Two Approximate Voting Schemes for Reliable Computing

Ke Chen, Member, Jie Han, Member Fabrizio Lombardi, Fellow IEEE

Abstract— This paper relies on the principles of inexact computing to alleviate the issues arising in static masking by voting for reliable computing in the nanoscales. Two schemes that utilize in different manners approximate voting, are proposed. The first scheme is referred to as inexact double modular redundancy (IDMR). IDMR does not resort to triplication, thus saving overhead due to modular replication. This scheme is crudely adaptive in its operation, i.e. it allows a threshold to determine the validity of the module outputs. IDMR operates by initially establishing the difference between the values of the outputs of the two modules; only if the difference is below a preset threshold, then the voter calculates the average value of the two module outputs. The second scheme (ITDMR) combines IDMR with TMR (triple modular redundancy) by using novel conditions in the comparison of the outputs of the three modules. Within an inexact framework, the majority is established using different criteria; in ITDMR, adaptive operation is carried further than IDMR to include approximate voting in a pairwise fashion. So, the validity of the three inputs is established and when only two of the three inputs satisfy the threshold condition, the IDMR operation is utilized. An extensive analysis that includes the voting circuits as well as a probabilistic framework is included. The proposed IDMR and ITDMR schemes improve the power dissipation and tolerance to variations compared to a traditional TMR. To further validate the applicability of the proposed schemes, inexact voting has been used in two applications (image processing and FIR filtering); the simulation results show that performance is substantially improved over TMR.

Index Terms— Voting, Approximate computing, Reliable system, Redundancy

1 Introduction

SOFT errors have become a major concern in the design of nanoscale digital integrated circuits [1]. A soft error may occur due to a strike by a high-energy particle and manifests itself as a transient bit reversal in the logic value of a circuit node. The bit reversal phenomenon (also commonly referred to as an event upset) can also affect the data stored in a memory as well as causing the execution of an erroneous computation. Over the years, different techniques have been proposed to protect electronic circuits against soft errors and to preserve data integrity [2].

Redundancy techniques are effective to address soft errors; they are commonly used for designing dependable systems to ensure high reliability and availability [3] [4]. One of the most effective fault-tolerant design schemes is the so-called N-modular redundancy (NMR); in a NMR scheme, N copies of a module are utilized [5]. A majority voter generates the voted output on the assumption that the number of erroneous modules is always the minority. Consider, for example, triple-modular redundancy (TMR) as the special case of NMR i.e. when N = 3. An error is detected if the outputs of the modules differ. The error is corrected by voting, i.e., taking the majority value as the correct result. This approach is effective when the rate of occurrence of soft errors is low and therefore, the probability of two modules both affected by soft errors is unlikely [6].

A well-known alternative to TMR is double modular redundancy (DMR), i.e. the original module is duplicated. This scheme reduces the cost of redundancy by providing error detection; however, error correction is not always possible, because comparison cannot always establish the erroneous module and therefore, additional circuitry is needed [5].

In general, redundancy approaches are best applicable provided failure independence is retained in the operations of the modules [5]. This assumption avoids the so-called common mode failure (CMF) [5]. CMF is a catastrophic failure that affects multiple modules in the same way. For example, if the modules are identical, the outputs, although erroneous, will be the same and the error will not be detected, so resulting in an incorrect majority. Design diversity is needed to resolve this problem, i.e. to employ different redundant implementations of the original module. Thus, when the CMF occurs, the modules can produce different outputs and the error can be detected. However, different implementations may cause small differences among the module values as outputs, thus resulting in the failure of a voting scheme such as TMR. This is caused by the strict relationship in finding the majority from the voter inputs when even slightly different values are provided. This property is often referred to as static masking and is one of the major disadvantages of a redundancy scheme [5]. Therefore, slight changes in module outputs can be encountered due to diversity, presence of soft errors and technology scaling.

Computing usually operates with a high degree of re-

- *K. Chen is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. E-mail: chen.ke1@husky.neu.edu.*
- *J. Han is with the ECE Department, University of Alberta, Edmonton, Canada. E-mail: jhan8@ualberta.ca.*
- *F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. E-mail: lombardi@ece.neu.edu.*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2653780, IEEE Transactions on Computers

liability and precision. However, many applications such as multimedia and image processing can tolerate errors and imprecision in computation and still produce meaningful and useful results [7]. Accurate and precise models and algorithms are not always suitable or efficient for use in these applications. The paradigm of inexact computation relies on relaxing fully precise and completely deterministic building modules when for example, designing energy-efficient systems. This allows imprecise (or inexact) computation to redirect the existing design process of digital circuits and systems by taking advantage of a decrease in complexity and cost with possibly a potential increase in performance and power efficiency [8]. Approximate (or inexact) computing relies on using this property to design simplified, yet approximate circuits operating at higher performance level and/or lower power consumption compared with precise (exact) logic circuits [7].

