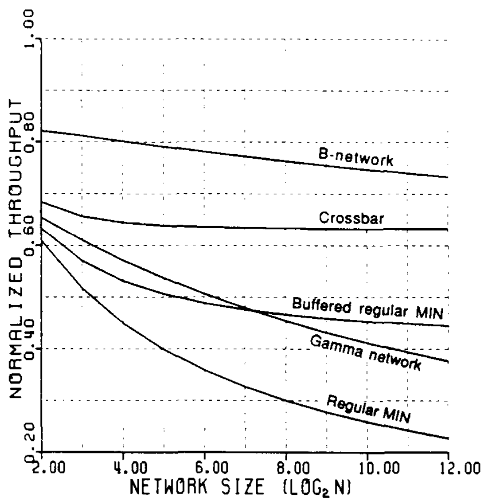

Fig. 5. Normalized throughput versus network size for various networks.

at the stage cycle i, $i > n + 1$:

$$\begin{aligned} m_n^i &= k(m_{n-1}^{i-1}) \\ m_j^i &= g(m_{j-1}^{i-1}, b_j^{i-1}), \qquad j = n-1, \cdots, 1 \\ m_0^i &= f(m, b_0^{i-1}) \\ b_{n-1}^i &= l(m_{n-1}^{i-1}) \\ b_j^i &= h(m_j^{i-1}, b_{j+1}^{i-1}), \qquad j = n-2, \cdots, 0. \end{aligned}$$

4) Numeric Results and Comparisons: The normalized throughputs of the *B*-networks of various sizes are shown in Fig. 5 together with those of other networks.[1] It shows that the normalized throughputs of the *B*-network are significantly higher than those of the crossbar, regular MIN's,[2] and the gamma network. The *B*-network's higher performance than the crossbar's is due to the elimination of memory conflicts via backward links at the last stage.

Fig. 5 also shows that the throughputs of the *B*-network are much higher than those of single-buffered regular MIN's. In buffered networks, the routing decision cannot be made locally at an SE, because the decision for a packet to move forward at one stage depends on the buffer states of all the remaining stages of the network [8], [7]. In the *B*-network, however, the routing decision is made locally at each SE. Moreover, each input (or output) port of an SE in buffered networks needs buffers as well as additional links to receive/pass the control signals (to prevent the overrun of the buffers). Considering these facts, the *B*-network can be a very cost-effective alternative to single-buffered networks.

IV. Conclusion

A new multistage interconnection network, the *B*-network, which employs backward links has been proposed and analyzed. A backward link provides an alternate path for a blocked request at a switch or memory. The bandwidth, the acceptance probability, and the throughput of the *B*-network have been analyzed. The *B*-network has been shown to exhibit much higher bandwidths and throughputs than the gamma network, single-buffered regular MIN's based on (2×2) switches, and the crossbar switches, while its hardware cost is the same as that of the gamma network. The *B*-network is controlled by the destination tag control scheme.

We were somewhat surprised by the performance results of the *B*-network exceeding those of the crossbar switch. The high performance is the combined effects of many causes: 1) the backward links function as implicit buffers, 2) the backward links at the very last switching stage of the *B*-network can handle memory contentions, which cannot be handled by the crossbar switch, 3) the original gamma network, on which the *B*-network is based, connects one network input point to a switch, effectively reducing the network input request rate.

[1] The values for the buffered regular MIN are from [8] and [11], and the values for the gamma network are from [11]. The buffered regular MIN denotes the single-buffered MIN based on (2×2) switches.

[2] MIN's such as the omega, the regular SW banyan, the baseline, and delta networks based on (2×2) switches.

References

[1] T. Feng, "A survey of interconnection networks," *IEEE Comput. Mag.*, vol. 14, pp. 12–27, Dec. 1981.

[2] C. Wu and T. Feng, Eds., *Tutorial: Interconnection Networks for Parallel and Distributed Processing*, IEEE Computer Society Press, 1984.

[3] D. S. Parker and C. S. Raghavendra, "The Gamma network," *IEEE Trans. Comput.*, vol. C-33, no. 4, pp. 367–373, Apr. 1984.

[4] H. Yoon and K. Y. Lee, "The B-network: A multistage interconnection network with backward links for multiprocessor systems," Tech. Rep. OSU-CISRC-TR22, Dep. Comput. Inform. Sci., Ohio State Univ.

[5] C. Wu and T. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-29, no. 8, pp. 694–702, Aug. 1980.

[6] J. H. Patel, "Performance of processor-memory interconnections for multiprocessors," *IEEE Trans. Comput.*, vol. C-30, no. 10, pp. 771–780, Oct. 1981.

[7] D. M. Dias and J. R. Jump, "Analysis and simulation of buffered delta networks," *IEEE Trans. Comput.*, vol. C-30, no. 4, pp. 273–282, Apr. 1981.

[8] Y. C. Jenq, "Performance analysis of a packet switch based on single-buffered banyan network," *IEEE J. Select. Areas Commun.*, vol. SAC-1, no. 6, pp. 1014–1021, Dec. 1983.

[9] C. P. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Trans. Comput.*, vol. C-32, no. 12, pp. 1091–1098, Dec. 1983.

[10] M. Kumar and J. R. Jump, "Performance of unbuffered shuffle-exchange networks," *IEEE Trans. Comput.*, vol. C-35, no. 6, pp. 573–578, June 1986.

[11] H. Yoon, K. Y. Lee, and M. T. Liu, "Performance analysis and comparison of packet switching interconnection networks," in *Proc. 1987 Int. Conf. Parallel Processing*, Aug. 1987, pp. 542–545.

[12] N. F. Tzeng, P. C. Yew, and C. Q. Zhu, "The performance of a fault-tolerant multistage interconnection network," in *Proc. 1985 Int. Conf. Parallel Processing*, 1985, pp. 458–465.

