Fault-Tolerant Ripple-Carry Binary Adder using Partial Triple Modular Redundancy (PTMR)

Rahul Parhi
Wayzata High School
Plymouth, MN 55446
Email: parhi003@umn.edu

Chris H. Kim
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455
Email: chriskim@umn.edu

Keshab K. Parhi
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455
Email: parhi@umn.edu

***Abstract*—Integrated circuit chips fabricated using nano-scale CMOS technologies will be prone to errors caused by fluctuations in threshold voltage, supply voltage, electromigration, random dopant fluctuations, aging, timing errors and soft errors. Design of nano-scale failure-resistant systems has drawn significant interest in past few years. One common approach to reducing errors is the use of triple modular redundancy (TMR). The hardware overhead associated with TMR is significantly high. This paper presents a novel *partial triple modular redundancy* (PTMR) approach that achieves the same or better fault-tolerance as that of TMR but with significantly less hardware overhead. In a weighted number system, the most significant bits carry greater weight and preserving these bits is more critical than the lower significant bits. In PTMR, only the P most significant bits of the result are computed using TMR as opposed to all the W bits, where W represents the word-length of the operands. The proposed PTMR approach is illustrated in the context of a ripple-carry adder. It is shown that the hardware overhead can be reduced by 75% to 87.5% with $P = 4$ as the word-length varies from 16 to 32, with average error power equal to or less than that of TMR. It is shown that $P = 3$ or 4 is sufficient for word-lengths varying from 16 to 32.**

I. Introduction

One common approach to reducing numerical errors due to faults is the use of triple modular redundancy (TMR) [1]. TMR was first introduced by Von Neumann in 1956 [1]. In triple modular redundancy, the same function is computed three times and a majority vote of these outputs is used as the final output. TMR has been used to design single-event upset (SEU) tolerant systems using field-programmable gate arrays (FPGAs) in [2], [3], [4]. Computing the same function three times requires a significant increase in hardware. One proposed method to reduce the hardware overhead of TMR in random logic computations is the use of selective TMR (STMR) [3][4]. However, no approach has yet been presented to reduce hardware overhead in TMR based arithmetic computing systems.

This paper addresses the design of arithmetic computing systems that can achieve the same reliabilty as that of TMR systems but with significantly less overhead. A novel method of redundancy, referred to as *partial triple modular redundancy* (PTMR), is presented in this paper. The underlying principle of PTMR is to use TMR only for few most significant bits (MSBs) of the computation instead of computing all the bits in triplicate. The reasoning behind this approach lies in the fact that the weights of the MSBs are significantly greater than that of the lower significant bits (LSBs). In digital computers, all numbers are represented using a weighted binary number representation. For example, in a 16-bit unsigned number, the weight of the MSB is $2^{15} = 32768$ while the weight of the LSB is $2^0 = 1$; thus, the error caused by the flipping of this bit is 32768 times the error caused by the flipping of the LSB. In addition, more gates using TMR are likely to fail with higher probability.

The paper is organized as follows. Section II introduces the ripple-carry binary adder, and the TMR and PTMR adders. Section III presents the experimental results of the reliability and hardware overhead of different adders. Section IV presents an analysis and discussion of the results.

II. Methods

The conventional ripple-carry adder of word length W is built with W one-bit adders. This one-bit adder adds the two input bits a_i and b_i with the input carry bit c_i. This generates the sum bit s_i and output carry c_{i+1} according to the equations:

$$g_i = a_i b_i \tag{1}$$

$$p_i = a_i \oplus b_i \tag{2}$$

$$s_i = p_i \oplus c_i \tag{3}$$

$$c_{i+1} = g_i + c_i p_i \tag{4}$$

The variables g_i and p_i represent the carry-generate and the carry-propagate variables, respectively. A ripple-carry adder of word length W is shown in Fig. 1.

A. TMR and PTMR

Triple modular redundancy (TMR) involves computing in triplicate followed by a majority vote. The voting circuit processes three binary inputs and generates one output that represents the majority of the inputs. The voting circuit implements the logic function:

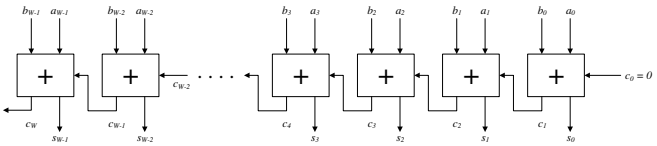

Fig. 1. A ripple-carry adder (RCA) with word length W. Each block implements the one-bit binary adder shown in Figure 1.

$$v = xy + xz + yz \tag{5}$$

where v is the majority vote of bits x, y, and z.

