TU/e technische universiteit eindhoven

Encoding Efficiency Bounds for Digital Number Representations Under Value Deviation Constraints

Phillip Stanley-Marbell

ES Reports
ISSN 1574-9517

ESR-2008-02
24 January 2008

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems

People who are really serious about software should make their own hardware. — Alan Kay

http://www.es.ele.tue.nl/esreports
esreports@es.ele.tue.nl

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems
PO Box 513
NL-5600 MB Eindhoven
The Netherlands

Encoding Efficiency Bounds for Digital Number Representations Under Value Deviation Constraints

Phillip Stanley-Marbell
Technische Universiteit Eindhoven
Den Dolech 2, 5612 WB Eindhoven, The Netherlands

Abstract— **When logic upsets affect machine words representing variables in programs, it is possible to consider their effect in terms of the *numeric magnitude* of the *value deviation* they induce. This is a result of the semantic interpretation that the bit-level layout of different language data types imposes.**

When some *distribution* of value deviations in variables may be acceptable, it is possible to consider techniques for encoding the representation of program variables, to limit the deviations of values in the presence of logic upsets, to acceptable distributions thereof. To quantify the best-case performance of encoding techniques for minimizing such value deviations, we investigate bounds for their encoding efficiency.

I. Introduction

Disruptions of machine state due to phenomena external to the computing process, is of increased interest in contemporary computing systems. Advances in semiconductor processes, while yielding ever greater availability of computing resources, have been accompanied by increased susceptibility of circuits to undesirable disruptive phenomena. These phenomena include on-chip power supply fluctuations, system-level electrical noise, and energetic species such as alpha particles and high-energy neutrons, in both terrestrial and space applications. In the ever-more-complex systems being built today, there is also increased concern for the integrity of data being communicated between multiple subsystems of an integrated circuit (IC), as well as between multiple ICs in a system.

In digital systems, the effects of these varied disruptive mechanisms may be abstracted by *logic upsets* or *faults* — undesirable changes in bit-level state. These faults may be seen as forcing a logic value in a circuit to a high (`1`) or low (`0`) value (possibly leading to `0`→`1` or `1`→`0` *inversion-upsets*). The upset phenomena may also lead to ostensibly invalid values, i.e., to logic states that are neither `0` nor `1`: *erasure-upsets*. For inversion-upsets, if the pre-existing value within the circuit is the same as the value induced, the fault is said to be *masked*.

It is typically the goal of forward error correction mechanisms to reduce the probability of logic upsets going undetected or uncorrected. For a given error detection and correction mechanism, there is an associated number of detectable or correctable logic upsets; for example, a single error correction, double error detection (SECDED) system will correct single faults and detect up to two faults. In systems employing error correction mechanisms of this form, no differentiation is made between the different cases of undetected or uncorrected faults, even though they might have differing effects on system behavior.

When faults affect digital representations of *numbers*, the *errors* induced may be considered from the viewpoint of the deviations in numeric value induced. In the case of microprocessors, the faults occurring in machine words (e.g., in registers, buses, memory cells) may be considered in terms of the deviations they introduce into program variables. Given a fault in a machine word, the numeric magnitude of deviation is dependent on the machine-level representation of values, e.g., unsigned or two's complement representation for integers, and IEEE-754 floating-point representation for approximate real numbers [4].

Variables of arithmetic types in programs (e.g., types `int` or `float` in the C programming language) can however often tolerate some amount of deviation in their values. For example, variables representing components of pixel values in an image-processing application might be tolerant of small variations resulting from faults. For such situations, it is possible to view the problem of forward error correction, from the perspective of the *guaranteed maximum value deviation*, rather than in terms of the *guaranteed maximum number of correctable faults*. For example, rather than attempting to find an encoding that will correct k faults, the problem might instead be to find an encoding that ensures that the value deviation is less than a constant C. This is essentially encoding under a fidelity criterion [7], based on the semantics of digital arithmetic representations of programming language variables of different data types.

In this paper, we investigate the bounds on attainable efficiency of encoding as a function of a constraint on value deviation, for encoding techniques that *satisfy*

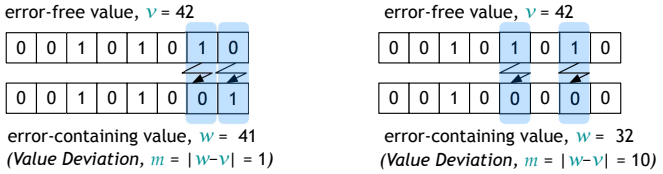

Fig. 1. Example: logic upsets and value deviations.

value deviation constraints on variables in programs. These bounds are analogous to the Shannon efficiency bound for encoding techniques *for correcting a fixed number of faults*. Relevant related research is reviewed in Section II. The bound derivations are presented in Section III, and the paper concludes in Section IV with a summary of contributions and directions for future research.