Analysis of Checksums, Extended-Precision Checksums, and Cyclic Redundancy Checks

NIRMAL R. SAXENA AND EDWARD J. McCLUSKEY

Abstract—**This paper presents an extensive analysis of the effectiveness of extended-precision checksums. It is demonstrated that the extended-precision checksums most effectively exploit natural redundancy occurring in program codes. Honeywell checksums and cyclic redundancy checks are compared to extended-precision checksums. Two's comple-**

Manuscript received November 8, 1987; revised August 12, 1988. This work was supported in part by the National Science Foundation under Grant MIP-8709128, in part by the Innovative Science and Technology Office of the Strategic Defense Initiative Organization and administered through the Office of Naval Research under Contract N00014-85-K-0600, and in part by Hewlett-Packard under the Honors Cooperative Program with Stanford University.

The authors are with the Center for Reliable Computing, Stanford University, Stanford, CA 93405.

IEEE Log Number 9035146.

0018-9340/90/0700-0969$01.00

ment, unsigned, and one's complement arithmetic checksums are treated in a unified manner. Results are also extended to any general radix-*p* arithmetic checksum. Asymptotic and closed form formulas of aliasing probabilities for the various error models are derived.

Index Terms—**Aliasing probability, checksums, cyclic redundancy check, generating functions, linear feedback shift registers.**

I. Introduction

Checksums are considered one of the least expensive methods of error detection. These methods are used for detecting errors in storage and transmission of data. System firmware or recovery routines in most of the commercial computer systems are checksum-protected [1]. The *checksum* of a block of words is formed by adding together all of the words in the block modulo-m, where m is arbitrary. These words could be data in memory or instructions in a checksum-protected code. The choice of m limits the number of bits in the checksum. This computed checksum validates the data or code integrity. Since checksums are essentially a form of compaction, error masking can occur. The metric used to quantify the effectiveness of checksums is the error masking probability (aliasing probability) under various error models.

Analysis of the effectiveness of *checksums* in [2] and [3] has largely been based on *unidirectional errors*. The detection of double and triple errors in low cost arithmetic codes has been analyzed in [4]. A closed form expression for the probability of checksum violation, assuming independent errors, appears in [5]. Error detection in serial transmission by arithmetic checksum, as an alternative to cyclic redundancy check (CRC), has been considered in [6]. The use of checksums is also considered in *watchdog processors* [7].

However, one problem that has not been examined in detail is the effectiveness of checksums under extended-precision arithmetic. Extended-precision checksums are described in [1], but their effectiveness has not been quantified. The effectiveness of extended-precision checksums as a function of the static distribution of instruction words in a program code is examined in Section III. The extended-precision checksums are effective when the static distribution of instructions is not uniform. Actual measurements on some firmware code show nonuniform distribution of instructions and it is believed that, in general, this will be true for other program codes. Results reported in [8] suggest this nonuniform distribution.

Restricted column errors or restricted word errors are analyzed. The motivation for this analysis is that some failures [2] in storage devices can be modeled as restricted column or word errors. The generating function presented in Section II provides a framework to analyze unsigned, two's complement, and one's complement arithmetic checksums. Results are extended to any general radix-p arithmetic checksums.

II. Generating Function

In this section, a generating function $f(X)$ is defined after the following definitions.

Definition 1: An n-bit word A_i is defined as a string of binary symbols $a_{j,i}$ and is denoted as $a_{n,i}a_{n-1,i}\cdots a_{1,i}$. The magnitude $|A_i|$ of A_i is given by

$$|A_i| = \sum_{j=1}^{n} 2^{j-1} a_{j,i}.$$

Definition 2: A length S column C_j is a column of binary symbols $a_{j,i}$ and is denoted by $[a_{j,1}, a_{j,2}, \cdots, a_{j,S}]^T$. The magnitude $|C_j|$ of C_j is given by

$$|C_j| = 2^{j-1}\sum_{i=1}^{S} a_{j,i}.$$

Definition 3: Block (n, S) is a block of S n-bit words A_i and is denoted as $[A_1, A_2, \cdots, A_S]^T$. Block (n, S) can also be defined in terms of columns. Block (n, S) is a block of n, length S, columns and is denoted as $[C_n, C_{n-1}, \cdots, C_1]$.

Definition 4: The checksum K of an (n, S) block is given by

$$K = \sum_{j=1}^{n} |C_j| = \sum_{i=1}^{S} |A_i|.$$

If the addition is performed without any loss of precision then K is the extended-precision checksum. $K \bmod 2^n$ and $K \bmod (2^n - 1)$ are two's complement and one's complement checksums, respectively.

Definition 5: $C(S, K, n)$ is defined as the number of possible distinct (n, S) blocks that have the same extended-precision checksum K.

The following example illustrates the use of a generating function in enumerating $C(S, K, n)$.

Example 1: Let $n = 2$, $S = 4$. Four 2-bit wide words are possible. The generating function in this case is

$$\begin{aligned} f(X) &= (1 + X + X^2 + X^3)^4 \\ &= X^{12} + 4X^{11} + 10X^{10} + 20X^9 + 31X^8 + 40X^7 + 44X^6 \\ &\quad + 40X^5 + 31X^4 + 20X^3 + 10X^2 + 4X + 1. \end{aligned}$$

There are four possible 2-bit (single-precision) checksums: 0, 1, 2, and 3. For example, the number of distinct ways of producing checksum 1 by adding 4 2-bit words is the sum of the coefficients of X^1, X^5, X^9. Adding the coefficients results in $4 + 40 + 20 = 64$, which is simply $2^{2\times 4-2}$. The same result is obtained for the other 2-bit checksum values. In the case of extended-precision, modulo-∞ addition is assumed. For example, the number of distinct ways of producing checksum 8 under extended-precision is the coefficient of X^8 which is 31.

