J. of Mult.-Valued Logic & Soft Computing, Vol. 23, pp. 337–363
Reprints available directly from the publisher
Photocopying permitted by license only

Error Correction via Restorative Feedback in M-ary Logic Circuits

CHRIS WINSTEAD, YI LUO, EDUARDO MONZON AND ABIEZER TEJEDA

Department of Electrical and Computer Engineering, Utah State University, Logan, UT 84322-4120, USA
E-mail: chris.winstead@usu.edu, yiluo.nku@gmail.com, eduardo.monzon@aggiemail.usu.edu, a.tejeda@aggiemail.usu.edu

Received: September 28, 2011. Accepted: May 7, 2012.

This paper presents an error correction method known as restorative feedback (RFB) that provides error-correction for both permanent and temporal logic faults in any M-ary logic system. The RFB method is a variant of triple modular redundancy (TMR), which achieves error correction in logic circuits by using three-fold redundancy. Unlike TMR, the RFB method has well-defined application to arbitrary M-ary logic systems as well as conventional binary logic circuits. The RFB method also uses a feedback mechanism to suppress transient errors, resulting in a lower error probability than TMR when considering transient upsets. The underlying theory of RFB is presented as an adaptation of stochastic error correction theory. Two circuit-level proof-of-concept demonstrations are presented, which include a binary implementation using Muller C-elements, and a ternary implementation based on Semi-Floating Gate logic circuits. The error-correcting performance of these circuits is evaluated using logic-level simulations as well as device-level simulations in Spectre. Bit and symbol error rates are also computed using Monte Carlo simulations which demonstrate that the RFB method is superior to traditional TMR for a variety of cases. An application of the RFB method is also demonstrated using redundant gate-level synthesis of multiple-valued ripple-carry adder circuits. The application circuits are simulated using an abstract noisy-logic model, and the RFB method is shown to significantly improve the circuits' noise immunity.

Keywords: Error correction, restorative feedback, triple modular redundancy, reliability.

1 INTRODUCTION

Manufacturing trends in CMOS logic have resulted in nanometer-scale transistor devices that are increasingly sensitive to noise fluctuations and manufacturing defects. At the same time, the density of integration is increasing both in terms of planar circuits and three-dimensional integration. This trend leads to higher thermal density, resulting in a harsher noise environment and an increased likelihood of damage to devices and interconnect in highly-integrated circuits. In addition to these trends in CMOS technology, several new technologies have emerged as candidates for "post-CMOS" logic devices. These technologies, which include carbon nanotube FETs and nanowire logic among others, are expected to be even more susceptible to defects.

For several decades the CMOS logic industry has been driven by a continual scaling of device dimensions and the resulting improvements in large-scale integration and switching speed. Although binary logic techniques have proven to be highly profitable during this period, there has been steady interest in alternative multiple-valued logic circuits. Multiple-valued logic systems may offer improvements in physical efficiency and, ultimately, may increase the effective density of computations compared to binary circuits. Unfortunately they are also inherently more sensitive to noise fluctuations than are binary circuits. While there has been considerable progress in developing error-correction and fault-compensation strategies for binary logic, there has been comparatively little progress in such techniques for multiple-valued circuits and systems. This paper presents a new error-correction method that is suitable for multiple-valued logic circuits as well as traditional binary circuits.

The new method – called Restorative Feedback (RFB) – is closely comparable to the method of Triple Modular Redundancy (TMR) [8], and may be considered to be an improved version of TMR. Among the available binary error-correction schemes, TMR is perhaps the simplest and most widely understood technique. Several researchers have proposed using modular redundancy strategies to compensate for faults in nano-scale CMOS and post-CMOS logic circuits [7,10,13,20]. In the standard TMR scheme, a logic operation is replicated three times, usually by three parallel, independent modules. Then, for each desired logic output x, there are three corresponding signals x_1, x_2, and x_3, each of which may be correct ($x_i = x$) or incorrect ($x_i \neq x$). The probability of error on each signal x_i is assumed to be independent from that of the other signals, and is equal to some small constant η. The TMR output y is obtained by taking the majority value over the three inputs. Hence the output y is correct if any single error occurs among the x_i, but two or more simultaneous errors will cause the error to propagate onto y.

In order to compensate for faults that occur within majority gates, the TMR scheme may be implemented using three parallel majority operations. The TMR system may then be used in a fully-redundant cascade configuration in which majority operations occur periodically at points dispersed throughout a logic pipeline [21, 22]. The rate of intrinsic errors within majority gates is then absorbed into the η parameter of some downstream majority operation. By using the fully-redundant cascade approach, the error rate in a large system is iteratively suppressed as a signal propagates through successive stages of computation and restoration.

Unfortunately the TMR method introduces some ambiguity when applied to multiple-valued logic systems. In a binary logic circuit, there is always a clear majority value. This is not the case in a multiple-valued circuit, because it is possible for each of the x_i signals to carry a distinct value. In this case the majority gate's output is undefined. Nanduri proposed using the "average" value of all three signals in this case [12]. The problem with this solution is that multiple-valued logic systems do not typically identify *logic levels* with *signal levels*. Rather, the *signal level* is an artefact of implementation, and the meaning of "average level" is not clear for abstract M-ary algebras. The Nanduri-TMR method is therefore limited to special cases in which there is a clear logic interpretation for the average signal level.

The RFB approach detailed in this paper is similar to fully-redundant cascaded TMR, but its behavior is well-defined for multiple-valued circuits. Hence, the RFB method can be applied as a general-purpose solution for error-correction in any M-ary logic system. The RFB method was previously introduced by the authors in [26]. In this paper, we provide additional details on the underlying theory of the RFB method, and also present additional examples of the RFB method applied to practical logic cases. The remainder of this paper is organized as follows. Section 2 presents the basic theory used to obtain the restorative feedback method and high-level simulation results. Section 3 presents a binary static CMOS circuit for the restorative feedback method, and provides simulation results obtained from Spectre. Section 4 proposes a CMOS semi-floating gate implementation for multiple-valued logic. Section 5 presents simulation results for the RFB method applied to multiple-valued ripple-carry adder circuits. Finally, Section 6 offers conclusions and items for further research.

2 ERROR SUPPRESSION THROUGH RESTORATIVE FEEDBACK

The restorative feedback method relies on the use of Muller C-elements instead of majority gates. The C-element is a well-known gate that has long

been used in asynchronous circuit design [11], and was more recently recognized for its inherent fault-compensating abilities [4, 5, 28]. The C-element is logically defined as a two-input latch with inputs x, y and output z. A new value is latched for z whenever $x = y$, so that the output equals the unanimous inputs, i.e. $z = x = y$. If $x \neq y$ then the output retains its latched value. Although the C-element is traditionally considered to be a binary logic gate, this definition applies equally well to multiple-valued logic systems.

Researchers previously described several error-correction methods based on C-elements. These include "Duplicated Dual Checking" (DDC), a dual-modular redundancy method intended for use in asynchronous logic [4,5], the BISER-FF method for compensating timing errors in synchronous circuits [28]. The BISER-FF method has been extended by several authors leading to a family of solutions for error-compensation. The DDC, BISER-FF and related methods are all feed-forward error correction techniques that provide compensation for soft faults, but are not suitable for protecting against hard faults. The Restorative Feedback method presented in this paper provides stronger protection against transient faults, and is able to correct hard faults as well.

In most of the work published to date on error correction with C-elements, a straightforward logic-case analysis is used, which can be generically described as follows. The C-element may be used to correct momentary faults if its inputs x and y are two redundant copies of the same logic value. Under normal conditions the two inputs should be equal and change at the same time. If a momentary error appears on only one of the signals, then the output remains unaffected. Hence the C-element can correct any single momentary error using only two redundant signals. As with TMR, two simultaneous errors will cause an error to propagate at the gate's output. This case analysis helps to visualize the error correcting capabilities of the C-element, but a more precise understanding is obtained by analyzing the gate's error statistics. To perform this analysis, we propose using an approach derived from stochastic iterative decoding, as explained in the next subsection.

2.1 The stochastic decoding interpretation.

In 2003 Gaudet and Rapley proposed *stochastic decoding* as a probabilistic approach to iterative error correction [3]. Stochastic decoders implement soft error correction algorithms by filtering streams of randomly toggling bits. This concept has been further developed in recent years by one of the authors (Winstead), and by Tehrani, Gross, Mannor and others [15–19,24]. Beginning in 2005 the author proposed using stochastic decoding methods to process and reduce the native error probabilities in switching devices [23, 25, 27]. The restorative feedback method proposed in this paper is derived using that strategy.