II. Related Research

Considering the numeric / semantic value of the effect of upsets during encoding is related to the idea of encoding with a fidelity criterion (rate distortion theory) [7], [3] and to *unequal error protection (UEP)*, joint source-channel coding with a fidelity criterion, which uses knowledge about how errors induced by the channel in the data (source representation) affect the quality of the reconstructed data on the other end of the channel [2]. The value deviations considered in this paper may be viewed as a distortion function defined for the *values* of integer and floating point approximate real numbers represented as bit vectors in programs. Source coding while taking into consideration the semantic interpretation of values is the basic principle behind perceptual encoding techniques, e.g. [5].

In a detailed gate-level simulation of an entire embedded microprocessor, Saggese *et al.* [6] show that upsets in the register file, load-store unit and bus interface account for more than 50% of the upsets that are not masked. Rather than attempting to eliminate the effects of logic upsets in these structures, it is possible to take advantage of the fact that the values they contain may represent variables in programs, to enable tradeoffs between the overhead of encoding and the effects of errors.

III. Bounds on Encoding Overhead

Digital representations of numbers define the contribution of individual binary digits of a bit vector, to its numeric value. When logic upsets occur within these digital representations, the value deviation they induce is a function of the location of the logic upset and the representation format. Figure 1 illustrates examples of value deviations caused by logic upsets in unsigned 8-bit integer representations.

Reduction of the error rate of a noisy channel (or equivalently, an encoded machine word in the presence of a source of logic upsets) to an arbitrarily small value, can be achieved by increasing the message redundancy. For this, Shannon's channel coding theorem [8] gives the upper bound on the efficiency, in the presence of upsets with a given bit upset probability. If however, the goal is not to reduce the overall error (i.e., non-masked upset) rate, but rather to reduce the magnitude of resulting value deviation, we can obtain similar bounds on encoding efficiency. Obtaining those bounds, for the digital arithmetic representations of program variable values, are the subject of the following sections.

A. Preliminary definitions

When k upsets in L-bit variables are possible, the value deviations, m, are in the range

$$m \in \left[0 \quad \sum_{i=L-k}^{L-1} 2^i \right]. \tag{1}$$

For each possible value deviation m, there are multiple counts, locations, and types (i.e., $0 \rightarrow 1$ versus $1 \rightarrow 0$) of upsets (up to k upsets in all), a total of z configurations, that may lead to the particular deviation. In other words, *z is the number of ordered pairs of values between 0 and $2^L - 1$ which differ in value by m, and differ in at most k bits in their digital arithmetic representation*[1]. If one ignores the upset-free word values and upset types, then there is a smaller number of *unique upset locations* to consider for a given word length L, value deviation m and maximum number of simultaneous upsets k. We denote this number of unique upset locations with y.

For example, for $L = 3$ and $k = 2$ (i.e., 3-bit variables with a maximum of two faults occurring simultaneously in a variable), there are seven possible values of value deviation m, i.e., $m \in [0, \ldots, 6]$ (Equation 1). The details of the cases for $m \in \{1, 2, 3\}$ are shown in Figure 2. For a value deviation of $m = 1$, there are $z = 12$ unique combinations of upset locations, upset types, and original upset-free word values leading to this value deviation. On the other hand, there are only $y = 2$ unique locations in which upsets can occur — in the least-significant bit alone, or in the two least-significant bits together.

If the underlying fault mechanism is that of inversion upsets, then all the error correcting code needs is the *location* of the upsets, and reversing the extant bit values provides the correction mechanism. The number of unique upset locations, y, gives a measure of the best case syndrome size for correcting inversion upsets, and hence a bound on the encoding efficiency; this is the subject of Section III-D. If however the fault mechanism leads to erasures, then the error-correcting code needs more information (it cannot simply "reverse" the bits as it would in the case of inversions). The total number of possible upset vectors, z, is thus of interest, as it contains information regarding the original and corrupted bit

[1] In this work, we only consider unsigned values. The extension to other number representations, such as two's-complement integer and floating-point representations is straightforward.

For z possible upset vectors which can lead to a deviation of m, there are only y of which are unique.