Generalizing Example 1: The number of distinct (n, S) blocks, $A_1', \cdots, A_S'$, having the same checksum K is enumerated by the coefficient of X^K in $f(X)$. The function $f(X)$ is given by the following relation:

$$f(X) = (1 + X + X^2 + \cdots + X^{2^n-1})^S. \tag{1}$$

There are S identical factors, $f_{i=1,\cdots,S} = (1+X+\cdots+X^{2^n-1})$, in $f(X)$ corresponding to S words. The exponents of X in each factor represent 2^n distinct values $(0, \cdots, 2^n - 1)$ a word can assume. The exponent $\Sigma\alpha_i$ of the product of S terms, picking a term X^{α_i} from each f_i, will be the checksum of S words. Therefore, the coefficient, denoted by $C(S, K, n)$, of X^K in $f(X)$ will be the number of ways of obtaining the same checksum K.

The generating function $f(X)$ can also be written as

$$f(X) = \frac{(1 - X^{2^n})^S}{(1 - X)^S}. \tag{2}$$

Let k' be the number formed by considering only the n least significant bits of K. Thus, k' is a single-precision checksum and $0 \le k' \le 2^n - 1$. The number of distinct blocks of S words that produce the same checksum k' is the sum of the coefficients of $X^{k'}, X^{k'+2^n}, \cdots, X^{k'+\beta 2^n}$, where β is the greatest integer less than or equal to $(S(2^n - 1) - k')/2^n$. This number is evaluated by

$$\sum_{w=0}^{\beta} C(S, k' + w2^n, n). \tag{3}$$

Expression (3) can be evaluated by using a coset counting argument. Function $f(X)$ can be reinterpreted as

$$f(X) = (1 + X + \cdots + X^{2^n-1})^{S-1}(1 + X + \cdots + X^{2^n-1}).$$

Powers of X in $(1 + X + \cdots + X^{2^n-1})^{S-1}$ can be reduced to 2^n modulo-2^n residue classes: class(0), $\cdots$, class(i), $\cdots$, class($2^n - 1$). Class(i) contains all those terms of $(1 + X + \cdots + X^{2^n-1})^{S-1}$ such that the powers of X are i modulo-2^n. For every class(i) a unique term X^j in $(1+X+\cdots+X^{2^n-1})$ can be picked such that

$$X^j X^{i \bmod 2^n} = X^{k' \bmod 2^n}.$$

Since the classes exhaust all possible powers of X in $(1+X+\cdots+X^{2^n-1})^{S-1}$, the summation $\Sigma C(S, k'+w2^n, n)$ is simply the sum of the coefficients in $(1+X+\cdots+X^{2^n-1})^{S-1}$. The foregoing expression at $X = 1$ evaluates to the sum of the coefficients. Therefore, the following identities are obtained:

$$\sum_{w=0}^{\beta} C(S, k'+w2^n, n) = 2^{nS-n}. \tag{4}$$

From (4), the error masking probability P for single-precision checksums with modulo 2^n addition can be derived as

$$P = \frac{2^{nS-n}-1}{2^{nS}-1}. \tag{5}$$

In the case of single-precision checksums, it is clear that the masking probability is independent of the value of k' when all error patterns are possible. As shall be seen later, this will not be true for restricted classes of errors. Relation (5) also applies for two's complement arithmetic; however, for one's complement a different relation is obtained. The onset of data dependency as the checksum precision is increased from n-bits is illustrated in the next example. By data dependency, it is meant that the number of possible ways of obtaining a particular checksum not only depends on the type of checksum computation but also depends on the checksum value. This will be illustrated by Example 2.

Example 2: Let $S = 5$, $n = 2$. The generating function is

$$\begin{aligned} f(X) &= (1+X+X^2+X^3)^5 \\ &= X^{15}+5X^{14}+15X^{13}+35X^{12}+65X^{11}+101X^{10} \\ &\quad +135X^9+155X^8+155X^7+135X^6 \\ &\quad +101X^5+65X^4+35X^3+15X^2+5X+1. \end{aligned}$$

Following the approach in Example 1, it can be shown that the number of ways of obtaining the same 2-bit checksum is 256. Extending the precision of the checksum result by 1 bit, we have modulo-2^{n+1} addition. With this new precision, there are eight possible checksums: $0, \cdots, 7$. For example, the number of ways of producing checksum 7 is the sum of the coefficients of X^7 and X^{15}, which is 156 (155 + 1). The number of ways of producing checksum 3 is the sum of the coefficients of X^3 and X^{11}, which is 90 (35 + 65). It can readily be seen that the number of ways of producing $n+1$ bit precision checksum also depends on the checksum value.

III. Extended-Precision Checksums

It is easy to show that the checksum value for a block of S n-bit wide words cannot exceed $S(2^n-1)$. In a checksum-protected code, it is extremely unlikely that S would exceed 2^n. For instance, in a 32-bit computer ($n = 32$) it is very unlikely that the number of words in a checksum-protected code would exceed 2^{32}. Thus, almost always, only a $2n$-bit precision result is needed. Furthermore, only an n-bit precision adder and a counter to count the *overflow carries* from this n-bit adder is required. The counter is incremented whenever an *overflow carry* is generated by the n-bit adder while computing the checksum. Unbounded precision can be achieved by using an n-bit adder and a binary counter of variable length. The number of ways checksum K is preserved is simply the coefficient of X^K in the series expansion of $f(X)$. In this section, exact and asymptotic values of the coefficient of X^K are derived.