The stochastic decoding approach is a realization of the widely used sum-product algorithm for iterative error correction. In a stochastic decoder, all signals are considered as *stochastic streams* which vary randomly over time. A stochastic stream may be defined as a signal $x(t)$ which takes values randomly from a discrete symbol alphabet $\mathcal{X}$. At any given time t, the value $x(t)$ is an independent sample of a random variable $X(t)$ with probability mass function $\rho_x(x, t)$. The stochastic stream is also assumed to be a memoryless process, so that two samples $x(t_1)$ and $x(t_2)$ are statistically independent if they are observed at different times $t_1 \neq t_2$.

In this paper we consider a logic signal to be a stochastic stream for which one value, x^*, is correct, but other erroneous values may occur at random due to transient noise fluctuations. By using this interpretation, we can employ the methods of stochastic decoding to improve error probability at the gate level in a logic system.

The equality operation

In this paper we are concerned with one specific type of sum-product operation, which is often called the *equality node* operation, using terminology introduced by Forney's "normal" factor graph approach [2]. The equality node's behavior is described as follows. Suppose the node has two inputs, $\mathbf{X}$ and $\mathbf{Y}$, which are random variables that represent independent noisy observations of the same "true" value x^*.

The variables $\mathbf{X}$ and $\mathbf{Y}$ are characterized by probability mass functions $\rho_x(x)$ and $\rho_y(y)$, respectively, where $\rho_x(x) = \Pr(\mathbf{X} = x)$. The equality operation computes the outgoing message for $\mathbf{Z}$ as

$$\rho_z(z) = \alpha \rho_x(z)\, \rho_y(z), \tag{1}$$

where α is a normalizing constant given by

$$\alpha = \left(\sum_{x \in \mathcal{X}} \rho_x(x)\, \rho_y(x) \right)^{-1}. \tag{2}$$

The result of this operation is that the output probability mass function, ρ_z, tends to be more concentrated around the true value. To show this, consider the likelihood ratio l_x, defined as

$$l_x = \frac{\Pr(\mathbf{X} = x^*)}{\Pr(\mathbf{X} \neq x^*)} = \frac{\rho_x(x^*)}{\sum_{x \neq x^*} \rho_x(x)} \tag{3}$$

and we similarly define the log-likelihood ratio (LLR) as $L_x = \log l_x$. We may interpret the LLR as a measure of *confidence* that an observation of $\mathbf{X}$

is correct. If $L_x \to \infty$, then we can be absolutely certain of a correct observation. On the other hand, if $L_x = 0$, then the observation is no better than a fair coin toss. Based on this interpretation, the circuits developed in this paper rely on the following theorem.

Theorem 1. *The output from an equality operation satisfies* $L_z \geq L_x + L_y$.

Proof. It is easily shown that $l_z \geq l_x l_y$, because

$$l_x l_y = \frac{\rho_x(x^*)\,\rho_y(x^*)}{\left(\sum_{x \neq x^*} \rho_x(x)\right)\left(\sum_{y \neq x^*} \rho_y(y)\right)} \tag{4}$$

$$\leq \frac{\rho_x(x^*)\,\rho_y(x^*)}{\sum_{x \neq x^*} \rho_x(x)\,\rho_y(x)} \tag{5}$$

$$= l_z \tag{6}$$

Based on this theorem, the equality operation may be described as a *confidence-amplifying* operation. If L_x and L_y are both greater than zero, then the output confidence L_z exceeds the confidence of either input.

The Muller C-element

To provide a stochastic realization of the equality operation, Gaudet described a flip-flop circuit called the "stochastic equality gate," which is equivalent to the Muller C-element. Because of Theorem 1, the Muller C-element may be considered as a confidence amplifying device under certain conditions. Figure 1 shows the usual graphical depiction of the equality operation and its C-element equivalent for the binary case. The figure also indicates the statistical transformations that occur in the binary case. The probability mass function $\rho_z(\zeta)$ is considered to be the frequency with which $z(t) = \zeta$. In the binary case, this function can be represented by a single statistic, $p_z = \rho_z(1)$, which is equal to the time-average $\overline{z}$.

It is easily shown that the C-element's output statistics are equal to those computed by the sum-product equality operation. This result relies on the statistical independence of $x(t)$, $y(t)$ and $z(t)$. This differs from the standard sum-product algorithm, which requires independence only between $\mathbf{X}$ and $\mathbf{Y}$. To explain this difference, suppose that the C-element has a delay of τ. Then, in the binary case, the output probability for $z(t)$ can be expressed as

$$p_z(t+\tau) = p_x(t)\,p_y(t) + \left[1 - p_x(t)\,p_y(t)\right] p_z(t). \tag{7}$$

The right-most term of (7) includes a product of p_z with $\left[1 - p_x(t)\,p_y(t)\right]$, representing the case where $x(t) \neq y(t)$ and $z(t)$ retains a logic 1 state. This

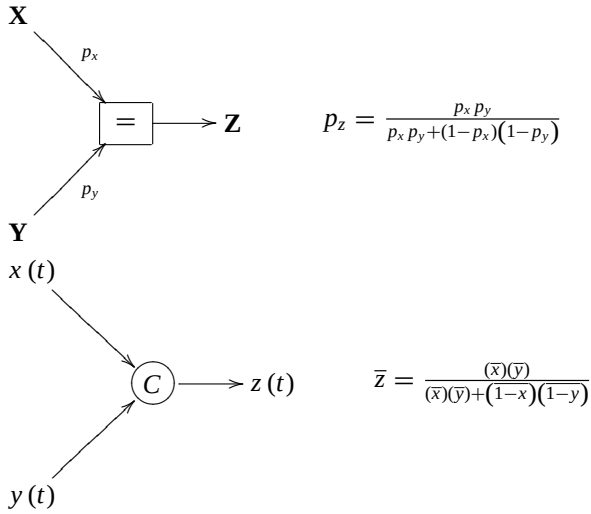


FIGURE 1
The *equality* operation and its stochastic implementation using a Muller C-element. **X** and **Y** are independent binary random variables. The probabilities p_x, p_y, p_z indicate the frequency with which the associated signals $x(t)$, $y(t)$ and $z(t)$ are equal to 1. The overline mark, as in $\overline{z}$, indicates the average signal level taken over all time. For Bernoulli-distributed variables, $\overline{x} = p_x$.

product is only valid if $p_z(t)$ is independent of the other variables at time t. Note that $p_z(t)$ can depend on values of x and y *prior to* time t, but it must be *instantaneously independent* of their values at the precise time t. Then, assuming that $p_z(t+\tau) = p_z(t)$, we may solve for p_z to obtain the result shown in Figure 1. The same analysis may also be repeated for the general non-binary case, yielding results identical to (1) and (2).

Restorative Feedback topology

The idea of restorative feedback is to connect the C-element's output back around to one of its inputs, forming a positive feedback loop. If this is done in a way that satisfies the independence conditions, then we would expect the confidence (L_y) to increase without bound. Unfortunately the independence condition is violated by direct feedback in a C-element, as shown in Figure 2. When direct feedback is used, the C-element's output state cannot ever change unless an upset error occurs in the C-element's state memory, and is therefore a useless configuration.

As an alternative, the C-element feedback may be distributed across several gates, as shown in Figure 3. This approach increases the *cycle girth* of the feedback path, which weakens the statistical dependence between each C-element's output and the feedback signal. As shown in the next subsection,

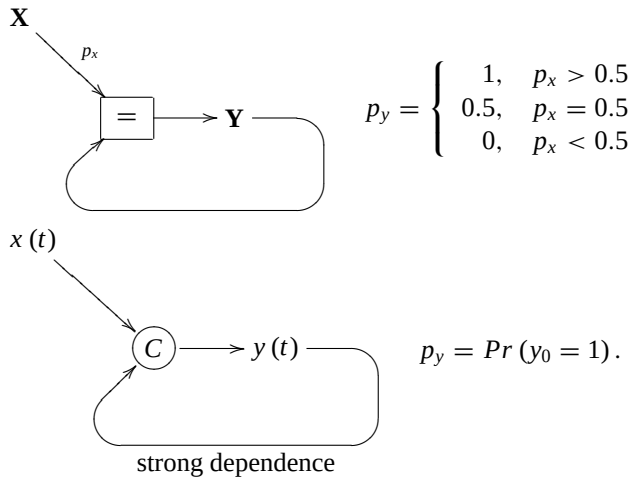


FIGURE 2
The *equality* operation is connected in a positive feedback configuration, driving L_y toward ∞. This process fails when used with the C-element realization, because the C-element inputs are required to be independent.

the three C-elements can be initialized in a reliable state. Then the feedback signal can be activated, driving the outputs' confidence toward infinity. In practice, the confidence reaches a limit set by the intrinsic error rate of the C-element's state memory. Hence the C-element can be used to restore the inputs' error probability down to that of the C-element itself.

