*VLSI DESIGN*
1998, Vol. 5, No. 4, pp. 385–392
Reprints available directly from the publisher
Photocopying permitted by license only



# Self Checking Design Technique for Numerical Computations

F. S. VAINSTEIN

*Department of Electrical Engineering, North Carolina A&T State University, 551 McNair Hall, Greensboro, NC 27411*

The objective of this paper is to develop an efficient method for testing of numerical computations based on algebraic concepts such as transcendental degree of field extensions.

A class of polynomially checkable functions is introduced, and for computation of the functions from this class a new method for error detection/error correction is proposed. This class of functions is shown to be large. The proposed method can also be extended to testing of computations of functions which are not polynomially checkable.

The preliminary results show great potential of this approach. In particular the proposed approach will lead to substantial reduction in hardware overhead required for multiple error detection and correction, as compare to the check sum method and other existing techniques.

*Keywords:* Testing, Error detection/error correction, Numerical functions, Fault tolerant computing, Algebraic methods

## 1. INTRODUCTION

Computation of numerical functions is a very common and widely used type of computing procedures. Testing of numerical computations, is, therefore, an important problem which up to now has not been satisfactorily solved.

By testing we mean here detection and, if possible, correction of errors resulting from different causes: software faults (bugs in the program), hardware design faults, or faults caused by technological imperfections, random failures, etc.

As far as we are aware of, there are four different approaches to error detection/correction in numerical computation and computer memories. The first of them treats all the data obtained by computation or stored in the memory as independent. Then the error control capability can be achieved by introducing a substantial space and/or time redundancy. This redundancy grows linearly with the size of the data array (for a fixed error density).

Typical methods of this kind are:

1) Replication with voting. This method, though the simplest one for implementation, require an extremely large redundancy[1].
2) Methods based on error-correcting codes (Hamming codes, Chinese remainder method, Check sum tests) (e.g.[2–4]). For the method developed in[2], a typical example requires the storage-space overhead of 40%.

The second approach makes use of the specific properties of the function to be computed. Since

this approach exploits the "hidden redundancy" of the values of the function itself, it, in principle, requires much smaller redundancy. The most advanced of previously known methods of this kind is liner checks for polynomials[5–7]. It is based on the properties of linear error-correcting codes. However, this method is limited to polynomial functions only, and is difficult for on-line implementation.

3) Methods based on the probabilistic concepts[10,11]. They deal primarily with faults in software systems.
4) Interval arithmetic methods[12–14]. They deal primarily with the errors introduced by the rounding in computations. It usually takes about as much computation as evaluation the function twice. This approach requires unacceptably high hardware redundancy (about 200%).

The proposed method which was first published in[9], is different from those mentioned above. It is based on certain algebraic concepts such as transcendental degree of field extension, and employs the specific structure of the function to be computed.

This method has important advantages, especially for the case of random errors: it requires small hardware redundancy (typically less than 5%), provides good fault coverage, and has very good fault location capability. It can be applied to check the computation of a very broad class of functions.

### 1.1 Example

Before outlying the general method let us consider an example.

*Example 1* Suppose we have to compute the function $f(x)=e^{-x}Sin5x$. Let $a_1, a_2 \in \mathbf{R}$. Denote by

$$f_0 = f(x+0) = e^{-x}Sin5x,$$
$$f_1 = f(x+a_1) = e^{-(x+a_1)}Sin5(x+a_1),$$
$$f_2 = f(x+a_2) = e^{-(x+a_2)}Sin5(x+a_2).$$

Denote by

$$p_1 = e^{-a_1}Cos5a_1; q_1 = e^{-a_1}Sin5a_1;$$
$$p_2 = e^{-a_2}Cos5a_2; q_2 = e^{-a_2}Sin5a_2;$$
$$A = p_1q_2 - p_2q_1; B = -q_2; C = q_1.$$

Then

$$Af_0 + Bf_1 + Cf_2 = 0 \text{ for every } x \in \mathbf{R}. \quad (1)$$

It is very important that $A, B$ and $C$ do not depend on $x$ and depend only on $a_1$ and $a_2$. Taking (1) into consideration we can consider the following method for **error detection**.