Approximate computing has been extensively applied to arithmetic circuits. Addition and multiplication are widely used operations in computer arithmetic; so, full-adder cells have been analyzed for approximate computing [9] [10]. [7] has compared these types of adder and proposed several new metrics for evaluating approximate and probabilistic adders with respect to unified figures of merit for design assessment of inexact computing under various applications. The tradeoff between precision and power has also been quantitatively evaluated in [7]. Inexact voting can be used in a redundant scheme with relaxed precision requirements, because an inexact voter offers advantages for toleranting and approximately correcting errors. However, when exactness and a precise result are strict requirements, an inexact voter may not be suitable.

This paper relies on the principles of inexact computing to alleviate the issues arising in static masking by voting. Two schemes that utilize in different manners approximate voting are proposed. The first scheme whose operation was initially proposed by the same authors in [11], is referred to as inexact double modular redundancy (IDMR). IDMR does not resort to triplication, thus saving overhead due to modular replication; this scheme is crudely adaptive in its operation, i.e. it allows a threshold to determine the validity of the module outputs. IDMR operates by initially establishing the difference between the values of the outputs of the two modules; only if the difference is below a preset threshold, then the voter calculates the average value of the two module outputs.

The second scheme combines IDMR with TMR by using novel conditions in the comparison of the outputs of the three modules, i.e. ITDMR. Within an inexact framework, the majority is established using different criteria; in ITDMR, adaptive operation is carried further than IDMR to include approximate voting in a pair wise fashion. The validity of the three inputs from the modules is established and if only two of the three inputs satisfy the threshold condition, then IDMR operation is utilized. Different applications of the proposed voting schemes are investigated in depth; an assessment of image processing and filtering using the proposed schemes is presented to show the quantitative and qualitative features of approximate voting for reliable computing.

2 Review

A brief review of redundant schemes for reliable computing is then pursued next; previous works on duplication (as well as majority voting and variants of it) are presented as relevant to the proposed techniques based on approximate voting.

2.1 DMR Scheme

In DMR, the outputs of two modules are compared as shown in Fig.1; an error is detected if the outputs differ, therefore, a traditional DMR does not provide error correction. [12] Recently, the use of design diversity within DMR has been investigated to provide low cost detection and correction of radiation-induced soft errors [13]. The principle of this approach is that the two modules are

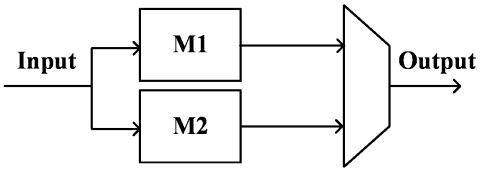

Fig. 1. DMR Scheme

implemented using designs that provide different output error patterns when a soft error hits. These error patterns can be detected as a series of mismatches between the module outputs; by recognizing these patterns, the module-in-error can be identified and the output from the other module is used as the final error-protected output. Thus, this approach is application dependent, because error detection and correction require a dedicated unit that intelligently assesses the outputs of the two redundant modules.

2.2 Majority Voter

Triple modular redundancy (TMR) uses three copies of the original module [5]. In a TMR, each module operates in a disjoint (independent) mode; so, the three modules compute in parallel. If a module produces an output that is different from the outputs of the other two modules, then the system output is established by voting. Voting assumes the majority of the modules to have the correct output; hence, the single disagreeing module (corresponding to the erroneous output) is masked by utilizing a voter.

TABLE I
EXAMPLES OF VOTING IN A TMR

Module	Error-free Scenario Output Value (Z_1 Z_2)	Erroneous Scenario Output Value (Z_1 Z_2)
1	01	10
2	01	11
3	01	01
Bit-wise Voting	01	11
Word-wise Voting	01	No majority

In a bit-wise voter, majority voting is performed on a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2653780, IEEE Transactions on Computers

bit-by-bit basis; as an example, the TMR outputs are listed in Table I for two-bit module outputs. In bit-wise voting, the voter compares each bit; it then finds the majority of each bit to form the final output value (in Table I, the correct output is 01). So if for example there are bit errors in modules 1 and 2, then the output of the bit-wise voter is 11. It is well known that the bit-wise TMR voter has bad performance for data integrity [14].

A word-voter has been proposed in [14] (Fig.2); in this scheme, voting considers the entire word, i.e. a word majority voter requires the output signals to be exactly the same when calculating the majority (as its output). The

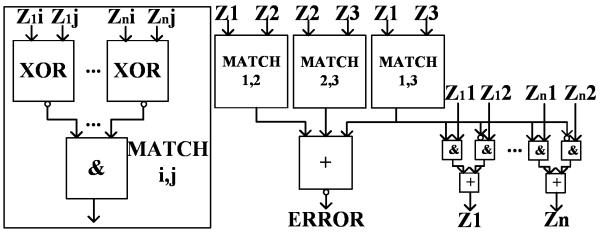

Fig. 2. Word-wise voter proposed in [14]

MATCH module compares every pair of the 3 input words. If the signals in an input pair are exactly the same, the MATCH module generates a match signal. Only when none of the three pairs generates a match signal, then the error signal is '1'. Otherwise, the output is given by the majority of the inputs; however, this type of voter is not efficient when there are slight variations in the outputs of the modules (as often occurring in the nanoscales).