$m = 1$: $\{v = {}^{000}0, w = {}^{001}\bullet 1\}, \{v = {}^{001}1, w = {}^{010}\bullet\bullet 2\}, \{v = {}^{010}2, w = {}^{011}\bullet 3\}, \{v = {}^{011}3, w = {}^{100}\bullet\bullet\bullet 4\}, \{v = {}^{100}4, w = {}^{101}\bullet 5\}, \{v = {}^{101}5, w = {}^{110}\bullet\bullet 6\},$
$\{v = {}^{110}6, w = {}^{111}\bullet 7\}, \{v = {}^{111}7, w = {}^{110}\bullet 6\}, \{v = {}^{110}6, w = {}^{101}\bullet 5\}, \{v = {}^{101}5, w = {}^{100}\bullet 4\}, \{v = {}^{100}4, w = {}^{011}\bullet\bullet\bullet 3\}, \{v = {}^{011}3, w = {}^{010}\bullet 2\},$
$\{v = {}^{010}2, w = {}^{001}\bullet\bullet 1\}, \{v = {}^{001}1, w = {}^{000}\bullet 0\}$
$(\mathbf{z = 12, y = 2})$

$m = 2$: $\{v = {}^{000}0, w = {}^{010}\bullet 2\}, \{v = {}^{001}1, w = {}^{011}\bullet 3\}, \{v = {}^{010}2, w = {}^{100}\bullet\bullet 4\}, \{v = {}^{011}3, w = {}^{101}\bullet\bullet 5\}, \{v = {}^{100}4, w = {}^{110}\bullet 6\}, \{v = {}^{101}5, w = {}^{111}\bullet 7\},$
$\{v = {}^{110}6, w = {}^{100}\bullet 4\}, \{v = {}^{111}7, w = {}^{100}\bullet 5\}, \{v = {}^{101}5, w = {}^{011}\bullet\bullet 3\}, \{v = {}^{100}4, w = {}^{010}\bullet\bullet 2\}, \{v = {}^{011}3, w = {}^{001}\bullet 1\}, \{v = {}^{010}2, w = {}^{000}\bullet 0\}$
$(\mathbf{z = 12, y = 2})$

$m = 3$: $\{v = {}^{000}0, w = {}^{011}\bullet\bullet 3\}, \{v = {}^{001}1, w = {}^{100}\bullet\bullet 4\}, \{v = {}^{010}2, w = {}^{101}\bullet\bullet\bullet 5\}, \{v = {}^{011}3, w = {}^{110}\bullet\bullet 6\}, \{v = {}^{100}4, w = {}^{111}\bullet\bullet 7\}, \{v = {}^{101}5, w = {}^{010}\bullet\bullet\bullet 2\},$
$\{v = {}^{110}6, w = {}^{011}\bullet\bullet 3\}, \{v = {}^{111}7, w = {}^{100}\bullet\bullet 4\}, \{v = {}^{100}4, w = {}^{001}\bullet\bullet 1\}, \{v = {}^{011}3, w = {}^{000}\bullet\bullet 0\}$
$(\mathbf{z = 8, y = 2})$

Fig. 2. Enumeration of all possible cases of value deviation, $m = \{1, 2, 3\}$, for $L = 3$-bit words and a maximum of $k = 2$ simultaneous upsets, for pairs of error-free values v and error-containing values w. Pairs of values that are not possible with a maximum of $k = 2$ upsets are shown in a lighter color. Bit positions which are in error are shown with a "•" above them. Note that, for $L = 3$ and $k = 2$, there is no combination of three upsets that would lead to a value deviation of $m = 2$.

values, necessary for correcting erasures. The encoding efficiency bounds in this case are the subject of Section III-E. An ideal encoding scheme will only employ as many bits as are necessary to identify these possible cases of value deviation, and the *syndrome* for the error-correcting code will need $\lceil \log_2(z) \rceil$ bits in the case of erasures, and $\lceil \log_2(y) \rceil$ in the case of inversion upsets.

