where \( U_{\mu \alpha} \) is determined by [11] and \( J \) is determined by the linear relation
\[
W_{\psi \alpha} (I + T^\theta \alpha) \Phi = \Phi.
\]
(53)
Finally, we state this code capability as a theorem.

**Theorem 7**
The code given by the parity check matrix of (49) corrects all single random byte errors and all double-error-byte errors with pointers.

**Acknowledgment**
The authors wish to thank their colleagues Dr. D. C. Bossen, Dr. M. Y. Hsiao, and Dr. D. L. Ostapko for their encouragement and useful discussions in the course of this work; and Prof. R. T. Chien for a constructive review of the results and related discussions; in particular, he suggested the alternative construction of GMC using the fire codes.

**References**

**Error-Control Techniques for Logic Processors**

**DHIRAJ K. PRADHAN** and **SUDHAKAR M. REDDY**

---

**Abstract**—A new error-control technique for logic processors is given. The proposed technique uses Reed–Muller codes (RMC’s). The design scheme given has better efficiency than the schemes proposed earlier. The improved efficiency is obtained by relaxing a basic assumption originally made by Elias. Furthermore, it is shown that the efficiency of the proposed scheme asymptotically approaches the maximum efficiency achievable by a practical though restricted class of error-control schemes. Reliability of the proposed scheme is studied.

**Index Terms**—Error control, expurgated codes, Hamming code decoder, logic processors, modulo-2 sum of products form, Reed–Muller codes.

**I. Introduction**

In the past some researchers have suggested error-control techniques for logic processors in computers [1]-[6]. We will reformulate the problem and derive error-control techniques which are better than the ones derivable from earlier techniques.

For the sake of simplicity in comparing the proposed technique with the earlier techniques, we will assume a processor configuration originally suggested by Peterson and Rabin (shown in Fig. 1). The \( k \)-digit vectors \( X \) and \( Y \) are encoded into \( U(X) \) and \( V(Y) \), respectively. \( W(Z) \) is the encoded version of \( Z \), where \( Z = X \ast Y \). \( \ast \) is any one of the digit-by-digit Boolean operations. In the configuration of the processor shown in Fig. 1, the decoder is assumed to be completely reliable.

In [6] under the assumption that \( W(Z) \) is a single valued function of \( Z \), in the absence of errors in \( W(Z) \), it has been shown that if \( \ast \) is any nontrivial binary Boolean operation other than EXCLUSIVE OR or EQUIVALENT, then the length \( n \) of \( W(Z) \) has to be at least \( d \cdot k \), if \( \lfloor (d-1)/2 \rfloor \) errors in the output of the processor (i.e., \( W(Z) \)) can be corrected (\( \lfloor x \rfloor \) is the integer part of \( x \)). Equivalently, if we define \( k/n \) as the efficiency of the error-control scheme, the efficiency of this scheme is no better than the one obtained by replicating [1] the input vectors and processing them independently.

---

**Fig. 1.** Block diagram of a digit-by-digit processor.
We will assume that $W(Z)$ is not a single valued function of $Z$ (though for a given $W(Z)$, $Z$ is unique). Under this assumption we will show that it is possible to derive error-control schemes with higher efficiency than the results of [6] tend to indicate. We will derive these results by using what are known as Reed-Muller codes (RMC's) [9]. For the sake of completeness we will give a brief review of RMC's.

Reed-Muller Codes

Reed-Muller codes are binary linear codes; that is, an RMC is a subspace of the vector space of all binary $n$-bit vectors. An RMC can be described in terms of a generator matrix, the linear combinations of the rows of which over a field of two elements are the code words of RMC. The 0th-order RMC of length $2^n$ has a single row in the generator matrix. The $i$th order, $0 \leq i \leq m$, RMC is a linear code of length $2^m$ and is iteratively defined as below. The generator matrix of the 1st-order RMC is obtained by adding $m$ more rows to the 0th-order RMC generator matrix such that the columns of the generator matrix in the portion of the $m$ rows now added, are all the $2^m$ binary $m$-tuples.

Example 1: Let $m = 4$, the generator matrix of the 1st-order RMC of length 16 is given below:

$$G = \begin{bmatrix}
1111 & 1111 & 1111 & 1111 \\
1111 & 1111 & 0000 & 0000 \\
1111 & 0000 & 1111 & 0000 \\
1100 & 1100 & 1100 & 1100 \\
1010 & 1010 & 1010 & 1010
\end{bmatrix}$$

Definition: The rows of a generator matrix for linear codes are called the generators of the code.