2.3 Other Approximate TMR Schemes

In a conventional TMR scheme, three modules compute the same function (albeit implementation diversity is usually employed to ensure statistical independence). In [15], a diversity-based TMR scheme is proposed; it employs three different implementations of a module with the same function to prevent the so-called common mode failure. In [16][17], a novel TMR scheme is proposed; unlike a conventional TMR, only a module computes the original function, the other two additional modules implement approximate functions. The first (second) module computes a so-called under-approximation (over-approximation) of the original function.

Therefore, the outputs from the modules with the approximations combined with the output of the original function module are used to mask an error that occurs in the two approximate modules and some errors of the original function module. This technique is generalizing method which is particularly suitable for FPGAs for implementing the under/over-approximations required for the two modules; good reductions in implementation area and power have been reported, while still retaining a high reliability. However, this approach is only suitable for programmable systems due to the requirement of the under/over-approximations.

3 INEXACT DMR (IDMR)

In this section, the first inexact voting scheme is considered; an initial analysis was pursued in [11]. This scheme is referred to as *inexact double modular redundancy* (IDMR) [11]. The basic principle of the IDMR is to initially estab-

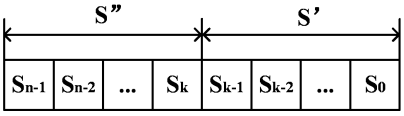

Fig. 3. Input data S

lish the difference between the outputs of two modules. If the difference is less than a *preset threshold*, the voter calculates the average value of the two module outputs as outcome. If the difference is larger than the threshold, the voter generates an error signal. The value of the threshold is dependent on the level of accuracy that is required as

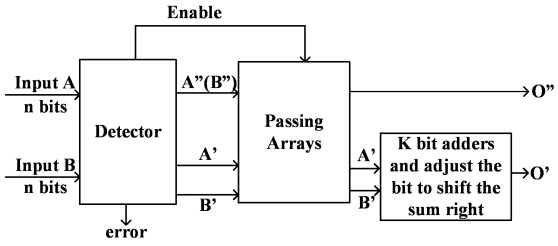

Fig. 4. IDMR voting scheme hardware

output in a reliable computing system. This scheme is crudely adaptive [11] in its operation, i.e. it allows a threshold to determine the validity of the module outputs. The averaging of the two module outputs ensures that variation in values is mediated by adjusting the final value as outcome. IDMR does not resort to triplication, thus saving overhead due to a smaller modular replication.

Let the input (parallel) data word be denoted by S; this word is made of n bits. Let the subset of the lower k bits be denoted as S', while the upper n-k bits be given by the subset S", i.e. S=(S",S') as shown in Fig.3. The block diagram of the IDMR scheme [11] is shown in Fig.4; IDMR consists of the blocks as discussed next.

3.1 Detector

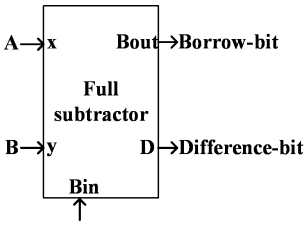

Fig. 5. Subtractor structure

The function of the detector block is to compare its two input signals corresponding to the two received outputs from the modules. An error signal is generated if the difference of the two values (denoted as Input A and Input B) is larger than the threshold. Else, the detector considers the two values to be valid and the following two cases are applicable:

- When the upper n-k bits of Input A and Input B are the same (i.e. A"=B"), then the largest possible difference between them is 2^k-1 .

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2653780, IEEE Transactions on Computers

- If the absolute value of A"-B" (i.e $|A"-B"|$)is 1, then the largest difference is 2^{k+1}-1 .

Thus, the detector is designed by utilizing a n-k bit subtractor (Fig. 5) to find A"-B" . Three scenarios are possible as validity conditions.

- A"-B"=0 : In this case, the value of the borrow of the first bit is '0'. The difference of each bit is also '0'.
- A"-B"=1: In this case, the value of the borrow of the first bit is '0'. The difference of each bit is '0' except the last bit.
- A"-B"= -1 : In this case, the value of the borrow of the first bit is '1'. The difference of each bit is also '1'.

Consider n=8, k=4; three examples for A" and B" are shown in Table II.

TABLE II
Examples of IDMR

	Example 1	Example 2	Example 3
A"	1010	1011	1010
B"	1010	1010	1011
A"-B"	B=0,D=0000	B=0,D=0001	B=1,D=1111

Let $D"=A"-B"=(B_{n-1})\overline{D}_{n-1}\overline{D}_{n-2}...\overline{D}_{k+1}\overline{D}_k$ where B_{n-1} is the borrow bit of the most significant bit; for the inputs to be valid, the following condition must be met.