Denote the computed values of function $f$ at the points $x$, $x + a_1$, $x + a_2$ by $\tilde{f}_0$, $\tilde{f}_1$, $\tilde{f}_2$ respectively. Then if the computation is correct

$$A\tilde{f}_0 + B\tilde{f}_1 + C\tilde{f}_2 = 0 \text{ (\textbf{independently of} } x!) \quad (2)$$

### 1.2 Error Correction (single error)

Consider $a_1 = a$; $a_2 = 2a$ and let $A\tilde{f}_0 + B\tilde{f}_1 + C\tilde{f}_2 \neq 0$ because one of $\tilde{f}_i \neq f_i$; $i=0,1,2$. Suppose for example that $\tilde{f}_0 \neq f_0, \tilde{f}_1 = f_1, \tilde{f}_2 = f_2$. Then the correct value is given by the following formula

$$f_0 = -\frac{B}{A}\tilde{f}_1 - \frac{C}{A}\tilde{f}_2 \quad (3)$$

Location of the error can be obtained by using (2) for the following triples: $x-2a$, $x-a$, $x$; $x-a$, $x$, $x + a$; $x$, $x + a$, $x + 2a$.

It should be taken into consideration that computations are done in practice with a certain degree level of accuracy. Hence the formula (2) should be substituted by the formula $|A\tilde{f}_0 + B\tilde{f}_1 + C\tilde{f}_2| \leq \delta$, (2′) where $\delta$ is a small positive number specified by the precision of the computation.

### 1.3 Hardware Implementation

A general block diagram for the implementation of the proposed technique for our example is shown in Fig. 1. The address mapper in Fig. 1 provides the

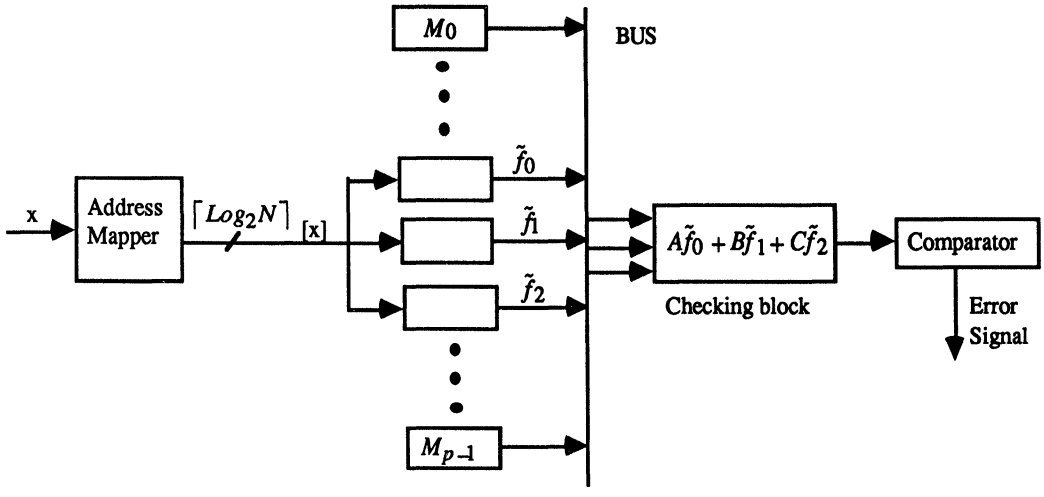


FIGURE 1 Block diagram for error detection.

address for the value of the function at point $x$. Let $N$ be the total size of the memory. The values of the function $f(x)$ are stored in $p$ different memory modules. Let $m$ be the size of each memory module, and $\Delta x$ be increment of $x$ i.e. the difference between two consecutive values of the argument $x$. Thus, $N = pm$, and $N\Delta x$ is the length of the domain of the function. We also require that $m\Delta x \leq a$ so that the values of the function at the points $x$, $x + a$, $x + 2a, \ldots$ be stored at different memory modules for simultaneous memory access. If $m\Delta x = a$, then the values of the function at the points $x$, $x + a$, $x + 2a$ will be stored in the same base address in different memory module. Therefore, the address can be divided in two parts: the line number and the module number.

$M_0, M_1, M_2, \ldots, M_{p-1}$ are the different memory modules. The checking block evaluates the number $e = A\tilde{f}_0 + B\tilde{f}_1 + C\tilde{f}_2$. An error signal is generated by the comparator if $|e| > \delta$. Error correction capability can be obtained by slight modification of this block diagram.