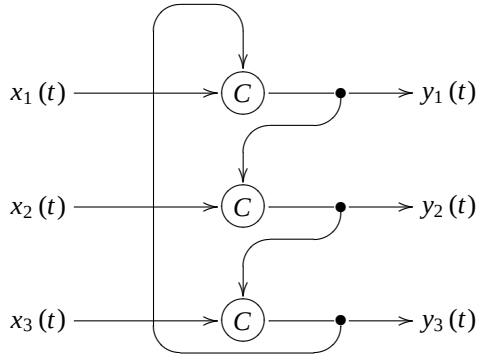


FIGURE 3
The Restorative Feedback (RFB) topology distributes feedback across three C-elements, so that strong statistical correlations are avoided.

2.2 Two-phase operation.
To successfully implement the restorative feedback concept, the state of the C-element must be initialized to an appropriate value. Since the C-element is a latch, it must somehow be assigned an initial condition before the feedback transformation can be considered meaningful. To resolve this problem, the circuit is operated in two phases, called *setup* and *restoration*, as shown in Figure 4. To initialize the C-element states, the inputs $x_1,\ x_2,\ x_3$ are

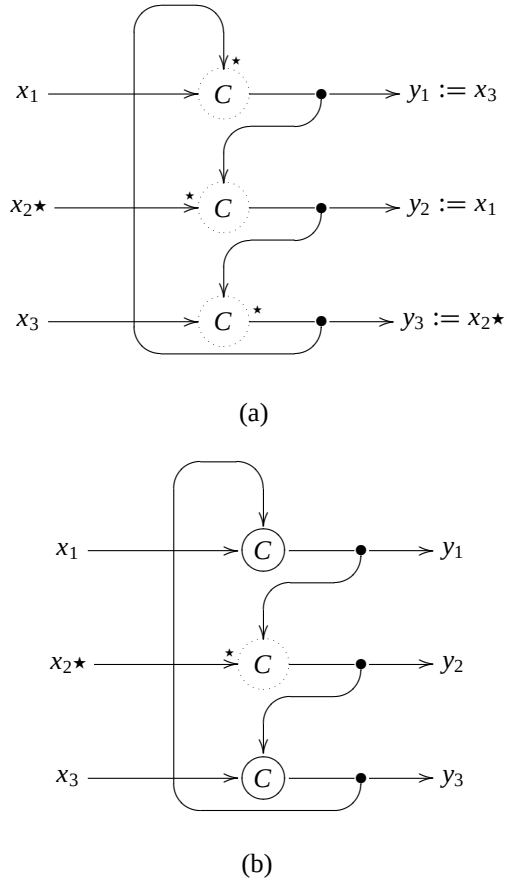


FIGURE 4
Two-phase error correction in the restorative feedback circuit. The dotted circles indicate C-elements which are inactive (i.e. they are not actively driving their outputs). The star indicates an error location. (a) In the setup phase, the outputs are initialized with barrel-shifted copies of the input signal values. (b) In the restoration phase, the feedback is activated, causing C-elements to activate. Errors are corrected as the C-elements become active. In the final, stable state, correct values are reinforced by the C-elements, and any further transient errors are suppressed.

barrel-shifted onto the outputs, so that $y_1 = x_3$, $y_2 = x_1$ and $y_3 = x_2$. After the states are initialized, the feedback paths are activated.

Like TMR, the two-phase design is able to correct a single initial error among the x_i. To show this, suppose that x_2 is in error while x_1 and x_3 are correct (the error position is indicated by a star ($\star$) in Figure 4(a)-(b)). Then during the initialization phase, the error is transferred to y_3, while y_1 and y_2 are correct. During the restoration phase, the C-element feedback is activated. The third C-element has inputs x_3 and y_2, both of which are correct. Since these inputs equal each other, the C-element forces y_3 to change, hence correcting the error. The corrected signal y_3 propagates to the first C-element, where it reinforces the correct value on y_1.

Unlike traditional TMR, the restorative feedback method suppresses additional transient errors that occur during the restoration phase. Suppose that a momentary error appears on signal x_1 during the restoration phase in Figure 4(b). Because y_3 is already correct, the C-element has mixed inputs ($x_1 \neq y_3$), and consequently the output y_1 does not change. Similarly, any single momentary error on x_2 or x_3 is prevented from propagating to the outputs. Furthermore, simultaneous double or triple errors are blocked during the restoration phase.

2.3 Simulation results using an iterative model.

In order to quantify the error-correcting capabilities of restorative feedback, as compared to TMR, the bit error rate (BER) is measured using Monte Carlo simulations. A discrete-time model is used to represent signal and error events in the respective circuits. The C-elements and majority gates are assumed to have a delay of one time unit. In these simulations, three types of errors are accounted for:

1. Transient errors on the input signals. Each of the redundant inputs is assumed to be of the form $x_i(t) = x^* \oplus e_{xi}(t)$, where x^* is the correct logic value, $e_{xi}(t)$ is a Bernoulli distributed error event that occurs with probability η, $\oplus$ indicates addition modulo 2, and $t \in \mathbb{N}_0$.
2. Momentary internal faults. Errors may also originate from within the TMR and RFB circuits. These intrinsic gate errors are represented by Bernoulli distributed error events e_{yi} which occur with probability ϵ. We assume that $\epsilon < \eta$ because η represents the composite error rate compounded over many logic gates. Then the circuits' actual signal outputs are $y_i' = y_i \oplus e_{yi}(t)$.
3. Persistent state upsets. In the restorative feedback circuit, an additional error type may appear if a persistent state-inversion occurs within the C-element. In this simulation, state upset errors are included and are assumed to occur with probability $\delta = \epsilon/10$.

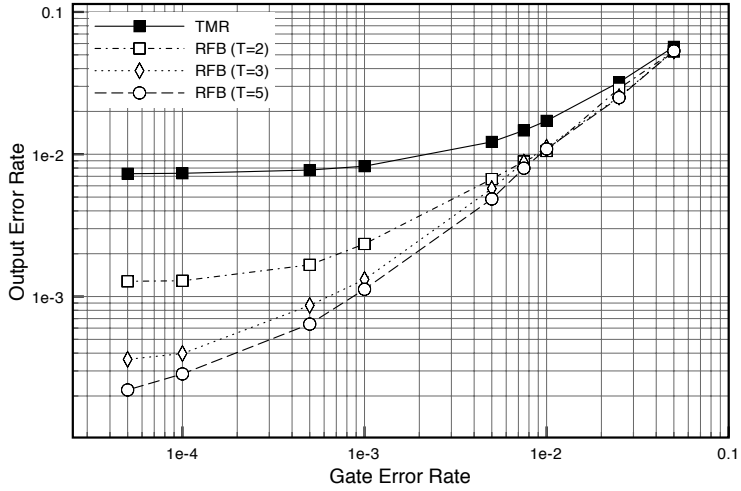


FIGURE 5
Simulated BER results obtained using a discrete-time model. Input transient errors appear at a fixed rate of $\eta = 0.05$ errors per time unit. The intrinsic gate error rate (ϵ) is varied from 5×10^{-5} to 0.05. The circuits' outputs are sampled and errors counted after 5 time units. The restorative feedback BER is found to approach ϵ, which is lower than the TMR limit.

The simulation results are shown in Figure 5. In this figure, the input error rate η is held constant while the intrinsic error rate ϵ is varied. As the intrinsic error rate is reduced, the TMR method approaches a limit approximately equal to the rate of double and triple error events, $3\eta^2 - 2\eta^3 + \epsilon$. The restorative feedback circuit approaches a limit approximately equal to ϵ. Because the restorative feedback circuit relies on feedback, its error statistics improve over time. In Figure 5, the circuit is allowed to settle for five time units. For each additional time-unit, the circuit's BER moves closer to the ϵ limit.