B. Efficiency bounds for singly-occurring logic upsets

For the special case when only a single upset may occur within an L-bit word (i.e., $k = 1$), there are $L - \lfloor \log_2(m) + 1 \rfloor$ bit positions that can cause a constraint on value deviation m to be violated. The intuition is that, for a constraint m, there are $\lfloor \log_2(m) + 1 \rfloor$ bit positions at which a single logic upset occurring would yield a deviation less than m. The *syndrome* of the error correcting code must contain at least as many bits as required to locate upsets in these bit positions, thus,

$$2^{L_C - L} \geq \text{number of locations in which upsets can cause value deviation to exceed } m, +1 \text{ (for the case when no bits are in error),}$$

$$= L - \lfloor \log_2(m) + 1 \rfloor + 1.$$

Thus,

$$L_C \geq L + \lceil \log_2(L - \lfloor \log_2(m) + 1 \rfloor + 1) \rceil. \quad (2)$$

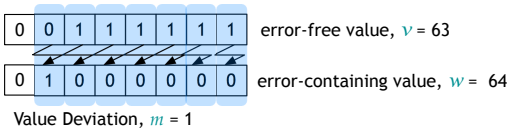

Fig. 4. When multi-bit upsets occur, $\lfloor \log_2(m) + 1 \rfloor$ no longer characterizes the bit positions that must be protected to prevent value deviations of m.

In the above, $L - L_C$ is the number of syndrome bits, and 2^{L-L_C} is the number of bit positions in which upsets can be identified by the syndrome. Thus, for example, for $L = 8$-bit values in which the constraint is $m \leq 64$, only $8 - \lfloor \log_2(64) + 1 \rfloor = 1$ bit is critical, and the syndrome must be $\lceil \log_2(2) \rceil = 1$ bit. This single syndrome bit will be used to differentiate between no upset, and the critical bit being incorrect. Figure 3 plots the limiting efficiency of encoding $(L/L_C,)$ as a function of L (number of bits in data value) and m (upper bound on tolerable value deviation).

When more than one upset may occur however, the above analysis is insufficient. To observe why this is so, consider an 8-bit value that has incurred a value deviation of $m = 1$. The foregoing analysis would imply that the upset must have occurred in the least significant $\lfloor \log_2(m) + 1 \rfloor$ bits. Consider however the value 63, which as a result of a multi-bit upset, is now the value 64. This would require a 7-bit upset vector as shown in Figure 4. The construction of appropriate bounds for the more

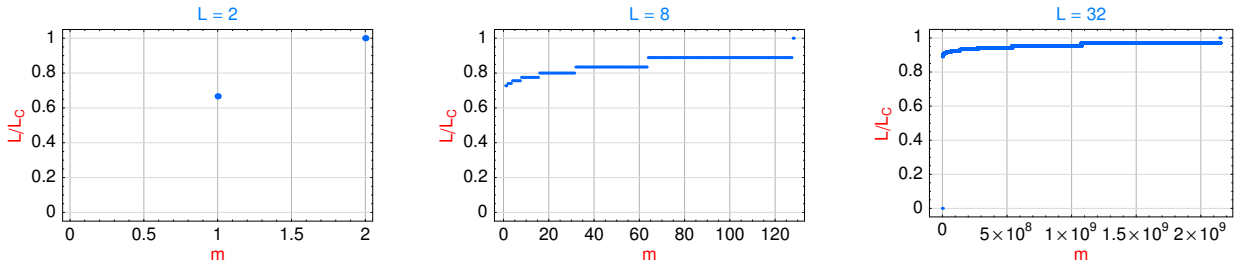

Fig. 3. Limiting efficiency of encoding length for single-bit upsets, L/L_C, as a function of number of bits of data value, L, for different values of tolerable value deviation, m.

general case of multi-bit upsets is described next.

C. Efficiency bounds for multiple logic upsets — general case

From the example of Figure 2, it can be easily determined that an upper bound on the number of logic upset cases leading to a value deviation of m (albeit a loose one) is given by

$$\begin{aligned} z &\leq \underbrace{2^L - m}_{(1)} + \underbrace{2^L - m}_{(2)} \\ &= 2^{L+1} - 2m, \end{aligned} \tag{3}$$

where term (1) accounts for all the cases where the error-free value v is increased to the error-containing value of at most $2^L - 1$, and likewise the term (2) for when the error-free value of at least m is decreased by m to yield the error-containing value. The above is an inequality, since, for a given L, k and m, it does not consider the restriction on number of bit positions (fewer or equal to k) which must have associated upsets.