## 2. POLYNOMIAL CHECKING

For the readers convenience let us present the following definitions and results from the field extension theory[8].

DEFINITION 1 Let $K \subset L$ be a field extension and $K[T_1, \ldots, T_n]$ be the set of all polynomials in $T_1, \ldots, T_n$ over $K$. The elements $a_1, \ldots, a_n \in L$ are called algebraically dependent over $K$, if there exists a polynomial $P \in K[T_1, \ldots, T_n]$, $P \neq 0$, such that $P(a_1, \ldots, a_n) = 0$. The elements $a_1, \ldots, a_n \in L$ are called algebraically independent over $K$, if they are not algebraically dependent. By $K(T_1, \ldots, T_n)$ we denote the quotient field of the ring $K[T_1, \ldots, T_n]$.

*Example 2* Consider the field extension $Q \subset \mathbf{R}$. The numbers $\sqrt{2}$ and $\sqrt{3}$ are algebraically dependent over $Q$. $P(T_1, T_2) = T_1^2 + T_2^2 - 5$. The numbers $1, \pi \in \mathbf{R}$ are algebraically independent over $Q$.

DEFINITION 2 Let $K \subset L$ be a field extension. Transcendental degree (Tr. deg.) of this field extension is by definition the maximum possible number of elements from $L$ algebraically independent over $K$.

If Tr. deg. of $K \subset L$ is equal to $n$ and $m > n$, then any subset $\{a_1, \ldots, a_m\} \subset L$ is algebraically dependent.

*Example 3* Tr. deg. of $\mathbf{R} \subset \mathbf{R}(T)$ equals to 1.

Tr. deg. of $\mathbf{R} \subset \mathbf{R}(x, e^x)$ equals to 2.

Tr. deg. of $\mathbf{R}(x, Sinx, e^x) \subset \mathbf{R}(x, Sinx, Cosx, e^x)$ equals to 0.

DEFINITION 3 A field $L$ is called algebraically closed if any polynomial $P \in L[T]$ has a root in $L$.

*Example 4* $\mathbf{R}$ is not algebraically closed. $P(T) = T^2 + 1$ does not have roots in $\mathbf{R}$.

DEFINITION 4 A field $\bar{K}$ is called an algebraic closure of the field $K$ if

1) $\bar{K}$ is algebraically closed.
2) Tr. deg. $K \subset \bar{K}$ is equal to 0.

THEOREM 1 For any field $K$ its algebraic closure $\bar{K}$ exists and is unique up to isomorphism.

### 2.1 The Main Result

DEFINITION 5 A function $f: \mathbf{R} \to \mathbf{R}$ is called polynomially checkable (PC) if there exists an integer $k$, such that for any $a_1, \ldots, a_k \in \mathbf{R}$ the functions

$$f_0(x) = f(x+0),$$
$$f_1(x) = f(x+a_1), \ldots, f_k(x) = f(x+a_k)$$

are algebraically dependent, i.e. there exists a polynomial $P \in K[T_0, \ldots, T_k]$, $P \neq 0$ such that $P(f_0, \ldots, f_k) = 0$ (for any $x \in \mathbf{R}$). The polynomial $P$ is called a **checking polynomial** of the function $f$.

It is shown in[9] that the class of PC functions is very broad even for small $k$.

Computation of a PC function can be readily verified. For a given value of $x \in \mathbf{R}$, denote by $\tilde{f}_0$, $\tilde{f}_1, \ldots, \tilde{f}_k$ the values of $f$ at the points $x$, $x + a_1, \ldots$, $x + a_k$ respectively. Then, if all the values are computed correctly, the following equality holds:

$$P(\tilde{f}_0, \tilde{f}_1, \ldots, \tilde{f}_k) = 0 \qquad (5)$$

This property provides an unified approach to the problem of error detection/correction in computation of numerical functions. Indeed we can consider inequality similar to (2′)

$$|P(\tilde{f}_0, \tilde{f}_1, \ldots, \tilde{f}_k)| \leq \delta \qquad (5')$$

where $\delta$ is a small positive number specified by the precision of the computation. We have to note, however, that even if (5′) is satisfied it doesn't give us 100% warrantee that the computation is correct. There are some faults that cannot be detected by using (5′).

