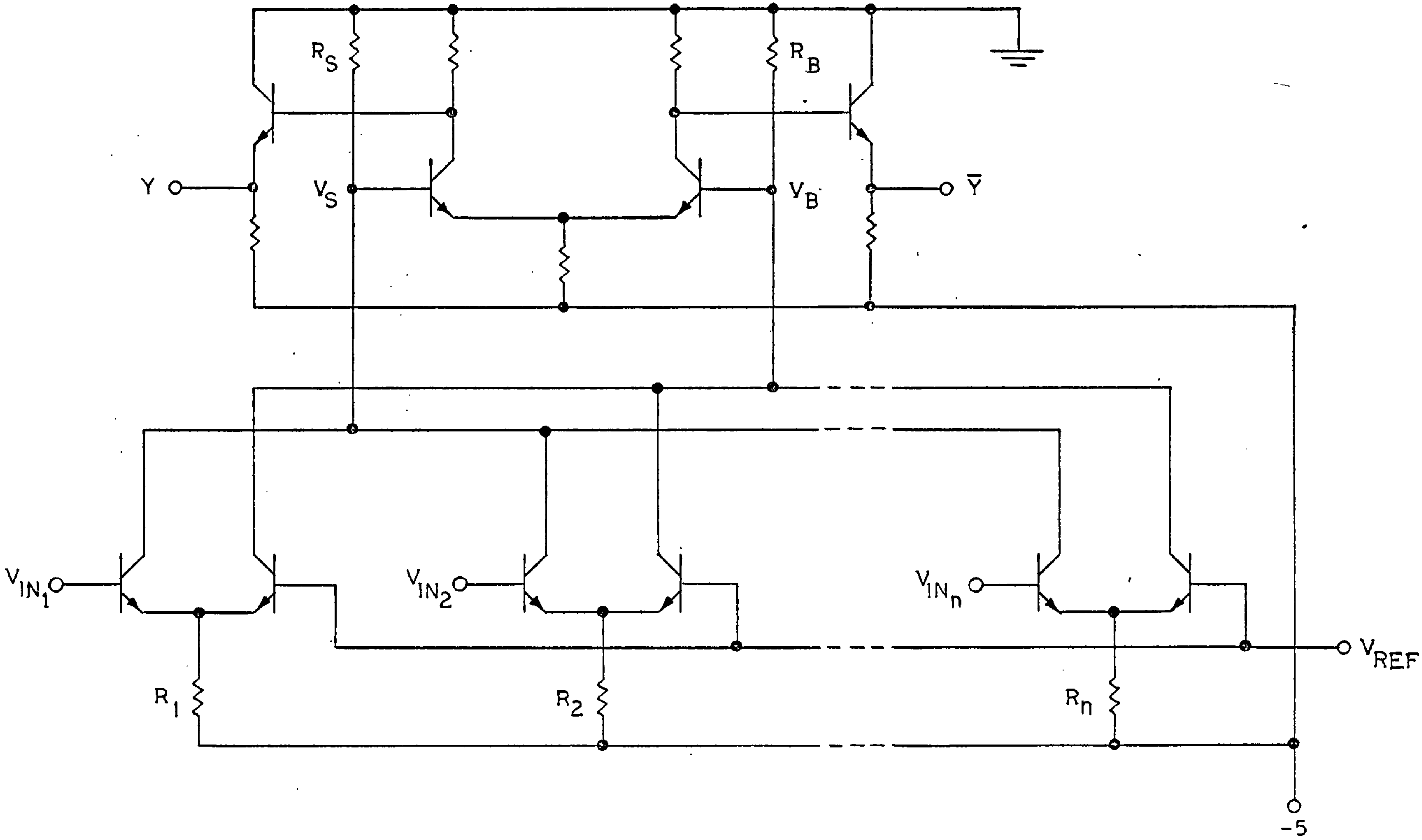


Fig. 2.3. Current Switching Threshold Gate Circuit

is switched from the point V_s or the point V_B depending on whether $V_{IN_i} > V_{REF}$ or $V_{IN_i} < V_{REF}$, respectively. Note that unlike the other two gates considered, this gate does not require input voltages that are constant levels V_0 or V_1 for n inputs equal to 0 or 1. For a positive logic system, a 1 may be any voltage greater than V_{REF} and a 0 may be any voltage less than V_{REF}.

The two summed currents at V_B and V_s develop a voltage difference across the output differential amplifier which causes the output to be a 0 or 1 according to whether $V_s > V_B$ or $V_s < V_B$, respectively. The threshold is determined by the ratio of R_s and R_B. The weights w_i are determined by R_i, V_{REF}, and the characteristics of the i-th input differential amplifier. The separating function may be written as

$$f(p) = \sum_{i=1}^{n} w_i z_i$$

where $w_i = \frac{1}{R_i}$ and $z_i = V_{REF} - V_{BE_i}$. Let the z_i's be called internal inputs. As in the resistor-transistor gate, the w_i's and the z_i's may be considered to be random variables. In the current switching gate, however, the variations in z_i are due to V_{REF} and the base-emitter voltage of the input amplifier, and not directly to the input variations. The variations in V_{REF} and V_{BE_i} are generally much smaller than the input variations. Thus, this type of gate produces a decoupling of the inputs from the separating function. Note that V_{BE_i} will still depend slightly on the input voltage variation.

Probability Model of a Threshold Gate

In a threshold gate, the inputs and weights are random variables. Each weight w_i may be written as

$$w_i = \eta_{w_i} + w_i' \tag{2.7}$$

where η_{w_i} is the mean value of w_i, and w_i' is a random variable with zero mean. Likewise, each input x_i may be written as

$$x_i = \eta(x_i) + x_i' \tag{2.8}$$

where $\eta(x_i)$ is the mean value of x_i given the logical value of x_i, and x_i' is a random variable with zero mean. Because each input and each weight is the sum of several independent random variables, the following are reasonable assumptions.

a. $\{x_1, x_2, \ldots, x_n\}$ is a set of jointly normal random variables with covariance matrix M_x.

b. $\{w_1, w_2, \ldots, w_n, w_{n+1}\}$ is a set of jointly normal random variables with covariance matrix M_w.

c. All x_i and w_i are independent.

d. The variance of a weight is proportional to the mean of that weight.

e. The variance of an input is constant.

CHAPTER III

PROBABILITY OF ERROR OF A THRESHOLD GATE

Due to variations in inputs and weights, a threshold gate may not always operate as designed. Associated with a threshold gate is the probability that it will not give the desired logical output for given logical inputs. This probability is the probability of error of the gate for a given input combination.

The total probability of error P_E of a threshold gate is the sum over all possible input combinations of the probability of error for a given input combination $\underline{x}_j$, Pr $\{error \mid \underline{x} = \underline{x}_j\}$, multiplied by the probability of occurrence of that input combination Pr $\{\underline{x} = \underline{x}_j\}$. Thus

$$P_E = \sum_{j=1}^{m} \Pr\,\{error \mid \underline{x} = \underline{x}_j\} \Pr\,\{\underline{x} = \underline{x}_j\} \qquad (3.1)$$

where m is the number of possible input combinations. For n input variables, the maximum value of m is 2^n. If all possible input combinations are equally probable the probability of error is

$$P_E = \frac{1}{2^n} \sum_{j=1}^{2^n} \Pr\,\{error \mid \underline{x} = \underline{x}_j\} \qquad (3.2)$$

The problem is reduced to finding the probability of error for a given input combination.

Consider any combination of n inputs $\underline{x}_j$. If the logic function $F(\underline{x}_j)$ equals 0 the probability of error is given by

$$Pr\ \{y = \left\langle f(p) \right\rangle_0 = 1\} = Pr\ \{f(p) \geq 0\} \qquad (3.3)$$

where $f(p) = \underline{x}_j^T \underline{w}$. If the logic function $F(\underline{x}_j)$ equals 1 the probability of error is given by

$$Pr\ \{y = \left\langle f(p) \right\rangle_0 = 0\} = Pr\ \{f(p) < 0\} \qquad (3.4)$$

Therefore, if the distribution of f(p) is known, the probability of error may be calculated.

The random variables $\underline{x}^T$ and $\underline{w}$ may be written

$$\underline{x}^T = \underline{n}^T{}_x + \underline{x}'^T \quad \text{and}$$
$$\underline{w} = \underline{n}_w + \underline{w}' \qquad (3.5)$$

where $\underline{x}^T = \{x_1, x_2, \ldots, x_n, -1\}$, $\underline{n}^T{}_x = \{n_{x_1}, n_{x_2}, \ldots, n_{x_n}, -1\}$,

$\underline{x}'^T = \{x'_1, x'_2, \ldots, x'_n, 0\}$, $\underline{w}^T = \{w_1, w_2, \ldots w_{n+1}\}$,

$\underline{n}^T{}_w = \{n_{w_1}, n_{w_2}, \ldots, n_{w_{n+1}}\}$, and $\underline{w}'^T = \{w'_1, w'_2, \ldots, w'_{n+1}\}$. Using this notation, the separating function can be written

$$f(p) = \underline{x}^T \underline{w} = \underline{\eta}^T{}_x \underline{\eta}_w + \underline{\eta}^T{}_x \underline{w}' + \underline{x}'^T \underline{\eta}_w + \underline{x}'^T \underline{w}'$$

or (3.6)

$$f(p) = \sum_{i=1}^{n+1} (\eta_{x_i} \eta_{w_i} + \eta_{x_i} w'_i + x'_i \eta_{w_i} + x'_i w'_i)$$

The expected value η_f of the separating function is given by

$$\eta_f = E(\underline{x}^T \underline{w}) = E\left(\sum_{i=1}^{n+1} (\eta_{x_i} \eta_{w_i} + \eta_{x_i} w'_i + x'_i \eta_{w_i} + x'_i w'_i)\right)$$

$$= \sum_{i=1}^{n+1} [E(\eta_{x_i} \eta_{w_i}) + E(\eta_{x_i} w'_i) + E(x_i' \eta_{w_i})$$

$$+ E(x'_i w'_i)]$$

Since all x'_i and w'_i have zero mean values, and all η_{x_i} and η_{w_i} are constants,

$$\eta_f = \sum_{i=1}^{n+1} \eta_{x_i} \eta_{w_i} = \underline{\eta}^T{}_x \underline{\eta}_w \tag{3.7}$$

Equation (3.6) may be rewritten as

$$f(p) = \underline{\eta}^T{}_x \underline{\eta}_w + \sum_{i=1}^{n+1} \eta_{x_i} w'_i + \sum_{i=1}^{n} x'_i \eta_{w_i} + \sum_{i=1}^{n} x'_i w'_i \tag{3.8}$$

For a non-trivial threshold gate $n \geq 2$. For a practical threshold gate $n \geq 3$ as the only gates realized for $n = 2$ and the AND and OR gates. Thus, the last term in Eqn. 3.8 is the sum of at least three random variables. However, these random variables are not independent unless all inputs and weights are uncorrelated.

In general, the inputs $\underline{x} = \{x_1, x_2, \ldots, x_n\}$ and weights $\underline{w} = \{w_1, w_2, \ldots, w_n\}$ are each normally distributed with covariance matrices M_x and M_w, respectively. There exists a nonsingular n x n matrix Q and a nonsingular n+1 x n+1 matrix P such that if

$$\underline{y} = Q^{-1}\underline{x}$$

and

$$\underline{z} = P^{-1}\underline{w}$$

The $\underline{y} = \{y_1, y_2, \ldots, y_n\}$ and $\underline{z} = \{z_1, z_2, \ldots, z_{n+1}\}$ are each normally distributed with all terms independent. (Miller) If $\underline{n}_x$ and $\underline{n}_w$ are the means of $\underline{x}$ and $\underline{w}$, respectively,

$$\underline{n}_y = E(\underline{y}) = Q^{-1}\underline{n}_x$$

and

$$\underline{n}_z = E(\underline{z}) = P^{-1}\underline{n}_w$$

Let $\underline{y} = \underline{n}_y + \underline{y}'$ and $\underline{z} = \underline{n}_z + \underline{z}'$. The separating function can be written

$$f(p) = \underline{x}^T\underline{w} = \underline{y}^T Q^T P\underline{z} = \underline{y}^T H\underline{z}$$

or

$$f(p) = \underline{\eta}_y^T H\underline{\eta}_z + \underline{\eta}_y^T H\underline{z}' + \underline{y}'^T H\underline{\eta}_z + \underline{y}'^T H\underline{z}'$$

where $H = Q^T P$. Note that the sum $\underline{\eta}_y^T H\underline{z}' + \underline{y}'^T H\underline{\eta}_z$ is the sum of independent normally distributed random variables and is therefore normally distributed. The term $\underline{y}'^T H\underline{z}'$ is not normally distributed.

Assume that the standard deviation σ_{w_i} of each weight w_i is small compared to its mean η_{w_i}, that is,

$$\sigma_{w_i} << \eta_{w_i} \tag{3.9}$$

Since $z_i' = \sum_{j=1}^{n+1} p_{ij}w_j'$ and $\eta_{z_i} = \sum_{j=1}^{n+1} p_{ij}\eta_{w_j}$, then $\sigma_{z_i} << \eta_{z_i}$ for for all z_i. Note that

$$\underline{y}'^T H\underline{\eta}_z = \sum_{i=1}^{n}\sum_{j=1}^{n} y_i' h_{ij}\eta_{z_j}$$

and

$$\underline{y}'^T H\underline{z}' = \sum_{i=1}^{n}\sum_{j=1}^{n} y_i' h_{ij} z_j'$$

Therefore, $y_i' h_{ij} z'_j << y_i' h_{ij}\eta_{z_j}$ for all $i = 1,\ldots,n$ and $j = 1,\ldots,n$. Hence $\underline{y}'^T H\underline{z}'$ is small compared to $\underline{y}'^T H\underline{\eta}_z$ and $\underline{x}'^T\underline{w}$ is small compared to $\underline{x}'^T\underline{\eta}_w$. Now Eq. 3.8 becomes

$$f(p) \simeq \underline{\eta}_x^T\underline{\eta}_w + \underline{\eta}_x^T\underline{w}' + \underline{x}'^T\underline{\eta}_w \tag{3.10}$$

The separating function is now a sum of normal random variables and is therefore normally distributed.

The variance of the separating function σ_f^2 can now be calculated:

$$\sigma_f^2 = \sigma^2(\underline{n}_x^T\underline{n}_w + \underline{n}_x^T\underline{w}' + \underline{x}'^T\underline{n}_w)$$

Since $\underline{x}'$ and $\underline{w}'$ are independent and $\underline{n}_x$ and $\underline{n}_w$ are constant,

$$\sigma_f^2 = \sigma^2(\underline{n}_x^T\underline{w}') + \sigma^2(\underline{x}'^T\underline{n}_w)$$

The variances of linear combinations of normal random variables, $\underline{n}_x^T\underline{w}'$ and $\underline{x}'^T\underline{n}_w$, respectively are

$$\sigma^2(\underline{n}_x^T\underline{w}') = \underline{n}_x^T M_w \underline{n}_x$$

$$\sigma^2(\underline{x}'^T\underline{n}_w) = \underline{n}_w^T M_x \underline{n}_w$$

where M_w and M_x are the covariance matrices of $\underline{w}'$ and $\underline{x}'$, respectively. (Morrison) The variance of the separating function is then

$$\sigma_f^2 = \underline{n}_x^T M_w \underline{n}_x + \underline{n}_w^T M_x \underline{n}_w \tag{3.11}$$

The mean and variance of the separating function are now known; therefore the probability of error can be calculated.

For those points of the n-cube where F = 0, the probability of error is the probability that f(p) is greater that or equal to zero.

$$P(f(p)>0) = \int_0^\infty \frac{1}{\sqrt{2\pi}\sigma_f} e^{-(t-\eta_f)^2/2\sigma_f^2}\, dt = P(\infty) - P(-\frac{\eta_f}{\sigma_f})$$

where $P(x_1) = \Pr\{X \leq x_1\}$, X is a normal random variable with zero mean and unit variance. Note that $P(-\infty) = 0$, $P(\infty) = 1$, and $P(-x) = 1-P(x)$. Thus

$$P(f(p)>0) = 1 - P(-\frac{\eta_f}{\sigma_f}) = P(\frac{\eta_f}{\sigma_f}) \tag{3.12}$$

For those points of the n-cube where F=1, the probability of error is the probability that f(p) is less than 0.

$$\begin{aligned} P(f(p)<0) &= \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}\,\sigma_f} e^{-(t-\eta_f)^2/2\sigma_f^2} \, dt \\ &= P(-\frac{\eta_f}{\sigma_f}) - P(-\infty) \\ &= P(-\frac{\eta_f}{\sigma_f}) \end{aligned} \tag{3.13}$$

The total probability of error P_E is given by

$$P_E = \sum_{p_i \varepsilon P(0)} \Pr\{p_i\} P(\frac{\eta_f}{\sigma_f}) + \sum_{p_i \varepsilon P(1)} \Pr\{p_i\} P(-\frac{\eta_f}{\sigma_f}) \tag{3.14}$$

where $P(0) = \{p_i | F=0\}$ and $P(1) = \{p_i | F=1\}$. For equally likely input combinations

$$P_E = \frac{1}{2^n}\left(\sum_{p_i \varepsilon P(0)} P(\frac{\eta_f}{\sigma_f}) + \sum_{p_i \varepsilon P(1)} P(-\frac{\eta_f}{\sigma_f}) \right) \tag{3.15}$$

Thus, when the means and variances of the weights and inputs are known, the probability of error for each input combination may be calculated. The total probability of error may be calculated if the probability of occurrence of the input combination is known.