Rewriting (2)

$$\begin{aligned} f(X) &= (1-X)^{-S}(1-X^{2^n})^S \\ &= \left(\sum_{j=0}^{\infty} \binom{S+j-1}{S-1} X^j\right) \left(\sum_{q=0}^{S} \binom{S}{q} (-1)^q \left(X^{2^n}\right)^q\right) \end{aligned}$$

$$C(S, K, n) = \sum_{j=0}^{j \le \lfloor K/2^n \rfloor} \binom{S}{j} (-1)^j \binom{S+K-j2^n-1}{S-1}. \tag{6}$$

In Example 1, the coefficient of X^{10} is 10. The same result is obtained using (6). Substituting $S = 4$ and $n = 2$ in (6),

$$\begin{aligned} C(4, 10, 2) &= \binom{4}{0}\binom{13}{3} - \binom{4}{1}\binom{9}{3} + \binom{4}{2}\binom{5}{3} \\ &= 286 - 336 + 60 = 10. \end{aligned}$$

$C(S, K, n)$ now can be defined recursively. Rewriting (1),

$$f(X) = (1+X+\cdots+X^{2^n-1})^{S-1}(1+X+\cdots+X^{2^n-1}).$$

From the above factorization of $f(X)$, the following recurrence relations are obtained.

If $K > 2^n - 1$ then

$$C(S, K, n) = C(S-1, K, n) + \cdots + C(S-1, K-2^n+1, n). \tag{7}$$

If $K \le 2^n - 1$ then

$$\begin{aligned} C(S, K, n) &= C(S-1, K, n) \\ &\quad + C(S-1, K-1, n) + \cdots + C(S-1, 0, n). \end{aligned} \tag{8}$$

Some of the boundary conditions for (7) and (8) are as follows:

$$C(S, 0, n) = 1 \text{ for all } S > 0.$$

$$C(0, K, n) = 1 \text{ if } K = 0 \text{ else } C(0, K, n) = 0.$$

$$C(1, K, n) = 1 \text{ for all } 0 \le K \le 2^n - 1.$$

$$C(S, S(2^n-1), n) = 1.$$

The effect of K on error masking is not readily apparent from the exact relation (6), unless a simple closed form for $C(S, K, n)$ exists. It appears that there is no simple closed form expression for $C(S, K, n)$; in fact, a special case of this is an open research problem in [9]. Therefore, an asymptotic formula for $C(S, K, n)$ is derived. As a result of this analysis certain important observations with regard to checksum-protected codes are made.

A. Asymptotic Formula for $C(S, K, n)$

Functions of the type $f(X)$ belong to a class of functions called *unimodal functions*. The coefficients in the polynomial representation of these functions can be approximated by *Gaussian distribution density* functions. The following is an asymptotic approximation of $C(S, K, n)$

$$C(S, K, n) \approx \frac{\sqrt{6}\,2^{nS}}{\sqrt{S}\sqrt{\pi(2^{2n}-1)}} e^{\frac{-6\left(K-\frac{S}{2}(2^n-1)\right)^2}{S(2^{2n}-1)}}.$$

The following is a sketch of the derivation of the asymptotic formula: substituting $X = 1$ in $f(X)$,

$$f(1) = 2^{nS}.$$

Define $g(X) = f(X)/f(1)$; then $g(1) = 1$.

The function $g(X)$ has the characteristics of a probability generating function. The mean μ_x is given by $g'(1)$, and the variance σ_x^2 is given by $g''(1) - (g'(1))^2 + g'(1)$.

$$g'(X)]_{X=1} = \frac{1}{2^{nS}}[S(1 + X \cdots + X^{2^n-1})^{S-1} \cdot (1 + 2X \cdots + (2^n - 1)X^{2^n-2})]_{X=1}.$$

Simplifying the above expression,

$$\mu_x = \frac{S}{2}(2^n - 1). \quad (9)$$

Taking the second derivative of $g(X)$ and following the above procedure the variance σ_x^2 is obtained.

$$\sigma_x^2 = \frac{S}{12}(2^{2n} - 1). \quad (10)$$

The coefficient $C(S, K, n)$ can be approximated by (using normal distribution)

$$C(S, K, n) \approx \frac{\sqrt{6}2^{nS}}{\sqrt{S}\sqrt{\pi(2^{2n} - 1)}} e^{\frac{-6\left(K - \frac{S}{2}(2^n-1)\right)^2}{S(2^{2n}-1)}}. \quad (11)$$

The coefficient of X^9 in Example 2 can be estimated by (11) by substituting $S = 5$, $K = 9$, and $n = 2$. From expression (11), $C(5, 9, 2)$ evaluates to 136.48 which closely agrees with the exact value 135. Error masking probability closely relates to $C(S, K, n)$; therefore, it strongly depends on the value of K. This is evident from expression (11). Modest deviations of K from the mean value, $S/2(2^n - 1)$, can reduce error masking probability significantly. If the data are program code then the performance of checksums under extended-precision clearly depends on the static instruction distribution. The static instruction frequency can influence the checksum value K and therefore the error masking probability.

For extended-precision checksums to be effective it is desirable that the static distribution of instructions be nonuniform. Extensive study of the distribution of instructions in program codes of the various computer architectures has been done in [8]. Results in [8] show a nonuniform static distribution of high-level language statements. It is quite likely that instructions in the machine code of these high-level programs would also have nonuniform distribution characteristics.