The first class of faults (we can call them software faults) are the result of the fact that some other PC function $g(x) \neq f(x)$ can have the same checking polynomial. For instance if $g(x) \neq f(x+b)$, where $b$ is a constant, then $g(x)$ and $f(x)$ have the same checking polynomials. Preliminary results, however, show the class of functions having the same checking polynomial is not big, which make it possible to fight the software faults. We are going to address this issue in the further publications.

The second class of faults which can not be detected by using (5′) are some of hardware faults. The hardware faults are the results of physical defects of a device which performs the computation of function. Random faults are hardware faults. The fault coverage of random faults is calculated below for an important case.

Denote by $S$ the set of three functions $x$, $Sinx$, $e^x$. Let $A \subseteq S$; denote by $\mathbf{R}(A)$ the field of all rational functions in $g_j \in A$ and by $\overline{\mathbf{R}(A)}$ its algebraic closure.

*Example 5* (a) $A = \{x\}, \mathbf{R}(A) = \{P_i(x)/Q_j(x)\}$, where $P_i$, $Q_j$ are polynomials of one variable with real coefficients. Its algebraic closure $\overline{\mathbf{R}(A)}$ includes as a special case any function $g(x)$ that is a solution of an equation $P_n(x)g^n + P_{n-1}(x)g^{n-1}(x) + \cdots + P_0(x) = 0$, where $P_i(x)$, $i = 0, 1, \ldots, n$ are polynomials of one variable with real coefficients. In particular, $\overline{\mathbf{R}(A)}$ includes the set of all functions that can be obtained by the application of finite number of additions, subtractions, multiplications, divisions, and raising to a rational power to the function $g(x) = x$. (b) $A = \{x, Sinx\}; \mathbf{R}(A) = \{P_i(x, Sinx)/Q_j(x, Sinx)\}$, where $P_i$, $Q_j$ are polynomials of two variable with real coefficients.

THEOREM 2 Let $f: \mathbf{R} \to \mathbf{R}$ belong to the field $\overline{\mathbf{R}(A)}$, $A \subseteq \{x, Sinx, e^x\}$. Then $f$ is polynomially checkable with $k = |A|$.

*Proof* We will prove the theorem for the case $A=\{x, Sinx, e^x\}$. For the other cases the proof is analogous.

Let $f(x) \in \overline{\mathbf{R}(x, Sinx, e^x)}$ and $a_1, a_2, a_3 \in \mathbf{R}$.

Denote

$$f_0(x)=f(x+0), f_1(x)=f(x+a_1),$$
$$f_2(x)=f(x+a_2), f_3(x)=f(x+a_3).$$

We have to show that $f_0, \ldots, f_3$ are algebraically dependent. This follows from the statements:

1) Tr. deg. of $\mathbf{R} \subset \mathbf{R}(x, e^x, Sinx)$ equals to 3.
2) For every $a \in \mathbf{R}$, $f(x+a) \in \overline{\mathbf{R}(x, e^x, Sinx)}$.

Indeed

$f(x) \in \overline{\mathbf{R}(x, Sinx, e^x)} \Longleftrightarrow$ there exists a polynomial $A \in \mathbf{R}(x, e^x, Sinx)[T]$, such that $A(f)=A_n(x)f^n(x) + \cdots + A_1(x)f(x) + A_0(x)=0$, where $A_i(x) \in \mathbf{R}(x, e^x, Sinx)$.

Let us denote $\varphi(x)=A_n(x)f^n(x) + \cdots + A_1(x)f(x) + A_0(x)$.

Then $\varphi(x+a)=A_n(x+a)f^n(x+a) + \cdots + A_1(x+a)f(x+a)+A_0(x+a), A_i(x+a) \in \mathbf{R}(x, e^x, Sinx, Cosx)$. Hence $f(x+a) \in \overline{\mathbf{R}(x, Sinx, Cosx, e^x)}$. But $\overline{\mathbf{R}(x, Sinx, Cosx, e^x)} = \overline{\mathbf{R}(x, Sinx, e^x)}$ and therefore $f_0, \ldots, f_3 \in \overline{\mathbf{R}(x, e^x, Sinx)}$. But the Tr. deg. of $\mathbf{R} \subset \overline{\mathbf{R}(x, Sinx, e^x)}$ equals to 3 and therefore $f_0, \ldots, f_3$ are algebraically dependent.