CHAPTER IV

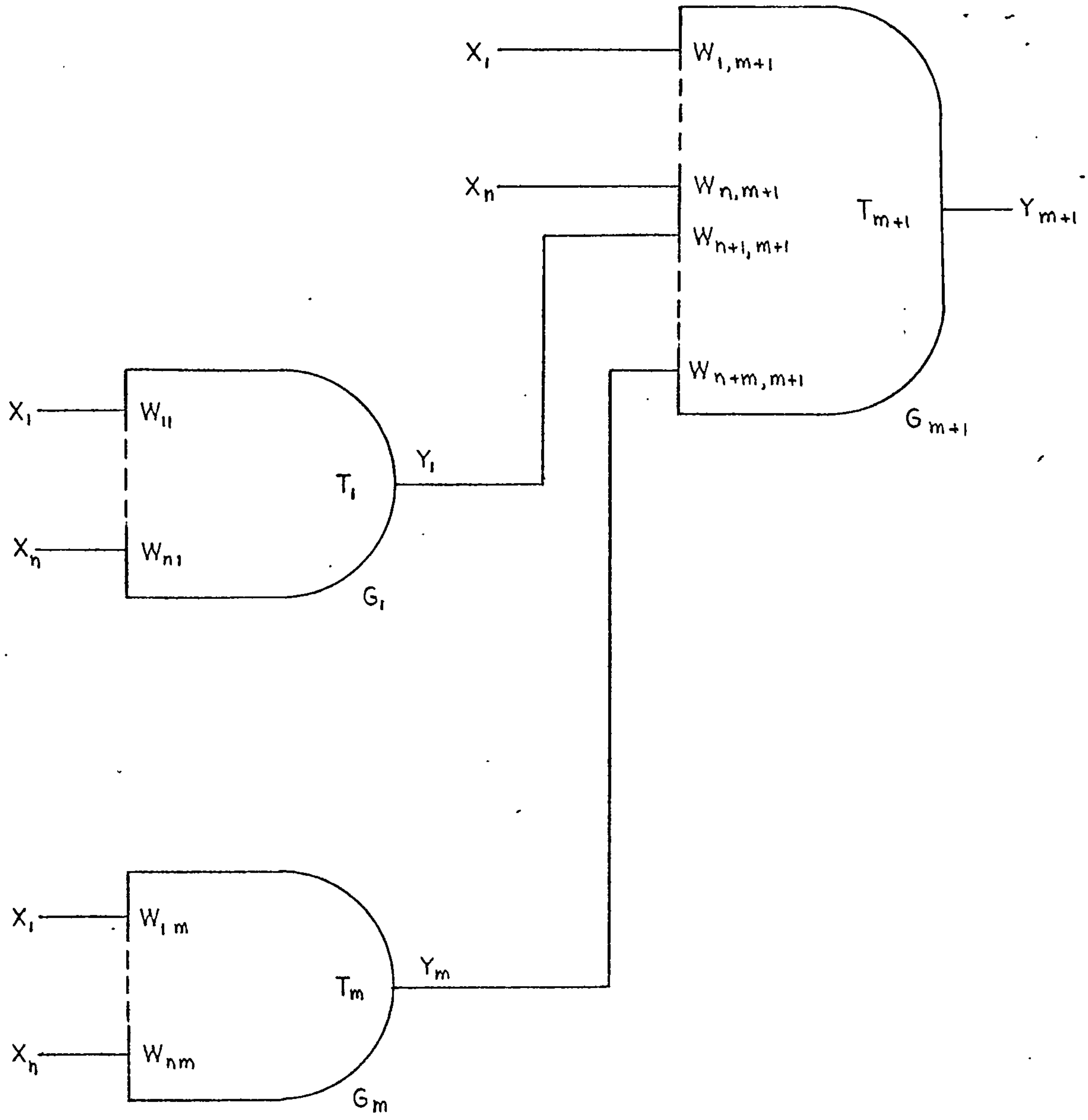
PROBABILITY OF ERROR OF THRESHOLD GATE NETWORKS

An error in a threshold gate network is caused by the unreliability of the gates that comprise the network. In this chapter, nonsequential, single output networks are analyzed and a procedure for finding the probability of error is derived.

A network of logic gates can be subdivided into levels. The first level contains gates whose inputs are network inputs and not outputs of any gate in the network. Since the network is nonsequential, the n-th level contains gates whose inputs are network inputs, or outputs of any gates in any lower level. The output level is a gate whose inputs are network inputs, or outputs of any other gate in the network.

A general two-level threshold network consisting of m gates with n inputs is shown in Fig. 4.1. Note that in general every network input goes to every gate in the network. If a certain gate G_j does not require a network input x_i, the weight w_{ij} is set equal to zero.

Let $G_1,\ldots,G_k,\ldots,G_m$ be threshold gates with weights $\{W_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$ and thresholds $T_1,\ldots,T_m$ such



<u>Fig. 4.1.</u> Two Level Threshold Gate Network

that the separating function $f_k(p)$ of the k-th gate is

$$f_k(p) = \sum_{i=1}^{n} w_{ik}x_i - T_k \tag{4.1}$$

Let G_{m+1} be a threshold gate with weights $w_{1,m+1},\ldots,$ $w_{m+n,m+1}$ and threshold T_m such that its separating function $f_{m+1}(p)$ is

$$f_{m+1}(p) = \sum_{i=1}^{n} w_{i,m+1}x_i + \sum_{j=1}^{m} w_{n+j,m+1}y_j - T_{m+1} \tag{4.2}$$

Let $x'_i,\ldots,x'_m$ be the inputs to G_{m+1} such that

$$x'_i = x_i \qquad 1 \leq i \leq n$$

$$x'_{n+j} = y_j \qquad 1 \leq j \leq m$$

Eqn. (4.2) now becomes

$$f_{m+1}(0) = \sum_{i=1}^{n+m} w_{i,m+1}x'_i - T_{m+1} \tag{4.3}$$

The probability of error of the network is the probability of error of gate G_{m+1} for a given combination of its inputs, mulitplied by the probability of occurrence of that input combination. The probability of occurrence of the inputs to G_{m+1} depends upon the probability of occurrence of the inputs to the network and the probability of occurrence of the outputs of gates $G_1, \ldots, G_m$.

The probability of occurrence of the output y_k of gate G_k for a given input combination $\underline{x}_j$ is

$$\begin{aligned} Pr\ \{y_k=0 \mid \underline{x}=\underline{x}_j\} &= Pr\ \{f_k(p)<0 \mid \underline{x}=\underline{x}_j\} \\ Pr\ \{y_k=1 \mid \underline{x}=\underline{x}_j\} &= Pr\ \{f_k(p)\geq 0 \mid \underline{x}=\underline{x}_j\} \end{aligned} \tag{4.4}$$

Thus, the probability of occurrence of the output is

$$\begin{aligned} Pr\ \{y_k=0\} &= Pr\ \{f_k(p)<0 \mid \underline{x}=\underline{x}_j\}\ Pr\ \{\underline{x}=\underline{x}_j\} \\ Pr\ \{y_k=1\} &= Pr\ \{f_k(p)\geq 0 \mid \underline{x}=\underline{x}_j\}\ Pr\ \{\underline{x}=\underline{x}_j\} \end{aligned} \tag{4.5}$$

where $Pr\ \{\underline{x}=\underline{x}_j\}$ is the probability of occurrence of the input combination $\underline{x}_j$. For any input combination $\underline{x}_j$, the distribution of $f_k(p)$ may be found from the statistics of the weights and inputs. Hence, $Pr\ \{y_k=0 \mid \underline{x}=\underline{x}_j\}$ and $Pr\ \{y_k=1 \mid \underline{x}=\underline{x}_j\}$ may be calculated without regard to the function to be realized by the network.

The probability of error of gate G_{m+1} for a given combination of its inputs $\underline{x}'^T_i = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ is

$$P_E = Pr\ \{\text{error} \mid \underline{x}'=\underline{x}'_i\}\ Pr\ \{\underline{x}'=\underline{x}'_i\} \tag{4.6}$$

For each combination of network inputs $\underline{x}_j$, there are 2^m possible combinations of the outputs $y_1,\ldots,y_m$. The probability of occurrence of a combination of inputs to G_{m+1} is

$$\Pr\{\underline{x}' = \underline{x}'_i\} = \Pr\{\underline{x}' = \underline{x}'_i \mid \underline{x} = \underline{x}_j\} \Pr\{\underline{x} = \underline{x}_j\} \tag{4.7}$$

Therefore the probability of error is

$$P_E = \sum_{j=1}^{2^n} \Pr\{\text{error} \mid \underline{x}' = \underline{x}'_i\} \Pr\{\underline{x}' = \underline{x}'_i \mid \underline{x} = \underline{x}_j\} \Pr\{\underline{x}=\underline{x}_j\} \tag{4.8}$$

where

$\Pr\{\text{error} \mid \underline{x}'=\underline{x}'_i\} \Pr\{\underline{x}'=\underline{x}'_i|\underline{x}=\underline{x}_j\}$

$$= \Pr\{f_{m+1}(\underline{x}')\geq 0|\underline{x}'=\underline{x}'_i\} \Pr\{\underline{x}'=\underline{x}'_i|\underline{x}=\underline{x}_j\} \text{ if } F(\underline{x}_j) = 0$$

or

$$= \Pr\{f_{m+1}(\underline{x}')<0|\underline{x}'=\underline{x}'_i\} \Pr\{\underline{x}'=\underline{x}'_i|\underline{x}=\underline{x}_j\} \text{ if } F(\underline{x}_j) = 1 \tag{4.9}$$

If $\Pr\{\underline{x}'=\underline{x}'_i|\underline{x}=\underline{x}_j\}$ is known, the probability of error can be calculated as shown in Chapter III.

For each network input combination $\underline{x}_j$, there are 2^m combinations of $\underline{x}_i$ corresponding to the 2^m combinations of the outputs of gates $G_1,\ldots,G_m$. Let

$$P_b = \Pr\{\underline{x}'=\underline{x}'_i|\underline{x}=\underline{x}_j\} \tag{4.10}$$

where b is a binary number $b_m b_{m-1} \ldots b_1$ such that b_i equals y_i given $\underline{x}$ equals $\underline{x}_j$. For m gate outputs, b lies in the range $0 \leq b \leq 2^m-1$. The 2^m probability P_b must now be found.

Although Pr $\{y_k = 0\}$ and Pr $\{y_k = 1\}$ can be found for each first level gate G_k,

$$P_b = \prod_{k=1}^{n} \Pr \{y_k = b_k\} \tag{4.11}$$

only if no input goes to more than one first level gate and the inputs and weights are all uncorrelated, thus making all the first level gate outputs independent.

In order to find the probability P_b, 2^m independent equations relating the P_b's can be found. Since all 2^m possible combinations of b are mutually exclusive and one b must occur,

$$\sum_{b=0}^{2^m-1} P_b = 1 \tag{4.12}$$

Each of the Pr $\{f_k(p) \geq 0\}$ yields an equation

$$\Pr \{f_k(0) \geq 0\} = \Pr \{y_k = 1\} = \sum_{b=1}^{2^m-1} b_k P_b \tag{4.13}$$

for a total of m equations. Now consider all possible products of 2 of the m separating functions $f_1 f_2, f_1 f_3, \ldots,$

$f_i f_j, \ldots, f_{m-1} f_m$. There are $\binom{m}{2}$* such products. Similarly, there are $\binom{m}{n}$ products $\Pi(b)$ of n of the m separating functions, where

$$\Pi(b) = \prod_{i=1}^{m} (b_i(f_i - 1) + 1) \tag{4.14}$$

For example, if there are 5 gates in the first level, $\Pi(22) = \Pi(10110) = f_5 f_3 f_2$. Considering all possible combinations, there are $2^m - m - 1$ possible products of 2 or more of the m separating functions. Therefore, there are $2^m - m - 1$ probabilities

$$P_o(b) = \Pr\{\Pi(b_i) \geq 0\} \tag{4.15}$$

Note that $\Pi(b)$ is positive only if an even number r of the n separating functions in the product are negative, that is, when r of the n outputs are equal to zero. Thus, these probabilities are related to the P_b's. In Eqn. 4.12, 4.13, and 4.14, there are 2^m equations in 2^m unknowns P_b. If this set of equations is linearly independent, we may solve for P_b.

---

* $\binom{m}{n} = \frac{m!}{n!(m-n)!}$ are the number of combinations of m objects taken n at a time, also known as binomial coefficients.

Three cases will now be considered. For m equal to two, there are four unknowns $p_{00}, p_{01}, p_{10}, p_{11}$ which are related by the following set of linear equations.

$$
\begin{aligned}
p_{00} + p_{01} + p_{10} + p_{11} &= 1 &&= P_o(0) \\
p_{01} \qquad\quad + p_{11} &= \Pr\{f_1 \geq 0\} &&= P_o(1) \\
p_{10} + p_{11} &= \Pr\{f_2 \geq 0\} &&= P_o(2) \\
p_{00} \qquad\qquad\quad + p_{11} &= \Pr\{f_1 f_2 \geq 0\} &&= P_o(3)
\end{aligned}
$$

or in matrix notation:

$$
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{bmatrix}
=
\begin{bmatrix} P_o(0) \\ P_o(1) \\ P_o(2) \\ P_o(3) \end{bmatrix}
$$

$$B_2\underline{P} = \underline{P}_0 \tag{4.16}$$

For m equal to one, there are two unknowns, p_0 and p_1, which are related by the following set of linear equations:

$$p_o + p_1 = 1 \qquad = P_o(0)$$

$$p_1 = \Pr\{f_1 \geq 0\} = P_o(1)$$

or

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_o \\ p_1 \end{bmatrix} = \begin{bmatrix} P_o(0) \\ P_o(1) \end{bmatrix}$$

$$B_1\underline{P} = \underline{P}_0$$

For m equal to three, the equation may be written:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{000} \\ p_{001} \\ p_{010} \\ p_{011} \\ p_{100} \\ p_{101} \\ p_{110} \\ p_{111} \end{bmatrix} = \begin{bmatrix} P_o(0) \\ P_o(1) \\ P_o(2) \\ P_o(3) \\ P_o(4) \\ P_o(5) \\ P_o(6) \\ P_o(7) \end{bmatrix}$$

$$B_3\underline{P} = \underline{P}_0$$

The set of 2^m equations has a unique solution if B_m is non-singular.

Notice that for each case the matrix B_m may be formed from a matrix H_{m-1} such that

$$B_m = \begin{bmatrix} H_{m-1} & H_{m-1} \\ \\ \overline{H}_{m-1} & H_{m-1} \end{bmatrix} \tag{4.17}$$

where $\overline{H}_{m-1}$ is formed from H_{m-1} by logically complementing each element. Thus,

$$H_0 = 1 \qquad H_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = B_1 \qquad H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} = B_2.$$

This reproducing property has been verified for m less than or equal to four.

The B_m matrices are related to the Hadamard matrix H'_q where $q = 2^m$. A Hadamard matrix H'_q is a q x q orthogonal matrix whose elements are the real numbers +1 and -1. It is evident that

$$H'_2 = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

It has been proved that if H'_q is a q x q Hadamard matrix then,

$$H'_{2q} = \begin{bmatrix} H'_q & H'_q \\ -H'_q & H'_q \end{bmatrix} \tag{4.18}$$

is a 2q x 2q Hadamard matrix (Peterson). The existence of Hadamard matrices has been proved for $q = 2^k$ where k is an integer. The matrix B_m is related to the Hadamard matrix H'_q by

$$B_m = \frac{1}{2}[H'_q + U_q] \tag{4.19}$$

where U_q is a q x q matrix with every element equal to 1. Solving Eqn. 4.18 for H'_q results in

$$H'_q = 2B_m - U_q \tag{4.20}$$

Since the first row of the B_m matrix always has all elements equal to one, subtracting the U_q matrix from $2B_m$ is equivalent to subtracting the first equation from all the other equations multiplied by a constant. The resulting set of linear equations is independent, being related by the nonsingular matrix H'_q. H'_q is orthogonal, hence, $H'^{-1}_q = H'^{T}_q$. Thus the original set of equations related by B_m are independent, therefore, B_m is nonsingular.

Since Hadamard matrices are orthogonal,

$$H'_q H'^T_q = q^{-1} I_q \tag{4.21}$$

where I_q is the q x q identify matrix and q^{-1} is a normalizing factor. Substituting Eqn. 4.19 into Eqn. 4.20 results in

$$I_q = q^{-1}[2B_m - U_q]\ [2B_m^T - U_q^T]$$

$$I_q = q^{-1}[4B_m B_m^T - 2B_m U_q - 2U_q B^T_m + U_q^2]$$

Note that $U_q^T = U_q$. Premultiplying the above equation by B_m^{-1} yields:

$$B_m^{-1} = q^{-1}[4B_m^T - 2U_q] - q^{-1}B^{-1}[2U_q B_m^T - U_q^2]$$

Note that

$$U_q^2 = U_q U_q = qU_q.$$

Notice that the first column of B_m^T has q elements equal to one and that all other columns have q/2 elements equal to one. Hence,

$$2U_q B_m^T = q \begin{bmatrix} 2 & 1 \ldots\ldots & 1 \\ 2 & 1 & 1 \\ \vdots & & \vdots \\ 2 & 1 \ldots\ldots & 1 \end{bmatrix}$$

Therefore,

$$2U_qB_m^T - U^2{}_q = q\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & & 0 \\ \vdots & & & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

Since $B_m^{-1}B_m = I_q$ and since all elements in the last column of B_m are equal to one,

$$B^{-1}\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

and therefore

$$A_q = B_m^{-1}\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & & 0 \\ \vdots & & & \\ 1 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 0 & & & 0 \\ \vdots & & & \\ 0 & 0 & & 0 \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

Finally,

$$B_m^{-1} = \frac{1}{2^{m-2}}\ [B_m^T - \frac{1}{2}\ U] - A_q \qquad (4.22)$$

Thus, B_m^{-1} may be easily calculated from B_m, even when m is large.

The $P_o(b)$'s must now be found. Eqn. 4.15 states

$$P_o(b) = \Pr\{\Pi(b) \geq 0\}$$

$P_o(b)$ may be calculated if the distribution of $\Pi(b)$ is known. $\Pi(b)$ is a product of separating functions of the form

$$f_i(p) = \underline{\eta}_{x_i}^T \underline{\eta}_{w_i} + \underline{\eta}_{x_i}^T \underline{w}'_i + \underline{x}'^T_i \underline{\eta}_{w_i} + \underline{x}'^T_i \underline{w}'_i$$

For $\sigma^2_{w_i} << \eta_{w_i}$ the separating function may be written as

$$f_i(p) \simeq \underline{\eta}_{x_i}^T \underline{\eta}_{w_i} + \underline{\eta}_{x_i}^T \underline{w}'_i + \underline{x}'^T_i \underline{\eta}_{w_i}$$

Consider the product of two separating functions f_1 and f_2:

$$\begin{aligned}
\Pi(b) = f_1 f_2 \simeq{} & \eta_1 \eta_2 + \eta_1 \underline{\eta}_{x_2}^T \underline{w}'_2 + \eta_1 \underline{x}'^T_2 \underline{\eta}_{w_2} \\
& + \underline{\eta}_{x_1}^T \underline{w}'_1 \eta_2 + \underline{\eta}_{x_1}^T \underline{w}'_1 \underline{\eta}_{x_2}^T \underline{w}'_2 + \underline{\eta}_{x_1}^T \underline{w}'_1 \underline{x}'^T_2 \underline{\eta}'_{w_2} \\
& + \underline{x}'^T_1 \underline{\eta}_{w_1} \eta_2 + \underline{x}'^T_1 \underline{\eta}_{w_1} \underline{\eta}_{x_2}^T \underline{w}'_2 + \underline{x}'^T_1 \underline{\eta}_{w_1} \underline{x}'^T_2 \underline{\eta}'_{w_2} \qquad (4.23)
\end{aligned}$$

where $\eta_1 = \underline{\eta}_{x_1}^T \underline{\eta}_{w_1}$ and $\eta_2 = \underline{\eta}_{x_2}^T \underline{\eta}_{w_2}$. For $\sigma^2_{w_1}$ and $\sigma^2_{w_2}$ small compared to every component of $\underline{\eta}_{w_1}$ and $\underline{\eta}_{w_2}$, Eqn. 4.23 becomes

$$\Pi(b) = f_1 f_2 \simeq \eta_1 \eta_2 + \eta_1 \underline{\eta}_{x_2}^T \underline{w}'_2 + \eta_1 \underline{x}'^T_2 \underline{\eta}_{w_2} + \underline{\eta}_{x_1}^T \underline{w}'_1 \eta_2 + \underline{x}'^T_1 \underline{\eta}_{w_1} \eta_2$$

$$+ \underline{x}'^T_1 \underline{\eta}_{w_1} \underline{x}'^T_2 \underline{\eta}_{w_2} \tag{4.24}$$