The problem of finding the number of cases that may exist for a given word length L, number of upsets k and value deviation m can be re-stated as the problem of finding the number of pairs of values of length L, that differ by m and have Hamming distance k. For unsigned L-bit values, this is given by the *number of solutions* to the simultaneous Diophantine equations:

$$\left| \sum_{i=0}^{L-1} a_i 2^i - \sum_{i=0}^{L-1} b_i 2^i \right| = m, \tag{4a}$$

$$\sum_{i=0}^{L-1} (a_i(1-b_i) + b_i(1-a_i)) = k. \tag{4b}$$

The intuition behind Equations 4a and 4b is that two L-bit unsigned values differ in k bit positions, and differ in value by m, when their arithmetic representations differ in value by m, and their bit-wise XOR (i.e., $a_i(1-b_i) + b_i(1-a_i)$) sums to k. For other number representations (e.g., two's complement, or floating-point approximate real-number representations like the IEEE-754 floating point format), the first of the above two equations will be replaced with an equation capturing the (different) role bit-positions play in the said number representation.

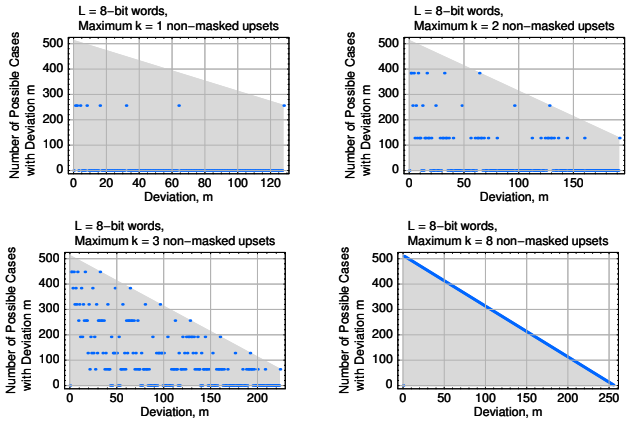

Fig. 5. Number of solutions to Equation 4, for $L = 8$, with varying k and m. The plots are overlaid on the loose bound to the number of solutions of Equation 4, given by Equation 3, shown as the shaded grey inequality region.

The general solution of Diophantine equations is undecidable. Equations 3 provided a loose upper bound on the number of solutions satisfying the above equations. More insight can be attained by the use of exhaustive enumeration to study the behavior of the number of solutions for Equation 4, as a function of L, k and m. The results for $L = 8$ are shown in Figure 5 (number of solutions, z) and Figure 6 (number of unique upset locations, y). Several observations may be made from Figure 5:

1) The number of possible solutions is always a multiple of 2^{L-k+1}.
2) For each value of m, there are 2^k possible numbers for the count of possible cases.
3) For a given value of k, the coefficients — the ratio of the number of possible solutions to 2^{L-k+1} — across the possible values of m always begin with the Mersenne numbers $(2^k - 1)$ for $m = 1$, and end in 1, for largest $m = \sum_{L-k}^{L-1} 2^i$.
4) For all L, at $k = 1$, the coefficients are 1 when m is a power of two.
5) For all L, at $k = L - 1$, the coefficients are monotonically decreasing after each second value of m.
6) For all L, at $k = L$, the coefficients monotonically decrease for each increase in m.

The of values of y for a given L and k as a function of m

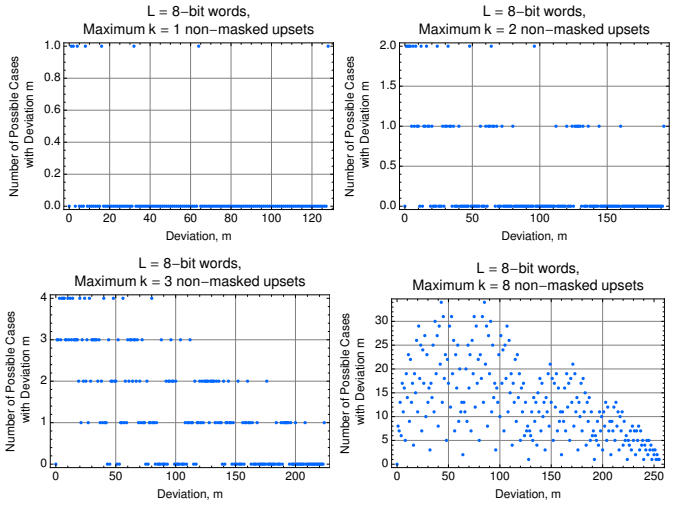

Fig. 6. Number of unique upset locations, y, for $L = 8$, with varying k and m.