## 2.2 Corollary

Let $f$ be a result of application of a finite number of additions, subtractions, multiplication, divisions and raising to a rational power to the following functions: *Const*, $e^x$, $Sin(r_i x + b_i)$, $Cos(r_j x + b_j)$, where $r_i$, $r_j$ are rational numbers. Then $f$ is a PC function with $k \leq 3$.

*Example 6* The function

$$f(x)=\frac{Sin(x/11+\pi/7+e^x)^{3/5}x^2Cos^4x}{x^5+(x^4+x^2(Sin\,2x+xe^x)^3)^{2/7}}$$

is a PC function with $k=3$.

*Example 7* Consider the function

$$f(x)=\frac{((Sinx)^{1/3}+Cosx)^{1/2}}{Sin(x+7)-(Cos3x\,Sin(3x+5)-4)^{3/17}};$$

$f(x) \in \overline{\mathbf{R}(Sinx)}$.

Tr. deg. of extension $\mathbf{R} \subset \overline{\mathbf{R}(Sinx)}$ equals to 1, therefore $f(x)$ is a PC function with $k=1$.

*Note* The theorem 2 states that the class of PC functions is very big. We have to note, however, that the number of commonly used functions like $\ln(x)$, $Sin^{-1}(x)$, $Cos^{-1}(x)$ are not PC functions.

## 2.3 How to Find a Checking Polynomial

Suppose we are given a function $f: \mathbf{R} \to \mathbf{R}$. How to find a checking polynomial $P(T_0, \ldots, T_k)$ for it? The first step is to consider the set of all shifted functions $S(f)=\{f(x+a) | a \in \mathbf{R}\}$. Denote by $B(f)$ the smallest field of real functions having $S(f)$ as a subset and by $k$ the Tr. deg. $\mathbf{R} \subset B(f)$. If $k$ is finite then $f$ is a PC function and there exists a checking polynomial for it with the number of variables equal to $k+1$.

Now we have to find the coefficients of $P(T_0, \ldots, T_k)$. If we know the degree of this polynomial, then the coefficients can be easily obtained by the method of indefinite coefficients.

Indeed $P(f(x_i), f(x_i+a_1), \ldots, f(x_i+a_k))$ for every $x_i \in \mathbf{R}$. Hence this equality can be considered as a linear equation for the unknown coefficients of the polynomial $P(T_0, \ldots, T_k)$ for every $x_i \in \mathbf{R}$.

We can form a sufficient number of linear equations by choosing different $x_i \in \mathbf{R}$. Then the coefficients of the checking polynomial are found as the solution of this system.

*Example 8* Consider the function $f(x)=e^x/x$. For this case $S(f)=\{(e^{x+a}/x+a) | a \in \mathbf{R}\}$. Tr. deg. $\mathbf{R} \subset B(f)$ equals to 2, therefore $k=2$. (This follows from the corollary as well). We do not know the degree $d$ of the checking polynomial. Let us start with $d=1$. Consider $a_1=1$, $a_2=2$. We have

$$f_0(x)=\frac{e^x}{x}, f_1(x)=\frac{ee^x}{x+1}, f_2(x)=\frac{e^2e^x}{x+2}.$$

For $x_i = i$ we can consider the linear equations $Af_0(x_i) + Bf_1(x_i) + Cf_2(x_i) = 0$ with unknowns $A$, $B$, $C$. We obtain the following system of infinitely many equations:

$$Ae + B\frac{e^2}{2} + C\frac{e^3}{3} = 0$$
$$A\frac{e^2}{2} + B\frac{e^3}{3} + C\frac{e^4}{4} = 0$$
$$A\frac{e^3}{3} + B\frac{e^4}{4} + C\frac{e^5}{5} = 0$$
$$\vdots$$

Obviously, the system is inconsistent. Hence $d > 1$. Considering $d = 2$, after computations we can find the checking polynomial

$$P(T_0, T_1, T_2) = 2eT_0T_2 - T_1T_2 - e^2T_0T_1.$$

It is clear that any PC function has infinitely many different checking polynomials that form an ideal in the ring of polynomials. Determination of a checking polynomial of the smallest degree is still an open and challenging problem.