$0=B_{n-1}+D_{n-1}+D_{n-2}...+D_{k+1}$ for scenarios 1 and 2 or $1=B_{n-1}\overline{D}_{n-1}\overline{D}_{n-2}...\overline{D}_{k+1}\overline{D}_k$ for scenario 3

Fig. 6. AND gate in passing array

Thus,

$$\overline{B_{n-1}+D_{n-1}+D_{n-2}...+D_{k+1}}+B_{n-1}\overline{D}_{n-1}\overline{D}_{n-2}...\overline{D}_{k+1}\overline{D}_k=1 \quad (1)$$

If the two inputs are valid, then the error signal is negated, i.e. it is given by '0'.

3.2 Passing Array

If following subtraction there is no error signal, the input must be propagated for further processing. Thus, an array made of AND gates (referred to as the passing array) is needed; this array is controlled by the Enable signal (Fig.6). When the Enable signal is '1', the inputs are propagated; else, no propagation is allowed.

If no error occurs, the output for the upper n-k bits of the AND gates is given by O"=A"=B" , i.e. each of the upper n-k bits at the output (denoted by Out) is equal to the corresponding Input A (or B) bit. So,

$$Out_n = Enable \cdot Input_n \quad (2)$$

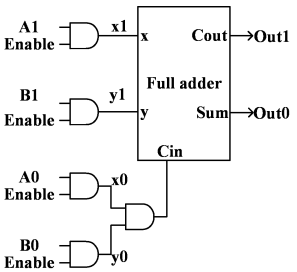

Fig. 7. Word-wise voter proposed in [14]

3.3 Full Adder

Let O' denote the average value of A' and B' . In the proposed design, full adders are used to calculate the sum of A' and B' . The average is found by shifting right the sum. However, the shift circuit is not necessary, because the first k bits (inclusive of the carry bit for k-1) can be used as result for this operation. For the last bit, only the carry bit needs to be considered; thus, a NAND gate is used to replace the last full adder. Fig.7 shows the adder structure when k=2.

4 Proposed Inexact TMR-DMR (ITDMR)

The second proposed scheme combines IDMR with TMR,

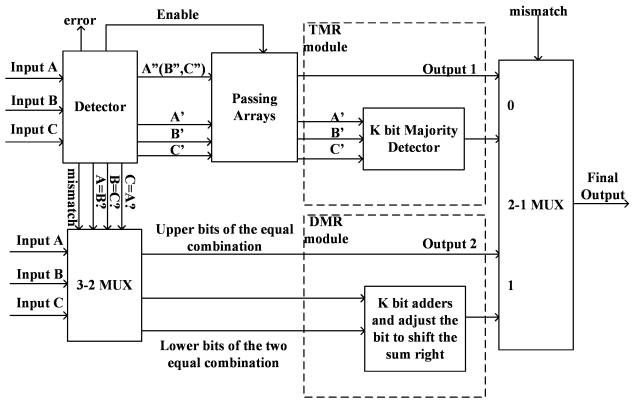

Fig. 8. Inexact TMR-DMR (ITDMR) voting scheme hardware

but it uses novel conditions in the comparison of the outputs of the three modules. Different from a DMR voter, a TMR voter compares three inputs to establish the majority as the correct output. Within an inexact framework, the majority is established using different criteria. As for IDMR, the validity of the inputs must be established first; in the proposed scheme, if the differences between all three pairs of inputs are not larger than the threshold, then the three inputs are considered valid. However, in some cases, only two of the three inputs satisfy the threshold condition, so a different scheme must be used. Hence in ITDMR, adaptive operation involves TMR with approximate voting in a pair-wise fashion followed by IDMR as a further voting configuration. Fig.8 shows ITDMR in block diagram form.

4.1 Detector

The detector compares the upper bits for the three module outputs (A, B and C). They are given by (A" , B"), (B" , C") and (C" , A"), i.e. on a pair-wise basis. For each pair, if the absolute value of the difference is less than 1, then the corresponding detection signal (AB, BC and CA) is generated (of '0' value in this design). The following cases are therefore possible for the operation of the detector.

- If $|A"-B"| \le 1, |A"-C"| \le 1, |B"-C"| \le 1$, the passing array propagates all three of these signals as inputs

to the TMR voter. The k bits majority voter determines the lower k bits; the mismatch and error signals are both '0'.

- If $|A"-B"| \leq 1$, $|A"-C"|>1$, $|B"-C"|>1$, the detector identifies the validity of A" and B" and generates the signal AB. This is provided to the 3-2 MUX. The mismatch signal is '1', while the enable signal is '0'; so, the TMR is disabled, i.e. there is no error and therefore, the error signal is '0'.
- If two of the three detection signals are valid, the following rules are used: (1) If AB and BC are valid, AB is chosen; (2) If AB and CA are valid, CA is chosen; (3) If BC and CA are valid, BC is chosen. The mismatch signal is '1' and the error signal is '0'.
- If $|A"-B"|>1$, $|A"-C"|>1$, $|B"-C"|>1$, the error signal is '1'.