1) TMR Based RCA (TMR): A commonly used form of fault-tolerance is through triple modular redundancy. In the context of a ripple-carry adder, the sum of two binary numbers of word length W is computed three times and the three corresponding sum bits are then voted using the vote logic in (5). This adder is referred to as TMR and is illustrated in Fig. 2.

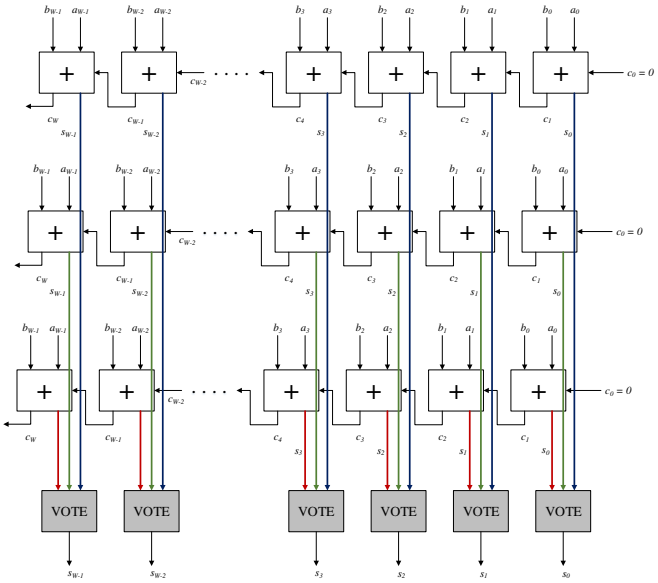

Fig. 2. A triple modular redundancy based ripple-carry adder (TMR).

2) Partial TMR Based RCA (PTMR): Similar to conventional TMR, this *partial* TMR based RCA computes the P MSBs of the sum three times, where $P < W$. These P bits are then voted using the voting circuit to compute the P MSBs of the result. The $(W-P)$ LSBs are computed only once. This is referred to as PTMR and is illustrated in Fig. 3.

Table I analyzes the number of computational and voting gates in the three adders: RCA, TMR and PTMR. The overhead gates represent the overhead due to redundancy. The ratio r represents the ratio of the number voting gates to the number of computational gates. The voting gates are likely to fail at r times the failure rate of the computational gates due to soft errors. A smaller r implies a smaller probability of failure of a voting circuit.

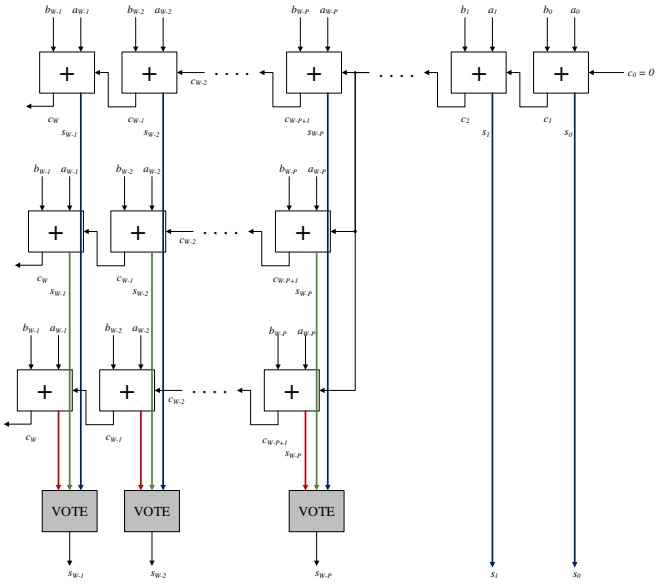

Fig. 3. A ripple-carry adder using partial triple modular redundancy (PTMR).