Definition: The set of generators of a 1st-order RMC, excluding the generator of the 0th-order RMC is denoted by $S_1$ and the generator for the 0th-order RMC is denoted by $S_0$.

The generators of $i$th-order RMC are obtained by taking the union of $S_0$, $S_1$, $S_2$, ..., $S_{i-1}$, $S_i$, where $S_i = \{ s_{i} \circ s_{i-1} \circ \cdots \circ s_1 \}$ where $s_{i} \in S_i \ 1 \leq k \leq j$ and $s_{i} \circ s_{i-1} \circ \cdots \circ s_1$ is the vector obtained by taking the bit-by-bit of the elements of the $j$ vectors and $1 \leq j \leq i$, up to no larger than $m$. It is known that the set of vectors $\cup_{i=0}^{m} S_i$ are linearly independent [9] and hence the number of generators for $i$th-order RMC is $\Sigma_{i=0}^{m} \binom{m}{i}$.

Example 2:

$$m = 4$$

$S_0 = \{1111 \ 1111 \ 1111 \ 1111 \}$

$S_1 = \begin{bmatrix}
1111 & 1111 & 0000 & 0000 \\
1111 & 0000 & 1111 & 0000 \\
1100 & 1100 & 1100 & 1100 \\
1010 & 1010 & 1010 & 1010
\end{bmatrix}$

The generators of the 2nd-order RMC of length $2^4$ are $\{ S_0 \cup S_1 \cup S_2 \}$ and they are 11 in number.

It is known that the minimum Hamming distance between any two words of an $i$th-order RMC, $0 \leq i \leq m$, is equal to $2^{m-i}$. Therefore an $i$th-order RMC can be used to correct $[(2^{m-i}-1)/2]$ errors and detect $2^{m-i}/2 = 2^{m-i-1}$ errors.

Encoding: If $G$ is the generator matrix of a linear code with $k$ rows and $n$ columns, then a $k$-dimensional vector $X$ can be encoded (i.e., the corresponding code word can be obtained) by computing $XG$, where $X$ stands for matrix multiplication. Therefore, any code word generated by $G$ is a linear combination of the rows of the $G$ matrix. It is advantageous in implementation to derive the code word in what is known as systematic form.

Definition: A code generated by a matrix $G$ is said to be a systematic code if $XG = XP$, where $P$ is an $n-k$ vector containing the redundant bits.

It is well known that for any linear code an equivalent systematic code can be obtained by suitable linear operations on the rows of the generator matrix and/or rearrangement of the columns of the generator matrix.

II. An Upper Bound on the Efficiency of Some Schemes

In this section we would like to find an upper bound on the efficiency of some error-control schemes.

Let $B$ be the set $\{1, 0\}$ and let $B^j$ be the set of $j$-dimensional vectors. Referring to Fig. 1 we describe the various functions given there:

$$U: B^k \rightarrow B^k$$

$$V: B^k \rightarrow B^k$$

$$W: B^k \rightarrow B^k.$$

Let $U(X) = U_1(X)U_2(X)$, i.e., the concatenation of the vectors $U_1(X)$ and $U_2(X)$ of lengths $k$ and $(n-k)$, respectively, and similarly

$$V(X) = V_1(X)V_2(X),$$
and let $U_1(X) = V_1(X)$ for every $X \in B^k$. Furthermore let us assume the following.

**Assumption 1**: $U$ and $V$ are one-to-one into functions.

**Assumption 2**: $U_1$ and $V_1$ are one-to-one onto functions from $B^k$ onto $B^t$.

**Assumption 3**: $U_1(X_i) \circ U_1(Y_j) \neq U_1(X_k) \circ U_1(Y_l)$ if and only if $X_i \circ Y_j = X_k \circ Y_l$ for every $X_i$, $Y_j$, $X_k$, and $Y_l \in B^t$.

The above conditions are valid for linear codes where $X \circ Y$ can be obtained from $U_1(X) \circ U_1(Y)$ (in the absence of errors), for every $X, Y \in B^k$. For example, if $U$ and $V$ are represented in systematic form, then these conditions are obviously satisfied.