From an information theoretic standpoint most of the program codes (at the machine instruction level) have inherent redundancy. This is so because not all instruction encodings are meaningful; for example, a 32-bit instruction may not have meaning for all 2^{32} encodings of the instruction word. Also, in most of the program codes some instructions occur more often than the others. An analogy can be drawn with the encoding of decimal digits. Again from an information theoretic standpoint only 3.3219 ($\log_2 10$) bits are required to represent decimal digits. However, the number of bits must be a whole number; therefore, 4 bits are chosen to represent decimal digits. This inherent or natural redundancy cannot be avoided if a regular representation of decimal numbers is desired. Sometimes this natural redundancy is well suited for error detection. In checksum-protected program codes, it is highly desirable to have checksum values to be far away from the mean value; this will decrease the error masking in extended-precision checksums significantly. Next, it will be shown how this natural redundancy in program codes can be exploited to further enhance error detection in extended-precision checksums.

A typical instruction word is a structured field having an *opcode*, *register*, and other *opcode-extension* fields. Assume that the opcode field is the most significant field in the instruction word. If opcodes are assigned such that

- an all zero opcode is assigned to that instruction which has, on the average, a high frequency of occurrence in program codes,

TABLE I

Checksum Measurements

Code	Code Length S	CheckSum Value	Expected Mean μ_x	Absolute Deviation in terms of σ_x
A	2558	$5.2343..\times10^{12}$	$5.493..\times10^{12}$	$4.128\sigma_x$
B	949	$2.237..\times10^{12}$	$2.037..\times10^{12}$	$5.213\sigma_x$
C	2617	$3.290..\times10^{12}$	$5.619..\times10^{12}$	$36.73\sigma_x$
D	2498	$3.167..\times10^{12}$	$5.364..\times10^{12}$	$37.93\sigma_x$
E	3140	$5.571..\times10^{12}$	$6.743..\times10^{12}$	$16.86\sigma_x$

- and increasing values of opcodes are assigned to instructions that, on the average, occur in decreasing order of the frequency of occurrence

then this will accomplish the task of moving the checksum value away from the mean μ_x. Likewise, compilers can be designed to allocate registers in the following manner: registers are allocated starting with the register which has the smallest allowed binary encoding (for example, register 0) and the rest of the registers are allocated in increasing order of their binary value.

In Table I, the measured checksum values of five different checksum-protected codes in the HP-9000 series/840 computers are listed. The instructions in these codes are based on the *HP Precision Instruction Set* [10]; these instructions are 32-bit wide. Therefore, $n = 32$ in this case. It is interesting to note that all the measured checksum values differ significantly from their respective mean values listed in the table. This would make the extended-precision checksums very effective. In fact, for code D, the extended-precision checksums would be most effective. The deviations of the measured values are given (Table I) in terms of the standard deviation. The deviations are considered significant if they are greater than $3\sigma_x$.

B. Equivalent CRC Length

One way to quantify the effectiveness of extended-precision checksums is to compare it to *CRC*. *CRC* in this case would be equivalent to multiple input signature analysis (MISA) [7]. It is known that for MISA with signature polynomial degree L the number of masking errors [11] is equal to 2^{nS-L}, when all error patterns are equally likely. *Masking errors* are those errors that escape detection. *Equivalent CRC length* L_e is defined similar to that discussed in [11]. L_e is the length of the MISA signature register that would mask the same number of errors as extended-precision checksums would for a given block of data. Following a procedure similar to that discussed in [11] and using expression (11),

$$L_e \approx n + \frac{1}{2}\log_2(\pi S) - 1.29 + 6(\log_2 e)\frac{\left(K - \frac{S}{2}(2^n - 1)\right)^2}{S(2^{2n} - 1)}. \quad (12)$$

L_e grows as the square of the difference between K and $S/2(2^n - 1)$. The number of bits required to store extended-precision checksums is at most $\lceil \log_2(S(2^n - 1) + 1) \rceil$. For example, in code C, L_e evaluates to 1010; the extended-precision checksum value for this code requires only 45 bits. Table II lists the equivalent lengths for the various codes.

For values of K close to $S/2(2^n - 1)$, *CRC* would perform better than extended-precision checksums.

Actual measurements on program codes do show significant difference between measured K and $S/2(2^n - 1)$.

Conceivably all the node signatures in *control flow checking* [7] could be replaced by extended-precision checksums. However, the tradeoff between the cost of adder and the cost of LFSR must be considered.

TABLE II
CRC EQUIVALENT LENGTH

Code	Code Length S	Equivalent Length L_e	Number of bits for extended-precision checksum
A	2558	50	45
B	949	57	44
C	2617	1010	45
D	2498	944	45
E	3140	242	46

C. Honeywell Checksums

The *equivalent length* measure is also useful in comparing the relative effectiveness of extended-precision checksums to *Honeywell checksums* [1], [2]. Honeywell checksums are a modified form of double-precision checksums. In Honeywell checksums, all pairs of successive words are concatenated and are treated as double-precision words. These double-precision words are summed to accumulate a double-precision checksum. This is equivalent to analyzing a single-precision type checksum where there are $S/2$ words and each word is $2n$ bits wide. For both S odd or even, the number of possible ways (assuming all possible error patterns) of obtaining the same Honeywell checksum is 2^{nS-2n}. This can be easily derived by using an approach similar to that developed in Section II.

The form of the number of masking errors in Honeywell checksums resembles the form of the number of masking errors, 2^{nS-L}, in MISA. Therefore, the *equivalent length* measure can be easily extended to Honeywell checksums. When L_e is less than $2n$, Honeywell checksums will be more effective than extended-precision checksums (for example, in Codes A, B). However, for cases where L_e is greater than $2n$, extended-precision checksums are more effective. It is important to note that a $2n$-bit adder is required to compute Honeywell checksums as opposed to an n-bit adder in extended-precision checksums. It is also important to note that the effectiveness measure developed in this section is dependent on the error model. Equally likely errors were considered for the L_e effectiveness measure.

If the *unidirectional-error* model [1], [2] is assumed then extended-precision checksums will be most effective because they guarantee complete coverage under this model. Honeywell checksums and MISA do not have complete coverage [1] with respect to this error model.