TABLE I
NUMBER OF GATES AND HARDWARE OVERHEAD IN EACH ADDER IN TERMS OF W AND P

Adder	RCA	TMR	PTMR
# of Gates	$5W$	$19W$	$5W + 14P$
# of Overhead Gates	0	$14W$	$14P$
# of Voting Gates	0	$4W$	$4P$
# of Computational Logic Gates	$5W$	$15W$	$5W + 10P$
Ratio r	0	$\frac{4}{15}$	$\frac{4P}{5W+10P}$

B. Fault Simulation

A bit-level simulator was developed to simulate faults at logic gate level. The failure rate of each logic gate is a pre-specified parameter that is input to the simulator. If an α–particle hits the integrated circuit, one or more logic gates will be affected. The probability that a voting circuit is affected is significantly less than the computational logic gates, as the total number of gates for voting circuits is much less than that of the total computational gates. The failure rate of a voting gate is r times that of the computational gate. The failure rate of the entire voting circuit can be shown to be $23qr/8$ where q is the failure rate of each gate. Define R as:

$$R = \frac{\text{Failure rate of the voting circuit}}{\text{Failure rate of computational logic gates}} = \frac{q_v}{q} = \frac{23r}{8}. \tag{6}$$

In this paper, simulation results are presented for $R = 0.5$ or 0.1. These two values of R do not consider the effect of r. The simulation results are used to validate the results of the theoretical analysis (not presented in this paper due to lack of space). The theoretical analysis is carried out to take into account the effect of r for each adder and the SER values are derived. It is important to point out that no prior work has considered different failure rates for combinational gates and

for voting circuits. Most prior results are based on ideal voting circuits that never fail. This assumption is very unrealistic.

The addends are random unsigned binary numbers where each bit is equally likely to be either 0 or 1. The MSB of each addend is taken to be 0 to avoid overflow. Ten million Monte Carlo simulations are carried out for each experiment. The simulator can be executed with five input parameters: word-length (W), percent failure rate of each computational logic gate (q), percent failure rate of voting circuit ($q_v = Rq$), number of MSB bits to compute in triplicate (P), and number of Monte Carlo simulations (N).

C. Performace Metrics

The performance characteristics of the various adders are characterized with respect to their errors in addition. Let A and B represent the two addends, and S represent the correct sum. The incorrect sum computed by a faulty adder is represented by $\hat{S}$. Define E as the *magnitude* of the error in computing the sum, i.e., $E = |S - \hat{S}|$.

Let the number of Monte Carlo simulations be denoted as N. The fault-resiliency of a design is measured by the *signal to error power ratio* denoted as SER. The SER is defined as the ratio of the average signal power and the variance of the magnitude of the error, and is given by:

$$SER = 10\log \frac{\frac{1}{N}\sum_{j=1}^{N}|S(j)|^2}{\frac{1}{N}\sum_{j=1}^{N}|E(j)-\overline{E}|^2} = 10\log \frac{\sum_{j=1}^{N}|S(j)|^2}{\sum_{j=1}^{N}|E(j)-\overline{E}|^2}$$

where $\overline{E}$ represents the mean of the magnitude of the error, and E and SER are measured in decibels (dB). The higher the SER, the more fault-resilient the adder is. The number of logic gates of each adder design is considered as another metric.

III. Results

Table II describes the SER for different adder designs for fault rates of computational logic gates varying from 0.001% to 5% for 16-bit addition. The ratio, R, of the failure rate of the voting circuit and the failure rate of a computational logic gate is assumed to be one-half. The values of P that maximize the SER for PTMR are noted in Table II.

Table III describes the SER and the number of logic gates for different PTMR adders as P is varied from 0 to 5 for fault rates of computational logic gates varying from 0.001% to 1%. Two different R values are considered: 0.5 and 0.1. Thus, the failure rates of the voting circuit are either one-half or one-tenth of the computational logic gates.

Table IV describes the SER values for different adders as a function of W. The word-length, W, is varied from 16 to 32. In this table, the fault rate of the computational logic is 0.01% and $R = 0.5$.

TABLE II
SER of different adders vs. fault rates of combinational logic gates where $W = 16$ and $R = 0.5$

Fault %	RCA	TMR	PTMR
0.001	42.70	52.89	52.60 ($P = 5$)
0.010	32.83	42.24	42.27 ($P = 5$)
0.050	25.59	34.73	36.80 ($P = 5$)
0.100	23.00	31.19	31.32 ($P = 5$)
0.500	16.52	21.94	22.23 ($P = 4$)
1.000	13.98	17.65	18.24 ($P = 3$)
2.000	11.64	13.83	14.42 ($P = 2$)
3.000	10.29	11.78	12.52 ($P = 2$)
4.000	9.72	10.43	11.15 ($P = 2$)
5.000	9.39	9.86	10.43 ($P = 2$)

TABLE IV
Signal to error power ratio for different adders vs. word length at 0.01% failure rate for $R = 0.5$.