In real circuit implementations, the settling time may differ depending on how the feedback is implemented. In the remainder of this paper, we consider two example designs. The first design is a binary CMOS implementation in which the feedback propagates in continuous time, resulting in non-iterative dynamics. The second design is a multiple-valued implementation based on semi-floating gate (SFG) circuits. In the SFG implementation, the feedback signal is expressly clocked, resulting in iterative dynamics that more closely match the discrete-time simulation model.

3 BINARY CMOS IMPLEMENTATION

The Muller C-element has been used for many years in CMOS logic design, particularly in the field of asynchronous circuits. A typical C-element circuit

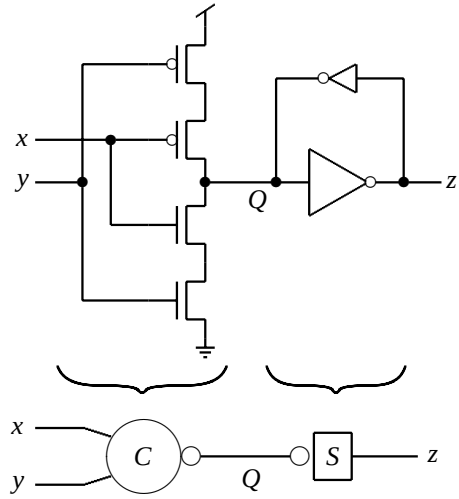


FIGURE 6
Composition of the C-element in terms of "C-not" and "S" gates.

implementation is shown in Figure 6 [14]. This circuit consists of two parts, which we will call the C-not and S gates. The C-not gate becomes active whenever its two inputs are equal. Once activated, the C-not gate drives a new value onto node Q, and overrides the state of the memory element S. When $x \neq y$, the C-not gate enters a high-impedance state at Q. In this case, the value at Q is retained by weak feedback in S. The weak feedback signal is indicated by the small inverter (also known as a "keeper") in Figure 6. The weak inverter is sized to ensure that the C-not gate always dominates when there is signal contention.

3.1 RFB circuit description.

C-element components are used to construct the RFB circuit shown in Figure 7. In this circuit, a multiplexer is inserted between the C-not and S gates. The multiplexer is assumed to be comprised of ideal electrical switches, so that the selected input signal is shorted to the multiplexer's output terminal. Hence, if the C-not gate's output is high-impedance ($Q = Z$), then when $\phi = 1$ the multiplexer's output is also high-impedance.

The RFB circuit operates in two phases. During the initialization phase, the multiplexer selects the adjacent input signal, so that the state of S is directly forced into the desired initial value. During the restoration phase, the multiplexer selects the output from the C-not gate, hence activating the feedback.

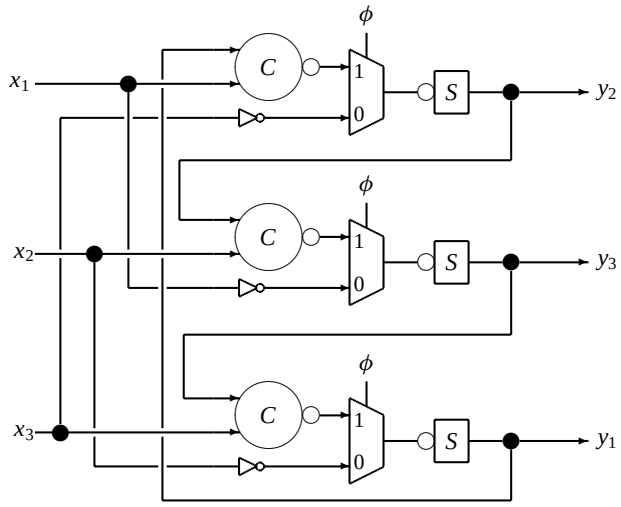


FIGURE 7
Implementation of the restorative feedback method using binary CMOS gates.

3.2 Simulation results.

The CMOS restorative feedback circuit was designed for a 32nm CMOS logic process with a supply voltage of 1V. For comparison, a cascaded-TMR circuit was also designed. Both circuits were simulated in Spectre using Predictive Technology SPICE models [29]. Because signal errors are very rare in CMOS technology, the "`noisescale`" parameter was set to a large value (100 to 200) in order to induce a high frequency of observable errors. During a transient simulation, all noise sources are multiplied by the `noisescale` parameter, resulting in highly exaggerated noise conditions.

Figure 8 shows an overlay of 100 Monte Carlo transient simulation runs from both the RFB and TMR simulations. An error is said to occur whenever a signal crosses the 0.5V threshold. In the TMR case, errors appear quite frequently, as indicated by numerous threshold-crossings in Figure 8. In the RFB case, the rate of errors is significantly reduced after a delay of 0.5ns. The amplitude of output noise fluctuations is also significantly reduced in the restorative feedback outputs. The 0.5ns delay is attributed to the propagation delay along the feedback path.

BER results were extracted from the transient simulations by counting *all* error events that occur after the 0.5ns delay. With this counting method, multiple transient errors may be counted within a clock cycle. The results are shown in Figure 9. As expected, the restorative feedback circuit always has a lower BER than the TMR circuit under equivalent conditions.

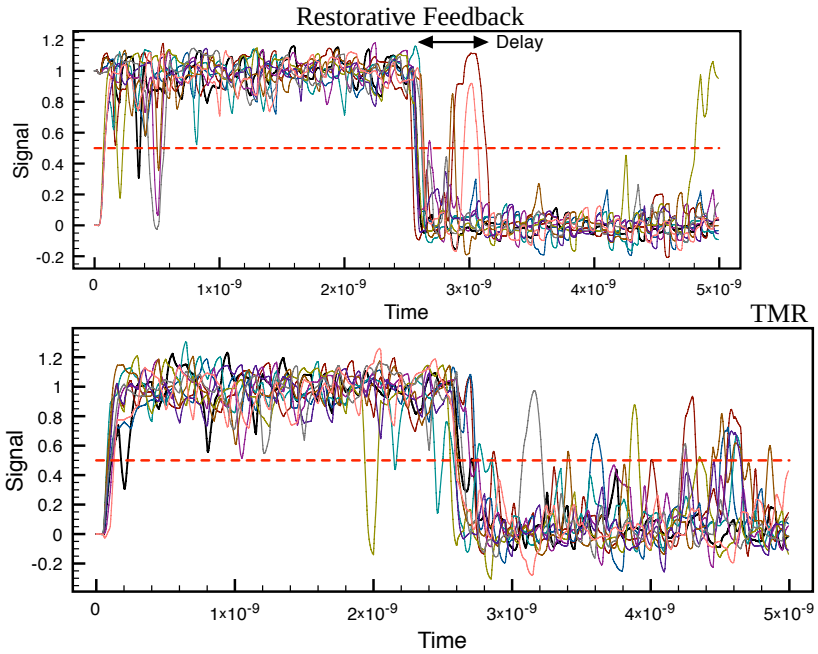


FIGURE 8
Transient simulations using a 32nm noisy-CMOS model for restorative feedback and TMR. The restorative feedback is more reliable after a delay of 0.5ns.

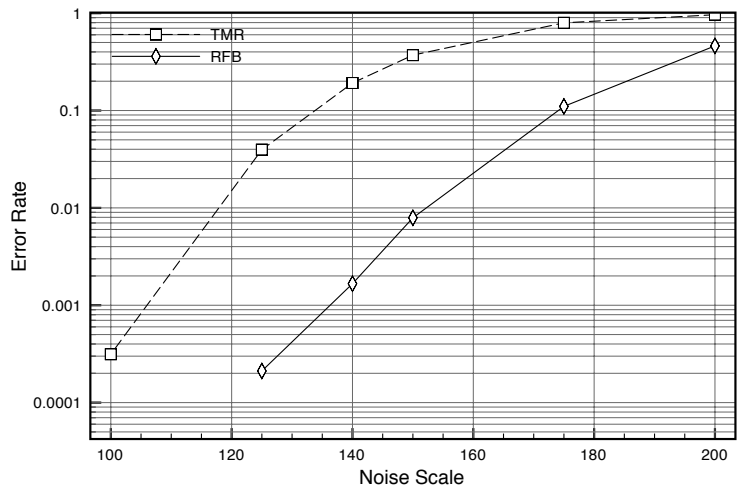


FIGURE 9
Simulated BER results obtained from Spectre. The "`noisescale`" parameter is set very high in order to induce enough errors to measure and compare the BER for the two methods.