D. Incremental Precision Analysis

The effect of increasing the checksum precision on the masking probability is examined in this section. Let K be the checksum value assuming extended-precision, then under *incremental precision* the checksum value will be k', where k' is the least $n+\alpha$ significant bits of K. The number of distinct ways of preserving k' is given by the following expression:

$$\sum_{j=0}^{\beta} C(S, k' + j2^{n+\alpha}, n) \tag{13}$$

where β in this case is the greatest integer less than or equal to $(S(2^n-1)-k')/2^{n+\alpha}$. For large S, expression (11) can be used to evaluate $C(S, k'+j2^{n+\alpha}, n)$. In Example 2, $\alpha = 1$. Let us compute (13) using approximation (11) for $k' = 6$. Notice that the approximate value, 142.04, evaluated using (11) closely agrees with the exact value 141.

IV. ONE'S COMPLEMENT CHECKSUMS

Thus far, the results presented were for unsigned arithmetic checksums. In so far as two's complement checksums are concerned they are equivalent to unsigned arithmetic checksums. This is so, because in both cases addition is done in the same manner. The difference lies only in the way the checksum number is interpreted. However, in one's complement arithmetic, addition is modulo 2^n-1. These distinctions do not arise in extended-precision checksums because addition is done without loss of precision. In this section, an analysis for single-precision one's complement arithmetic checksums using the generating function $f(X)$ is presented. Let $Q(S, k')$ be the number of possible ways of producing the checksum k' in single-precision one's complement arithmetic for a block of S n-bit words. There are 2^n possible checksum values. In one's complement addition, the only way block of S n-bit words produce an all zero n-bit checksum is when all the S words are zero. This will become clear as Example 3 is discussed, later in this section. To enumerate $Q(S, k')$ for k' not equal to zero, a coset counting argument similar to that in Section II will be used. The generating function $f(X)$ can be factored as follows:

$$f(X) = (1 + X + \cdots + X^{2^n-1})^{S-1}(1 + X + \cdots + X^{2^n-1}).$$

Let $a(X) = (1 + X + \cdots + X^{2^n-1})^{S-1}$ and $b(X) = (1 + X + \cdots + X^{2^n-1})$. Therefore, $f(X) = a(X)b(X)$.

Powers of X in $a(X)$ can be reduced to $2^n - 1$ mod$-(2^n-1)$ residue classes: class(0), $\cdots$, class(i), $\cdots$, class(2^n-2). Class(i) contains all those terms of $a(X)$ such that the powers of X are i mod $-(2^n-1)$. For every class(i), a unique term X^j in $b(X) - X^{2^n-1}$ can be picked such that

$$X^j X^{i \bmod (2^n-1)} = X^{k' \bmod (2^n-1)}.$$

Note that X^{2^n-1} was not included in $b(X) - X^{2^n-1}$. Terms of the form $X^{k' \bmod (2^n-1)}$ can be found in $a(X)$. The sum of the coefficients of these terms will be $Q(S-1, k')$. Multiplying the left out X^{2^n-1} term by the foregoing terms $X^{k' \bmod (2^n-1)}$ in $a(X)$ will preserve their powers to $k' \bmod (2^n-1)$ and also the sum of their coefficients to $Q(S-1, k')$. Thus, the following recurrence on Q:

$$Q(S, k') = Q(S-1, k') + 2^{n(S-1)}. \tag{14}$$

Solving (14),

$$Q(S, k') = \frac{2^{nS}-1}{2^n-1}, \qquad \text{for } k' > 0. \tag{15}$$

$Q(S, k')$ is independent of checksum value when $k'$is greater than zero. However, $Q(S, 0) = 1$. The following example will illustrate the counting procedure for one's complement checksums.

Example 3: Let $S = 2$, $n = 2$. There are four possible checksums: 0, 1, 2, and 3. These checksums correspond to 2-bit patterns 00, 01, 10, and 11, respectively. The only way checksum 00 is obtained is by adding 00 and 00. However, checksum 11 can be produced in the following five distinct ways: $11+11 = 11$, $01+10 = 11$, $10+01 = 11$, $11+00 = 11$, and $00+11 = 11$. This is also enumerated by (15) which is 5 for $n = 2$ and $S = 2$. Enumeration for other checksum values can also be verified.

From (15) the probability of error masking with one's complement checksums is

$$\frac{1}{2^n-1} - \frac{1}{2^{nS}-1}, \qquad \text{for } k' > 0$$

when $k' = 0$, the error masking probability is zero.

V. WORD AND COLUMN ERRORS

This section examines the error masking probability for single-precision and extended-precision checksums under restricted errors. For restricted word errors, a straightforward extension of the results derived in the previous sections is possible. To analyze restricted column errors, the generating function $f(X)$ is useful.

A. Restricted Word Errors

Assume that only r specific words in a block of S words, $A_1, \cdots, A_S$, are in error. Extending previous results for single-precision checksums,

1) For unsigned arithmetic or two's complement checksums the error masking probability is $(2^{nr-n} - 1)/(2^{nr} - 1)$.

2) For one's complement checksum the masking probability is $1(2^n - 1) - 1/(2^{nr} - 1)$.

The following can be inferred from 1) and 2). All single word errors ($r = 1$) in two's complement or one's arithmetic checksums are detected. If the restricted r words have sum K_r under extended-precision the number of masking errors is obtained by substituting $S = r$ and $K = K_r$ in expression (11).