4.2 3-2 MUX

The 3-2 MUX is enabled when the mismatch signal is '1' and the error signal is '0'. The 3-2 MUX utilizes the three module outputs (A, B, C) and the three flag signals (A=B , B=C , C=A) as inputs. The outputs are two equality signals based on the values of the three flag signals (A=B , B=C , C=A). There are two scenarios when the 3-2 MUX is used: 1. Only one detection signal is valid. For example, the flag A=B is '0' (0 is the valid condition), then the two outputs are A and B. 2. Two detection signals are valid. The 3-2 MUX follows the rules presented in Section 4.1. The output for each case is listed in Table III.

4.3 2-1 MUX

The 2-1 MUX is controlled by the mismatch signal; if the mismatch signal is '1', the final output is the voted output by the DMR. Otherwise, the final output is the TMR output.

TABLE III
Output of 3-2 MUX

AB	BC	CA	Output
0	1	1	AB
0	0	1	AB
1	0	1	BC
1	0	0	BC
1	1	0	CA
0	1	0	CA

5 Simulation Results

The designs of the proposed inexact voting schemes are evaluated in the section; PTMs at different CMOS feature sizes are used in HSPICE for the transistors.

5.1 Delay

Consider the following definitions for the delay.

- *Output delay*: The output delay is defined as the largest delay of each bit when no error is detected; so, the delay is the timing latency from inputs to the outputs of the voting hardware.
- *Enable delay*. The enable delay is defined as the time latency from the comparator to the Enable signal when no error is detected, i.e. the Enable signal is '1'.

The largest delay occurs when A"-B"= -1; in this case, D"=11...1 ; each bit in the difference between Input A and Input B (as calculated by the subtractor) must be '1'. Tables IV to VI show the output delays for IDMR, ITDMR and WordTMR by varying k and n.

TABLE IV
IDMR Output Delay

	n=8	n=16	n=24	n=32
		32nm (ns)		
k=1	10.99	12.39	15.18	20.51
k=2	8.66	10.14	13.75	18.17
k=4	6.06	7.85	11.50	15.64
k=8	-	6.72	9.94	14.28
		22nm (ns)		
k=1	8.92	10.06	12.33	16.65
k=2	7.03	8.23	11.17	14.75
k=4	4.92	6.37	9.34	12.70
k=8	-	5.46	8.07	11.60
		16nm (ns)		
k=1	7.19	8.10	9.93	13.41
k=2	5.66	6.63	8.99	11.88
k=4	3.96	5.13	7.52	10.23
k=8	-	4.39	6.50	9.34

TABLE V
ITDMR Output Delay

	n=8	n=16	n=24	n=32
		32nm (ns)		
k=1	16.74	23.07	27.65	31.07
k=2	14.44	21.88	26.12	30.35
k=4	10.66	19.87	25.74	29.36
k=8	-	18.49	24.11	28.79
		22nm (ns)		
k=1	13.88	19.13	22.92	25.76
k=2	11.97	18.14	21.65	25.16
k=4	8.84	16.47	21.34	24.34
k=8	-	15.33	19.99	23.87
		16nm (ns)		
k=1	11.35	15.64	18.75	21.07
k=2	9.79	14.83	17.71	20.58
k=4	7.23	13.47	17.45	19.91
k=8	-	12.54	16.35	19.52

TABLE VI
Word-based TMR Delay

	n=8	n=16	n=24	n=32
32nm (ns)	9.45	10.66	13.05	17.64
22nm (ns)	7.67	8.65	10.60	14.32
16nm (ns)	6.18	6.97	8.54	11.53

5.2 Power

Power dissipation has also been evaluated for IDMR, ITDMR and WordTMR. Tables VII-IX show the simulation results at different values of feature size, n and k. As expected, due to its more complex operation and sophisticated voting scheme, ITDMR incurs a nearly 100% overhead in power consumption compared to IDMR.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2653780, IEEE Transactions on Computers

TABLE VII
IDMR POWER

	n=8	n=16	n=24	n=32
		32nm (uW)		
k=1	7.9	8.7	9.8	10.4
k=2	8.6	9.7	11	12.8
k=4	10.1	10.9	13.5	19.9
k=8	-	12.6	15.8	20.8
		22nm (uW)		
k=1	6.49	7.14	8.05	8.54
k=2	7.06	7.96	9.03	10.51
k=4	8.29	8.95	11.08	16.34
k=8	-	10.34	12.97	17.9
		16nm (uW)		
k=1	4.91	5.4	6.09	6.46
k=2	5.34	6.02	6.83	7.95
k=4	6.27	6.77	8.38	12.36
k=8	-	7.82	9.81	14.5

TABLE VIII
ITDMR POWER

	n=8	n=16	n=24	n=32
		32nm (uW)		
k=1	14	15.39	17.31	18.36
k=2	15.43	16.87	18.85	19.93
k=4	17.45	18.98	20.12	21.35
k=8	-	19.88	21.1	22.59
		22nm (uW)		
k=1	11.84	13.02	14.64	15.53
k=2	13.05	14.27	15.95	16.86
k=4	14.76	16.06	17.02	18.06
k=8	-	16.82	17.85	19.11
		16nm (uW)		
k=1	8.41	9.25	10.4	11.03
k=2	9.27	10.14	11.33	11.98
k=4	10.49	11.41	12.09	12.83
k=8	-	11.95	12.68	13.58