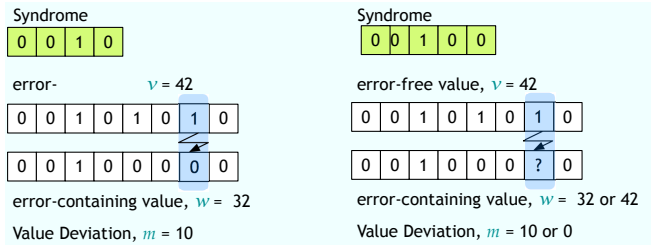

Fig. 7. Differing role of the syndrome for bit-reversals versus bit-erasures.

can however be shown to belong to the integer sequence defined by $a(2n+1) = a(n), a(2n) = a(n) + a(n-1)$ (Sloane's integer sequence number A020951 [1]).

D. Efficiency bounds for multiple logic inversion upsets

For a value deviation m, there are y bit positions that must be protected by the error-correcting code. The number, y of bit positions that must be protected in order to ensure a value deviation less than m, for a given L and k were enumerated in Figure 6. The number of syndrome bits, $L_C - L$, needed to protect against a value deviation m, for a given L and k is thus $\lceil \log_2(y) \rceil$. The limiting efficiency, L/L_C, attainable in the ideal case is shown in Figure 8 (points, upper curve).

E. Efficiency bounds for multiple erasures

When considering erasures, the syndrome of an error-correcting code must distinguish between the different cases of possible underlying bit-values prior to the erasure, in addition to identifying the *location* of the upsets.

For example, for an 8-bit word, and assuming singly-occurring upsets, four bits can be used to identify which bit position (if any) has incurred a bit-reversal — there are nine cases, eight of which correspond to a particular one of the bits being incorrect, and the last corresponding to the case of all bits correct. For erasures, the syndrome must not only denote which bit position is incorrect, but

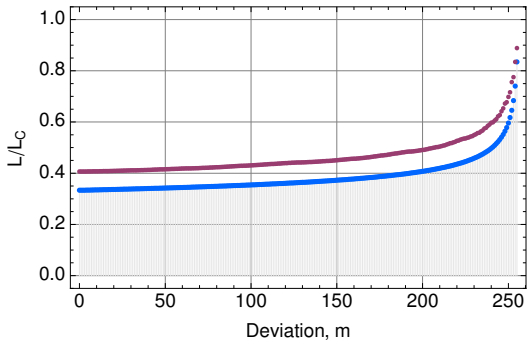

Fig. 8. Limits on encoding efficiency for erasures in $L = 8$-bit words with at most $k = 8$ simultaneous distinct inversion upsets (points, upper curve) and erasures (stem plots, lower curve).

must also be able, based on information in the syndrome (which is constructed from the original source word) to determine the correct bit value. There are sixteen possible single-erasures, e.g., bit 4 should have value 0 but has suffered an erasure, bit 4 should have value 1 but has suffered an erasure, and so forth, plus the additional case of no incurred erasures. The syndrome thus needs five bits; this is illustrated in Figure 7.

For a word length $L = 8$ and up to $k = 8$ distinct erasures, the lower curve (stem plots) in Figure 8 shows the limiting efficiency of a deviation-bounded k-erasure correcting code. The lower limiting encoding efficiency for erasures, as compared to bit inversions, follows directly from the preceding arguments — intuitively, more information must be maintained to enable correction of erasures in contrast to bit inversions. As the tolerable deviation, m, resulting from non-corrected upsets increases, inversion-upset- and erasure-correcting codes need fewer bits in their syndromes to locate and correct upsets.

IV. Summary

When bit-level value reversals (inversions) and erasures occur within machine state representing arithmetic program variables, it is possible to consider their effect in terms of the *value deviation* incurred. This permits the use of forward error correction schemes that limit the amount of such value deviation, by taking into account the role individual bits play in the digital arithmetic representation of numbers. This paper presented an investigation of the bounds on achievable efficiency for such *deviation-tolerant* encoding.

References

[1] A020951. In *The On-Line Encyclopedia of Integer Sequences*.
[2] Berger and Gibson. Lossy source coding. *IEEETIT: IEEE Transactions on Information Theory*, 44, 1998.
[3] T. Goblick. A coding theorem for time-discrete analog data sources. *Information Theory, IEEE Transactions on*, 15(3):401–407, May 1969.
[4] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–47, Mar. 1991.

[5] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, October 1993.

[6] G. P. Saggese and A. Vetteth. Microprocessor sensitivity to failures: Control vs execution and combinational vs sequential logic. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 760–769, Washington, DC, USA, 2005. IEEE Computer Society.

[7] C. E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record*, 7(4):142–163, 1959.

[8] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Illinois, 1963.