B. Restricted Column Errors

Let the n-bit pattern of word A_i be represented as $A_i = a_{n,i}a_{n-1,i}\cdots a_{1,i}$, where $1 \le i \le S$. For column t, the column sum c_t is

$$c_t = \sum_{j=1}^{S} a_{t,j}.$$

The checksum value K under extended-precision can be interpreted as

$$K = \sum_{j=1}^{n} 2^{j-1} c_j.$$

Analogous to this interpretation of K is the following form of generating function:

$$f(X) = (1 + X^{2^{n-1}})^S (1 + X^{2^{n-2}})^S \cdots (1 + X)^S.$$

The number of possible sums provided by column t corresponds to the factor $(1 + X^{2^t})^S$ in $f(X)$. Columns vary from 0 to $n - 1$. With this formulation of the generating function $f(X)$, restricted column errors can be analyzed very easily. Consider single column errors, the number of ways the two's complement checksum is preserved when errors occur in column t is given by the expression

$$\sum_{j=-h}^{\beta} \binom{S}{c_t + j2^{n-t}} \tag{16}$$

$\beta = \lfloor (S - c_t)/2^{n-t} \rfloor$ and $h = \lfloor c_t/2^{n-t} \rfloor$. Similarly, the number of ways in which the single-precision one's complement checksum is preserved when errors occur in column t is enumerated by

$$\sum_{j=-h}^{\beta} \binom{S}{c_t + j(2^n - 1)}. \tag{17}$$

Here $\beta = \lfloor (S - c_t)/(2^n - 1) \rfloor$ and $h = \lfloor c_t/(2^n - 1) \rfloor$. In two's complement checksum, error masking is worst for the $(n - 1)$th column. That is, the error masking probability for single column errors also depends on the column position t. This, however, is not true for one's complement checksums. Expression (17) is independent of the column position t. In extended-precision checksums the number of masking errors is given by

$$\binom{S}{c_t}. \tag{18}$$

If columns $t_m, t_{m-1}, \cdots, t_1$ are in error then the following results can be derived.

One's complement single-precision checksum: In this case, the number of masking errors is the sum of the coefficients of X^α, for all $\alpha = (\Sigma_{j=1}^{m} 2^{t_j - 1} c_{t_j}) \bmod (2^n - 1)$, in the generating function

$$(1 + X^{2^{t_m - 1}})^S (1 + X^{2^{t_{m-1} - 1}})^S \cdots (1 + X^{2^{t_1 - 1}})^S.$$

Two's component single-precision checksum: In this case, the number of masking errors is the sum of the coefficients of X^α, for all $\alpha = (\Sigma_{j=1}^{m} 2^{t_j - 1} c_{t_j}) \bmod 2^n$, in the generating function

$$(1 + X^{2^{t_m - 1}})^S (1 + X^{2^{t_{m-1} - 1}})^S \cdots (1 + X^{2^{t_1 - 1}})^S.$$

Extended-precision checksum: In this case, the number of masking errors is the coefficient of X^α, where $\alpha = \Sigma_{j=1}^{m} 2^{t_j - 1} c_{t_j}$, in the generating function

$$(1 + X^{2^{t_m - 1}})^S (1 + X^{2^{t_{m-1} - 1}})^S \cdots (1 + X^{2^{t_1 - 1}})^S.$$

VI. Other Extensions

In the foregoing sections, the checksums analyzed were for radix-2 number system. These results can be easily generalized for general radix-p arithmetic checksums.

The generating function $f(X)$ in radix-p can be generalized to

$$f(X) = (1 + X + X^2 + \cdots + X^{p^n - 1})^S \tag{19}$$

or with removable poles the form of $f(X)$ will be

$$f(X) = \frac{(1 - X^{p^n})^S}{(1 - X)^S}. \tag{20}$$

All the results obtained for radix-2 can be derived for radix-p by using various forms of (19) and (20). For example, an equivalent expression for (11) in radix-p would be

$$C(S, K, n) \approx \frac{\sqrt{6} p^{nS}}{\sqrt{S}\sqrt{\pi(p^{2n} - 1)}} e^{\frac{-6\left(K - \frac{S}{2}(p^n - 1)\right)^2}{S(p^{2n} - 1)}}. \tag{21}$$

VII. Conclusions

The generating function approach provides a unified framework to analyze various forms of checksums. The ability to express $f(X)$ in different forms has the mathematical elegance and the physical significance of directly relating to the error model. For example, the form of $f(X)$ presented in Section V-B directly relates to the column error model; in contrast, the form of $f(X)$ presented in the beginning of Section II directly relates to the word error model. It is not difficult to express $f(X)$ into a form that relates to a combination of word and column error model.

Extended-precision checksums most effectively exploit natural redundancy occurring in program codes. In addition, a clever encoding of instructions, by way of opcode assignments and register allocations, would further enhance the effectiveness of extended-precision checksums. Single-precision checksums, Honeywell checksums, and cyclic redundancy checks cannot, effectively, take advantage of this natural redundancy.

Acknowledgement

The authors would like to thank the anonymous referees for their valuable comments.

References

[1] D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*. Bedford, MA: Digital, 1982.

[2] Jack *et al.*, "Coverage analysis of self-check techniques in semiconductor memories," Honeywell Corp. Tech. Rep. MR12399, Minneapolis, MN, 1975.

[3] A. M. Usas, "Checksum versus residue codes for multiple error detection," in *Proc. FTCS-8*, 1978, p. 224.

[4] J. Wakerly, *Error Detecting Codes, Self-Checking Circuits and Applications*. New York: North-Holland, 1978.

[5] S. C. Tzou Chen and G. S. Fang, "A closed-form expression for the probability of checksum violation," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, no. 7, pp. 407-410, July 1980.

[6] J. G. Fletcher, "An arithmetic checksum for serial transmissions," *IEEE Trans. Commun.*, vol. COM-30, no. 1, pp. 247-252, Jan. 1982.

[7] A. Mahmood and E. J. McCluskey, "Concurrent error detection using

watchdog processors—A survey," *IEEE Trans. Comput.*, vol. C-37, no. 2, pp. 160–174, Feb. 1988.