With the exception of the last term, this is a sum of normal random variables. It is possible that under certain conditions this term is small compared with the other terms and can be neglected. The worst case occurs when all inputs are equal to zero. The separating function then becomes

$$f \simeq -w_{n+1} + \underline{x}'^T \underline{\eta}_{w_i} + w'_{n+1}$$

and the product $\Pi(b)$ becomes

$$\Pi(b) = f_1 f_2 \simeq T_1 T_2 - T_1 \underline{x}'^T_2 \underline{\eta}_{w_2} - T_2 \underline{x}'^T_1 \underline{\eta}_{w_1} - T_1 T'_2 - T_2 T'_1$$

$$+ \underline{x}'^T_1 \underline{\eta}_{w_1} \underline{x}'^T_2 \underline{\eta}_{w_2}$$

where T_1 and T_2 are the thresholds for G_1 and G_2, respectively. To be able to neglect the last term, the variance of $\underline{x}'^T_1 \underline{\eta}_{w_1}$ and $\underline{x}'^T_2 \underline{\eta}_{w_2}$ must be small compared to T_1 and T_2, respectively, that is,

$$\begin{aligned} \underline{\eta}_{w_1}^T M_{x_1} \underline{\eta}_{w_1}^T &<< T_1 \\ \underline{\eta}_{w_2}^T M_{x_2} \underline{\eta}_{w_2} &<< T_2 \end{aligned} \tag{4.25}$$

where M_{x_1} and M_{x_2} are the covariance matrices of $\underline{x}_1'$ and $\underline{x}_2$, respectively. The above conditions are met for small input variances $\sigma_{x_1}^2$ and $\sigma_{x_2}^2$. The product $\Pi(b)$ is approximately

$$\Pi(b) = f_1 f_2 \simeq \eta_1\eta_2 + \eta_1 \underline{\eta}_{x_2}^T \underline{w}'_2 + \eta_1 \underline{x}'^T_2 \underline{\eta}_{w_2} + \eta_2 \underline{\eta}_{x_1}^T \underline{w}'_1 + \eta_2 \underline{x}'^T_1 \underline{\eta}_{w_1}$$

$$\simeq \eta_1\eta_2 \left(1 + \frac{\underline{\eta}_{x_2}^T \underline{w}'_2}{\eta_2} + \frac{\underline{x}'^T_2 \underline{\eta}_{w_2}}{\eta_2} + \frac{\underline{\eta}_{x_1}^T \underline{w}'_1}{\eta_1} + \frac{\underline{x}'^T_1 \underline{\eta}_{w_1}}{\eta_1} \right)$$

In like manner, a product $\Pi(b)$ of n separating functions is approximately

$$\Pi(b) \simeq \Pi_\eta \left(1 + \sum_{i=1}^{m} b_i \cdot \frac{\underline{\eta}_{x_i}^T \underline{w}'_i}{\eta_i} + \sum_{i=1}^{m} b_i \cdot \frac{\underline{x}'^T_i \underline{\eta}_{w_i}}{\eta_i} \right) \tag{4.26}$$

where $\Pi_\eta = \prod_{i=1}^{m} (b_i(\eta_i - 1) + 1)$. Now $\Pi(b)$ is approximately a sum of jointly normal random variables. Therefore, it is reasonable to assume that $\Pi(b)$ is normally distributed. Since $E(\underline{w}_i)$ and $E(\underline{x}_i)$ are both zero for all $i=1,\ldots,m$,

$$E(\Pi(b)) = \prod_{i=1}^{m} (b_i\ (\eta_i - 1) + 1) \tag{4.27}$$

In general, every network input goes to every gate in the network. Therefore,

$$\underline{\eta}_{x_i} = \underline{\eta}_x \text{ and } \underline{x}'_i = \underline{x}'$$

for all values of i. Eqn. 4.26 now becomes

$$\Pi(b) = \Pi_\eta \left[1 + \underline{\eta}_x^{*T} \underline{w}^* + \underline{x}'^T \underline{\eta}^*_w \right] \tag{4.28}$$

where $\underline{\eta}_x^{*T} = \dfrac{b_1 \underline{\eta}_x^T}{\eta_1}, \dfrac{b_2 \underline{\eta}_x^T}{\eta_2}, \ldots, \dfrac{b_m \underline{\eta}_x^T}{\eta_m}$

$$\underline{w}^{*T} = \underline{w}'^T_1, \underline{w}'^T_2, \ldots, \underline{w}'^T_n$$

$$\underline{\eta}_w^{*T} = \sum_{i=1}^{m} \frac{b_i \underline{\eta}_{w_i}}{\eta_i}$$

Since $\underline{x}'$ and $\underline{w}^*$ are sets of jointly normal random variables, their covariance matrices M_x and M_w^* can be computed if the variances and correlations of the weights and inputs are known. The variance of $\Pi(b)$ is given by

$$\sigma^2(\Pi(b)) = (\eta_\Pi)^2 [\underline{\eta}_x^{*T} M_w^* \underline{\eta}_x^* + \underline{\eta}_w^{*T} M_x \underline{\eta}_w^*] \tag{4.29}$$

The distribution of $\Pi(b)$ is now known, hence $\Pr\{\Pi(b) \geq 0\}$ can be calculated. Thus, $\Pr\{\underline{x}' = \underline{x}'_i | \underline{x} = \underline{x}_j\}$ can be found for each input combination $\underline{x}_j$, and the probability of error can be computed.

For threshold gate networks with more than two levels, the probability of error can be calculated by successive applications of the techniques shown above. From the separating functions of the gates in the second level, the probability of occurrence of the inputs to the third level can be calculated. It is then possible to calculate the probability of error of the third level or the distribution of the products of the third level separating function and proceed to the next level. The process is repeated for the remaining higher levels in the network.

For each network input combination, any level n may be considered as a system which transforms the probability of occurrence of its n_i inputs into the probability of occurrence of the n_{i+1} inputs to the next level. By successive transformations, the probability of occurrence of the inputs to the highest level can be found and thus the probability of error of the network.

CHAPTER V

SIMULATION OF THRESHOLD GATE NETWORKS

A threshold gate can be simulated by generating random numbers for the inputs and weights, computing the separating function, comparing it to the threshold, and computing the probability of error. A network of threshold gates can then be simulated by simulating each gate in the network. If the separating function of the output gate of a network is compared to its threshold, the probability of error of the network can be computed. It is also possible to examine the density function of the separating function of a single gate and the density function of the products of the separating functions of several gates.

Generation of Correlated, Normally Distributed Random Numbers

The distribution of a sum of n independent random numbers approaches the normal distribution as n approaches infinity. In fact, if n independent, uniformly distributed random numbers are summed the distribution of the sum is approximately normal for $n \geq 3$. Let $\{x_1, x_2, \ldots, x_n\}$ be a set of n independent random numbers, uniformly distributed on the interval $\{0 \leq x_i \leq 1\}$. The mean of any x_i is equal to 1/2 and the variance is equal to 1/12.

When $n = 12$, the sum v

$$v = \sum_{i=1}^{12} x_i$$

is approximately normally distributed with mean

$$E(v) = \sum_{i=1}^{12} E(x_i) = 6$$

and variance

$$\sigma^2(v) = \sum_{i=1}^{12} \sigma^2(x_i) = 1$$

Thus,

$$y = \sum_{i=1}^{12} x_i - 6 \tag{5.1}$$

is approximately normally distributed with zero mean and unit variance.

The numbers x_i can be generated by a digital computer using a power residue method utilizing the following equation.

$$n_{i+1} = m\, n_i \bmod w$$

where $x_i = \frac{n_i}{w}$, m is a constant, and w is the largest integer that can be stored on a word. For a computer with a word length of k bits, the maximum integer that can be stored in a word is 2^k-1. The constant m is selected to give the longest possible sequence of numbers before repeating.

Note that $-6 \leq y \leq 6$. A polynomial correction can be applied to y in order to obtain a better approximation to the normal distribution. Thus if

$$y = \frac{1}{4}\left(\sum_{i=1}^{12} x_i - 6\right)$$

we can find the constants a_1, a_3, a_5, a_7, a_9 so that

$$z = a_1 y + a_3 y^3 + a_5 y^5 + a_7 y^7 + a_9 y^9$$

is a better approximation to the normal distribution than y. By successive application of the techniques described above, a sequence of independent, approximately normal random numbers can be generated.

Let $\underline{Z}^T = z_1, z_2, \ldots, z_n$ be a vector of independent random variables each normally distributed with zero mean and unit variance. The random variables are jointly normal with covariance matrix M_Z equal to the n x n identity matrix I_n. If $\underline{W}^T = w_1, w_2, \ldots w_n$ is normally distributed in n dimensions with zero mean and covariance matrix M_X, there exists a transformation such that

$$\underline{W} = Q^T \underline{Z}$$

and

$$M_w = Q^T M_z Q = Q^T Q$$

where Q is a nonsingular matrix (Miller). Thus, a correlated set of normal random numbers can be generated from an independent set of normal random numbers by a linear transformation. The matrix Q can be found by a Gauss elimination matrix inversion procedure which transforms M_w into the identity matrix such that

$$Q^{T-1} M_w Q^{-1} = I \tag{5.2}$$

Since Q is nonsingular, Q^{-1} always exists.

Simulation of a Threshold Gate

In a threshold gate, the weights $\underline{w}$ and inputs $\underline{x}$ are random variables which may reasonably be assumed to be correlated sets of jointly normal random variables. As shown above, these correlated sets can be generated from independent sets of normal random variables by a linear transformation. Let

$$\begin{aligned} \underline{x}' &= P^T \underline{y} \quad \text{and} \\ \underline{w}' &= Q^T \underline{z} \end{aligned} \tag{5.3}$$

where $\underline{y}$ and $\underline{z}$ are sets of independent random variables with zero mean and unit variance. The matrices P and Q are related to the covariance matrices of the inputs and weights, M_x and M_w, respectively.

In order to simulate a threshold gate the means, variances, and correlations of the inputs and weights must be known. From this information, the covariance matrices can be calculated. In this thesis, simulations were carried out for threshold gates for which the following conditions are satisfied:

(a) The standard deviations of the inputs are all equal to σ_x.

(b) The correlation coefficient ρ_x is the same for any two inputs x_i and x_j.

(c) The standard deviation of a weight w_i is equal to its mean η_{w_i} multiplied by a constant σ_w.

(d) The correlation coefficient ρ_w is the same for any two weights w_i and w_j.

The covariance matrices are given by

$$M_x = [\rho_{ij}\sigma_x^2] \qquad \rho_{ij} = \rho_x,\ i \neq j,\ \rho_{ii} = 1$$

$$M_w = [\rho_{ij}\eta_{w_i}\eta_{w_j}\sigma_w^2] \quad \rho_{ij} = \rho_w,\ i \neq j,\ \rho_{ii} = 1 \tag{5.5}$$

The transformation matrices are found from a Gauss elimination matrix inversion subroutine which calculates P and Q such that

$$Q^{T-1} M_x^{-1} Q^{-1} = I$$

and (5.4)

$$P^{T-1} M_w^{-1} P^{T} = I$$

The matrices P and Q can then be computed from P^{-1} and Q^{-1} by using the same subroutine. By generating the sets of independent normal random variables $\underline{y}$ and $\underline{z}$ and applying the transformation, $\underline{x}'$ and $\underline{w}'$ can be generated. Since the random variables in $\underline{y}$ and $\underline{z}$ have zero means, $\underline{x}'$ and $\underline{w}'$ have zero mean. The inputs and weights can be generated from $\underline{x}'$ and $\underline{w}'$ by adding the mean of the inputs and weights such that

$$x_i = \eta_{x_i} + x'_i$$

$$w_j = \eta_{w_j} + w'_j$$

The separating function of the threshold gate can be calculated from

$$f(p) = \sum_{i=1}^{n} w_i x_i - w_{n+1}$$

where n is the number of inputs.

The simulation procedure was performed as follows:

(1) Set up the covariance matrices M_x and M_w.

(2) Find the transformation matrices P and Q.

(3) Generate random numbers for $\underline{y}$ and $\underline{z}$.

(4) Compute $\underline{x}$ and $\underline{w}$.

(5) Compute f(p), compare to zero.

(6) Repeat steps 3 through 5 for N iterations.

The probability of error is given by

$$P_E = n_e/N$$

where n_e is the number of times the output of the gate was in error.

Simulation of Density Functions

The density function of the separating function can also be determined from the simulation. Consider the set of values of the separating function $\{f_i(p)\}$ to be a sample from the distribution of f(p). The sample mean η_s and variance σ^2_s can easily be calculated.

$$\begin{aligned} \eta_s &= \frac{1}{N} \sum_{i=1}^{n} f_i(p) \\ \sigma^2_s &= \frac{1}{N} \sum_{i=1}^{n} (f_i(p) - \eta_s)^2 \end{aligned} \tag{5.6}$$

An approximation f(k) of the density function can be obtained by counting the number N_k of $f_i(p)$ that lie in the interval $c\,k \le \frac{f_i(p)-\eta_s}{\sigma_s} < c(k+1)$

$$f(k) = \frac{N_k}{N}$$

where k = 0, ±1, ±2, ... and c is a constant. The product of the separating functions of several gates can be studied by replacing $f_i(p)$ by the product.

The simulation program described in Appendix A calculates an approximation of the density function of the separating function. An approximation n(k) of a normal density function $N(\eta,\sigma)$ with mean η and variance σ^2 equal to η_s and $\sigma^2{}_s$, respectively, is also calculated, using the same interval as for $f_i(p)$. Both densities are plotted versus c_k for c = 0.2 and k = 0, ±1,..., ±25. The mean square error d is then computed

$$d = \sum_{k=-25}^{25} (n(k) - f(k))^2 \tag{5.7}$$

Table 5.1 shows the result of a simulation study of the separating functions of several threshold gates. The values of the mean and variance are calculated using the techniques developed in Chapters III and IV including the approximation

TABLE 5.1. Results of Simulation Study

Function	Input and Weight Variances	Input and Weight Correlations	Calculated Mean	Simulated Mean	Calculated Variance	Simulated Variance	Mean Square Error with Respect to Normal Density
$f=x_1+x_2+x_3+x_4-3.5$							
$\underline{x}=1,1,1,1$	.01	.0	0.500	0.502	0.202	0.198	$1.1\mathrm{x}10^{-4}$
$f=x_1+x_2+x_3-1.5$							
$\underline{x}=1,1,1$	.01	.9	1.500	1.497	0.1095	0.1097	$1.9\mathrm{x}10^{-4}$
$\underline{x}=1,1,0$	.01	.9	0.500	0.498	0.0905	0.0902	$0.8\mathrm{x}10^{-4}$
$f=f_1f_2$, where							
$f_1=x_1+x_2-1.5$							
$f_2=x_3+x_4-1.5$							
$\underline{x}=1,1,1,1$	.01	.0	0.250	0.252	$2.52\mathrm{x}10^{-2}$	$2.79\mathrm{x}10^{-2}$	$3.3\mathrm{x}10^{-3}$
$\underline{x}=1,1,1,1$	.01	.9	0.250	0.296	$7.61\mathrm{x}10^{-2}$	$5.76\mathrm{x}10^{-2}$	$1.2\mathrm{x}10^{-2}$
$\underline{x}=0,0,0,0$	.0001	.0	2.250	2.249	$1.35\mathrm{x}10^{-3}$	$1.38\mathrm{x}10^{-3}$	$1.1\mathrm{x}10^{-4}$
$\underline{x}=0,1,0,0$	.0001	.0	0.750	0.749	$9.75\mathrm{x}10^{-4}$	$9.63\mathrm{x}10^{-4}$	$1.7\mathrm{x}10^{-4}$
$\underline{x}=0,1,0,1$	.0001	.0	0.250	0.250	$2.00\mathrm{x}10^{-4}$	$1.95\mathrm{x}10^{-4}$	$1.2\mathrm{x}10^{-4}$
$\underline{x}=1,1,0,0$	.0001	.0	-0.750	-0.749	$1.20\mathrm{x}10^{-3}$	$1.20\mathrm{x}10^{-3}$	$0.7\mathrm{x}10^{-4}$
$\underline{x}=1,1,0,1$	.0001	.0	-0.250	-0.250	$2.25\mathrm{x}10^{-4}$	$2.22\mathrm{x}10^{-4}$	$1.2\mathrm{x}10^{-4}$
$\underline{x}=1,1,1,1$	.0001	.0	0.250	0.252	$2.50\mathrm{x}10^{-4}$	$2.52\mathrm{x}10^{-4}$	$1.1\mathrm{x}10^{-4}$
$f=f_1f_2f_3$ where							
f_1 & f_2 are as above							
and $f_3=x_5+x_6-1.5$							
$\underline{x}=0,0,0,0,0,0$	.0001	.0	-3.375	-3.374	$4.56\mathrm{x}10^{-3}$	$4.60\mathrm{x}10^{-3}$	$0.8\mathrm{x}10^{-4}$

of Eqn. 3.10 and Eqn. 4.26. The simulation means and variances are very close to the calculated values, indicating that the approximations can be made with little error.

A simulated density function is plotted in Figure 5.1. for $y = \left\langle x_1+x_2+x_3 \right\rangle_{1.5}$ with $\sigma^2_x = \sigma^2_w = 0.01$ and $\rho_x = \rho_w = 0.9$ for the inputs $\underline{\eta}_x^T = \{1,1,1\}$.

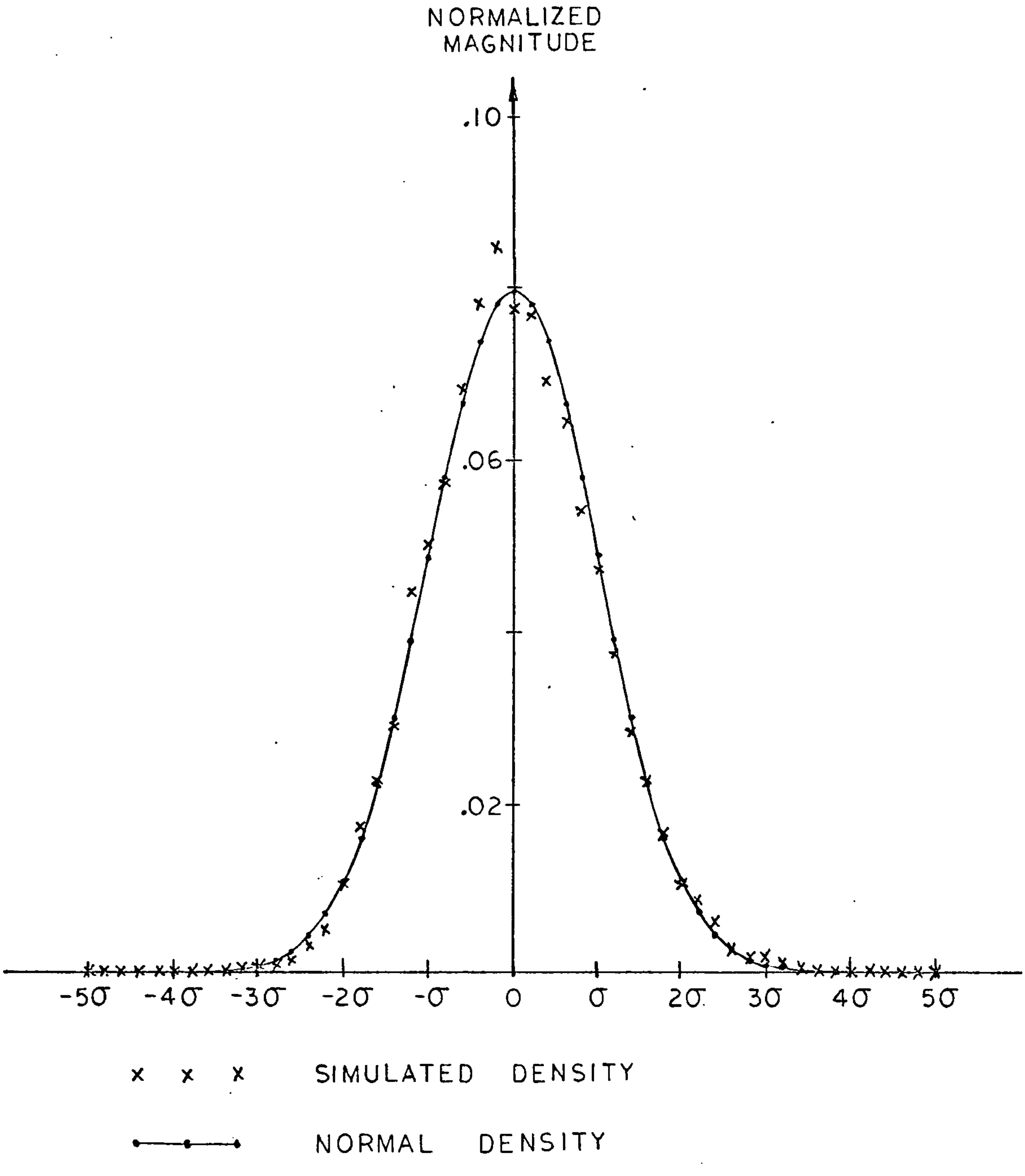


Fig. 5.1. Simulated Density Function Compared with Normal Density Function

CHAPTER VI

MINIMIZATION OF PROBABILITY OF ERROR

In the threshold gate realizations of a given logic function F, the weights are not unique. If the threshold is fixed, there are many combinations of the other weights that will realize F. By adjusting weights, it is possible to find a realization of F that minimizes the probability of error of the gate for a particular circuit. Minimization by adjusting weights can be implemented by a multi-dimensional search technique called "pattern search" developed by Hooke and Jeeves.