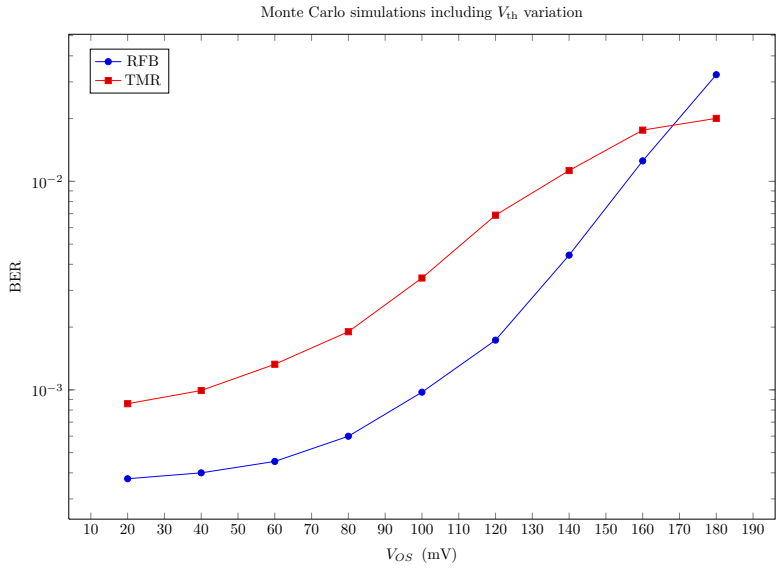


FIGURE 10
Simulated BER results obtained from a Spectre Monte Carlo simulation that includes variation in device threshold voltages. The "`noisescale`" parameter is fixed at 170.

A second simulation was performed to evaluate the circuit's sensitivity to process variation. In this simulation, process variation is modelled as a Gaussian-distributed offset-voltage V_{OS} applied to the threshold voltage of each transistor. Each transistor is assumed to have an independent offset. The simulation was performed using Spectre's Monte Carlo simulation capability, and the results are shown in Figure 10.

When using the simulator in the Monte Carlo mode, it was necessary to count errors in a slightly different way from the method used to obtain Figure 9. In the Spectre Monte Carlo simulations, a single error is counted if *any* transient cross-over event occurs during a clock cycle (as opposed to counting all such crossings as separate errors).

3.3 Discussion of results.

The BER curves in Figure 9 do not perfectly correspond to those obtained using the discrete-time model in Figure 5. This is possibly due to the fact that, in the device-level CMOS implementation, the input error rate (η) and intrinsic error rate (ϵ) are both non-trivial functions of the `noisescale` parameter, so it is not possible to hold one fixed while sweeping the other. Hence the selection of noise parameters cannot be precisely matched with those used in the discrete-time model.

The differences between the discrete-time and CMOS results can be further explained by considering continuous-time analog dynamics which are accounted for in the Spectre simulation. The inputs x_i are subject to significant noise fluctuations, so that $v(x_i(t)) = v(x^*) + v_{ex,i}(t)$, where $v(x)$ indicates the voltage signal associated with x. The noise waveforms $v_{ex,i}$ may propagate to a C-element's output only if there is a *simultaneous* fluctuation in $y_i(t)$, such that $v_{ey,i}$ has the same sign as $v_{ex,i}$. Furthermore the fluctuations $v_{ex,i}$ and $v_{ey,i}$ must both be large enough to approach the inverter's threshold region. Such a double-fluctuation event is considerably more rare than any single-fluctuation event. Fluctuations in y_i are therefore suppressed, which further suppresses fluctuations in y_{i+1}, and so on.

In the TMR case, the continuous-time dynamics permit the propagation of noise fluctuation from the x_i. This is evident by considering the logic of a majority operation: $y = x_1 * (x_2 + x_3) + x_2 * x_3$. Clearly any significant double-fluctuation among the x_i will induce a fluctuation in y. For example, suppose the correct logic value is $x^* = 0$, and there is a simultaneous positive fluctuation in x_1 and x_2. In this case, we expect to observe some positive fluctuation in y. In the RFB circuit, however, the double-fluctuation in x_1, x_2 has no consequence, unless there are corresponding fluctuations in y_1, y_2, which has very low probability.

Based on this discussion, the comparative BER between RFB and TMR is evidently sensitive to particular device models for noise and analog dynamics. The results are also sensitive to how "error" is defined, which may be application-dependent. In Figure 10, a single error is counted per clock cycle, regardless of the number of level crossings that may occur during the cycle. Judging from the transient simulation shown in Figure 8, the TMR method exhibits a higher rate of level-crossings throughout a clock cycle compared to RFB. By counting at most one error per cycle, the TMR error rate is made to appear much lower.

The particular BER performance is dependent on particular device models, their unique noise characteristics, and application-driven definitions of "error." The collective results nevertheless demonstrate that the RFB method is consistently superior in performance to TMR, with BER gains ranging from $4\times$ to $100\times$, depending on the particular models and definitions.

4 MULTIPLE-VALUED LOGIC IMPLEMENTATION USING SFG CIRCUITS

In the restorative feedback architecture shown in Figure 7, the input and output signals need not be binary. If the C-not and S gates are implemented using a multiple-valued logic approach, then the restorative feedback circuit provides a solution for error correction in multiple-valued logic systems. By

definition, a C-element latches the value of its inputs whenever they are equal, and retains its stored state whenever the inputs are not equal. This definition is compatible with multiple-valued operation. To demonstrate a multiple-valued implementation, we use the recharged semi-floating gate (SFG) approach, in which dynamic logic functions are implemented via switched capacitors and CMOS inverters [1, 6, 9]. In this paper, we use previously described SFG designs for sample-and-hold (S/H) and comparator circuits. We also introduce a new ΔV circuit, which is used in implementing the SFG C-element.

4.1 SFG circuit theory.

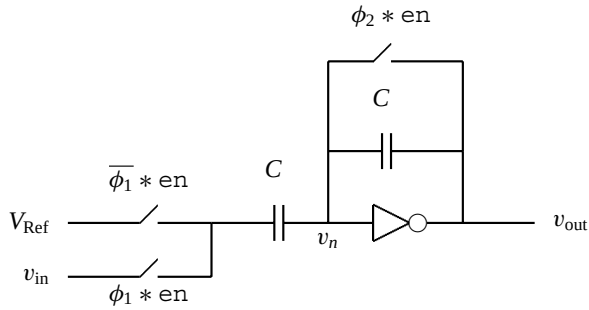
SFG circuits implement a class of dynamic multiple-valued logic. During each clock cycle, every SFG signal returns to a reference voltage level V_{Ref} for one half-cycle, and then acquires its proper logic value during the subsequent half-cycle. SFG signals therefore appear to oscillate, always returning V_{Ref} for half of each cycle. The reference voltage is equal to the cross-over voltage of a CMOS inverter. The operating principle of each SFG gate is to first balance an inverter at V_{Ref}, and then "tip" the inverter up or down (or neutral) depending on the value of input signals. Three example SFG gates are shown in Figure 11. Each gate contains an inverter and one or more capacitors. The SFG gates are switched using non-overlapping clock phases ϕ_1 and ϕ_2. In each gate, the CMOS inverter is balanced at V_{Ref} during phase ϕ_2. Logic operations occur during phase ϕ_1.

Figure 11(a) shows a S/H circuit which operates as follows. When the S/H gate is enabled (i.e. `en` is asserted), the inverter is balanced at V_{Ref} during ϕ_2, and the capacitors are discharged to zero. During this phase, $v_{\text{out}} = V_{\text{Ref}}$. Then the inverter's input node, v_n, is charged to equal V_{Ref}. Then, during ϕ_1, node v_n is left floating, so that both capacitors retain zero charge. The voltage drop across both capacitors is therefore zero, resulting in $v_{\text{out}} = v_{\text{in}}$.

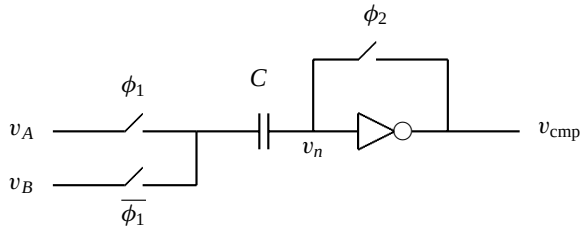
Figure 11(b) shows a ternary comparator circuit which is very similar to the S/H design. As with the S/H, the inverter is balanced at V_{Ref} during ϕ_2, and C is charged to $q_C = C\,(v_B - V_{\text{Ref}})$. Then during ϕ_1, the top-plate of C is shifted to v_A, causing the inverter's input voltage to change, $v_n \rightarrow v_A - v_B + V_{\text{Ref}}$. If $v_A > v_B$, then $v_n > V_{\text{Ref}}$, resulting in $v_{\text{cmp}} = \text{L}$ (low logic level). If $v_A = v_B$, then $v_n = V_{\text{Ref}}$, resulting in $v_{\text{cmp}} = V_{\text{Ref}}$. In the remaining case where $v_A < v_B$, $v_{\text{cmp}} = \text{H}$ (high logic level).