Word Length	RCA	TMR	PTMR
16	32.83	42.24	42.27 ($P = 5$)
24	32.78	42.39	42.18 ($P = 5$)
32	32.97	42.21	42.11 ($P = 5$)

IV. Discussion and Future Work

The computational logic gates are assumed to fail at a rate varying from 0.001% to 5%. For example, a fault rate of 0.001% implies that gates fail at the rate of 10 parts per million (ppm), i.e., in 10 million Monte Carlo runs, a specific logic gate flips in about 100 instances on average out of 10 million instances. Each instance can be thought of as an adder chip, and thus, 10 million chips for each design are simulated in this paper. Furthermore, the voting circuit is *not* assumed to be ideal in this paper. Two failure rates for the voting circuit are considered for a specified computation logic gate failure rate. The failure rates for the voting circuit are assumed to be one-half and one-tenth of that of the computational logic gate.

From Table II, it is observed that TMR has higher SER than RCA. However, the improvement in SER due to TMR varies from about 10 dB for logic gate fault of 0.001% to about 4 dB at logic gate fault rate of 1%. The advantage of TMR disappears for logic gate fault rates of 3% or higher. The performance of PTMR is the same or better than TMR with a P value of 5 at fault rates in the range of 0.001% to 0.1%. The PTMR performance is better due to the fact that it contains fewer gates for the voting circuits than that of TMR, thus having fewer errors. The observation that PTMR with $P = 3$ or 4 performs similar to or better than the TMR implies that replicating as few as 3–5 MSBs in triplicate is sufficient to protect the sum and decrease error.

Fig. 4 shows the SER of a 16-bit adder for fault rates of computational gates varying from 0.001% to 0.1% for two different values of the ratio R as the parameter P, i.e., the number of bits triplicated and voted, increases from 0 to 5. This figure illustrates that for all cases the SER approximately becomes horizontal at or before $P = 4$. Furthermore, Table IV illustrates that the performance of different adders is independent of word-length as the word-length changes from

TABLE III
THE NUMBER OF LOGIC GATES AND SER FOR DIFFERENT PTMR ADDERS AS A FUNCTION OF P FOR $R = 0.5$ AND 0.1 AND FOR COMBINATIONAL LOGIC GATE FAULT RATES VARYING FROM 0.001% TO 1%.

P	**# Gates**	**0.001%**		**0.01%**		**0.05%**		**0.1%**		**0.5%**		**1%**	
		$R = 0.5$	$R = 0.1$	$R = 0.5$	$R = 0.1$	$R = 0.5$	$R = 0.1$	$R = 0.5$	$R = 0.1$	$R = 0.5$	$R = 0.1$	$R = 0.5$	$R = 0.1$
0	80	42.70	43.15	32.83	32.85	25.96	25.94	23.00	23.01	16.52	16.51	14.25	13.97
1	94	43.89	44.33	34.18	34.58	27.13	27.58	24.22	24.62	17.60	17.98	14.98	15.33
2	108	48.56	50.16	38.48	40.19	31.51	33.04	28.46	29.99	21.15	22.44	17.85	18.94
3	122	50.64	55.20	40.85	44.66	33.85	37.11	30.63	33.65	22.20	24.09	18.24	19.55
4	136	52.30	57.74	41.82	47.17	34.63	39.19	31.16	35.26	22.24	24.17	17.99	19.25
5	150	52.60	59.62	42.27	48.46	34.80	39.69	31.32	35.53	22.12	24.02	17.84	19.00

16 to 32. Thus, PTMR with $P = 4$ is shown to be the best design with least hardware overhead and the highest SER for word lengths at least upto 32 bits.