## 3. FAULT COVERAGE

Suppose for the simplicity that the values of the function are stored in a $b$-bit ROM. The function $f(x)$ is normalized in such way that $f(x) \leq 1 - 2^{-b}$ in the domain where we are going to detect the errors of the computation and the numbers are stored as fixed point numbers. It means that the number $y$ is given by a string $y = sy_1y_2 \ldots y_b$ where $s$ presents the sign of $y$ and $y_b$ is the least significant bit. In other words $y = sign(2^{-1}y_1 + 2^{-2}y_2 + \cdots + 2^{-b}y_b)$. Assume that for error detection/correction we use the formula

$$\left|\sum_0^k c_i\tilde{f}_i\right| \leq \delta; c_0 = 1 \qquad (6)$$

The value of $\delta$ depends on the function $f(x)$ and on the interval where it is computed.

If the function $f(x)$ is a PC function with $P(T_0, \ldots, T_k) = \sum_{i=0}^k c_iT_i$ then $\delta$ is defined only by the accuracy of the computation. In this case $\sum_{i=0}^k c_if_i = 0$. Denote by $\hat{f}(x)$ the stored value of the function at the point $x$ in a fault free ROM. Denote $\alpha_i = \hat{f}_i(x) - f_i(x)$. It is clear that $|\alpha_i| \leq 2^{-b}$. We can neglect the inaccuracy in $c_i, i = 0, 1, \ldots, k$ using $b + d$ bit adders and multipliers, where $d \geq 1$, for the calculation of $\sum_{i=0}^k c_if_i$. In case of correct computation

$$\sum_{i=0}^k c_i\tilde{f}_i = \sum_{i=0}^k c_i\hat{f}_i = \sum_{i=0}^k c_i(f_i + \alpha_i) = \sum_{i=0}^k c_i\alpha_i.$$

Therefore $\left|\sum_{i=0}^k c_i\tilde{f}_i\right| \leq \sum_{i=0}^k |c_i| \cdot |\alpha_i| \leq \sum_{i=0}^k |c_i| \cdot 2^{-b}$, and we can set

$$\delta = \sum_{i=0}^k |\, c_i \,| \cdot 2^{-b} \qquad (7)$$

For example for the function $f(x) = e^x$ we have $f_0 - e^{-1}f_1 = 0$ and therefore $\delta = 2^{-b}(1 + e^{-1})$.

Now consider the case when $f(x)$ doesn't have a checking polynomial of degree 1. In this case, we can neglect the effect of a limited accuracy, and the value of $\delta$ is obtained from the formula

$$\delta = \max\left|\sum_0^k c_if_i\right|, (x \in [a, b] - \text{the domain}). \qquad (8)$$

Let us now evaluate the portion of the faults detected by using the formula (6). In case of an error occurred at the point $x$ we have $\hat{f}(x) \neq \tilde{f}(x)$. Denote by $e_i = \tilde{f}_i - \hat{f}_i$, $i = 0, 1, \ldots, k$. The error is not detected if $|\sum_0^k c_i\tilde{f}_i| \leq \delta$.

We have $|\sum_0^k c_i\tilde{f}_i| = |\sum_0^k c_i(\hat{f}_i + e_i)| = |\sum_0^k c_i(f_i + \alpha_i + e_i)| = |\sum_0^k c_i(\alpha_i + e_i)|$, where $\alpha_i \leq 2^{-b}$. Hence the error is not detected if

$$\left|\sum_0^k c_i(\alpha_i + e_i)\right| \leq \delta \qquad (9)$$

Assume that the number $b$ is big enough and all the faults are equiprobable (we assume the latter only for the sake of evaluation of the portion of all undetected faults). Under this assumption we can consider $\beta_i = \alpha_i + e_i$ as independent random variables uniformly distributed in the interval $[-1, 1]$.

To proceed we need the following theorem.

THEOREM 3 Let $X$ and $Y$ be independent random variables and the density function $f_Y(y)$ take the maximum value at $y=0$. Then $f_Z(0) \leq f_Y(0)$, where $Z=X+Y$.