The ΔV circuit, is shown in Figure 11(c), yields an output equal to the difference of its inputs, $v_{\text{out}} = V_{\text{Ref}} + v_A - v_B$. During phase ϕ_2, $v_n = V_{\text{Ref}}$ and C is charged to $q_C = (v_B - V_{\text{Ref}})\,C$. During ϕ_1, v_n is left floating so that the total capacitive charge cannot change. Hence

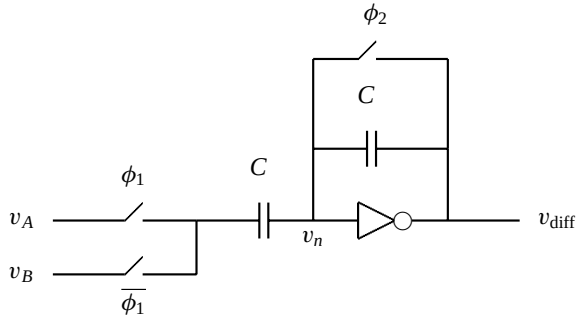
$$\begin{aligned} q_C &= (v_A - v_n)\,C + (v_n - v_{\text{diff}})\,C \\ &= (v_B - V_{\text{Ref}})\,C \\ \Rightarrow v_{\text{diff}} &= V_{\text{Ref}} + v_A - v_B. \end{aligned}$$



(a) Sample-and-Hold circuit.



(b) Comparator circuit.



(c) ΔV circuit, $v_{\text{diff}} = V_{\text{Ref}} + v_A - v_B$.

FIGURE 11
Basic SFG components used to implement a multiple-valued C-element. "V_{Ref}" refers to the value of an inverter's cross-over voltage, which represents a middle logic level between the traditional "low" and "high" values.

4.2 SFG C-element implementation.

The S/H, ΔV and Comparator components can be used to implement an SFG C-element as shown in Figure 12. In this circuit, the ΔV component is used to sense the difference between the two input voltages. The ΔV output is then compared against two thresholds, V_H and V_L. If $v_{\text{diff}} > V_H$ or $v_{\text{diff}} < V_L$, then

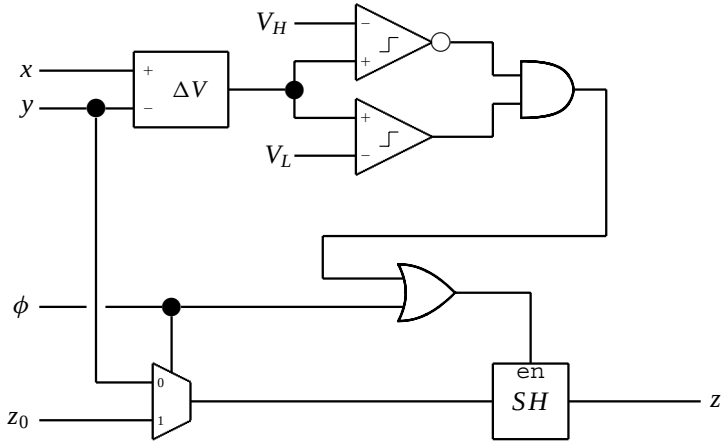


FIGURE 12
A multiple-valued C-element circuit comprised of the SFG components in Figure 11. This circuit substitutes for the binary C-element in the restorative feedback architecture shown in Figure 7.

the input signals are deemed unequal. Otherwise they are deemed equal, and the S/H component is enabled to sample the input signal. In principle, this circuit can work for arbitrarily many logic levels. In practice, the number of levels is limited by offset errors and non-linearity in the SFG components. If the spacing between logic levels is V_{step}, then the threshold voltages should be $V_H = V_{\text{Ref}} + 0.5V_{\text{step}}$ and $V_L = V_{\text{Ref}} - 0.5V_{\text{step}}$.

4.3 Simulation Results

As a demonstration of the concept, a ternary RFB circuit was designed using SFG circuits. The circuits were implemented using models for a 0.5μm CMOS technology. Transient simulation results are shown in Figure 13. In this simulation, two of the input signals (x_1 and x_2) always agree, while the third signal is in error. The input signals are stepped through all signal values and error cases. In each case, the error is corrected after one clock cycle. Valid data appears during the ϕ_1 phase, which is indicated by the check-mark (✓). During the ϕ_2 phase, the signals return to V_{Ref}, resulting in oscillatory waveforms.

5 APPLICATION TO MORE COMPLEX CIRCUITS

In the preceding sections, the RFB method was shown to significantly improve reliability when applied to isolated modules. In this section we examine the application of RFB to a more complex circuit. It will be shown

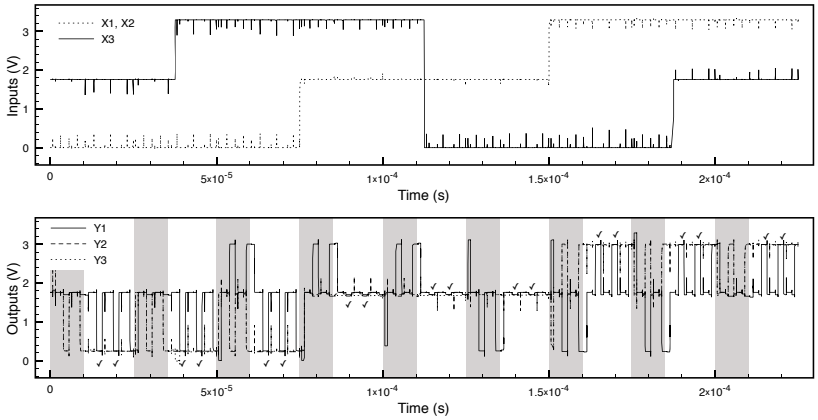


FIGURE 13
Simulations results for a ternary SFG implementation in a 0.5μm CMOS technology. The shaded intervals indicate the setup phase. The input sequence includes all single-error patterns. Signals x_1 and x_2 are correct, while x_3 is in error. The check-marks indicate the time intervals in which output signals are valid and correct. All errors are corrected after a one-cycle delay. The outputs alternate between V_{Ref} (during phase ϕ_2) and the corrected value (phase ϕ_1).

that the RFB method can sustain reliable function across a larger system composed of several RFB-protected submodules. It is difficult to perform a general analysis of large-scale circuits due to complex dependencies among logic signals in a design. In some cases a single error event may propagate to many gates, leading to a large number of correlated signal faults. In other cases an error may have no effect on the circuit's computational result. The severity of an error depends heavily on the circuit's topology, on the location of the error, and on the particular operations being performed. It is therefore desirable to validate an error-correction method in the context of important case examples, such as common arithmetic circuits, allowing for random errors to occur at any time and location within the simulation.

To demonstrate the application of RFB to a larger circuit, we use an M-level pipelined adder circuit as shown in Figure 14. Adders are used in implementing a variety of arithmetic operations, making them suitable as a demonstration case for RFB. The modules shown in Figure 14 include a half-adder (HA), full-adders (FA) and delay (D) registers. The HA and FA modules perform modulo-M operations, and the D module provides M-level storage.