[8] J. C. Huck, "Comparative analysis of computer architectures," Stanford Ph.D. dissertation, Stanford, CA, Mar. 1983.

[9] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, Research Problem 7.56. Reading, MA: Addison-Wesley, 1988, p. 366.

[10] HP 3000/930 and HP 9000/840 Computers, *Precision Architecture and Instruction Reference Manual*, Hewlett-Packard Co., 2nd ed., June 1987.

[11] N. R. Saxena and J. P. Robinson, "Accumulator compression testing," *IEEE Trans. Comput.*, vol. C-35, no. 4, pp. 317–321, Apr. 1986.

Markovian Queueing Network Models for Performance Analysis of a Single-Bus Multiprocessor System

IBRAHIM H. ÖNYÜKSEL AND KEKI B. IRANI

***Abstract*—An exact solution for the performance analysis of a typical single-bus multiprocessor system is presented. The multiprocessor system is modeled by a Markovian queueing network. An r-stage hypoexponential distribution or an r-stage hyperexponential distribution is used to represent the nonexponential service times. Consequently, the equilibrium probabilities of two-dimensional Markov chains are expressed by simple recurrence relations. Processing efficiency is used as the primary performance measure. To investigate the effects of different service time distributions on the system performance, comparative results have been obtained for a large set of input parameters. The numerical results illustrate that processing efficiency attains its maximum value for a constant (deterministic) service time, and if the service time of the common memory is hypoexponentially distributed, then approximating the service time by an exponential distribution produces less than 6% error on the system performance.**

***Index Terms*—Markovian queueing networks, multiprocessor systems, nonexponential service times, performance modeling and analysis, processing efficiency.**

I. Introduction

In early multiprocessor systems, a crossbar switch was used to connect the processors to the common memory. For example, a widely known crossbar system is C.mmp multi-minicomputer [1]. The performance of a crossbar multiprocessor system has been analyzed in recent years [2]–[5]. However, crossbar interconnection networks are becoming less and less interesting due to their complexity and their cost. Recent proposals and implementations show that a more attractive alternative would be a bus-structured interconnection network [6], [7].

The performance of bus-oriented multiprocessor systems has been studied by many researchers. Fung and Torng [8] developed a deterministic model for the analysis of memory contention and bus conflicts in multiple-bus multiprocessor systems. Goyal and Agerwala [9] proposed two generic classes of multiple-bus systems, and they analyzed the performance of these systems using the independence approximation method introduced by Hoogendoorn [4]. Ajmone Marsan *et al.* [10] examined four different single-bus multiprocessor architectures, and they showed how to build analytic models for the performance analysis—exponential assumptions were introduced to simplify the analysis of these models. Irani and Önyüksel [11] extended the analysis to multiple-bus systems, and they obtained an exact, closed-form solution by using a Markovian queueing network model.

The contents of this paper are twofold. First, a simple set of recurrence relations for equilibrium probabilities for the closed queueing network model with hypo- or hyperexponential service times are obtained, and then these results are used to analyze the performance of a typical single-bus multiprocessor system (Fig. 1). The architecture of the single-bus multiprocessor system is characterized by a common memory external to all processors and accessible only through a time-shared global bus. Each processor has its own local memory and processors execute segments of programs in their local memories until they need to access the common memory. After issuing a request for the global bus, a processor may have to wait for the bus to become available.

Section II presents an $M/G/1//N$ queueing network model for the performance analysis of a typical single-bus multiprocessor system, and then in Section III, this model is analyzed for hypo- and hyperexponential service times. Finally, in Section IV, the effects of the input parameters on the multiprocessor system's performance are discussed by several numerical examples.

II. Performance Modeling

Queueing theory has been a widely recognized major tool for computer performance modeling and analysis of multiprocessor computer systems [12]–[14]. Because of its elegant mathematical structure and because it can be easily applied to a large class of computer systems, a queueing model is usually preferred to a simulation model.

A single-bus multiprocessor system can be modeled by a closed queueing network to analyze its performance. This model is known as *the machine repair queueing system with one repairman* [15], [17]. Each machine in the model corresponds to a processor in the multiprocessor architecture and the repairman represents the common memory. The population of potential customers of this queueing system consists of N identical devices, each of which has an operating time of O time units between breakdowns. If O has an exponential distribution, then this system can be represented by an $M/G/1//N$ queueing network model.

The simplest case in an $M/M/1//N$ model. Its steady-state analysis can be found in [17]. Although the exponential assumption considerably simplifies the analysis, it may not be very realistic for some systems. Therefore, to extend the application area, this assumption is relaxed in this paper.

The following assumptions are made for the queueing model of the single-bus multiprocessor system.

1) When a processor requests access to the common memory, a connection is immediately established between the processor and the common memory, provided that the global bus is available for connection—only one processor can access to the common memory at each point in time.

2) A processor cannot have another request if its present request has not been granted.

3) The duration between the completion of a request and the generation of the next request to the common memory is an independent, exponentially distributed random variable with the same mean value of $1/\lambda$ for all processors.

4) The duration of an access by a processor to the common memory is an independent random variable with the mean value of $1/\mu$.

The goal of the analysis of the queueing network model is to determine the values of a performance measure for a given set of input parameters. A performance measure is merely an index which can be used to represent the performance of a system. In this paper, *processing efficiency* (PE), which is equal to the expected value of the percentage of ACTIVE processors, is used as a direct measure

Manuscript received November 12, 1987; revised October 28, 1988.

I. H. Önyüksel is with the Department of Computer Science, Northern Illinois University, DeKalb, IL 60115.

K. B. Irani is with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 9035145.

0018-9340/90/0700-0975$01.00