*Proof*

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(z-y) f_Y(y) dy;$$

$$f_Z(0) = \int_{-\infty}^{\infty} f_X(-y) f_Y(y) dy$$

$$\leq f_Y(0) \int_{-\infty}^{\infty} f_X(-y) dy = f_Y(0) \cdot 1 = f_Y(0)$$

The theorem is proved.

COROLLARY Let $X_1, \ldots, X_n$ be independent random variables such that for every $i=1, \ldots, n$ $f_{X_i}(x) \leq f_{X_i}(0)$, and $Z=X_1 + \cdots + X_n$.

Then $f_Z(z) \leq f_Z(0)$ and $f_Z(0) \leq \min_i f_{X_i}(0)$.

*Proof* The first statement is obvious. To prove the second statement assume that $f_{X_n}(0) \leq f_{X_i}(0)$ for every $i=1, 2, \ldots, n-1$. Denote by $X=X_1 + \cdots + X_{n-1}$ and by $Y=X_n$. Applying the theorem 3 we complete the proof.

Denote by $X_i=c_i\beta_i=c_i(\alpha_i+e_i)$, $i=0, \ldots, k$ and by $Z=X_1 + \cdots + X_k$. The error is not detected if $|Z| = |\sum_0^k X_i| \leq \delta$. Therefore the portion of undetected faults is equal to the probability

$$\text{Prob}(|Z| \leq \delta) \leq 2\delta \cdot (2 \max_i C_i)^{-1} = \frac{\delta}{\max_i C_i}.$$

We have proved the following theorem.

THEOREM 4 If we use the formula (6) for detecting errors in computation then the portion of undetected errors is smaller or equal than $\delta/\max_i C_i$.

*Example 9* Testing the computation of the function $f(x)=e^x$ by the formula $\tilde{f}_0 - e^{-1}\tilde{f}_1 = 0$ ($k=1$, $a_1=1$) we detect more than $(1-2^{-b+1})$ 100% of all faults.

*Example 10* Consider computation of the function $f(x)=Sinx$. Let $a_1=1$, $a_2=2$. It is easy to show that $f_0(x) - (Sin2/Sin1)f_1 + f_2 = 0$. Hence the formula $|\tilde{f}_0(x) - (Sin2/Sin1)\tilde{f}_1 + \tilde{f}_2| \leq \delta$ can be used for error detection. Using formula (7) we obtain the value of $\delta = 2^{-b}(1 - (Sin2/Sin1) + 1) < 3.1 \cdot 2^{-b}$. The portion of all undetected faults is greater of equal than

$$\left(1 - \frac{2^{-b} \cdot (1 + Sin2/Sin1 + 1)}{Sin2/Sin1}\right) \cdot 100\%.$$

For $b=32$ the portion of undetected faults is smaller or equal than $6.6 \cdot 10^{-10}$.

## 4. HARDWARE IMPLEMENTATION OF THE CHECKING POLYNOMIAL TECHNIQUE

A general block diagram for the implementation of the proposed method can be obtained by slight modification of the diagram presented in Fig. 1. Address mapper generates now $k+1$ addresses and the numbers $\tilde{f}_0, \tilde{f}_1, \ldots, \tilde{f}_k$ are sent to checking block via bus. The checking block evaluates the number $e=P(\tilde{f}_0, \tilde{f}_1, \ldots, \tilde{f}_k)$. An error signal is generated by the comparator if $|e| > \delta$. Error correction capability for the case when $\deg P=1$ can be obtained by slight modification of this block diagram.

## 5. APPLICATIONS

The proposed method of error detection/correction can be effectively used for on-line computation of numerical functions. It can be used for off-line acceptance-testing of programs in the course of development, or of devices in the course of manufacturing. For example, this approach can be used for testing a ROM containing the value $f(x)$ in cell whose address is $x$. In the case of testing a RAM we first have to choose $f(x)$, and then write in every cell the value $f(x)$ corresponding to its address $x$. After this we scan out the memory, verify the checks (according to inequality 5′) and detect or locate errors by analyzing the results of these checks.

This approach is applicable to stuck-at faults in ROM and RAM, stuck-at faults at the outputs of an address decoder, bridging faults between output

lines of the decoder and faults that affect power supply or read/write circuits.