**Definition**: Let $\text{wht} (X)$ be the number of ones in $X$, then $d_h(X, Y) = \text{wht} (X \oplus Y)$ where $d_h$ is called the Hamming distance between $X$ and $Y$ and $\oplus$ is the bitwise EXCLUSIVE-OR operation.

**Theorem 1**: If $d_h(W(Z_1), W(Z_2)) \geq d$ for every $Z_1 \neq Z_2$, then $d_h(U_1(X_1), U_1(X_2)) \geq d-1$ for every $X_1 \neq X_2$.

**Proof**: Let $X_1$ and $X_2$ be such that $U_1(X_1)$ and $U_1(X_2)$ differ in the $i$th position. Let $Y \in B^t$ be such that $U_1(Y)$ is a $k$-dimensional vector with a one only in the $i$th position. $U_1(X_1) \circ V(Y)$ and $U_1(X_2) \circ V(Y)$ are such that $U_1(X_1) \circ V(Y)$ and $U_1(X_2) \circ V(Y)$ differ in exactly one position (the $i$th position). Furthermore from Assumption 3, $X_1 \neq X_2$ implies $X_1 \neq X_2$ and hence $d_h(U_1(X_1), U_1(Y)) \geq d$ by hypothesis. But $d_h(U_1(X_1), U_1(Y)) = 1$. Therefore $d_h(U_1(X_1) \circ V(Y), U_1(X_2) \circ V(Y)) \geq d-1$; i.e., $\text{wht}(V(Y) \circ (U_1(X_1) \oplus U_1(Y))) \geq d-1$; thus $d_h(U_1(X_1), U_1(X_2)) \geq d-1$ and therefore $d_h(U_1(X_1), U_1(X_2)) \geq d-1$.

**Corollary 1**: If for every $Z_1 \neq Z_2$, we have $d_h(W(Z_1), W(Z_2)) \geq d$, then $n \geq k + k_1$, where $k_1$ is the lower bound on the length of a code with $2^k$ code words and minimum Hamming distance $d-1$.

**Proof**: Corollary follows from Theorem 1, since $d_h(U_1(X_1), U_1(X_2)) \geq d-1$ for $X_1 \neq X_2$. Q.E.D.

**Corollary 2**: The maximum efficiency of any error-control scheme that can either detect or correct errors in $W(Z)$ and satisfies Assumptions 1–3 is $1/2$.

**Proof**: For any error-control scheme that can detect or correct errors in $W(Z)$, $d_h(W(Z_1), W(Z_2)) \geq 2$ whenever $Z_1 \neq Z_2$. Then from Corollary 1 we have $n \geq k + k$ and thus $k/n \leq k/2k = 1/2$. Q.E.D.

In the next section we will give an error-control scheme whose efficiency asymptotically approaches $1/2$ for fixed $d$.

### III. Design Technique

**Notation**: Let us designate $S_n = I$.

**Lemma 1**: If $X$ and $Y$ exist in an $i$th-order RMC then $X \ast Y$ exists in $2i$th-order RMC, where $\ast$ is any two variable Boolean function, performed bit-by-bit.

**Proof**: We can write $X \ast Y = C_1 \cdot I \oplus X \oplus C_1 \cdot Y \oplus C_2 \cdot X \circ Y$, where $C_1$ is a Boolean constant (0 or 1), $0 \leq i \leq 3$, and $\circ$ is the bit by bit EXCLUSIVE-OR operation and $\oplus$ is the bit by bit AND operation. (That this can be done follows from the fact that any two variable Boolean function has a Reed–Muller canonical expansion [10].) $X$ and $Y$ both are linear sum (bit-by-bit EXCLUSIVE OR) of the vectors of $S_1$ and of vectors which are obtained by taking bit-by-bit AND of at most $i$ vectors of $S_1$. Therefore $X \circ Y$ is a linear sum of vectors of $S_1$ and vectors that are obtained by taking bit-by-bit AND of at most $2i$ vectors of $S_1$ and hence $I, X, Y,$ and $X \circ Y$ all exist in the $2i$th-order RMC, so does $C_1 \cdot I \oplus C_1 \cdot X \circ C_1 \cdot Y \oplus C_1 \cdot X \circ Y$. Q.E.D.