TABLE IX
WORDTMR POWER

	n=8	n=16	n=24	n=32
32nm (uW)	16.58	18.22	20.50	21.74
22nm (uW)	14.02	15.42	17.33	18.39
16nm (uW)	9.96	10.95	12.31	13.06

This overhead is also due to the larger number of modules required for ITDMR and TMR versus IDMR, i.e. 3 versus 2. TMR incurs in the largest power dissipation (static and dynamic) as reflected by the larger circuit complexity (analyzed next).

5.3 Circuit Complexity

Consider an input of n-bits from each of the modules to the voting hardware and a k-bit threshold for the approximate schemes. Table X (XI) shows the number of transistors for each circuit in IDMR (ITDMR) as function of n and k. Table XII shows the complexity of WordTMR.

TABLE X
IDMR COMPLEXITY

Circuit	Circuit count	# of transistors
Subtractor	n-k	8 [18]
AND	1	2(n-k+1)
NOR	1	2(n-k)
Inverter	1	2
2-input NOR	1	4
2-input AND	n+k+1	6
Full adder	k-1	8 [18]

Thus, the circuit complexity of IDMR (as measured by the number of transistors required in its design) is given by

$$T(IDMR)=18n+2k+6 \quad (3)$$

TABLE X
IDMR COMPLEXITY

Circuit	Circuit count	# of transistors
Subtractor	n-k	8 [18]
AND	1	2(n-k+1)
NOR	1	2(n-k)
Inverter	1	2
2-input NOR	1	4
2-input AND	n+k+1	6
Full adder	k-1	8 [18]

The circuit complexity of ITDMR is given by

$$T(ITDMR)=54n+2k+26 \quad (4)$$

TABLE XII
WORDTMR COMPLEXITY

Circuit	Circuit count	# of transistors
2 input AND	2n	6
2 input OR	n	6
3 input NOR	1	6
2 input XNOR	3n	14
n input OR	3	2n

The circuit complexity of a Word-based TMR is given by

$$T(WordTMR)=66n+6 \quad (5)$$

TABLE XIII
VOTING CIRCUIT COMPLEXITY

	Circuit Complexity (transistor count)
WordTMR	66n+6
IDMR	18n+2k+6
ITDMR	54n+2k+26

Table XIII shows the expressions for the circuit complexity of the proposed schemes as well as WordTMR. The proposed schemes incur in a complexity smaller than TMR; the reduction is more pronounced at higher values of n. As linear with n, the circuit complexity of both proposed schemes decreases (slightly increases) with higher values of k for ITDMR (IDMR). Not surprisingly, ITDMR has a complexity higher than IDMR (but still less than TMR).

5.4 Process and supply voltage variations

Next, variations in the MOSFETs of the proposed

schemes are evaluated using Monte Carlo simulation. For Monte Carlo simulation, the process variations of a MOSFET consider the channel length and the threshold voltage. The variations in percentage for these parameters have been reported in [19] and shown in Table XIV at a 32nm feature size; as the most relevant metric in most high performance applications, the variability (in percentage) of the output delay is measured when all transistors are subject to variation. The simulation results (Table XV) show that the threshold voltage has a more pronounced effect than the channel length in the operation of an approximate or exact voting system.

Also as ITDMR is more complex than IDMR, its variability is larger too; however, the variation in output delay is mitigated by the approximate nature of the voting process in the proposed schemes. TMR has the largest percentage for variability, hence this is yet another negative feature that static masking causes.

TABLE XIV
VARIATION PERCENTAGE

	Vth	L
32nm	3%	2%

TABLE XV
VARIABILITY PERCENTAGE ON GLOBAL BEHAVIOR (OUTPUT DELAY)

3σ/μ(%)	IDMR	ITDMR	WordTMR
L	28.97%	33.54%	36.73%
V_{th}	36.91%	45.95%	47.18%

Another variation that has been analyzed for the proposed schemes, is the supply voltage, i.e. at 32nm feature size, the nominal value of the supply voltage is 0.9V. Table XVI shows the so-called critical value of the supply voltage, i.e. the least value (lower than the nominal supply value) such that the voting scheme can continue to operate correctly. Compared to IDMR, ITDMR shows a larger value of critical supply voltage, hence more dependent on this parameter. This is caused by the more complex operational modes of this scheme compared with the simpler adaptive mode of approximate operation of IDMR. However, TMR has the largest critical voltage as the worst performance.

TABLE XVI
CRITICAL SUPPLY VOLTAGE FOR 32NM

	IDMR	ITDMR	TMR
$V_{critical}$	0.778v	0.803v	**0.815v**

6 PROBABILITY ANALYSIS

Next, a probabilistic analysis is pursued for the proposed inexact schemes to assess the impact of bit- and value-wise errors on the functionality of the voting process. In the proposed approximate voters, if the difference of outputs from two modules is smaller than the threshold (as set by k) that the voters can tolerate, then the output is said to be valid. Only valid results are useful for voting in reliable computing.