An important feature of the method is that the required hardware and software overhead is limited to that necessary for computing the values of the checking polynomial. This computation can be performed by the same hardware system without any overhead. The method looks especially advantageous for the case when function values to be computed or stored has a large number of argument values. In case of on-line testing the time redundancy required for a error detection is $100/k \cdot t_1/t_2\%$, where $t_1$ and $t_2$ are the time intervals required for computing a single value of the checking polynomial and the function respectively, while error location needs $100 \cdot t_1/t_2\%$ time overhead. Moreover, when an error is located, it can be corrected by computing the root of the polynomial of one variable obtained from the checking polynomial by substituting $f_i$ for $T_i$ for those values of $i$ where computation is correct. This technique is especially simple and convenient when the checking polynomial is linear with respect to any variable $T_i$, in particular, when the polynomial is of degree 1 (see section 1.2).

The checking polynomial approach is a high-level functional technique which does not depend on the implementation of the program or the device computing the function $f(x)$.

### *Acknowledgment*

The author would like to thank Lev Levitin, Mark Karpovsky and Fadi Busaba for many useful suggestions.

### *References*

[1] Parag K. Lala, "Fault Tolerant & Fault Testable Hardware design" Prentice-Hall International, London (1985).
[2] G. R. Blakley and C. Meadows, "Security of Ramp Schemes", pp. 242–268 in the book "Advances in Cryptology: Proceedings of Cripto 84, Edited by G. R. Blakley and David Chaum (Volume 196 of Lecture Notes in Computer Science) Springer-Verlag. Berlin (1985).
[3] C. A. Asumth and G. R. Blakley, "Polling splitting and restituting information to overcome total failure of some channels of communication" Proc. of the 1982 Symposium on Security and Privacy, *IEEE Society*, pp. 156–169.
[4] Daniel P. Siewiorek and Robert S. Swarz, "The Theory and Practice of Reliable System Design", Digital Press (1982).
[5] M. G. Karpovsky, "Error Detection for Polynomial Computations", *IEEE J. Comput. & Digital Tech.*, **2**, (1), pp. 49–56 (1979).
[6] M. G. Karpovsky, "Testing for Numerical Computations", *IEEE Proc. E. Comput. & Digital Tech.*, **127**, (2), pp. 69–76 (1980).
[7] M. G. Karpovsky, "Detection and Location of errors by Linear Inequality Checks", *IEEE Proc.*, Vol. 129, Pt. E, No. 3 (1982).
[8] S. Lang, "Algebra" Addison-Wesley Publishing Co. (1992).
[9] F. S. Vainstein, "Error Detection and Correction in Numerical Computations by Algebraic Methods", Lecture Notes in Computer Science 539. Springer-Verlag (1991).
[10] M. Blum and S. Kannan, "Designing Programs That Check Their Work", 21st ACM Symposium on Theory of Computing, pp. 86–97 (1989).
[11] M. Blum, M. Luby and R. Rubinfeld, "Self-Testing and Self-Correcting Programs with Applications to Numerical Problems", 22nd ACM Symposium on Theory of Computing, pp. 73–83 (1990).
[12] R. E. Moore, Methods and Applications of Interval Analysis, SIAM, Philadelphia (1979).
[13] Alefeld and Herzberger, Introduction to Interval Computation, Academic Press, New York (1983).
[14] A. Neumaier, Interval Methods for Systems of Equations, Cambridge University Press (1990).

### *Author's Biography*

**Feodor S. Vainstein** received the M.S. degree in Electronics & Electrical Engineering from Moscow Institute of Electrical and Computer Engineering (USSR) in 1971 and M.S. in Applied Mathematics from the same Institute in 1974. In 1992 he received Ph.D. in Electrical and Computer Engineering from Boston University. In USSR he worked in a number research institutes and universities including Central Research Institute of Electronics, Moscow Institute of Electrical and Computer Engineering, Lenin Institute of Education and Likhachev Technological Institute.

In 1987 he emigrated to the USA and during 1988–1992 he taught at the Department of Electrical Computer and Systems Engineering and the Department of Computer Science of Boston University. Since 1992 he is an Associate Professor at the Department of Electrical Engineering of NC A&T State University.

Dr. Vainstein has over 20 years of research and teaching experience in various areas of engineering, computer science and applied mathematics. His interests include fault-tolerant computing, signal processing and applied mathematics. He has published about 30 papers.