Let the encoders in Fig. 1 be the encoders of a systematic $i$th-order RMC, $i \leq [m/2]$, and let the decoder be the decoder for a $2i$th-order RMC. Therefore $U(X)$ and $V(Y)$ exist in $i$th-order RMC and $W(Z)$ exists in $2i$th-order RMC.

**Theorem 2**: The design technique proposed can correct up to $[(2^{2i+1} - 1)/2]$ errors and detect $2^{2i+1} - 1$ errors in the output of the processor.

**Proof**: Proof is immediate from the fact that the maximum Hamming distance between any two words of a $2i$th-order RMC is equal to $2^{2i+1}$. Q.E.D.

We will illustrate this technique in the following example.

**Example 3**: Consider the $(8, 4)$ first-order RMC as described by the following generator matrix:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The generator matrix of the equivalent systematic code is given below:

$$G_{SS} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Consider bit-by-bit operation on the following two 4-dimensional vectors: $1101$ and $1010$. The encoded versions of these two vectors in the above code are $11010010$ and $10100011$, respectively. The bit-by-bit operation on these two code vectors will yield $11110011$, which is a valid code word in the second-order RMC of length 8.

As can be seen from the above example, if the code used is in systematic form then $X \ast Y$ appears as such in the first $k$ positions. Thus the computation of $X \ast Y$ is completed in the logic processor and the decoder corrects any error in the computation by using the information available from the rest of the $n-k$ digits.

### IV. Comparison

Let $m = 2l$, $l \geq 2$. Then the number of generators (which gives $k$) in an $(l-1)$th-order RMC is equal to
The minimum distance of the \(2(l-1)\)th-order RMC is equal to 4. Therefore we will have a scheme to correct single errors and detect double errors.

The efficiency of the proposed scheme is

\[
\frac{k}{n} = \frac{2^{m-1} - \binom{m}{m/2}}{2^m} = \frac{1}{2} - \frac{\binom{m}{m/2}}{2^{m+1}} = \frac{1}{2} \text{ as } n \to \infty.
\]

For the replication scheme to achieve minimum Hamming distance of 4 we have \(n = 4k\), therefore the efficiency is 1/4. The decoders of the proposed codes are also reasonably simple, since an \((m-2)\)th-order RMC is the single-error-correcting, double-error-detecting Hamming code.

As a specific example let \(n = 64\), then the efficiency of the proposed scheme is 22/64.

We have shown above that at least for \(d = 4\) the efficiency of the proposed scheme is asymptotically optimum. In the following we show that in general for fixed \(d\) the efficiency of the proposed scheme approaches 1/2.

Let \(d = 2^s\), \(s \geq 1\), \(n = 2^{2s}\), \(l \geq 1\) and then \(i\) for the proposed scheme will be \((l-s)\). The efficiency of the proposed scheme is

\[
k = \frac{2^{2s-1} - \left\{ \binom{2l}{l-s+1} + \binom{2l}{l-s+2} + \cdots + \binom{2l}{l} \right\}}{2^{2s}}
\]

\[
k = \frac{1}{2} \left\{ \frac{2^l}{l-s+1} + \frac{2^l}{l-s+2} + \cdots + \frac{2^l}{l} \right\}
\]

as \(n \to \infty\) the second term in the right-hand side tends to zero. Hence as \(n \to \infty\), \(k/n \to 1/2\). We can derive similar results when \(m\) is an odd integer.

**Theorem 3:** The efficiency of the proposed error-control scheme is asymptotically optimum for fixed \(d\).

**Proof:** For \(d = 2^s\) the result follows from the observations made above. For any \(d \neq 2^s\) there exists an \(s\) such that \(2^s > d\) and hence the proof. Q.E.D.

The improvement in efficiency \((k/n)\) of the proposed scheme over that given by Elias [5], Peterson and Rabin [6] is derived by not requiring \(W\) to be a one-to-one function. This has allowed increased efficiency \((k/n)\) but the decoders are more complex than the ones required for the replication schemes suggested earlier [1], [5], [6].