The pattern search is based on the premise that a set of parameter adjustments which has proved successful in minimizing a performance index will be worth trying again. The procedure is adaptive in the sense that prior success determines the next adjustment. The search begins at an initial base point $\underline{W}(0)$, and small adjustments are made from this point with repeated movement in the direction of improvement until improvement ceases. At that point, a search for a new direction of improvement is conducted.

Two types of parameter adjustments are made by the pattern search, the exploratory move and the pattern move. The exploratory move establishes the direction of improvement from a base point of the performance index. No attempt is made to estimate the gradients. The result of the exploratory move is a pattern of improvement. The pattern move utilizes the information gained in the exploratory move. The adjustments which proved successful in the exploratory move are repeated in the pattern move. If the performance index is decreased, the pattern move is repeated. When the pattern move is no longer successful, a new base point is established.

The exploratory move is carried out as follows. A single coordinate is increased or decreased by a predetermined step size to determine which change, if any, will produce a decrease in the performance index. If there is an improvement, the change is included in the pattern. When all coordinates have been examined, a pattern is established. If no change produces an improvement, the step size is reduced and the exploratory move is repeated until the step size is reduced below a predetermined minimum at which point the search is terminated.

If a pattern move fails to produce an improvement, the coordinates are restored to their values before the last pattern move and a new base point is established. From this new base point, a new pattern is established and a new series of pattern moves is started. The search continues in this way until terminated.

An example of a two-dimensional pattern search is shown in Figure 6.1. In this example, P_1, P_2, P_3, P_4, and P_5 are contours of the performance index with $P_1 > P_2 > P_3 > P_4 > P_5$.

The search begins at an initial base point $\underline{B}_0$, where w_1 and w_2 are equal to $w_1(0)$ and $w_2(0)$, respectively. The initial step size s_0 is chosen either arbitrarily or from some a priori knowledge of the performance index. A pattern move vector $\underline{I}_0 = \{i_1, i_2\}$ is now established. The first variable w_1 is incremented by s_0, that is, $w_1(1)$ equals $w_1(0)+s_0$. If there is a decrease in the performance index, the increment i_1 is set equal to s_0. If there is no decrease, $w_1(0)-s_0$ is tried. If this point produces a decrease in the performance index, i_1 is set equal to $-s_0$. If neither move produced a decrease, i_1 equals zero. In like manner, an increment i_2 is found for w_2. The pattern move is now the vector $\underline{I}$.

From the point $\underline{B}_0$, the pattern move is made n times until there is no further decrease in the performance index, that is,

$$\underline{B}_1 = \underline{B}_0 + n\underline{I}_0$$

where $\underline{B}_1$ is a new base point. Note that the point $\underline{B}_1 + \underline{I}_0$ has a greater value of the performance index than the point $\underline{B}_1$. From this new base point, a new pattern $\underline{I}_1$ is established. The pattern is used until a new base point $\underline{B}_2$ is found.

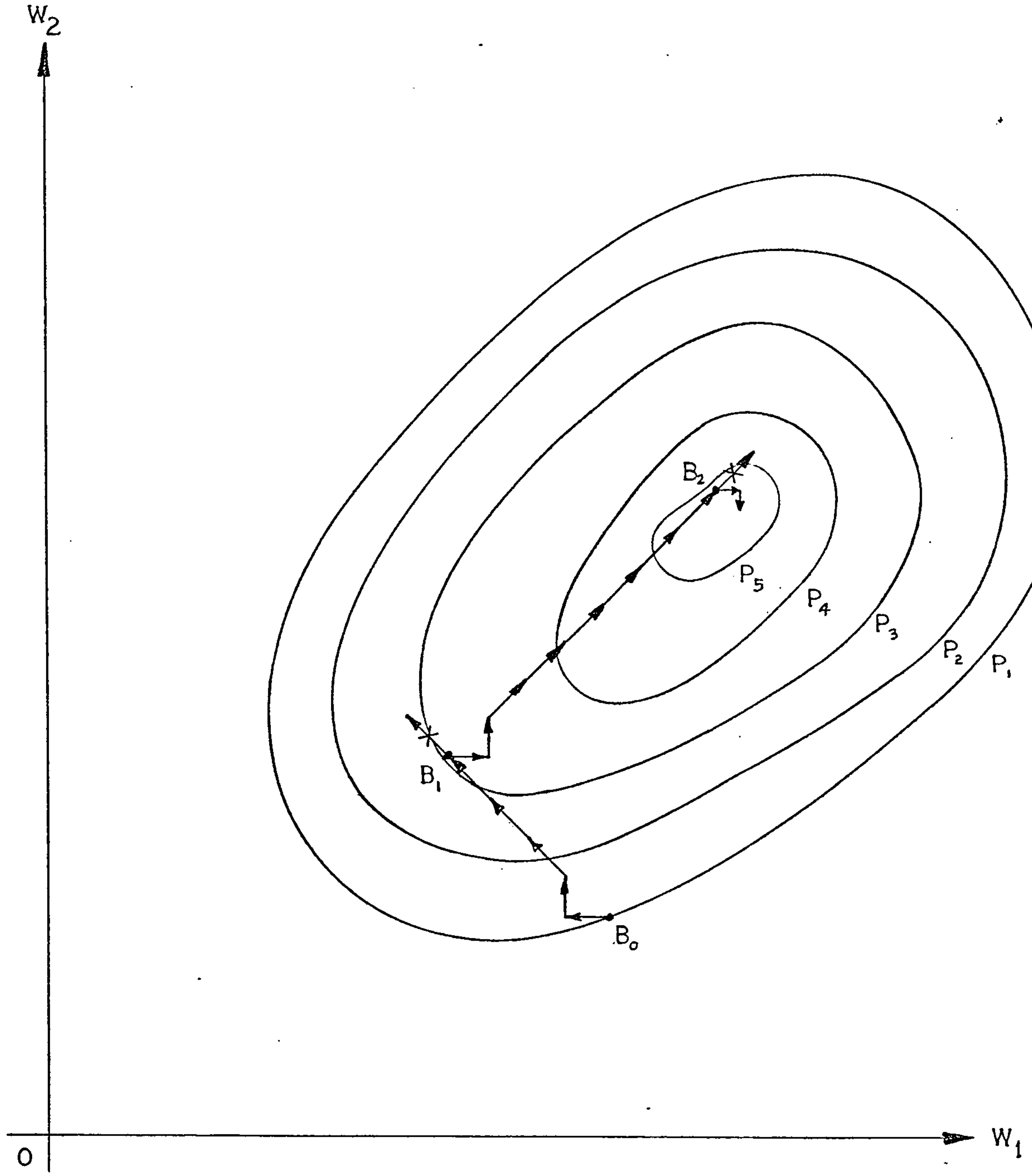


Fig. 6.1. Example of a Pattern Search

If a pattern $\underline{I}_n$ is found such that every component of $\underline{I}_n$ is equal to zero, then the step size s_0 is reduced to new step size s_1 and an exploratory move is made. The search continues using successively smaller step sizes as needed until the step size is reduced below a predetermined minimum at which time the search is terminated.

In this thesis, the pattern search is used to minimize the probability of error of a threshold gate. The mean values of the weights $\eta_{w_1},\ldots,\eta_{w_n}$ are adapted to find the combination $\underline{\eta}_w^*$ that minimizes the probability of error for a gate with a given set of statistics. As discussed in Chapter II, the standard deviation of a weight σ_{w_i} is set equal to the mean value of the weight η_{w_i} multiplied by a constant σ_w

$$\sigma_{w_i} = \sigma_w \eta_{w_i} \tag{6.1}$$

<u>Theorem 6.1.</u> If in a given threshold gate realization $y = \langle f(p) \rangle_0$ of a logic function F, the standard deviation of a weight σ_{w_i} is proportional to the mean of the weight η_{w_i}, and every weight including the threshold is multiplied by a positive constant k, the resulting realization $y = \langle k\ f(p) \rangle_0$ has the same probability of error as $y = \langle f(p) \rangle_0$.

<u>Proof.</u> Let

$$f(p) = \sum_{i=1}^{n+1} x_i w_i$$

therefore

$$k\ f(p) = \sum_{i=1}^{n+1} x_i k w_i$$

Note that

$$\eta_{kf} = k\eta_f \qquad (6.2)$$

The variance of k f(p) is given by Eqn. 3.11

$$\sigma_{kf}^2 = \underline{\eta}_x^T\ M'_w \underline{\eta}_x + k\eta_w^T\ M_x k\eta_w^T$$

where M'_w is the covariance matrix of $k\underline{w}$. Note that the elements of the covariance matrix are given by

$$M_{w_{ij}} = \sigma_{w_i}\sigma_{w_j}\rho_{ij}$$

Therefore,

$$M'_{w_{ij}} = k^2\sigma_{w_i}\sigma_{w_j}\rho_{ij}$$

hence

$$M'_w = k^2 M_w$$

The variance is

$$\sigma_{kf}^{2} = k^{2}[\underline{n}_{x}^{T}M_{w}\underline{n}_{x} + \underline{n}_{w}M_{x}\underline{n}_{w}] = k^{2}\sigma^{2}_{kf}$$

therefore

$$\sigma_{kf} = k\sigma_{f} \tag{6.3}$$

The probability of error given by Eqn. 3.15 becomes

$$P'_{E} = \frac{1}{2^{n}}\left[\sum_{p_i \varepsilon P(0)} P\left(\frac{\eta_{kf}}{\sigma_{kf}}\right) + \sum_{p_i \varepsilon P(1)} P\left(-\frac{\eta_{kf}}{\sigma_{kf}}\right)\right]$$

or since

$$\frac{\eta_{kf}}{\sigma_{kf}} = \frac{\eta_{f}}{\sigma_{f}}$$

the probability of error is the same for both realizations. Therefore, in the adaption techniques developed in this thesis, the mean of the threshold can be held constant without any loss of generality.

Using the techniques developed in Chapters III and IV, digital computer programs for calculation of the probability of error of a single gate and of a network have been written. The single gate program is described in Appendix B and the network program in Appendix C. These programs are used as subroutines to a pattern search program for minimizing probability of error.

CHAPTER VII

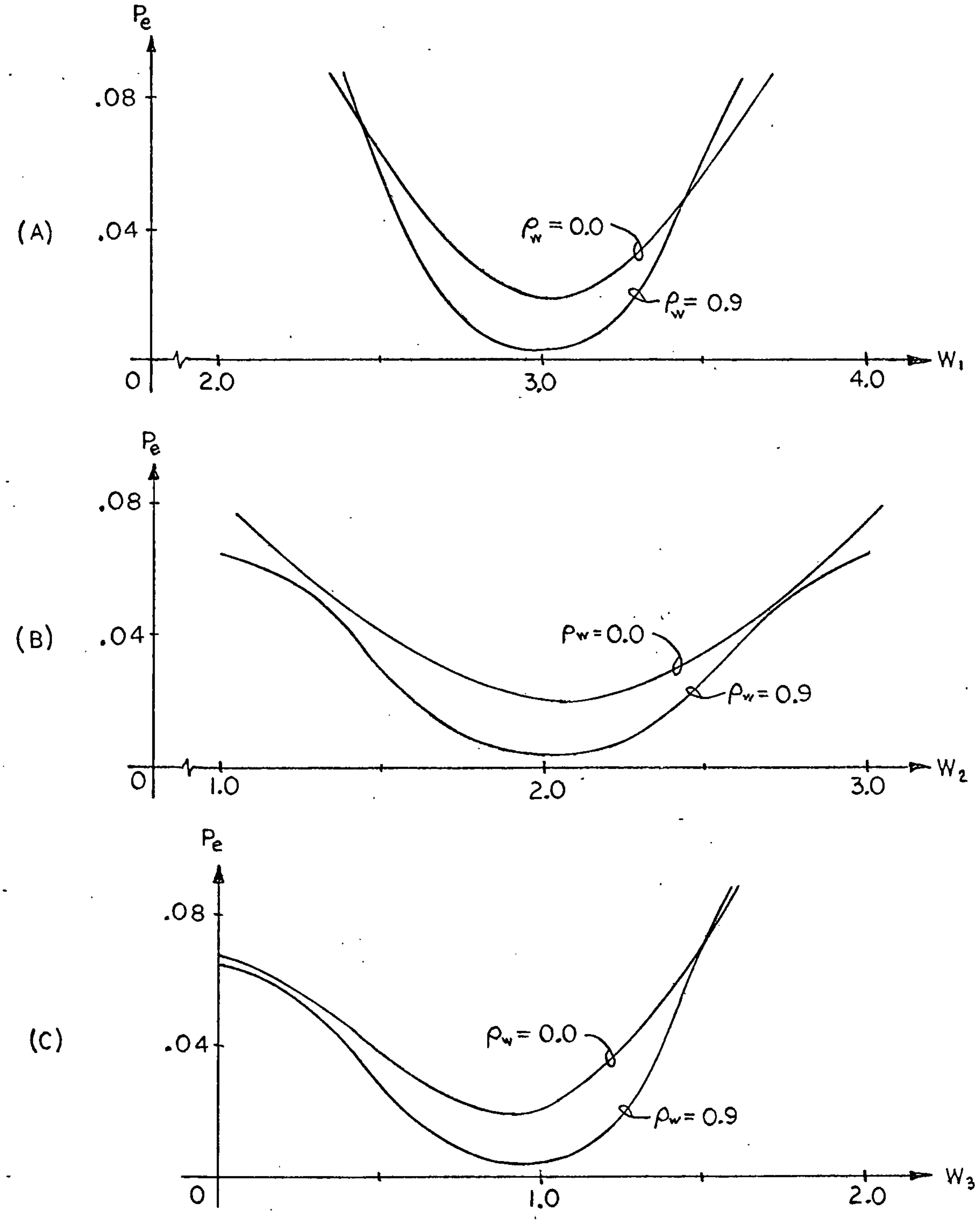
RESULTS AND CONCLUSIONS

The Probability of Error Surface

The probability of error of a threshold gate is a function of the statistics of the inputs and weights of the circuits which will implement the gate. For a given set of variances and correlations of the inputs and weights, the probability of error depends on the mean value of the weights and inputs and the probability of occurrence of the inputs. In the pattern search used in this thesis, the means of the inputs are constants, $\eta_x(0)$ and $\eta_x(1)$, equal to zero and one, respectively, and the input combinations are equally likely. The probability of error is a function only of the mean of the weights for constant input variances and input and weight correlations. It is desirable to examine this function to obtain some information as to the feasibility of the pattern search. Because it is difficult to display directly a function of more than two variables, the probability of error is evaluated by varying one weight of a realization while holding the others at their optimal values. The probability of error of the function

$$y = x_1x_2 + x_1x_3x_4$$

is plotted versus w_1 in Fig. 7.1a, versus w_2 in Fig. 7.1b,



<u>Fig 7.1.</u> Contours of Probability of Error of $y = x_1x_2 + x_1x_3x_4$

and versus w_3 in Fig. 7.1c, for $\sigma_x = \sigma_x = 0.0025$ and various correlation coefficients. Because the function is symmetric with respect to the inputs x_3 and x_4, and the variances and correlations of the inputs and weights are equal, w_3 equals w_4 in the optimal realization and the variation of the probability of error for a change in w_3 is the same as for a change in w_4. In each instance, the other weights are held at their optimal values for the appropriate correlation coefficient.

It is apparent from Fig. 7.1a-c that probability of error is not a convex function with respect to the weights. However, in this case, it is highly likely that there is only one set of weights which minimizes the probability of error. For all single gate cases investigated in which the probability of error for a given input combination was less than 0.5 for every possible input combination, the pattern search converges to a single minimum. It is conjectured that for any single gate realization which satisfies the condition such that the probability of error for any given input combination is less than 0.5, there is only one value of the weights such that the probability of error is minimized. Thus, if the starting point satisfies the above condition, the minimum obtained by the search should be the optimum realization of the function considered.

<u>Effects of Variance and Correlation on the Probability of Error of a Threshold Gate</u>

Changes in the variances and correlations of the inputs and weights, σ_x, σ_w, ρ_x, and ρ_w, respectively, produce changes in the variance of the separating function, and thereby affect the probability of error. In order to determine the dependence of probability of error on variances and correlations, σ_x, σ_w, ρ_x, and ρ_w are varied and the corresponding optimal realizations are found.

The functions

$$y_1 = x_1x_2x_3x_4$$

and

$$y_2 = x_1x_2 + x_1x_3x_4$$

are examined. These functions have the corresponding minimum integer realizations

$$y_1 = \left\langle x_1+x_2+x_3+x_4 \right\rangle_{3.5}$$

and

$$y_2 = \left\langle 3x_1+2x_2+x_3+x_4 \right\rangle_{4.5}$$

Table 7.1 and 7.2 show the variation of the probability of error of the optimum solutions with a change in ρ_w for constant σ_x, σ_w, and ρ_x.