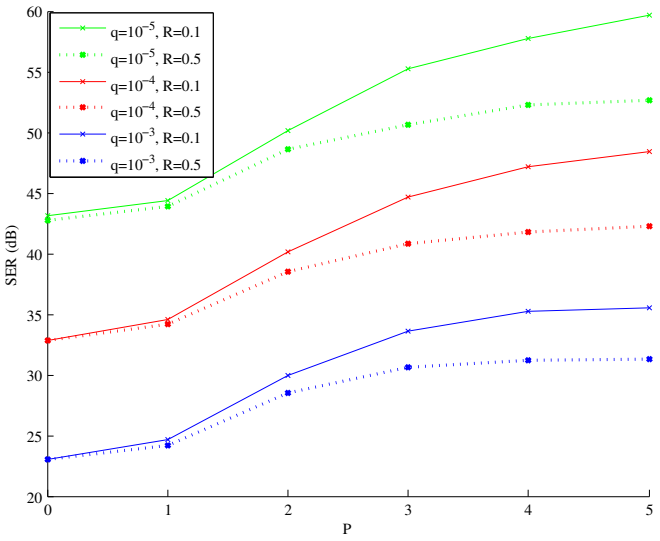

Fig. 4. SER vs. P for different fault rates for PTMR 16-bit adders.

A theoretical statistical analysis was carried out to compute SER for different adders with actual value of r for each adder from Table I; however, this analysis is not presented here due to lack of space. The SER of the PTMR is illustrated in Fig. 5 for $q = 10^{-3}, 10^{-4},$ and 10^{-5}, for a 16-bit adder. The SER of the PTMR adder is not a monotonically increasing function with respect to P, but has a maximum at $P = 3$ or 4, for very small q, independent of the word-length W. This proves that a PTMR adder is always more fault-tolerant than a TMR adder, while requiring significantly less hardware. This result is non-intuitive, but can be explained by the fact that more voting circuits are likely to be at fault with higher probability. Thus, there exists an optimal value of P. This observation is a key contribution of this paper.

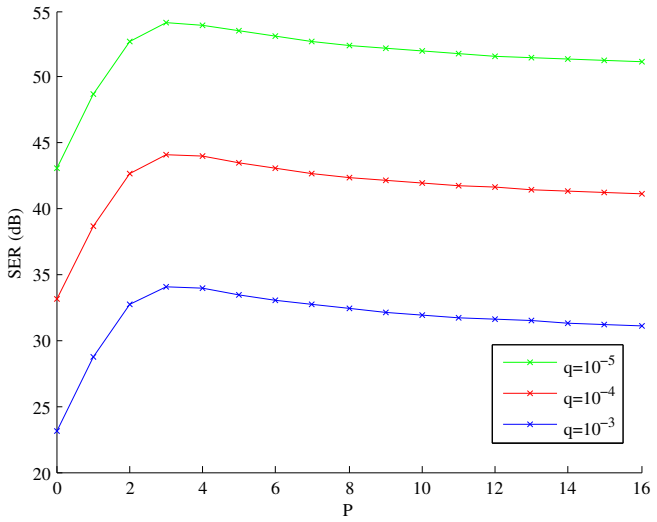

Fig. 5. SER of 16-bit PTMR adders as a function of P for three different gate failure rates.

The hardware overheads of the PTMR adder and the TMR adder are $14P$ and $14W$, respectively. The savings with PTMR as compared with TMR are given by $1 - \frac{P}{W}$. With $P = 4$, these savings are given by 75% and 87.5% for word-lengths of 16 and 32, respectively.

Future research will be directed towards the application of PTMR in fast adders such as the Kogge-Stone [5] adder, and the Wallace-tree multiplier [6].

REFERENCES

[1] J. von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, Automata Studies, pages 43–98. Princeton University Press, 1956.

[2] H. Quinn, P. Graham, and B. Pratt. An automated approach to estimating hardness assurance issues in triple-modular redundancy circuits in Xilinx FPGAs. *IEEE Transactions on Nuclear Science*, 55(6):3070–3076, December 2008.

[3] P. K. Samudrala, J. Ramos, and S. Katkoori. Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs. *IEEE Transactions on Nuclear Science*, 51(5):2957–2969, October 2004.

[4] I. Polian and J. P. Hayes. Selective hardening: Toward cost-effective error tolerance. *IEEE Design and Test of Computers*, 28(3):54–62, May/June 2011.

[5] P. M. Kogge and H. S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, C-22(8):786–793, August 1973.

[6] C. S. Wallace. A suggestion for a fast multiplier. *IEEE Transactions on Computers*, EC-13:14–17, February 1964.