Though we have shown that the efficiency of the proposed scheme is asymptotically optimum, for finite values of \(k\) the efficiency falls below the one given by Corollary 2. For example for \(k = 22\) and \(d = 4\), \(n = 64\) for the proposed scheme where as the lower bound on \(n\) given by Corollary 2 is 49. There are two reasons why this may be happening. Firstly, the lower bound on \(n\) of Corollary 2 was derived for a processor performing single operation (AND) and secondly, even in the absence of errors all the code words of a 2\(th\)-order RMC do not appear as \(W(Z)\) and hence we may not be using the 2\(th\)-order code very efficiently. For example the 2\(nd\)-order RMC of length 64 has \(2^{22}\) words and hence there can be at most \(2^{22} \times 2^{22} \times 16 = 2^{48}\) different \(W(Z)\)’s. But there are \(2^{48}\) words in the 4\(th\)-order RMC. In the next section we will show that for given encoders and decoders the logic processor can be more general than the one given in Fig. 1.

**V. DISCUSSION**

In this section we will indicate some useful modifications of the design technique given in the last section.

We have seen that to perform bit-by-bit Boolean operations between two vectors we can use an \(i\)th-order RMC and the result will be in the \(2\)\(i\)-th-order RMC. If bit-by-bit Boolean function on \(l\) vectors is to be evaluated, then the result will be an \((li)\)-th-order RMC. This can be seen by noting that the modulo-2 sum-of-products form [10], [11] using EXCLUSIVE-OR and AND logic primitives for any \(l\) variable function will contain products of at most \(l\) literals. For example if the function \(Y_1 \circ Y_2 \circ Y_3 \circ Y_4 \circ Y_5 \circ Y_6\) is to be evaluated bit-by-bit, then the result will be in a \(2\)\(th\)-order RMC only and not in a \(3\)\(th\)-order RMC, if \(Y_1, Y_2, and Y_3\) exist in \(i\)\(th\)-order RMC, since the maximum number of literals in a product is 2.

In general a fault-tolerant logic processor which computes a bit-by-bit Boolean function on \(l\) vectors can be designed as schematically indicated in Fig. 2. The encoders are for \(i\)th-order RMC and the decoder is for \(r\)th-order RMC, \(r \leq li\). The value of \(r\) is determined by the largest number of literals in a product term of the canonical Reed-Muller expansion [10], [11] for the Boolean function computed by the processor. We know that any arbitrary logic function \(f(x_1, x_2, \ldots, x_n)\) has a unique Reed-Muller expansion as given below in (1)

\[
f(x_1, x_2, \ldots, x_n) = c_0 \oplus c_1 x_1 \oplus c_2 x_2 \oplus \cdots \oplus c_{2^n} x_n \oplus C_{n+1} x_1 x_n \oplus \cdots \oplus C_{2^{n-1} - 1} x_1 x_2 \cdots x_n,
\]

(1)

where \(x_1\) is either \(x_i\) or the complement of \(x_i\) and \(c_j\) is a binary constant (1 or 0). It is assumed that the \(x_i\) at the output of the processor are independently generated by \(l\)-input combinational networks. The \(l\) inputs are the \(i\)th outputs of \(l\) encoders. If the processor is computing all logic functions with Reed-Muller expansions containing products with no more than \(r\) literals then all the code words of the \((ri)\)-th-order RMC are possible as \(W(Z)\).

It may be observed that the design technique also yields...
asymptotically optimum efficiency for any function which can be expressed in modulo-2 sum-of-products form such that the product terms contain at most two literals.

We have seen above an example of a Boolean function over three vectors with the result contained in 2\textsuperscript{nd}-order RMC. By choosing a subset of the generators of the \textit{i}th-order RMC (the modified codes are called expurgated [12] codes) we can assume that the results are contained in an RMC of smaller order than when all the generators of the \textit{i}th order are chosen to form the code to encode the input vectors. Example 3 will show such a case.

\textbf{Example 3:} Let \( n = 8 \).

\[
S_0 = \begin{bmatrix}
1111 \\
1111 \\
0000 \\
\end{bmatrix}
\]

\[
S_1 = \begin{bmatrix}
1100 \\
1100 \\
\end{bmatrix} = \{X_1, X_2, X_3\}.
\]

Now consider the code formed by the two generators \( \{1111 1111, 1111 0000\} \). The code is contained in the 1\textsuperscript{st}-order RMC and is \( \{0000 0000, 0000 1111, 1111 0000, 1111 1111\} \). It can be readily seen that any Boolean function evaluated bit-by-bit over any number of vectors chosen from the above code will be a vector from the same code; i.e., the results are contained in the expurgated 1\textsuperscript{st}-order RMC [12]. We give a special application of this in Theorem 4 without proof.