TABLE 7.1. Minimum Probability of Error of $y = x_1 x_2 x_3 x_4$

Variance of Inputs σ_w^2	Variance of Weights σ_w^2	Correlation of Inputs ρ_x	Correlation of Weights ρ_w					
			0	.1	.3	.5	.7	.9
.0001	.0001	0	2.48×10^{-30}	1.08×10^{-32}	7.35×10^{-39}	3.34×10^{-48}	-0-	-0-
.0004	.0004	.9	7.67×10^{-7}	4.32×10^{-7}	1.12×10^{-7}	2.04×10^{-8}	2.19×10^{-9}	1.03×10^{-10}
.0009	.0009	0	2.25×10^{-5}	1.19×10^{-5}	2.28×10^{-6}	1.88×10^{-7}	2.87×10^{-9}	5.50×10^{-13}
.0025	.0025	.9	8.68×10^{-3}	7.80×10^{-3}	6.09×10^{-3}	4.46×10^{-3}	2.98×10^{-3}	1.72×10^{-3}
.01	.01	0	3.14×10^{-2}	2.92×10^{-2}	2.42×10^{-2}	1.83×10^{-2}	1.16×10^{-2}	4.67×10^{-3}
.01	.01	.9	4.17×10^{-2}	4.04×10^{-2}	3.75×10^{-2}	3.41×10^{-2}	3.01×10^{-2}	2.54×10^{-2}

TABLE 7.2. Minimum Probability of Error of $y = x_1x_2 + x_1x_3x_4$

Variance of Inputs σ_x^2	Variance of Weights σ_w^2	Correlation of Inputs ρ_x	Correlation of Weights ρ_w: 0	.1	.3	.5	.7	.9
0	.0025	0	9.93×10^{-3}	7.99×10^{-3}	4.37×10^{-3}	1.53×10^{-3}	1.51×10^{-4}	4.55×10^{-9}
.0004	.0004	0	3.06×10^{-5}	1.82×10^{-5}	5.02×10^{-6}	8.35×10^{-7}	5.81×10^{-8}	7.16×10^{-10}
.0009	.0009	0	1.94×10^{-3}	1.52×10^{-3}	8.32×10^{-4}	3.62×10^{-4}	1.06×10^{-4}	1.41×10^{-5}
.0016	.0016	0	9.13×10^{-3}	7.89×10^{-3}	5.51×10^{-3}	3.36×10^{-3}	1.63×10^{-3}	5.02×10^{-4}
.01	.01	0	6.73×10^{-2}	6.46×10^{-2}	5.88×10^{-2}	5.22×10^{-2}	4.43×10^{-2}	3.46×10^{-2}
.01	.01	.9	*8.24×10^{-2}	*8.06×10^{-2}	*7.68×10^{-2}	*7.33×10^{-2}	*7.08×10^{-2}	*6.63×10^{-2}
		Correlation of Weights ρ_x	Correlation of Inputs ρ_x: 0	.1	.3	.5	.7	.9
.0025	0	0	1.42×10^{-3}	2.82×10^{-3}	6.49×10^{-3}	1.07×10^{-2}	1.48×10^{-2}	1.88×10^{-2}

*Probability of Error for one or more input combinations is greater than 0.5.

For both functions considered, the probability of error increases as the variance of the inputs and weights increases and the probability of error decreases as the correlation of the weights increases. The fact that the probability of error increased with increasing variances and input correlation is evident from Eq. 3.11. The elements of the covariance matrices increase as σ_x, σ_w and ρ_x increase, thus increasing the variance of the separating function for all input combinations. The decrease in probability of error with increasing correlation of the weights is due to the fact that the last element of $\underline{n}_x$ is negative and the fact that the standard deviation of a weight is proportional to the mean of that weight. As the correlation of the weights approaches one, the first term in Eqn. 3.11 approaches the difference of two highly correlated random variables. Thus, the contribution of the first term to the variance of the separating function is greatly reduced.

Table 7.3 shows the minimum integer and optimal realizations of $y = x_1x_2 + x_1x_3x_4$ for several variances and correlations. The optimum weights are close to the minimum integer weights for small variances, but differ significantly for large variances. Note that the improvement obtained by using the optimal realization instead of the minimum integer realization is only about ten per cent. Thus, in

TABLE 7.3. Realization of $y = x_1x_2 + x_1x_3x_4$

Variance of Inputs & Weights $\sigma_x^2 = \sigma_w^2$	Correlation of Inputs and Weights $\rho_x = \rho_w$	Mean Value of Weights ($w_5 = 4.5$) w_1	w_2	w_3	w_4	Probability of Error
.0001	0	3.000	2.000	1.000	1.000	$3.10\text{x}10^{-14}$
		3.001	2.018	0.999	0.999	$1.78\text{x}10^{-14}$*
.0001	.9	3.000	2.000	1.000	1.000	$1.29\text{x}10^{-13}$
		3.004	2.001	0.996	0.996	$1.17\text{x}10^{-13}$*
.0004	0	3.000	2.000	1.000	1.000	$3.31\text{x}10^{-5}$
		3.011	2.021	0.988	0.989	$3.06\text{x}10^{-5}$*
.0004	.9	3.000	2.000	1.000	1.000	$5.44\text{x}10^{-5}$
		3.018	2.000	0.981	0.981	$4.87\text{x}10^{-5}$*
.0025	0	3.000	2.000	1.000	1.000	$2.14\text{x}10^{-2}$
		3.064	2.047	0.917	0.918	$1.96\text{x}10^{-2}$*
.0025	.9	3.000	2.000	1.000	1.000	$2.38\text{x}10^{-2}$
		3.096	2.000	0.890	0.890	$2.08\text{x}10^{-2}$*
.01	0	3.000	2.000	1.000	1.000	$7.53\text{x}10^{-2}$
		3.151	2.219	0.679	0.680	$6.73\text{x}10^{-2}$*
.01	.9	3.000	2.000	1.000	1.000	$7.96\text{x}10^{-2}$
		3.212	2.192	0.584	0.584	$6.63\text{x}10^{-2}$* †

*Optimum Realization

†Function not realized for one input combination.

most single gate applications, the minimum integer realization would be a good compromise between reliability and the amount of work required to obtain the realization.

Minimal Probability of Error Threshold Gate Networks

The function $y = x_1x_2 + x_3x_4$ is not linearly separable and therefore cannot be realized with a single gate. However, this function may be realized by the following two gate network

$$y = \left\langle 2y_1 + x_3 + x_4 \right\rangle_{1.5}$$

where

$$y_1 = \left\langle x_1 + x_2 \right\rangle_{1.5}$$

or

$$y = \left\langle 2 \left\langle x_1 + x_2 \right\rangle_{1.5} + x_3 + x_4 \right\rangle_{1.5}$$

The function y is realized for $\sigma_x = \sigma_w = 0.1$ and $\rho_x = \rho_w = 0$. The best realizations for the individual gates is

$$y_1 = \left\langle 0.995\ x_1 + 0.995\ x_2 \right\rangle_{1.5}$$

and

$$y = \left\langle 2.239\ y_1 + 0.986\ x_3 + 0.937\ x_4 \right\rangle_{1.5}$$

The probability of error of the network was calculated for the above gates and the minimum integer gates. These are compared to the optimal realization for the overall network found by the pattern search by adapting weights in both gates simultaneously. These results are summarized in Table 7.4.

TABLE 7.4. Probability of Error of $y = x_1x_2 + x_3x_4$

Realization	Probability of Error
Minimum Integer	
$y = \left\langle 2 \left\langle x_1+x_2 \right\rangle_{1.5} + x_3 + x_4 \right\rangle_{1.5}$	$4.38\text{x}10^{-2}$
Gates optimized individually	
$y = \left\langle 2.239 \left\langle 0.995x_1+0.995x_2 \right\rangle_{1.5} + 0.986x_3+0.987x_4 \right\rangle_{1.5}$	$4.57\text{x}10^{-2}$
Optimal network	
$y = \left\langle 1.816 \left\langle 0.995x_1+0.996x_2 \right\rangle_{1.5} + 0.085x_1+0.085x_2+0.952x_3+0.952x_4 \right\rangle_{1.5}$	$4.26\text{x}10^{-2}$

It is evident from Table 7.4 that the realization using the individual gates with least probability of error does not give the optimal realization. In fact, for this example, it is worse than the integer realization. The optimal realization requires the use of a five input gate in the second level and is slightly more reliable than the minimum integer realization.

Conclusions

The following conclusions may be drawn:

1) It is feasible to use a digital computer to find the realization which minimizes the

probability of error of a threshold gate.

2) Probability of error increases with increasing input variance, weight variance, and input correlation.

3) Probability of error decreases with increasing weight correlation.

4) In most cases, the single-gate minimum integer realization has a probability of error which is almost optimal.

5) In a threshold gate network, optimizing each gate in the network does not minimize the probability of error.

6) In any application, gates with the smallest possible input and weight variances should be used.

7) For given statistics, the current switching gate should be the most reliable of the circuits considered in Chapter II. This is due to the reduction of input variance and correlation produced by the isolation of the inputs from the separating function.

BIBLIOGRAPHY

Amodei, J. J., D. Hampel, T. R. Mayhew, and R. O. Winder. "An Integrated Threshold Gate," Digest of the 1967 International Solid State Circuits Conference.

Canion, J. R. "Adaptive Threshold Logic" (unpublished Master's Thesis, University of Houston, Houston, Texas, 1968).

Eriksen, C. E. "A Minimal Probability of Error Threshold Gate" (unpublished Master's Thesis, University of Houston, Houston, Texas, 1968).

Hooke, R. and T. Jeeves. "'Direct Search' Solutions of Numerical and Statistical Problems," Association for Computing Machinery Journal, Vol. 8, Pt. 2, April, 1966, pp. 212-229.

Lewis, P. M. and C. L. Coates. Threshold Logic. New York: John Wiley and Sons, 1967.

Miller, K. S. Multidimensional Gaussian Distributions. New York: John Wiley and Sons, 1964.

McNaughton, R. "Unate Truth Functions," IRE Trans. Electron. Computers, Vol. EC-10, March, 1961, pp. 1-6.

Morrison, D. F. Multivariate Statistical Methods. New York: McGraw-Hill Book Co., 1967.

Papoulis, A. Probability, Random Variables and Stochastic Processes. New York: McGraw-Hill Book Co., 1965.

Peterson, W. W. Error-Correcting Codes. Cambridge, Massachusetts: The M.I.T. Press, 1961.

Rowe, W. D. "The Transistor NOR Circuit," IRE Wescon Convention Record, Pt. 4, 1957, pp. 231-245.

APPENDIX A

COMPUTER PROGRAM FOR THRESHOLD GATE SIMULATION

PROGRAM FOR SIMULATION OF THE PRODUCT OF THE SEPARATING FUNCTIONS OF SEVERAL THRESHOLD GATES

NG = Number of gates in a two level network

NG1 = Number of gates in the first level

NI(I) = Number of inputs to the I-th gate

NW(I) = Number of weights in the I-th gate

ITT = Number of simulation iterations

AW = Weight vector, $\underline{n}_w$

AX = Input vector, $\underline{n}_x$

IX = Input distribution vector

MI = Number of independent inputs

NWT1 = Total number of weights in the first level

VARW = Variance of weights, σ_w^2

VARX = Variance of inputs, σ_x^2

CORW = Correlation of weights, ρ_w

CORX = Correlation of inputs, ρ_x

MW = Covariance matrix of weights (NWT1 x NWT1)

MX = Covariance matrix of inputs (MI x MI)

Y = Independent random vector for inputs, $\underline{Y}'$

Z = Independent random vector for weights, $\underline{Z}'$

F(I) = Simulated value of I-th separating function

FP = Simulated product of separating function

EXS = Simulation mean of FP

VARS = Simulation variance of FP

DIFF = Mean square error of distribution of FP with respect to a normal distribution with mean EXS and variance EXS

SETUP (A,C) Sets up plot intervals and normal distribution

COVMAT (M,N,V,R) See subroutine THRESH

INVERS (A, B, N) Generates $B = A^{-1}$ when A is a nonsingular N x N matrix

EQUAL (A, B, N) Sets B = A when A and B are N X N matrices

NORMAL (X), Generates a normal random number X,

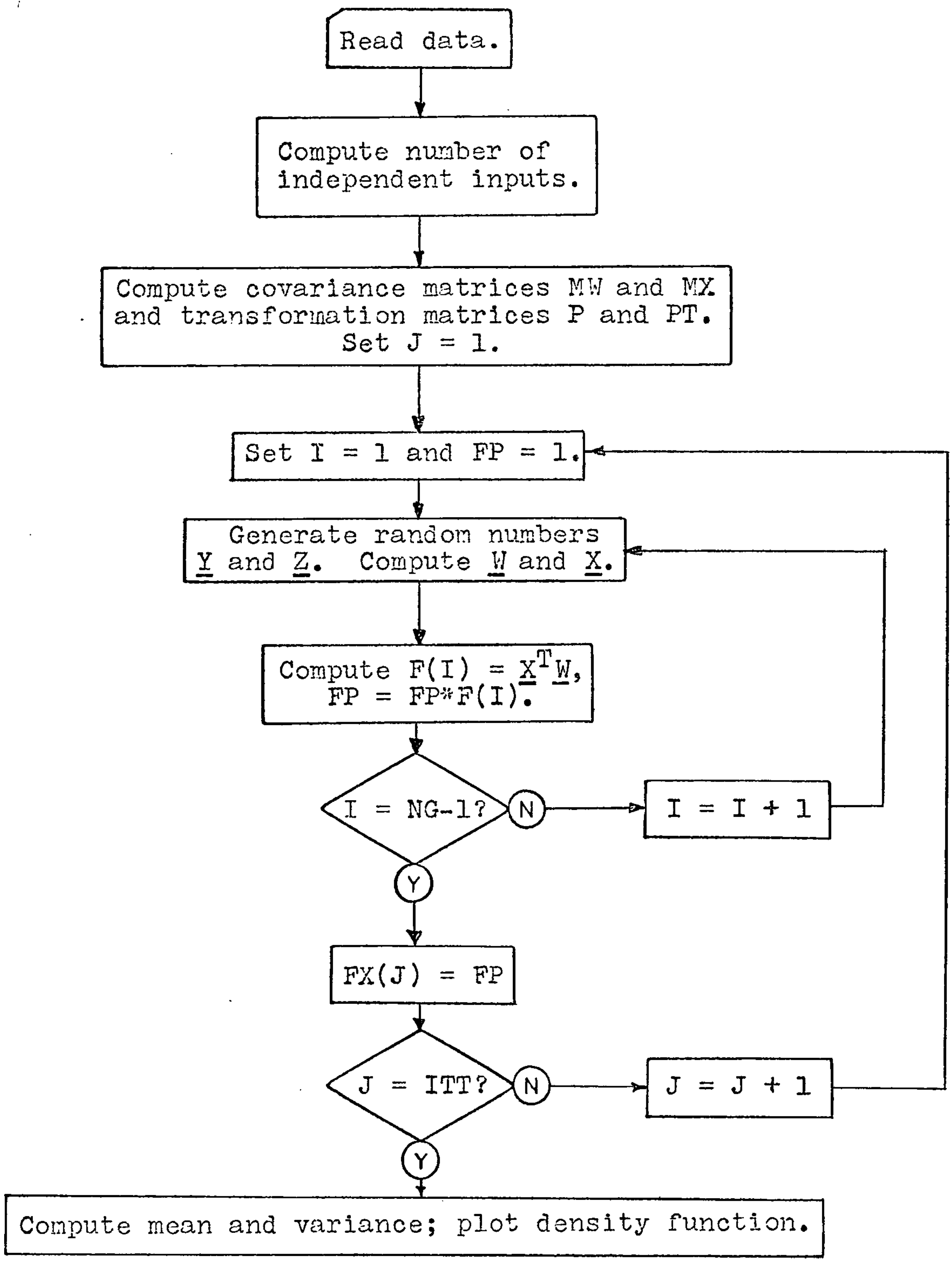
START (X) START sets up the subroutine initially

PROC (FX, EXS, VARS, ITT, B) Calculates VARS and the distribution of FX which is returned in B