If an error occurs in one of the carry signals in the pipelined adder, then the error will propagate down the carry path, potentially inducing a large number of faulty signals. To protect the circuit from errors, the RFB method is applied to all submodules, so that every object shown in Figure 14 becomes a bundle of three identical copies. Similarly, all signal wires represent bundles

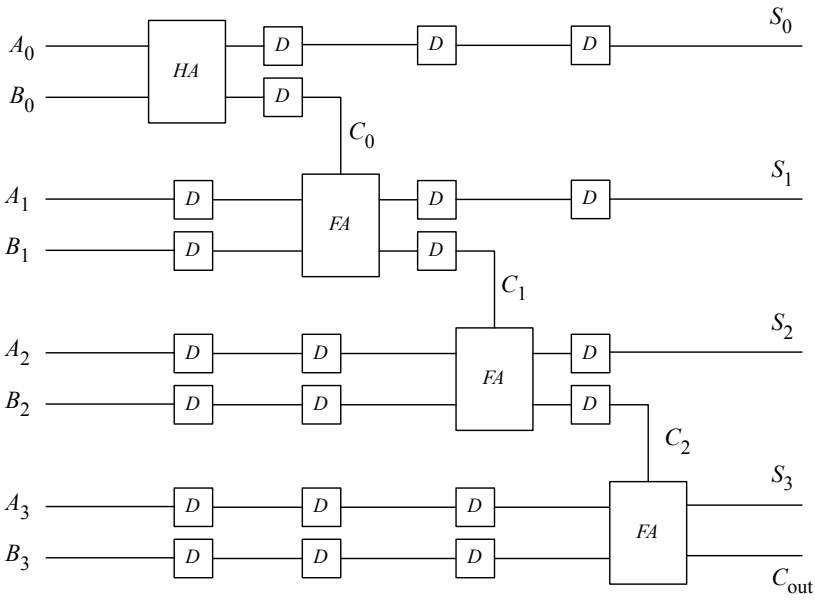


FIGURE 14
Pipelined architecture for an M-ary ripple-carry adder. Each half-adder (HA) and full-adder (FA) performs a sum modulo-M of its two inputs, i.e. $S_i = A_i + B_i \mod M$. An M-ary carry digit is also produced, which is transmitted to the subsequent adder block. The carry propagation is delayed by D registers where the RFB method may be applied to control errors in the carry path. In the RFB implementation, every module and signal is triplicated and the RFB circuit is folded into the D registers.

of three identical signals. The RFB circuit is embedded at the output of the D flip-flops at each stage of the pipeline.

To simulate the transient error rate of this circuit, we assume that M-level logic is physically encoded by a signal between 0 and 1, with arbitrary units. The available signal values are $x_i^* = i/(M-1)$, where $i = 0, \ldots, M-1$. Between each pair of signals x_i, x_{i+1}, there is a threshold point. The threshold values are given by $v_i = (i+1)/2(M-1)$. At the output of any M-level module, the actual signal value is given by $x(t) = x^* + n(t)$, where x^* is the module's correct output value. The noise process $n(t)$ is assumed to be zero-mean and Gaussian distributed, so that on average $x(t) = x^*$ The same noise model is also applied to the D registers, where momentary faults induce persistent upsets.

This model was implemented using the SystemC-AMS language. The noise process $n(t)$ was applied at a rate of $f_n = 20 f_c$, where f_c is the frequency of the main simulation clock. There are consequently ten noise samples per clock phase (ϕ_1 and ϕ_2), meaning there are 10 opportunities for

momentary faults to occur during each phase. Identical simulations were performed for adder circuits with RFB and without RFB. In each simulation, errors were detected by comparing the outputs of two adders. The first adder is the device under test, in which noise is applied at each module. The second adder is an identical circuit in which the noise standard deviation is $\sigma = 0$, resulting in ideal behavior. In the RFB circuit, only uncorrectable "bundle errors" are counted when computing the Symbol Error Rate (SER). An error is not counted if only one fault appears in a signal's output bundle (such a fault will be internally corrected). An error is counted whenever two or more signals are erroneous. The the symbol error rate is given by

$$\text{SER} = \frac{\text{\# of bundle errors}}{(\text{\# of operations performed}) \times (\text{\# of symbols per operation})}.$$

The SER simulation results are shown in Figure 15. In this figure, the SER is computed while sweeping the noise power of $n(t)$. The noise power is a function of the variance, and is computed by $10 \log\left(2\sigma^2\right)$. The performance gain is measured as the difference in noise power between the RFB and unprotected adders when operating at the same SER. From the data in Figure 15, it is clear that the RFB method yields a gain of about 1dB in the two-level and four-level cases. The gain is closer to 0.5dB in the eight-level case. These results indicate that the RFB method is most beneficial when M is small, with diminishing gains as M is increased.

6 CONCLUSIONS

The Restorative Feedback method provides a superior alternative to TMR for a variety of design cases, particularly those where transient upsets represent the dominant fault species. The feedback topology was shown to suppress transient fluctuations that may occur in the correction circuit itself, and masks transient upsets that may occur on the input signals. Furthermore the RFB method has well-defined application to non-binary logic systems. A binary implementation was proposed, and was shown via device-level simulations in Spectre to achieve a significantly better error rate than TMR. A ternary logic implementation was also presented, based on Semi-Floating Gate circuit techniques. The ternary implementation was shown to correct all input error patterns.

The two-phase design of the RFB circuit is compatible with dynamic logic families, including traditional dynamic CMOS logic and multiple-valued SFG circuits. To demonstrate the application of RFB to a dynamic logic design, a pipelined M-ary ripple carry adder circuit was simulated using a

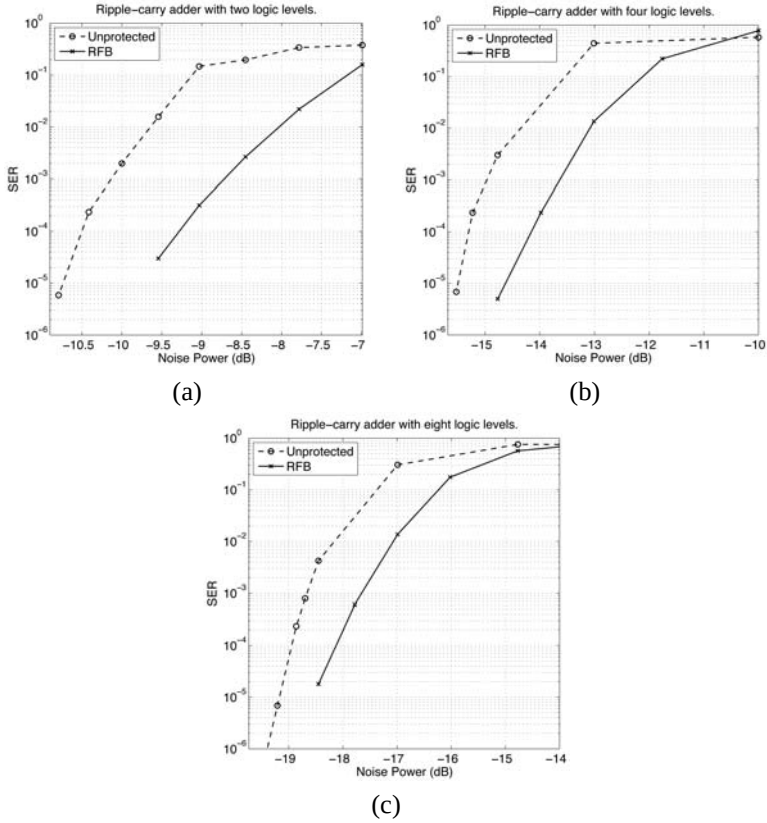


FIGURE 15
SER results for M-ary ripple-carry adder circuits with four digits. Performance gain is measured as the difference in noise power between protected and unprotected circuits at SER$= 10^{-5}$. Three cases are shown: (a) $M = 2$ (the binary case), with a performance gain of 1dB, (b) $M = 4$, with a gain of 1dB, and (c) $M = 8$, with a gain of 0.5dB.

high-level model. The RFB method was found to provide significant performance gain in all cases, although the benefits are reduced for larger M. Simulation results also indicated that the best error performance is obtained after some delay, which is not required in the TMR method. There is consequently a tradeoff between speed and error rate in the RFB method.

ACKNOWLEDGMENTS

The authors would like to thank Chris Myers, of the University of Utah, for his helpful suggestions on this topic. This work was supported by the

US National Science Foundation under awards CCF-0916105 and ECCS-0954747.