\textbf{Theorem 4:} If a Boolean function is evaluated bit-by-bit on two vectors \( Y_1 \) and \( Y_2 \), both contained in an expurgated \( i \)\textsuperscript{th}-order RMC whose generators are given by \( S_0 \cup S_1 \cup \cdots \cup S_i \), where \( S_i' = \{X_j \circ X_1 \circ X_1, \cdots \}
\]

\( o X_1 X_2, X_3 \rangle \) and is fixed, \( X_{i+1} \in S_i, 1 \leq i \leq i-1 \) and \( X_{i+1} \neq X_j \}, \) then the results will be in \( (2i-1)\text{-order RMC.} \)

We next give an example to illustrate these ideas. In Fig. 3 the generator matrix of an expurgated 2\textsuperscript{nd}-order RMC of length 32 is given. In computing \( S_i' \) we have chosen \( X_1 \) in all pairwise products. The bit-by-bit \( \text{AND} \) of any two rows of the generator matrix \( G_{RSS} \) (which is a rearrangement of the columns of \( G_Z \) to make the code systematic and is given in Fig. 4) will have all zeros in the first ten columns, but different entries in the last 22 columns. Therefore for \( Z = (0000000000) \) there are several \( \text{W}(Z) \). The bit-by-bit Boolean operation on any two vectors from this code will yield a code vector in 3\textsuperscript{rd}-order RMC. The efficiency of the proposed scheme is 10/32 and the decoder need only be the decoder for single-error-correcting and double-error-detecting Hamming code.

It is appropriate to point out that the RMC are majority decodable [11]; that is, the decoders use majority gates and EXCLUSIVE-OR gates.

\section*{VI. \textbf{Estimation of Reliability}}

We will estimate the reliability of the processor designed by the proposed technique. The reliability of a component will be defined as the probability that it will
work over some specified time interval. In estimating the reliability we will assume the processor configuration given in Fig. 2 and use the following parameters.

$\lambda_1$ Reliability of the least reliable output lead of any encoder. This is the same for all the encoders as they are identical.

$\lambda_2$ Reliability of any single output lead of the processor which is the same for all the outputs as they realize identical switching functions and hence they are the outputs of identical combinational logic circuits.

d = 2t + 2 Minimum distance of the output code.

$R_1$ Reliability of l encoders.

$R_2$ Reliability of the processor.

$R_3$ Reliability of the decoder.

$R$ Reliability of the complete organization = $R_1 R_2 R_3$.

$$R_1 \geq \left[ \binom{n}{0} \lambda_1^n + \binom{n}{1} \lambda_1^{n-1}(1 - \lambda_1) + \cdots \right.$$

$$+ \binom{n}{t} \lambda_1^{n-t}(1 - \lambda_1)^t \left. \right]^t$$

$$R \geq \left[ \binom{n}{0} \lambda_2^n + \binom{n}{1} \lambda_2^{n-1}(1 - \lambda_2) + \cdots \right.$$

$$+ \binom{n}{t} \lambda_2^{n-t}(1 - \lambda_2)^t \left. \right]^t$$

$$\quad \left[ \binom{n}{0} \lambda_3^n + \binom{n}{1} \lambda_3^{n-1}(1 - \lambda_3) + \cdots \right.$$  

$$+ \binom{n}{t} \lambda_3^{n-t}(1 - \lambda_3)^t \right] R_3.$$  

As we see for given $\lambda_1$, $\lambda_2$, and $R_3$, $R$ can be increased arbitrarily up to maximum of $R_3$ (reliability of the decoder) by increasing $t$. But we know that for RMC to have arbitrarily large minimum distance we have to have arbitrarily large $n$. Therefore, we conclude that as in the von Neumann [1] or Moore–Shannon [2] schemes we have to have an arbitrarily large number of unreliable elements to achieve arbitrarily high reliability for the processor.

VII. Conclusion

A new and efficient error-control technique for the design of logic processors has been given. The proposed technique achieves better efficiency than earlier known techniques. Furthermore, it has been shown that the efficiency of the proposed scheme asymptotically approaches the maximum efficiency achievable by a class of practical schemes. We have extended our results to apply RMC to arithmetic processors. These results might be presented in another paper.

ACKNOWLEDGMENT

The authors wish to thank the referees for many helpful suggestions.

REFERENCES