PLOT2(N,A,B,C,AMAX) Plots B and C versus A for N values of A

SOURCE LANGUAGE: SDS Sigma 7 FORTRAN IV - H

Flow Diagram of Threshold Gate Simulation Program



```
C     THIS PROGRAM WILL SIMULATE THE DENSITY FUNCTION OF THE SEPARATING
C     FUNCTION OF A THRESHOLD GATE OR THE PRODUCT OF THE SEPARATING
C     FUNCTIONS OF THE FIRST LEVEL GATES OF A THRESHOLD GATE NETWORK.
C     NG = NUMBER OF GATES IN THE TWO LEVEL NETWORK
C     NI(I) = NUMBER OF INPUTS TO THE I-TH GATE
C     AW = WEIGHT VECTOR
C     IX = INPUT DISTRIBUTION VECTOR
C     VARW,VARX,CORW,CORX ARE THE STATISTICS OF THE GATE OR GATES
C     ITT = NUMBER OF SIMULATION ITERATIONS
C     AX = INPUT VECTOR
C     REF: SETUP, COVMAT, INVERS, EQUAL, NORMAL, PROC, PLOT2, PROB
      REAL MW, MX
      DIMENSION MW(20,20), MX(20,20), WMW(20,20), WMX(20,20), P(20,20),
     1 PT(20,20), Q(20,20), QT(20,20), X(20), Y(20), Z(20), W(20),
     1 D(20,20), A(101), B(101), C(101), AW(20), AX(20), FX(10000),
     1 IX(20), NI(11), NW(11), F(11)
      COMMON Q,QT
      CALL START (1,0)
 1    CONTINUE
      READ (5,5, END = 99) NG
 5    FORMAT (8I10)
      READ (5,5) (NI(I), I = 1,NG)
      DO 10 I = 1,NG
10    NW(I) = NI(I) + 1
      I = 1
      J = 0
      DO 12 K = 1,NG
      J = J + NW(K)
      READ (5,6) (AW(L), L = I,J)
 6    FORMAT (8E10.3)
12    I = I + NW(K)
      I = 1
      J = 0
      DO 14 K = 1,NG
      J = J + NI(K)
      READ (5,5) (IX(L), L = I,J)
14    I = I + NI(K)
      NG1 = NG - 1
      NIT1 = 0
      DO 20 I = 1,NG1
20    NIT1 = NIT1 + NI(I)
      MI = 0
      DO 22 I = 1,NIT1
22    IF ( IX(I) .GT. MI ) MI = IX(I)
      NWT1 = NIT1 + NG1
      READ (5,6) VARW, VARX
      READ (5,6) CORW, CORX
      READ (5,5) ITT
25    CONTINUE
      READ (5,6,END = 1) (AX(I), I = 1,MI)
C     SET UP PLOT VALUES
      CALL SETUP (A,C)
C     SET UP COVARIANCE AND TRANSFORMATION MATRICES
      CALL COVMAT (MX,MI,VARX,CORX)
```

```
      CALL INVERS (MX,NMX,MI)
      CALL EQUAL (D,QT,MI)
      CALL INVERS (D,PT,MI)
      CALL COVMAT (MW,NWT1,VARW,CORW)
      CALL INVERS (MW,NMW,NWT1)
      CALL EQUAL (D,QT,NWT1)
      CALL INVERS (D,P,NWT1)
      EXS = 0.
C     SIMULATION LOOP
      DO 100 J = 1,ITT
C     GENERATE RANDOM VARIABLES
      DO 40 I = 1,NWT1
40    CALL NORMAL (Z(I))
      DO 42 I = 1,MI
42    CALL NORMAL (Y(I))
      DO 44 I = 1,MI
      X(I) = 0.
      DO 44 K = 1,MI
44    X(I) = X(I) + PT(I,K)*Y(K)
      DO 46 I = 1,NWT1
      W(I) = 0.
      DO 46 K = 1,NWT1
46    W(I) = W(I) + P(I,K)*Z(K)
      FP = 1.
      KW = 1
      KX = 1
      DO 50 I = 1,NG1
      F(I) = 0.
      N = NI(I)
      DO 48 K = 1,N
      F(I) = F(I) + (AW(KW) + W(KW))*(AX(KX) + X(KX))
      KW = KW + 1
48    KX = KX + 1
      F(I) = F(I) - (AW(KW) + W(KW))
      KW = KW + 1
50    FP = FP*F(I)
      EXS = EXS + FP
      FX(J) = FP
100   CONTINUE
C     CALCULATE SIMULATION MEAN AND VARIANCE
      EXS = EXS/FLOAT(ITT)
      CALL PROC (FX,EXS,VARS,ITT,B)
      DIFF = 0.
      DO 130 I = 1,51
130   DIFF = DIFF + (B(I) - C(I))**2
C     PLOT SIMULATED DENSITY
      CALL PLOT2 (51,A,B,C,.1)
C     PRINT RESULTS
      WRITE (6,150) VARW,CORW,VARX,CORX,MI,ITT
150   FORMAT (/2X,'VARW = ',E10.3,3X,'CORW = ',E10.3,3X,'VARX = ',E10.3,
     1  3X,'CORX = ',E10.3,5X,'MI = ',I10,4X,'ITT = ',I10)
      WRITE (6,160) EXS, VARS, EXF, VARF
160   FORMAT (/3X,'EXS = ',E15.8,3X,'VARS = ',E15.8,3X,'EXF = ',E15.8,
     1  3X,'VARF = ',E15.8)
```

```
      WRITE (6,170) DIFF
170   FORMAT (/1X,'MEAN SQUARE ERROR = ',E15.8)
      WRITE (6,180) (AX(I), I = 1,NWT1)
180   FORMAT (///,' THE WEIGHTS ARE',/,8(1X,F14.7))
      WRITE (6,182) (AX(I), I = 1,MI)
182   FORMAT (///,' THE INPUTS ARE',/,8(1X,F14.7))
      GO TO 25
99    WRITE (7,5)
      END



      SUBROUTINE INVERS ( A, B, N )
C     GAUSS ELIMINATION MATRIX INVERSION
C     B IS THE INVERSE OF A NONSINGULAR REAL SQUARE MATRIX A
C     IF THERE IS A ZERO ON THE DIAGONAL OF A, OR IF A IS SINGULAR,
C     B IS SET EQUAL TO THE IDENTITY MATRIX
C     Q AND QT ARE SIMILARITY TRANSFORMATION MATRICES
C     REF: MATRIX, EQUAL, PRDIAG
      DIMENSION A(20,20),B(20,20),Q(20,20),QT(20,20),P(20,20),PT(20,20),
     1  D(20,20)
      COMMON Q, QT
      L = 1
1     DO 5 I = 1,N
      DO 4 J = 1,N
      Q(I,J) = 0.0
      QT(I,J) = 0.0
4     B(I,J) = A(I,J)
      IF ( B(I,I) .EQ. 0. ) GO TO 90
      QT(I,I) = 1.0
5     Q(I,I) = 1.0
      M = N - 1
      DO 50 J = 1,M
      DO 10 I = 1,N
      DO 8 K = 1,N
      PT(I,K) = 0.0
8     P(I,K) = 0.0
      PT(I,I) = 1.0
10    P(I,I) = 1.0
      JX = J + 1
      DO 15 I = JX,N
      P(J,I) = -B(J,I)/B(J,J)
15    PT(I,J) = -B(I,J)/B(J,J)
      CALL MATRIX (PT,QT,D,N)
      CALL EQUAL (QT,D,N)
      CALL MATRIX (Q,P,D,N)
      CALL EQUAL (Q,D,N)
      CALL MATRIX (PT,B,D,N)
      CALL EQUAL (B,D,N)
      CALL MATRIX (B,P,D,N)
      CALL EQUAL (B,D,N)
50    CONTINUE
```

```
      IF ( L .EQ. 1 ) GO TO 52
      CALL PRDIAG (DA,B,N)
      B(1,1) = DA
      RETURN
52    DO 60 I = 1,N
      DO 55 J = 1,N
      PT(I,J) = 0.0
55    P(I,J) = 0.0
      IF (B(I,I)) 58,70,56
56    PT(I,I) = 1.0/SQRT(B(I,I))
      P(I,I) = 1.0/SQRT(B(I,I))
58    S(I,I) = 1.0/B(I,I)
60    CONTINUE
      CALL MATRIX (Q,B,D,N)
      CALL MATRIX (D,QT,B,N)
      CALL MATRIX (PT,QT,D,N)
      CALL EQUAL (QT,D,N)
      CALL MATRIX (Q,P,D,N)
      CALL EQUAL (Q,D,N)
      RETURN
70    WRITE (6,72)
72    FORMAT (//'*** ERROR: INVERSE OF SINGULAR MATRIX'//)
      DO 76 I = 1,N
      WRITE (6,75) (A(I,J), J = 1,N)
75    FORMAT (8(1X,E14.7))
76    CONTINUE
      WRITE (6,77)
77    FORMAT (//)
      DO 78 I = 1,N
      WRITE (6,75) (B(I,J), J = 1,N)
78    CONTINUE
73    WRITE (6,77)
      CALL PDUMP
      WRITE (6,77)
      DO 80 I = 1,N
      DO 79 J = 1,N
79    B(I,J) = 0.
80    B(I,I) = 1.
      RETURN
90    WRITE (6,91)
91    FORMAT (//'*** ERROR: ZERO ON DIAGONAL, INVERSE NOT COMPUTED'//)
      DO 92 I = 1,N
      WRITE (6,75) (A(I,J), J = 1,N)
92    CONTINUE
      GO TO 73
C     ALTERNATE ENTRY , B(1,1) = |B|
      ENTRY DETERM ( A, B, N )
      L = 0
      GO TO 1
      END
```

```
      SUBROUTINE MATRIX ( A, B, C, N )
C     C = AB, WHERE A, B, AND C ARE SQUARE MATRICES
      DIMENSION A(20,20), B(20,20), C(20,20)
      DO 10 I = 1,N
      DO 10 J = 1,N
      C(I,J) = 0.0
      DO 10 K = 1,N
10    C(I,J) = C(I,J) + A(I,K) * B(K,J)
      RETURN
      END


      SUBROUTINE EQUAL ( A, B, N )
C     A IS SET EQUAL TO B, WHERE A AND B ARE SQUARE MATRICES
      DIMENSION A(20,20), B(20,20)
      DO 5 I = 1,N
      DO 5 J = 1,N
 5    A(I,J) = B(I,J)
      RETURN
      END


      SUBROUTINE PRDIAG ( PRA, A, N )
C     PRA IS THE PRODUCT OF THE DIAGONAL TERMS OF THE SQUARE MATRIX A
      DIMENSION A(20,20)
      PRA = 1.0
      DO 10 I = 1,N
10    PRA = PRA * A(I,I)
      RETURN
      END


      SUBROUTINE SETUP ( A, C )
C     SETS UP PLOT VALUES FOR SIMULATION
C     CALCULATED DENSITY FUNCTION IS STORED IN C
      DIMENSION A(51), C(51)
      DO 10 I = 1,51
      A(I) = -5.0 + .2*FLOAT(I-1)
10    C(I) = PROB (A(I)+.1,0.,1.) - PROB(A(I)-.1,0.,1.)
      RETURN
      END
```

```
      SUBROUTINE PROC ( FX, EXS, VARS, ITT, B )
C     CALCULATES SIMULATION VARIANCE
C     SIMULATED DENSITY FUNCTION IS STORED IN B
      DIMENSION B(51), FX(10000)
      VARS = 0.
      DO 10 J = 1,ITT
10    VARS = VARS + (FX(J) - FXS)**2
      VARS = VARS/FLOAT(ITT)
      DO 20 I = 1,51
20    B(I) = 0.
      DO 30 J = 1,ITT
      G = (FX(J) - EXS)/SQRT(VARS)
      G = 5.*G + 26.5
      I = IFIX(G)
      IF ( I .LT. 1 ) I = 1
      IF ( I .GT. 51 ) I = 51
30    B(I) = B(I) + 1
      DO 40 I = 1,51
40    B(I) = B(I)/FLOAT(ITT)
      RETURN
      END



      SUBROUTINE PLOT2 ( N, X, Y, Z, AMAX )
C     Y AND Z ARE PLOTTED VERSUS X
C     N = NUMBER OF INCREMENTS, X = X(T), Y = F1(T), Z = F2(T)
      DIMENSION LINE(103), SCALE(6), X(101), Y(101), Z(101)
      DATA KBLNK, KSTAR, KPLUS, KMINU, KAEYE/ ' ', '*', '+', '-', 'I'/
      AMIN = 0.
      DO 14 I = 1,N
      IF ( Y(I) .GT. AMAX ) AMAX = Y(I)
      IF ( Y(I) .LT. AMIN ) AMIN = Y(I)
      IF ( Z(I) .GT. AMAX ) AMAX = Z(I)
      IF ( Z(I) .LT. AMIN ) AMIN = Z(I)
14    CONTINUE
C     PRINT TOP SCALES AND BORDER
      WRITE (6,602)
602   FORMAT (1H1)
      H1 = AMAX - AMIN
      H12 = 0.2 * H1
      H1 = 100./H1
      SCALE(1) = AMIN
      DO 10 I = 2,6
10    SCALE(I) = SCALE(I-1) + H12
      WRITE (6,600) (SCALE(I), I = 1,6)
600   FORMAT (19X,E10.3,5(10X,E10.3))
      DO 12 I = 1,103
12    LINE(I) = KMINU
      DO 13 I = 2,102,10
13    LINE(I) = KAEYE
      WRITE (6,601) (LINE(I), I = 1,103)
601   FORMAT (25X,103A1)
```

```
C       PLØT BØDY ØF GRAPH
        DØ 21 I = 2,102
21      LINE(I) = KBLNK
        DØ 23 I = 1,103,51
23      LINE(I) = KAEYE
        DØ 22 J = 1,N
        L1 = (Y(J) - AMIN)*H1 + 2.5
        L2 = (Z(J) - AMIN)*H1 + 2.5
        LINE(L1) = KSTAR
        LINE(L2) = KPLUS
        WRITE (6,603) X(J), Y(J), Z(J), (LINE(K), K = 1,103)
603     FØRMAT (1X,F7.2,1X,F7.4,1X,F7.4,1X,103A1)
        LINE(L1) = KBLNK
        LINE(L2) = KBLNK
        LINE(52) = KAEYE
22      CØNTINUE
C       PRINT BØTTØM LINE AND SCALES
        DØ 31 I = 1,103
31      LINE(I) = KMINU
        DØ 32 I = 2,102,10
32      LINE(I) = KAEYE
        WRITE (6,601) (LINE(I), I = 1,103)
        WRITE (6,600) (SCALE(I), I = 1,6)
        RETURN
        END



        SUBRØUTINE CØVMAT ( M, N, V, R )
        REAL M(12,12)
        DØ 10 I = 1,N
        DØ 10 J = 1,N
        M(I,J) = R
        IF ( I .EQ. J ) M(I,J) = 1.0
10      M(I,J) = M(I,J) * V
        RETURN
        END



        SUBRØUTINE NØRMAL ( X )
C       EACH CALL GENERATES A NØRMAL RANDØM NUMBER X
C       START MUST BE CALLED INITIALLY
C       'S' IN CØLUMN 1 DENØTES SYMBØLIC SDS SIGMA 7 ASSEMBLY LANGUAGE
C       INSTRUCTIØN
1       CØNTINUE
S       LI,11  , X'40000'
C       SUM 12 UNIFØRM RANDØM NUMBERS
S       SLS,11    12
S       LW,5      M
S       MI,5      65539
S       LW,9      5
```

```
S       STW,5     M
S       SLS,9     -8
S       AW,9      11
S       LI,1      2
S   20  MI,5      65539
S       LW,3      5
S       SLS,3     -3
S       AW,3      11
S       FAS,9     3
S       AI,1      1
S       CI,1      12
S       BLE       &20
C       APPLY POLYNOMIAL CORRECTION
S       FSS,9     SIX
S       FMS,9     FOURTH
S       STW,9     Y1
S       LW,3      Y1
S       FMS,3     Y1
S       STW,3     Y2
S       FMS,3     A9
S       FAS,3     A7
S       FMS,3     Y2
S       FAS,3     A5
S       FMS,3     Y2
S       FAS,3     A3
S       FMS,3     Y2
S       FAS,3     A1
S       FMS,3     Y1
S       LW,13     X
S       STW,3     *13
        RETURN
        ENTRY START ( X )
        SIX = 6.
        FOURTH = .25
        A1 = 3.949846138
        A3 = .252408784
        A5 = .076542912
        A7 = .008355968
        A9 = .029899776
        IF ( M .EQ. 0 ) M = 1
        GO TO 1
        END


          DEF          PROB
*   THIS FUNCTION CALCULATES THE PROBABILITY THAT A RANDOM X IS LESS THAN
*   (X - EX)/STD
*   FUNCTION PROB ( X, EX, STD )
*   SOURCE LANGUAGE IS SDS SIGMA 7 ASSEMBLY LANGUAGE--SYMBOL
          BOUND        8
A0        DATA,8       FL'1'
A1        DATA,8       FL'0.0498673470'
```

```
A2        DATA,8     FL'0.0211410061'
A3        DATA,8     FL'0.0032776263'
A4        DATA,8     FL'0.0000380036'
A5        DATA,8     FL'0.0000488906'
A6        DATA,8     FL'0.0000053830'
HALF      DATA,8     FL'0.5'
PROB      AI,13      1
          LW,1       *13
          AI,13      1
          LW,2       *13
          AI,13      1
          LW,3       *13
          LW,4       *1
          FSS,4      *2
          LW,3       *3
          FDS,4      3
          LI,2       1
          CI,4       0
          BGEZ       MAIN
          LI,2       0
          LCW,4      4
MAIN      LI,5       0
          LD,6       4
          FML,6      A6
          FAL,6      A5
          FML,6      4
          FAL,6      A4
          FML,6      4
          FAL,6      A3
          FML,6      4
          FAL,6      A2
          FML,6      4
          FAL,6      A1
          FML,6      4
          FAL,6      A0
          LI,3       1
LOOP      FML,6      6
          AI,3       1
          CI,3       4
          BLE        LOOP
          LD,4       A0
          FDL,4      6
          FML,4      HALF
          CI,2       0
          BE         RETURN
          LD,6       A0
          FSL,6      4
          LD,4       6
RETURN    LW,3       4
          AI,13      1
          B          *13
          END
```

APPENDIX B

COMPUTER PROGRAM TO CALCULATE THE PROBABILITY OF ERROR OF A THRESHOLD GATE

SUBROUTINE TO CALCULATE PROBABILITY OF ERROR OF A THRESHOLD GATE - THRESH

AW = Weight vector, $\underline{n}_w$

AX = Input vector, $\underline{n}_x$

VARW = Variance of weights, σ_w^2

VARX = Variance of inputs, σ_x^2

CORW = Correlation of weights, ρ_w

CORX = Correlation of inputs, ρ_x

N = Number of inputs, n

PE = Total probability of error for equally likely inputs, p_E

CW = Covariance matrix of the weights (N+1 x N+1)

CX = Covariance matrix of the inputs (N x N)

EXF = Expected value of f(p)

VARF = Variance of f(p)

PF = Probability of error given AX

PROB (X,EX,VAR) = Probability that a random, normally distributed X $\leq$ (X-EX)/$\sqrt{\text{VAR}}$

FN(X)= Value of the logic function FN(X)

COVMAT (M, N, V, R) generates an N x N covariance matrix

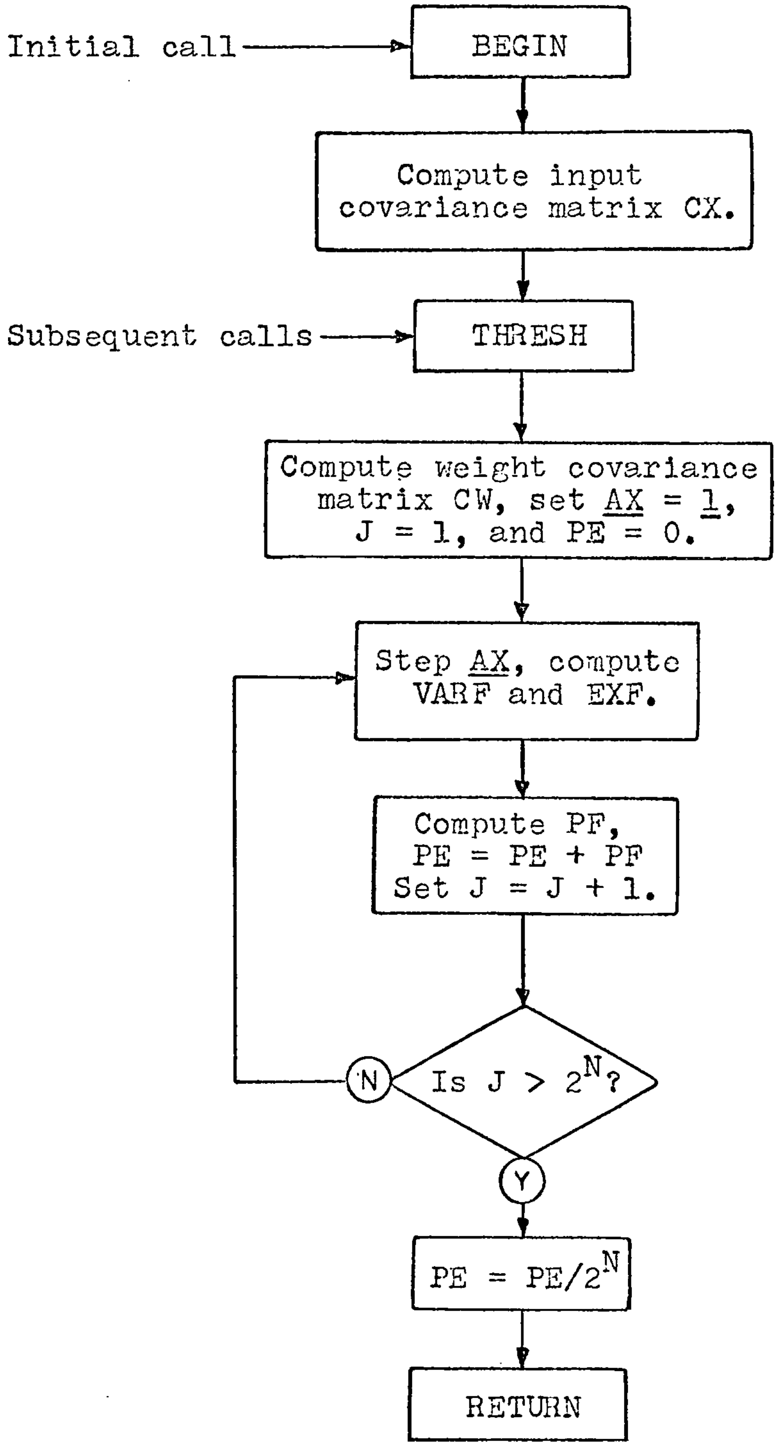
$M_{ij} = \text{cov}(x_i, x_j) = \rho_{ij}\sigma_{x_i}\sigma_{x_j}$, where $\rho_{ii} = 1$
and $\rho_{ij} = R$, $i \neq j$, and $\sigma_{x_i} = \sigma_{x_j} = \sqrt{V}$

MATRIX (A,B,C,N) forms the product C = AB where A, B, and C are N x N matrices.

STEP (X,N) considers X as an N place binary number and returns X + 1, neglecting any high order carry. Thus entering X = 1101, N = 4, returns X = 1110.