NOMENCLATURE

6.1 Symbols

M	The number of discrete logic levels used in an M-ary logic circuit.
$\mathcal{X}$	A discrete alphabet containing M levels, or symbols. In most cases $\mathcal{M} = \{0,\ 1,\ \ldots,\ M-1\}$.
$\mathbf{X}$	A discrete random variable with values from $\mathcal{X}$.
$\rho_X\,(x,\ t)$	The probability mass function for $\mathbf{X}$. May also be written as $\rho_X\,(x)$ if the function is time-invariant.
$x\,(t)$	A stochastic process representing independent samples of random variable $\mathbf{X}$, with distribution $\rho_X\,(x,\ t)$.
p_x	The probability that $x\,(t) = 1$. This symbol is used only when $\mathcal{X} = \mathbb{Z}_2$.
$\overline{x\,(t)}$	A real-valued time average of $x\,(t)$.
x^*	The "true" or correct value for $x\,(t)$.
α	A normalizing constant used in the sum-product algorithm.
l_x	Likelihood ratio, $l_x = \Pr(\mathbf{X} = x^*) \,/\, \Pr(\mathbf{X} \neq x^*)$.
L_x	Log-likelihood ratio $L_x = \log l_x$.
$e_{xi}\,(t)$	Discrete error process for signal x_i, so that $x_i\,(t) = x^* + e_{xi}\,(t)$.
$v_{ex,i}\,(t)$	Analog error process for signal x_i, so that $v\,(x, i\,(t)) = v\,(x^*) + e_{xi}\,(t)$.
$\mathbb{N}_0$	The set of non-negative integers.
$x_1,\ x_2,\ x_3$	Input signals to a TMR or RFB circuit.
$y_1,\ y_2,\ y_3$	Output signals from a TMR or RFB circuit.
η	Rate of errors (errors per time unit) that occur among the x_i inputs to a TMR or RFB circuit.
ϵ	Rate of momentary errors (errors per time unit) that occur at the output of an individual gate (C-element or Majority) in a TMR or RFB circuit.
δ	Rate of persistent errors (errors per time unit) that occur due to upsets in a C-element's state memory.
$\phi_1,\ \phi_2$	Non-overlapping clock phases (i.e. there is no time when both ϕ_1 and ϕ_2 are active).
V_{Ref}	Cross-over voltage, where $v_{\mathrm{out}} = v_{\mathrm{in}}$, for a standard CMOS inverter used in an SFG logic circuit.
v_{cmp}	Output from a ternary SFG comparator circuit, $v_{\mathrm{cmp}} \in \{0,\ V_{\mathrm{Ref}},\ v_{\mathrm{DD}}\}$.

ΔV	Name of an SFG circuit which computes the difference between two voltages v_A and v_B.
v_{diff}	Output from the ΔV circuit, $v_{\text{diff}} = V_{\text{Ref}} + v_A - v_B$.
`en`	An "enable" signal which is true when high.
`noisescale`	Parameter that specifies the noise multiplier for analog simulation in Spectre.
$n(t)$	Gaussian noise process applied to the output of multi-level gates for simulating ripple-carry adders.
σ	Standard deviation of $n(t)$. The noise power is $10 \log\left(2\sigma^2\right)$.

6.2 Abbreviations

TMR	Triple-Modular Redundancy method.
RFB	Restorative Feedback Method.
BER	Bit Error Rate (errors per clock cycle per bit).
SER	Bit Error Rate (errors per clock cycle per M-ary symbol).
SFG	Semi-Floating Gate logic.
CMOS	Complementary Metal Oxide Semiconductor transistors.
S/H	Sample-and-Hold circuit.
HA	Half-adder.
FA	Full-adder.
D	Delay register.

REFERENCES

[1] Y. Berg, S. Aunet, O. Minnotahari, and M. Hovin. (May 2003). Novel recharge semi-floating-gate cmos logic for multiple-valued systems. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 5, pages V–193–V–196 vol. 5.

[2] G.D. Forney. (February 2001). Codes on graphs: normal realizations. *IEEE Trans. on Infrom. Theory*, pages 520–548.

[3] V.C. Gaudet and A Rapley. (2003). Iterative decoding using stochastic computation. *Electronics Letters*, 39(3):299–301.

[4] Wonjin Jang and A.J. Martin. (March 2005). SEU-tolerant QDI circuits. *Asynchronous Circuits and Systems, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on*, pages 156–165.

[5] Wonjin Jang and Alain J. Martin. (April 2005). Soft-error robustness in QDI circuits. In *Proc. 1st Workshop on System Effects of Logic Soft Errors*.

[6] R. Jensen, Y. Berg, and J.G. Lomsdalen. (2005). Semi floating-gate s/h circuits. In *NORCHIP Conference, 2005. 23rd*, pages 176–179.

[7] F. Lima Kastensmidt, L. Sterpone, L. Carro, and M. Sonza Reorda. (2005). On the optimal design of triple modular redundancy logic for sram-based fpgas. In *Proceedings of the*

conference on Design, Automation and Test in Europe - Volume 2, DATE '05, pages 1290–1295, Washington, DC, USA. IEEE Computer Society.

[8] R. E. Lyons and W. Vanderkulk. (1962). The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development*, 6(2):200–209.

[9] O. Mirmotahari and Y. Berg. (May 2004). A novel d-latch in multiple-valued semi-floating-gate recharged logic. In *Multiple-Valued Logic, 2004. Proceedings. 34th International Symposium on*, pages 210–213.

[10] S. Mitra and E.J. McCluskey. (2000). Word-voter: a new voter design for triple modular redundant systems. In *VLSI Test Symposium, 2000. Proceedings. 18th IEEE*, pages 465–470.

[11] D.E. Muller and W.S. Bertky. (1959). A theory of asynchronous circuits. In *Proc. International Symposium on the Theory of Switching, Part 1*, pages 204–243.

[12] S.R. Nanduri, A.K. El Hakeem, and A.J. Al-Khalili. (mar 1990). Fault analysis of a tmr system using multiple valued logic. In *Computers and Communications, 1990. Conference Proceedings., Ninth Annual International Phoenix Conference on*, pages 23–29.

[13] K. Nikolic, A. Sadek, and M. Forshaw. (June 2002). Fault-tolerant techniques for nanocomputers. *Nanotechnology*, 13(3):357–362.

[14] M. Shams, J.C. Ebergen, and M.I. Elmasry. (Aug 1996). A comparison of CMOS implementations of an asynchronous circuits primitive: the C-element. *Low Power Electronics and Design, 1996., International Symposium on*, pages 93–96.

[15] S. Sharifi Tehrani, W.J. Gross, and S. Mannor. (Oct. 2006). Stochastic decoding of LDPC codes. *IEEE Communications Letters*, 10(10):716–718.

[16] S. Sharifi Tehrani, W.J. Gross, and S. Mannor. (Oct. 2006). Stochastic decoding of ldpc codes. *Communications Letters, IEEE*, 10(10):716–718.

[17] S. Sharifi Tehrani, C. Jego, Bo Zhu, and W.J. Gross. (Nov. 2008). Stochastic decoding of linear block codes with high-density parity-check matrices. *Signal Processing, IEEE Transactions on*, 56(11):5733–5739.

[18] S. Sharifi Tehrani, S. Mannor, and W.J. Gross. (Nov. 2008). Fully parallel stochastic ldpc decoders. *Signal Processing, IEEE Transactions on*, 56(11):5692–5703.

[19] S.S. Tehrani, C. Winstead, W.J. Gross, S. Mannor, S.L. Howard, and V.C. Gaudet. (2010). Relaxation dynamics in stochastic iterative decoders. *Signal Processing, IEEE Transactions on*, 58(11):5955–5961.

[20] J. Vial, A. Virazel, A. Bosio, P. Girard, C. Landrault, and S. Pravossoudovitch. (2009). Is triple modular redundancy suitable for yield improvement? *Computers Digital Techniques, IET*, 3(6):581–592.

[21] J. von Neumann. (1955). Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon, editor, *Automata Studies*, pages 43–98. Princeton University Press, Princeton, NJ.

[22] J.F. Wakerly. (1976). Microcomputer reliability improvement using triple-modular redundancy. *Proceedings of the IEEE*, 64(6):889–895.

[23] C. Winstead. (10 2009). C-element multiplexing for fault-tolerant logic circuits. *Electronics Letters*, 45(19):969–970.

[24] C. Winstead, V.C. Gaudet, A. Rapley, and C. Schlegel. (2005). Stochastic iterative decoders. In *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 1116–1120.

[25] C. Winstead and S. Howard. (2009). A probabilistic ldpc-coded fault compensation technique for reliable nanoscale computing. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 56(6):484–488.

[26] C. Winstead, Yi Luo, E. Monzon, and A. Tejeda. (may 2011). An error correction method for binary and multiple-valued logic. In *Multiple-Valued Logic (ISMVL), 2011 41st IEEE International Symposium on*, pages 105–110.

[27] Chris Winstead. (2005). Error-control decoders and probabilistic computation. In *Tohoku Univ. 3rd SOIM-COE Conference*.

[28] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, and S. J. Patel. (2006). Sequential element design with built-in soft error resilience. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(12):1368–1378.

[29] Wei Zhao and Yu Cao. (nov. 2006). New generation of predictive technology model for sub-45 nm early design exploration. *Electron Devices, IEEE Transactions on*, 53(11):2816–2823.