6.1 Bit-wise Error

Let the number of bits of a module (as inputs to the voter) be given by n; in this analysis, it is assumed that each bit has the same probability to change (i.e. to flip due to a soft error) and every bit is independent. The flip probability is denoted by P_f. In the proposed inexact voters, only the upper bits are considered and the last k bits can be ignored. So, the validity of the inputs to the voter is assessed by calculating the difference in their upper bits, i.e. if the absolute value of the difference is less than or equal to 1, then the inputs to the voter are valid. Let A" and B" be the upper bits of the two inputs. There are 4 cases for the valid inputs.

- If A"=B", then all n-k bits of A" and B" are the same. Thus, the probability is $(1-P_f)^{n-k}$
- If A"-B"=1and the last bit of B" is '0', then the other n-k-1 bits of A" and B" are the same and the last bit of A" becomes '1'. Thus, the probability is $P_f(1-P_f)^{n-k-1}$.
- If A"-B"= -1 and the last bit of A" is '0', then the other n-k-1 bits of A" and B" are the same and the last bit of B" becomes '1'. Thus, the probability is $P_f(1-P_f)^{n-k-1}$.
- For the other cases (such as A"-B"=1 and the last bit of A" is '0' or A"-B"= -1 and the last bit of B" is '0'), the other bits of A" and B" are totally different. In these scenarios, the probability is very small; so, these cases can be ignored as a first approximation.

Therefore, the probability of a module to generate a valid result is given by

$$\begin{aligned} P_{m,bit\text{-}wise} &= (1-P_f)^{n-k} + 2\times\frac{1}{2}P_f(1-P_f)^{n-k-1} \\ &= P_f(1-P_f)^{n-k-1} \end{aligned} \tag{6}$$

6.2 Value-wise Error

In this case, the value of each input from a module to the voter must be considered in its entirety, i.e. all n bits. Assume that the error-free value of an input to the voter is equal to Q; in the presence of a soft error, a module may generate an output (then becoming an input to the voter), that is different from Q. Assume that the distribution of the output space of a module is normally distributed with mean Q and variance σ. Therefore, the probability of a module to generate a valid result is given by

$$P_{m,value\text{-}wise} = \sum_{|x-Q|\le T} P(x) \tag{7}$$

where T is the threshold (i.e. the difference in the values of the inputs) and P(x) is the probability density of a normal distribution. The following cases can be distinguished for the inexact voting schemes proposed in this manuscript

- For IDMR, the probability to generate a valid output result is given by

$$P_{IDMR,bit\text{-}wise} = P_m{}^2 = (1-P_f)^{2(n-k-1)} \tag{8}$$

$$P_{IDMR,value\text{-}wise} = P_m{}^2 = \left[\sum_{|x-Q|\le T} P(x)\right]^2 \tag{9}$$

- For ITDMR, the probability of generating a valid

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2653780, IEEE Transactions on Computers

result is given by

$$P_{ITDMR,bit\text{-}wise} = (1-P_f)^{3(n-k-1)} + 3(1-P_f)^{2(n-k-1)} \times \left[1-(1-P_f)^{n-k-1}\right] \quad (10)$$

$$P_{ITDMR,value\text{-}wise} = \left[\sum_{|x-Q|\le T} P(x)\right]^3 + 3\left[\sum_{|x-Q|\le T} P(x)\right]^2 \left\{1-\left[\sum_{|x-Q|\le T} P(x)\right]\right\} \quad (11)$$

This probability (as defined above for the inexact schemes) is hereafter referred to as the correct probability of generating an output result from voting.

TABLE XVII
CORRECT PROBABILITY FOR BIT-WISE ERRORS

	P_F =1%	P_F =5%	P_F =10%	P_F =15%
ITDMR k=1	0.990	0.827	0.547	0.319
ITDMR k=2	0.993	0.870	0.634	0.416
ITDMR k=3	0.995	0.910	0.727	0.533
ITDMR k=4	0.997	0.945	0.819	0.668
IDMR k=1	0.886	0.540	0.282	0.142
IDMR k=2	0.904	0.599	0.349	0.197
IDMR k=3	0.923	0.663	0.430	0.272
IDMR k=4	0.941	0.735	0.531	0.377

TABLE XVIII
CORRECT PROBABILITY FOR VALUE-WISE ERRORS

	σ=1	σ=2	σ=4	σ=8
ITDMR k=1	0.999	0.775	0.615	0.338
ITDMR k=2	1.000	1.000	0.937	0.651
ITDMR k=3	1.000	1.000	1.000	0.947
ITDMR k=4	1.000	1.000	1.000	0.977
IDMR k=1	1.000	0.870	0.669	0.265
IDMR k=2	1.000	1.000	0.989	0.719
IDMR k=3	1.000	1.000	1.000	0.992
IDMR k=4	1.000	1.000	1.000	0.998

Tables XVII-XVIII show the correct probability for n=8 for the bit- and value-wise errors in ITDMR and IDMR; different values are utilized for the standard deviation and P_f respectively. The correct probability increases at higher values of k and lower variance; both ITDMR and IDMR have a higher correct probability under bit-wise errors (even at a variance of 2) than value-wise errors (for example at a P_f of 0.01), thus showing their applicability to improve existing (exact) voting schemes. The results confirm that an inexact voting scheme on a probabilistic basis is very effective in providing a voted output, thus overcoming the static masking feature of an exact voter.