SOURCE LANGUAGE: SDS Sigma 7 FORTRAN IV - H

Flow Diagram of Subroutine THRESH



```
      SUBROUTINE THRESH ( AW, VARW, CORW, VARX, CORX, N, PE, IFN )
C     THIS SUBROUTINE CALCULATES THE PROBABILITY OF ERROR OF AN N-INPUT
C     THRESHOLD GATE WITH WEIGHTS AW, AND STATISTICS VARW,CORW,VARX,CORX.
C     IFN IS THE NUMBER OF POINTS OF THE N-CUBE WHERE PE > 0.5
C     THE LOGIC FUNCTION REALIZED IS SPECIFIED IN FUNCTION SUBPROGRAM FN
C     BEGIN MUST BE CALLED INITIALLY
C     REF: COVMAT, MATRIX, STEP, PROB, FN
      DIMENSION CW(12,12),CX(12,12),Q(12,12),QT(12,12),RW(12,12),
     1  RX(12,12),VW(12,12),VX(12,12),AW(12),AX(12)
 1    CONTINUE
      AX(NP) = -1.
C     GENERATE MW
      DO 12 I = 1,NP
      DO 10 J = 1,NP
10    VW(I,J) = 0.
12    VW(I,I) = ABS(AW(I))*STDW
      CALL MATRIX (VW,RW,QT,NP)
      CALL MATRIX (QT,VW,CW,NP)
      DO 20 I = 1,N
20    AX(I) = 1.
      PE = 0.
      IFN = 0
C     STEP THRU ALL POSSIBLE INPUT COMBINATIONS
      DO 100 J = 1,M
      CALL STEP (AX,N)
C     CALCULATE VARIANCE OF SEPARATING FUNCTION
      VARF = 0.
      DO 22 I = 1,NP
      DO 22 K = 1,NP
22    VARF = VARF + AX(I)*CW(I,K)*AX(K)
      DO 24 I = 1,N
      DO 24 K = 1,N
24    VARF = VARF + AW(I)*CX(I,K)*AW(K)
C     CALCULATE MEAN OF SEPARATING FUNCTION
      EXF = 0.
      DO 28 I = 1,N
28    EXF = EXF + AW(I)*AX(I)
      X = EXF - AW(NP)
C     EVALUATE LOGIC FUNCTION
      F = FN (AX)
C     CALCULATE PROBABILITY OF ERROR
      IF ( X ) 31,30,31
30    PF = 0.5
      GO TO 36
31    IF ( F .EQ. 1. ) X = -X
      PF = PROB (X,0.,VARF)
36    PE = PE + PF
      IF ( X .GE. 0. ) IFN = IFN + 1
100   CONTINUE
      PE = PE/M
      RETURN
      ENTRY BEGIN ( AW, VARW, CORW, VARX, CORX, N, PE, IFN )
      NP = N + 1
      M = 2**N
```

```
C     GENERATE CORRELATION AND VARIANCE MATRICES
      STDX = SQRT(VARX)
      STDW = SQRT(VARW)
      CALL COVMAT (RW,NP,1.,CORW)
      CALL COVMAT (RX,N,1.,CORX)
C     GENERATE MX
      DO 52 I = 1,N
      DO 50 J = 1,N
50    VX(I,J) = 0.
52    VX(I,I) = STDX
      CALL MATRIX (RX,VX,Q,N)
      CALL MATRIX (VX,Q,CX,N)
      GO TO 1
      END



      FUNCTION FN ( X )
      DIMENSION X(12)
C     THIS FUNCTION EVALUATES THE FOLLOWING LOGIC FUNCTION
      FN = X(1)*X(2) + X(1)*X(3)*X(4)
      IF ( FN .GT. 0. ) FN = 1.0
      RETURN
      END



      SUBROUTINE STEP ( X, N )
C     THIS SUBROUTINE STEPS THRU THE POINTS OF THE N-CUBE
      DIMENSION X(20)
      CARRY = 1.
      DO 10 I = 1,N
      M = N - I + 1
      IF ( CARRY .EQ. 0.) GO TO 10
      IF ( X(M) .EQ. 1.) GO TO 20
      X(M) = 1.
      CARRY = 0.
      GO TO 10
20    X(M) = 0.
10    CONTINUE
      RETURN
      END



      SUBROUTINE MATRIX ( A, B, C, N )
C     C = A*B, WHERE A, B, AND C ARE N X N MATRICES
      DIMENSION A(12,12), B(12,12), C(12,12)
      DO 10 I = 1,N
      DO 10 J = 1,N
      C(I,J) = 0.0
```

```
      DO 10 K = 1,N
10    C(I,J) = C(I,J) + A(I,K) * B(K,J)
      RETURN
      END


      SUBROUTINE COVMAT ( M, N, V, R )
C     THIS SUBROUTINE GENERATES AN N X N COVARIANCE MATRIX M WITH
C     VARIANCE V AND CORRELATION R
      REAL M(20,20)
      C = R*V
      DO 12 I = 1,N
      DO 10 J = 1,N
10    M(I,J) = C
12    M(I,I) = V
      RETURN
      END


      FUNCTION PROB ( X, EX, VAR )
C     PROBABILITY THAT A RANDOM X IS LESS THAN (X - EX)/STD
      DOUBLE PRECISION S,T
      T = 0.0D0
      T = DBLE((X-EX)/SQRT(VAR))
      IF ( T .LT. -4.0D0 ) GO TO 20
      I = 1
      IF (T) 5,10,10
 5    T = -T
      I = 0
10    S = 0.0000053830D0*T + 0.0000488906D0
      S = S*T + 0.0000380036D0
      S = S*T + 0.0032776263D0
      S = S*T + 0.0211410061D0
      S = S*T + 0.0498673470D0
      S = S*T + 1.0D0
      S = S**16
      PROB = 1.0D0/(2.0D0*S)
      IF (I.EQ.1) PROB = 1.0D0 - PROB
      RETURN
20    CONTINUE
      IF ( T .LT. -16.0D0 ) GO TO 30
      T = -T
      S = 0.5D0*T**2
      ET = DEXP(S)
      PROB = 0.39894228*0.31789023/(0.33458084*T*ET)
      RETURN
30    PROB = 0.
      RETURN
      END
```

APPENDIX C

COMPUTER PROGRAM TO CALCULATE THE PROBABILITY OF ERROR OF A THRESHOLD GATE NETWORK

SUBROUTINE TO CALCULATE PROBABILITY OF ERROR OF A THRESHOLD GATE NETWORK - THRESM

AW = Weight vector, $\underline{n}_W$

AX = Input vector, $\underline{n}_X$

VARW = Variance of weights, σ_W^2

VARX = Variance of inputs, σ_X^2

CORW = Correlation of weights, ρ_W

CORX = Correlation of inputs, ρ_X

NI = Total number of network inputs

NO = Number of inputs that go only to output gate

NG = Number of gates in network

PE = Total probability of error for equally likely inputs, p_E

CW = Covariance matrix of the weights (N+1 x N+1)

CX = Covariance matrix of the inputs (N x N)

P = Probability of occurance of outputs of first level gates

NG1 = Number of gates in first level

NI1 = Number of inputs to each first level gate

EXF2 = Mean of output gate separating function

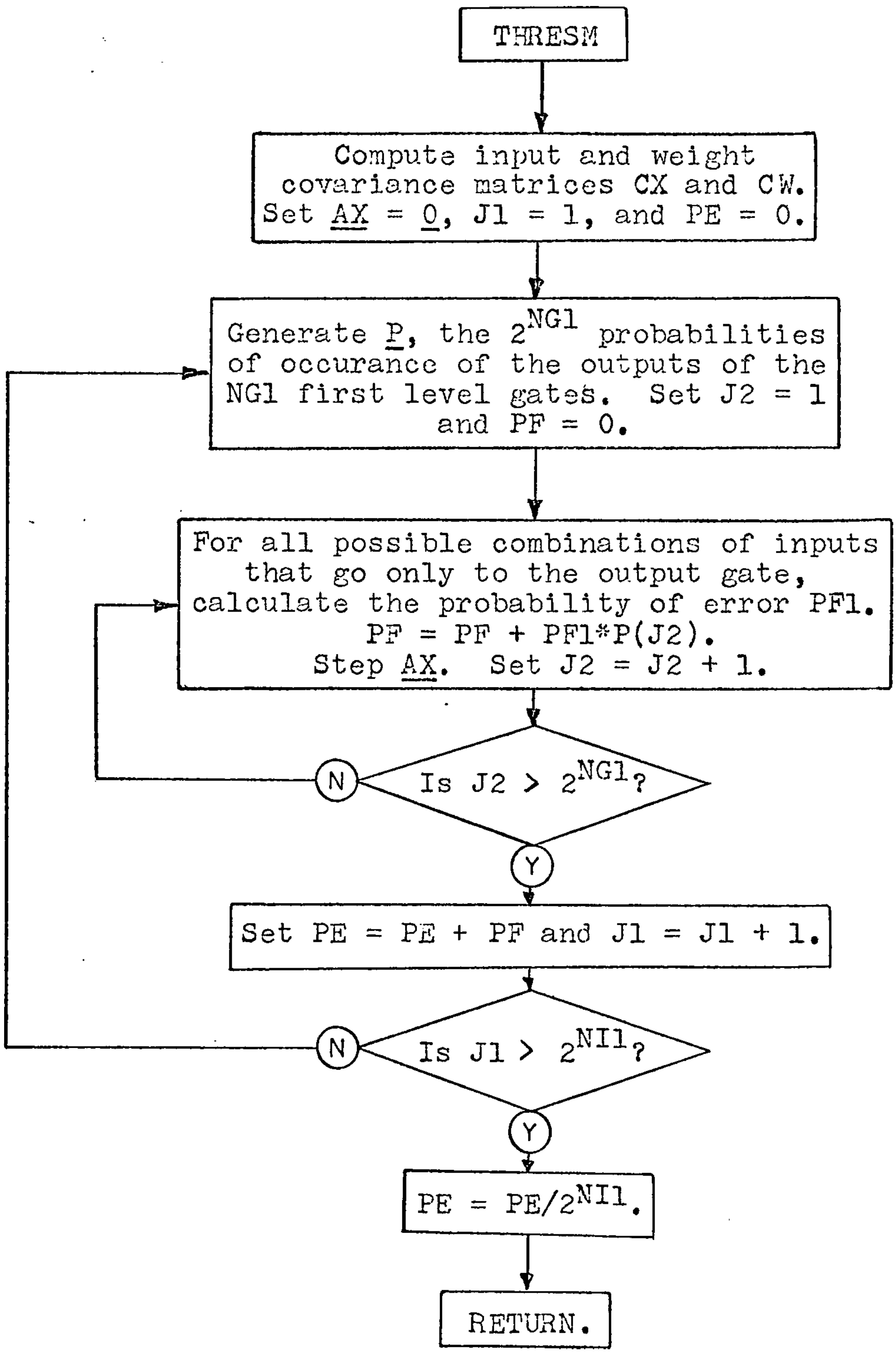
VARF2 = Variance of output gate separating function

GENP (AW, VARW, CORW, VARX, CORX, NG1, N, P) generates the 2^{NG1} probabilities of occurance of the outputs of NG1 first level gates

GENB (B, N) generates an n-th order B matrix

SOURCE LANGUAGE: SDS Sigma 7 FORTRAN IV - H

Flow Diagram of Subroutine THRESM



```
      SUBROUTINE THRESM (AW,VARW,CORW,VARX,CORX,NI,NO,NG,PE)
C     THIS SUBROUTINE CALCULATES THE PROBABILITY OF ERROR OF A TWO-LEVEL
C     THRESHOLD GATE NETWORK WITH NG GATES AND NI INPUTS.  (NI-NO) INPUTS
C     GO TO EVERY GATE.  (NO) INPUTS GO ONLY TO THE OUTPUT GATE.  AW
C     CONTAINS THE WEIGHTS, FIRST THE NI-NO+1 WEIGHTS OF EACH FIRST LEVEL
C     GATE, THEN THE NI-NO WEIGHTS CORRESPONDING TO THE INPUTS THAT GO TO
C     EACH FIRST LEVEL GATE, FOLLOWED BY THE NG WEIGHTS CORRESPONDING
C     TO THE OUTPUTS OF THE FIRST LEVEL GATES, THE NO WEIGHTS OF THE
C     INPUTS THAT GO ONLY TO THE OUTPUT GATE, AND THE OUTPUT GATE THRESHOLD.
C     THE LOGIC FUNCTION REALIZED IS SPECIFIED IN FUNCTION SUBPROGRAM FN
C     BEGINM MUST BE CALLED INITIALLY
C     REF: COVMAT, MATRIX, STEP, GENP, FN, PROB
      DIMENSION AW(20),AX(10),CW(20,20),CX(20,20),D(20,20),RW(20,20),
     1  RX(20,20),VW(20,20),VX(20,20),P(20),X(10)
C     SET UP CONSTANTS
      NG1 = NG - 1
      NI1 = NI - NO
      NW1 = NI1 + 1
      NI2 = NI + NG1
      NWT1 = NW1*NG1
      NWT2 = NI2 + 1
      NWT = NWT1 + NWT2
      MO = 2**NO
      MI1 = 2**NI1
      MG1 = 2**NG1
      NC = NI1 + NG1
      NCP = NC + 1
C     CALCULATE COVARIANCE MATRICES
      STDW = SQRT(VARW)
      STDX = SQRT(VARX)
      CALL COVMAT (RW,NWT,1.,CORW)
      DO 12 I = 1,NWT
      DO 10 J = 1,NWT
10    VW(I,J) = 0.
12    VW(I,I) = ABS(AW(I))*STDW
      CALL MATRIX (VW,RW,D,NWT)
      CALL MATRIX (D,VW,CW,NWT)
      CALL COVMAT (RX,NI2,1.,CORX)
      DO 22 I = 1,NI2
      DO 20 J = 1,NI2
20    VX(I,J) = 0.
22    VX(I,I) = STDX
      CALL MATRIX (VX,RX,D,NI2)
      CALL MATRIX (D,VX,CX,NI2)
      DO 30 I = 1,NI2
      X(I) = 0.
30    AX(I) = 0.
      AX(NWT2) = -1.
      PE = 0.
C     STEP THRU ALL POSSIBLE COMBINATIONS OF THE INPUTS THAT GO TO BOTH LEVEL
      DO 100 J1 = 1,MI1
C     GENERATE THE PROBABILITIES OF OCCURANCE OF THE FIRST LEVEL OUTPUT
C     COMBINATIONS FOR THE GIVEN INPUT COMBINATION
      CALL GENP (AW,VARW,CORW,AX,VARX,CORX,NG1,NI1,P)
```

```
      PF = 0.
C     STEP THRU ALL POSSIBLE COMBINATIONS OF FIRST LEVEL GATE OUTPUTS
      DO 90 J2 = 1,MO1
      PF1 = 0.
C     STEP THRU ALL POSSIBLE COMBINATIONS OF INPUTS THAT GO ONLY TO THE
C     OUTPUT GATE
      DO 80 J3 = 1,MO
C     CALCULATE VARIANCE OF THE OUTPUT GATE SEPARATING FUNCTION
      VARF2 = 0.
      DO 52 I = 1,NWT2
      DO 52 K = 1,NWT2
52    VARF2 = VARF2 + AX(I)*CW(NWT1+I,NWT1+K)*AX(K)
      DO 54 I = 1,NI2
      DO 54 K = 1,NI2
54    VARF2 = VARF2 + AW(NWT1+I)*CX(I,K)*AW(NWT1+K)
C     CALCULATE MEAN OF THE OUTPUT GATE SEPARATING FUNCTION
      EXF2 = 0.
      DO 56 I = 1,NWT2
56    EXF2 = EXF2 + AX(I)*AW(NWT1+I)
C     EVALUATE LOGIC FUNCTION
      F = FN(X)
C     CALCULATE PROBABILITY OF ERROR
      FX = EXF2
      IF ( F .EQ. 1. ) FX = -FX
      PF2 = PROB (FX,0.,VARF2)
      PF1 = PF1 + PF2
      CALL STEP (AX,NI2)
      CALL STEP (X,NI)
80    CONTINUE
      PF1 = PF1/FLOAT(MO)
      PF = PF + PF1*P(J2)
      DO 85 I = 1,NI1
85    X(I) = AX(I)
90    CONTINUE
      PE = PE + PF
100   CONTINUE
      PE = PE/FLOAT(MI1)
      RETURN
      END




      SUBROUTINE GENP (AW,VARW,CORW,AX,VARX,CORX,NG,N,P)
C     THIS SUBROUTINE GENERATES P, A VECTOR CONTAINING THE 2**NG
C     PROBABILITIES OF OCCURANCE OF THE OUTPUTS OF NG, N-INPUT
C     THRESHOLD GATES WITH STATISTICS VARW, CORW, VARX, AND CORX, WEIGHT
C     VECTOR AW, AND INPUT VECTOR AX.
      DIMENSION A(6,4),AW(20),AX(10),CW(20,20),CX(20,20),D(20,20),
     1 EXF(11),IB(5),P(20),PO(20),RW(20,20),RX(20,20),VW(20,20),
     1 VX(20,20),B(20,20),C(20,20),AWX(10),AXW(20),X(10)
      NP = N + 1
      NWT1 = NG*NP
      M = 2**NG
```

```
      P(1) = 1.
      IF ( NG .LT. 1 ) RETURN
      P0(1) = 1.
C     CALCULATE THE MEANS OF THE SEPARATING FUNCTIONS
      L = 0
      DO 120 I = 1,NG
      F = 0.
      DO 110 J = 1,N
110   F = F + AW(J+L)*AX(J)
      L = L + NP
120   EXF(I) = F - AW(L)
      F = 1.
      DO 130 I = 1,NG
130   F = F*EXF(I)
      EXF(NG+1) = F
      DO 5 I = 1,NG
 5    X(I) = 0.
C     STEP THRU ALL POSSIBLE INPUT COMBINATIONS
      DO 20 J = 2,M
      CALL STEP (X,NG)
C     CALCULATE MEAN OF PRODUCT OF SEPARATING FUNCTIONS
      EXFI = 1.
      DO 10 I = 1,NG
      IF ( X(I) .EQ. 0. ) GO TO 10
      EXFI = EXFI*EXF(I)
10    CONTINUE
C     CALCULATE MEAN VECTORS
      L = 0
      DO 220 I = 1,NG
      CONST = X(I)/EXF(I)
      DO 210 K = 1,N
210   AXW(K+L) = AX(K)*CONST
      L = L + NP
220   AXW(L) = -CONST
      L = 0
      DO 225 K = 1,N
225   AWX(K) = 0.
      DO 240 I = 1,NG
      IF ( X(I) .EQ. 0. ) GO TO 240
      DO 230 K = 1,N
230   AWX(K) = AWX(K) + AW(K+L)/EXF(I)
240   L = L + NP
C     CALCULATE COVARIANCE MATRICES
      STDW = SQRT(VARW)
      STDX = SQRT(VARX)
      CALL COVMAT (RW,NWT1,1.,CORW)
      DO 312 I = 1,NWT1
      DO 310 K = 1,NWT1
310   VW(I,K) = 0.
312   VW(I,I) = ABS(AW(I))*STDW
      CALL MATRIX (VW,RW,D,NWT1)
      CALL MATRIX (D,VW,CW,NWT1)
      CALL COVMAT (RX,N,1.,CORX)
      DO 322 I = 1,N
```

```
      D0 320 K = 1,N
320   VX(I,K) = 0.
322   VX(I,I) = STDX
      CALL MATRIX (VX,RX,D,N)
      CALL MATRIX (D,VX,CX,N)
C     CALCULATE VARIANCE OF PRODUCT OF SEPARATING FUNCTIONS
      VARFI = 0.
      D0 250 I = 1,N
      D0 250 K = 1,N
250   VARFI = VARFI + AWX(I)*CX(I,K)*AWX(K)
      D0 260 I = 1,NWT1
      D0 260 K = 1,NWT1
260   VARFI = VARFI + AXW(I)*CW(I,K)*AXW(K)
      VARFI = VARFI*EXFI*EXFI
C     CALCULATE PROBABILITY THAT PRODUCT IS GREATER THAN ZERO
      P0(J) = PROB (EXFI,0.,VARFI)
20    CONTINUE
C     GENERATE B MATRIX
      CALL GENB (B,NG)
C     FORM C, THE INVERSE OF B
      CONST = 4./FLOAT(M)
      D0 30 I = 1,M
      D0 30 J = 1,M
30    C(I,J) = (B(J,I) - .5)*CONST
      C(M,1) = C(M,1) - 1.
C     CALCULATE THE PROBABILITY OF OCCURANCE OF THE OUTPUT COMBINATION
      D0 40 I = 1,M
      P(I) = 0.
      D0 40 J = 1,M
40    P(I) = P(I) + C(I,J)*P0(J)
      RETURN
      END



      SUBROUTINE GENB ( B, N )
C     THIS SUBROUTINE GENERATES AN N'TH ORDER B MATRIX USED BY THE
C     MULTIGATE THRESHOLD NETWORK PROBABILITY OF ERROR ROUTINE--THRESM
C     REF: STEP
      DIMENSION B(20,20), X(4), Y(4)
      M = 2**N
      D0 5 I = 1,N
 5    Y(I) = 1.
      D0 100 J = 1,M
      CALL STEP (Y,N)
      K = 0
      D0 10 I = 1,N
10    K = K + IFIX(Y(I) + .1)
      D0 20 I = 1,N
20    X(I) = 1.
      D0 80 L = 1,M
      CALL STEP (X,N)
      B(J,L) = 0.
```

```
      F = 0.
      DO 30 I = 1,N
30    F = F + X(I)*Y(I)
      LA = IFIX(F + .1) + K
      LA = (LA + 1)/2 - LA/2
      IF ( LA .GT. 0 ) GO TO 80
      B(J,L) = 1.
80    CONTINUE
100   CONTINUE
      RETURN
      END
```

APPENDIX D

COMPUTER PROGRAMS FOR PATTERN SEARCH TO MINIMIZE PROBABILITY OF ERROR

COMPUTER PROGRAMS FOR PATTERN SEARCH TO MINIMIZE PROBABILITY OF ERROR

N = Number of inputs

W = Initial weight vector, $\underline{n}_w$

DWF = Initial step size

DWL = Minimum step size

VARW = Variance of weights, σ_w^2

VARX = Variance of inputs, σ_x^2

CORW = Correlation of weights, ρ_w

CORX = Correlation of inputs, ρ_x

WW = Adjusted weight vector

INC = Pattern vector

PE = Probability of error given WW

BEST = Minimum probability of error

IFN = Number of points at which PE > 0.5

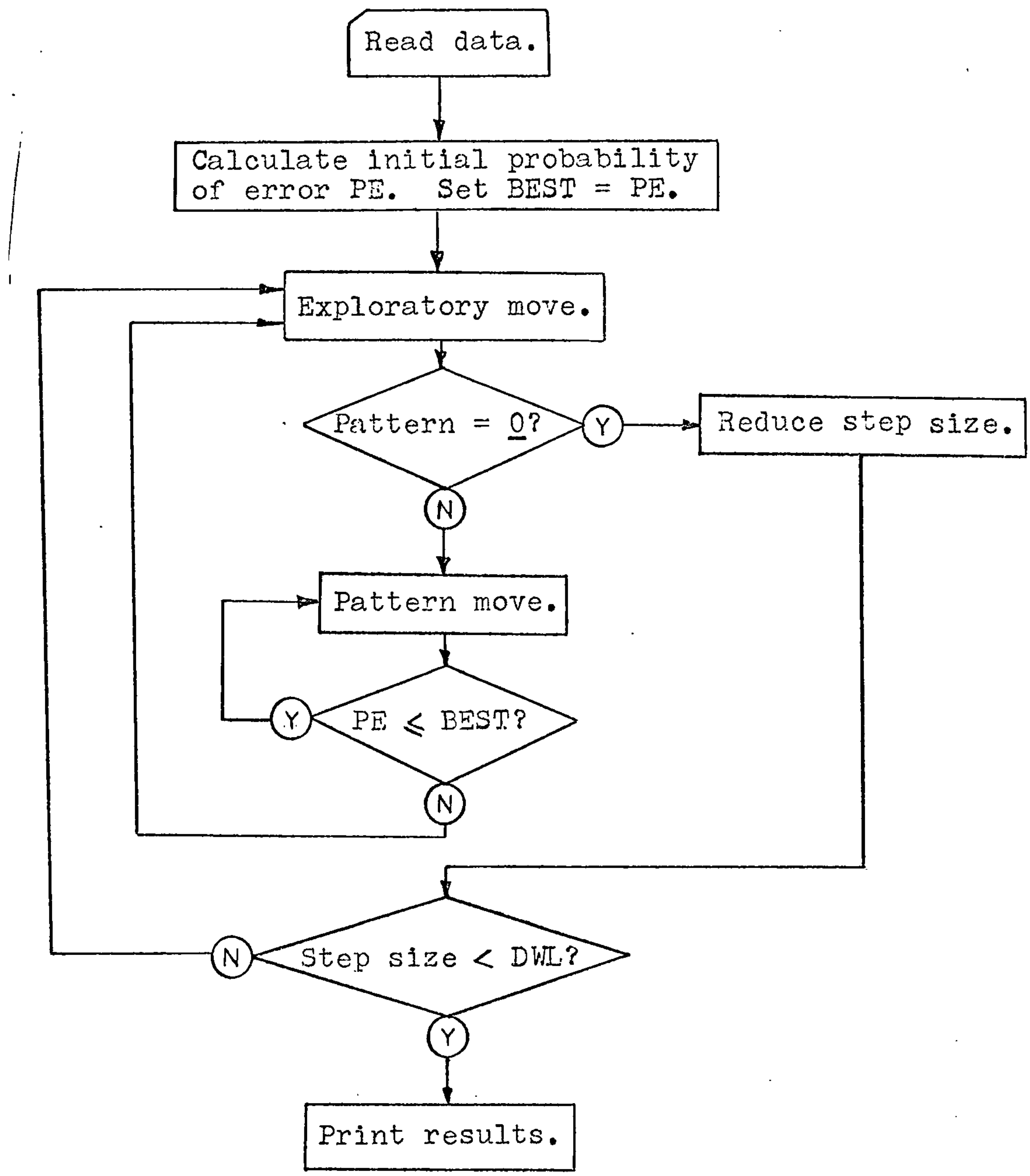
KA = Number of iterations

THRESH (WW, VARW, CORW, VARX, CORX, N, PE, IFN),
BEGIN (WW, VARW, CORW, VARX, CORX, N, PE, IFN) calculates the propability of error PE of an N-input threshold gate with weight vector WW and statistics VARW, CORW, VARX, and CORX. IFN is the number of points of the n-cube at which the probability of error is greater than 0.5. The initial call is made to BEGIN and subsequent calls are made to THRESH (see Appendix B).

THRESM (WW, VARW, CORW, VARX, CORX, N, 0, 2, PE),
BEGINM (WW, VARW, CORW, VARX, CORX, N, 0, 2, PE) calculates the probability of error PE of a two gate, N-input threshold network with weight vector WW and statistics VARW, CORW, VARX, and CORX. The initial call is made to BEGINM and subsequent calls are made to THRESM (see Appendix C).

SOURCE LANGUAGE: SDS Sigma 7 FORTRAN IV - H

Flow Diagram for Optimal Threshold Gate Search Program



```
C     OPTIMAL THRESHOLD GATE SEARCH PROGRAM
C     THIS PROGRAM WILL SEARCH TO FIND THE WEIGHTS WHICH WILL MINIMIZE
C     THE PROBABILITY OF ERROR OF AN N-INPUT THRESHOLD GATE WITH
C     STATISTICS VARW, CORW, VARX, AND CORX.  THE INITIAL WEIGHT VECTOR
C     IS W.  THE INITIAL STEP SIZE IS DWF, AND THE MINIMUM STEP SIZE
C     IS DWL.  THE THRESHOLD IS W(N+1). IFN IS THE NUMBER OF POINTS OF THE
C     N-CUBE FOR WHICH THE PROBABILITY OF ERROR > 0.5.
C     REF: THRESH
      DIMENSION INC(12), W(12), WW(12)
  1   READ (5,2,END=99) N
  2   FORMAT (I5)
      M = N + 1
      READ (5,5) (W(I), I = 1,M)
  5   FORMAT (8E10.3)
      READ (5,5) DWF, DWL
      READ (5,5,END=1) VARW, VARX
 10   CONTINUE
      READ (5,5,END=1) CORW, CORX
      CORX = 0.
      DO 11 I = 1,M
 11   WW(I) = W(I)
      DWM = DWF
C     PRINT INPUT DATA AND INITIAL PROBABILITY OF ERROR
      WRITE (6,12)
 12   FORMAT (1H1)
      WRITE (6,13) VARW,CORW,VARX,CORX
 13   FORMAT (/' INITIAL VARIANCES AND CORRELATIONS ',4(2X,E10.3))
      WRITE (6,14) DWM,DWL
 14   FORMAT (/' THE ITTERATION LIMITS ARE ',E10.3,2X,E10.3)
      WRITE (6,15) (WW(J), J = 1,M)
 15   FORMAT (/,' INITIAL WEIGHTS ARE ',/,(1X,10E13.6)/)
      NOW = 1
      DW = DWM
      DO 16 I = 1,M
 16   INC(I) = 0
      KODE = 0
      CALL BEGIN  (WW,VARW,CORW,VARX,CORX,N,PE,IFN)
      WRITE (6,17) PE
 17   FORMAT (/,' INITIAL PROBABILITY OF ERROR IS ',E15.8)
      BEST = PE
      KA = 1
      IF ( IFN .EQ. 0 ) GO TO 20
      WRITE (6,18) IFN
 18   FORMAT (/,' THE FUNCTION IS NOT REALIZED AT ',I4,' POINTS')
 20   CONTINUE
C     EXPLORATORY MOVE
      WW(NOW) = WW(NOW) + DW
      IF ( WW(NOW) .LT. 0. ) GO TO 29
      CALL THRESH (WW,VARW,CORW,VARX,CORX,N,PE,IFN)
      IF ( PE .LT. BEST ) GO TO 36
 29   WW(NOW) = WW(NOW) - DW
      DW = -DW
      KODE = KODE + 1
      IF ( KODE .EQ. 2 ) NOW = NOW + 1
```

```
      IF ( KODE .EQ. 2 ) KODE = 0
      IF ( NOW .GT. N ) GO TO 30
      GO TO 50
30    NOW = 1
      JG = 0
      DO 32 J = 1,N
32    JG = JG + IABS(INC(J))
      IF ( JG .EQ. 0 ) GO TO 52
      GO TO 38
36    BEST = PE
      IF ( KODE .EQ. 0 ) INC(NOW) = 1
      IF ( KODE .EQ. 1 ) INC(NOW) = -1
      NOW = NOW + 1
      KODE = 0
      DW = DWM
      IF ( NOW .GT. N ) GO TO 38
      GO TO 50
38    DW = DWM
      NOW = 1
C     PATTERN MOVE
      DO 40 J = 1,N
40    WW(J) = WW(J) + DW*FLOAT(INC(J))
42    KA = KA + 1
      DO 44 J = 1,N
44    IF ( WW(J) .LT. 0. ) GO TO 47
      CALL THRESH (WW,VARW,CORW,VARX,CORX,N,PE,IFN)
      IF ( PE .GE. BEST ) GO TO 47
      BEST = PE
      DO 45 J =1,N
45    WW(J) = WW(J) + DW*FLOAT(INC(J))
      GO TO 42
47    DO 48 J = 1,N
      WW(J) = WW(J) - DW*FLOAT(INC(J))
48    INC(J) = 0
      DW = DWM
50    KA = KA + 1
      GO TO 20
C     REDUCE STEP SIZE
52    DWM = DWM/10.
      IF ( DWM .LT. DWL ) GO TO 70
      DW = DWM
C     PRINT INTERMEDIATE RESULTS
      WRITE (6,65) KA,(WW(J), J = 1,M)
65    FORMAT(//,' AFTER ',I4,' ITTERATIONS,THE WEIGHTS ARE ',
     1  (1X,10E13.6))
      WRITE (6,67) BEST
67    FORMAT (/,' AND THE PROBABILITY OF ERROR IS ',E15.8,/)
      IF ( IFN .EQ. 0 ) GO TO 20
      WRITE (6,18) IFN
      GO TO 20
C     PRINT FINAL RESULTS
70    WRITE (6,74) KA
74    FORMAT (//,' AFTER ',I4,' ITTERATIONS,')
      WRITE (6,76) ( WW(J), J = 1,M )
```

```
76     FORMAT (/,' THE BEST WEIGHTS ARE ',(1X,10E13.6))
       WRITE (6,78) BEST
78     FORMAT (/,' THE LEAST PROBABILITY OF ERROR IS ',E15.8,////)
       IF ( IFN .EQ. 0 ) GO TO 10
       WRITE (6,18) IFN
       GO TO 10
99     STOP
       END


C      OPTIMAL THRESHOLD GATE SEARCH PROGRAM, TWO GATE
C      THIS PROGRAM WILL SEARCH TO FIND THE WEIGHTS WHICH WILL MINIMIZE
C      THE PROBABILITY OF ERROR OF A TWO-GATE THRESHOLD NETWORK WITH
C      N INPUTS AND STATISTICS VARW, CORW, VARX, AND CORX.  THE INITIAL
C      WEIGHT VECTOR IS W. THE THRESHOLD OF THE FIRST GATE IS W(N+1), AND
C      THE THRESHOLD OF THE SECOND GATE IS W(2*N+3).  W(2*N+2) IS THE
C      WEIGHT GIVEN THE OUTPUT OF THE FIRST GATE.  THE INITIAL STEP SIZE
C      IS DWF, AND THE MINIMUM STEP SIZE IS DWL.
C      REF: THRESM
       DIMENSION INC(12), W(12), WW(12)
 1     READ (5,2,END=99) N
 2     FORMAT (I5)
       NI1 = N
       NW1 = N + 1
       NI2 = N + 1
       NW2 = N + 2
       NWT = NW1 + NW2
       READ (5,5) (W(I), I = 1,NWT)
 5     FORMAT (8E10.3)
       READ (5,5) DWF, DWL
       READ (5,5,END=1) VARW, VARX
10     CONTINUE
       READ (5,5,END=1) CORW, CORX
       DO 11 I = 1,NWT
11     WW(I) = W(I)
       DWM = DWF
C      PRINT INPUT DATA AND INITIAL PROBABILITY OF ERROR
       WRITE (6,12)
12     FORMAT (1H1)
       WRITE (6,13) VARW,CORW,VARX,CORX
13     FORMAT (/' INITIAL VARIANCES AND CORRELATIONS ',4(2X,E10.3))
       WRITE (6,14) DWM,DWL
14     FORMAT (/' THE ITTERATION LIMITS ARE ',E10.3,2X,E10.3)
       M = NWT
       L = NWT - 1
       WRITE (6,15) (WW(J), J = 1,M)
15     FORMAT (/,' INITIAL WEIGHTS ARE ',/,(1X,10E13.6)/)
       NBW = 1
       DW = DWM
       DO 16 I = 1,M
16     INC(I) = 0
       KODE = 0
```

```
      CALL BEGINM (WW,VARW,CØRW,VARX,CØRX,N,0,2,PE)
      WRITE (6,17) PE
17    FØRMAT (/,' INITIAL PRØBABILITY ØF ERRØR IS ',E15.8)
      BEST = PE
      KA = 1
20    CØNTINUE
C     EXPLØRATØRY MØVE
      WW(NØW) = WW(NØW) + DW
      IF ( WW(NØW) .LT. 0. ) GØ TØ 29
      CALL THRESM (WW,VARW,CØRW,VARX,CØRX,N,0,2,PE)
      IF ( PE .LT. BEST ) GØ TØ 36
29    WW(NØW) = WW(NØW) - DW
      DW = -DW
      KØDE = KØDE + 1
      IF ( KØDE .EQ. 2 ) NØW = NØW + 1
      IF ( NØW .EQ. NW1 ) NØW = NØW + 1
      IF ( KØDE .EQ. 2 ) KØDE = 0
      IF ( NØW .GE. NWT ) GØ TØ 30
      GØ TØ 50
30    NØW = 1
      JG = 0
      DØ 32 J = 1,L
32    JG = JG + IABS(INC(J))
      IF ( JG .EQ. 0 ) GØ TØ 52
      GØ TØ 38
36    BEST = PE
      IF ( KØDE .EQ. 0 ) INC(NØW) = 1
      IF ( KØDE .EQ. 1 ) INC(NØW) = -1
      NØW = NØW + 1
      IF ( NØW .EQ. NW1 ) NØW = NØW + 1
      KØDE = 0
      DW = DWM
      IF ( NØW .GE. NWT ) GØ TØ 38
      GØ TØ 50
38    DW = DWM
      NØW = 1
      INC(NW1) = 0
C     PATTERN MØVE
      DØ 40 J = 1,L
40    WW(J) = WW(J) + DW*FLØAT(INC(J))
42    KA = KA + 1
      DØ 44 J = 1,L
44    IF ( WW(J) .LT. 0. ) GØ TØ 47
      CALL THRESM (WW,VARW,CØRW,VARX,CØRX,N,0,2,PE)
      IF ( PE .GE. BEST ) GØ TØ 47
      BEST = PE
      INC(NW1) = 0
      DØ 45 J =1,L
45    WW(J) = WW(J) + DW*FLØAT(INC(J))
      GØ TØ 42
      INC(NW1) = 0
47    DØ 48 J = 1,L
      WW(J) = WW(J) - DW*FLØAT(INC(J))
48    INC(J) = 0
```

```
      DW = DWM
50    KA = KA + 1
      GO TO 20
C     REDUCE STEP SIZE
52    DWM = DWM/10.
      IF ( DWM .LT. DWL ) GO TO 70
      DW = DWM
C     PRINT INTERMEDIATE RESULTS
      WRITE (6,65) KA,(WW(J), J = 1,NWT)
65    FORMAT(//'AFTER ',I4,' ITERATIONS, THE WEIGHTS ARE ',/,
     1  11(1X,F10.7))
      WRITE (6,67) BEST
67    FORMAT (/,' AND THE PROBABILITY OF ERROR IS ',E15.8,/)
      GO TO 20
C     PRINT FINAL RESULTS
70    WRITE (6,74) KA
74    FORMAT (//,' AFTER ',I4,' ITTERATIONS,')
      WRITE (6,76) ( WW(J), J = 1,NWT )
76    FORMAT (/' THE BEST WEIGHTS ARE ',/,11(1X,F10.7))
      WRITE (6,78) BEST
78    FORMAT (/,' THE LEAST PROBABILITY OF ERROR IS ',E15.8,/////)
      GO TO 10
99    STOP
      END
```