7 APPLICATIONS

7.1 Image Processing

In this section, image processing is considered as a first application of the proposed voting schemes. For analysis and ease of simulation, the system model is slightly changed to allow a controlled insertion of errors in module outputs. This allows a better understanding of the voting process for the underlying operations of the three modules in a redundant system, while still accounting for diversity in output values

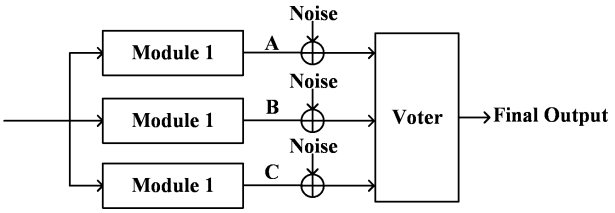

Fig. 9. Noise model of a voting scheme

for voting. The block diagram of this model for a voting scheme is shown in Fig.9. Therefore, a noise source is introduced at the outputs of each module prior to the voter. The noise sources are useful in introducing errors; if there is no error and noise, each module in a TMR generates the exact (correct) result.

7.1.1 Bit-wise noise

Bit-wise noise is defined as the noise affecting each bit of the inputs of the voting scheme with the same probability to flip (change) values. This probability is denoted by P_f (it is assumed that each bit is independent). For simulation, a random variable with a 0-1 uniform distribution is generated: if the value of this random variable is less or equal to P_f , then the corresponding bit of the input word is changed. The range of P_f in the simulations is from 0.01% to 10%. These values are higher than for example those currently encountered for soft error occurrence; however, at nanoscales bit-wise noise is not only due to external noise (such as soft and radiation-induced errors), but also to unavoidable variations in the fabrication process due to technology scaling. In either case, the increase in probability causes the occurrence of multiple errors, so the selected range is pessimistic but useful in validating the proposed schemes under very stringent conditions.

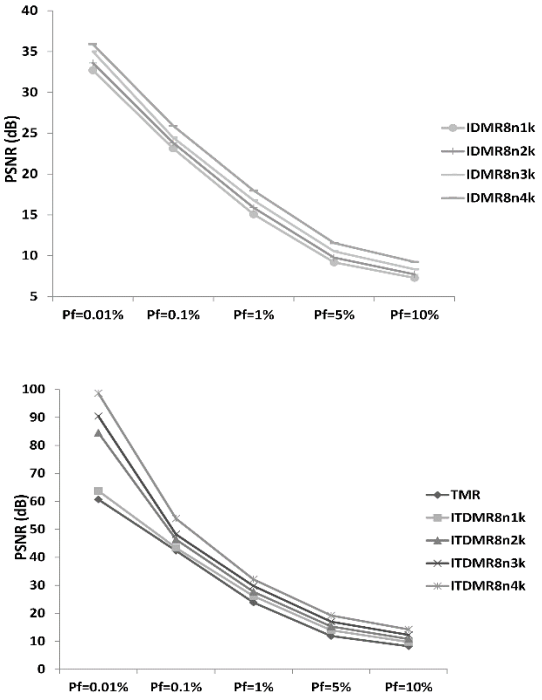

Fig. 10. PSNR of different approximate voting schemes (ITDMR and IDMR) with bit-wise noise vs P_f and variable k

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2653780, IEEE Transactions on Computers

Fig. 11. (a) Error-free image; (b) TMR with σ =1(PSNR=10.17dB); (c) IDMR with σ =1(PSNR=18.80dB) (k=1, n=8); (d) ITDMR with σ =1(PSNR=48.41dB) (k=1, n=8)

For example, an error seldom occurs at the output of a module when P_f is less than 0.01%, so this is of limited usefullness for evaluation. The peak signal-to-noise ratio (PSNR) [19] is established for the final output of the voter with respect to the error-free result; Fig.10 shows the simulation results for the PSNR of the voting schemes at the same bit-wise noise and for an 8-bit image (i.e. n=8).
From these results, the following conclusions can be drawn.

- When the value of k increases, the PSNR increases for IDMR or ITDMR, i.e. when k increases, then more inexactness is encountered in the operation of the approximate voter and thus, the probability of each module to generate a valid result increases too. However, by increasing k, there are few extreme scenarios (such as when all MSBs change) in which the PSNR will experience a significant decrease too.
- The ITDMR scheme always outperforms the TMR, because the probability of a module with an error-free (exact) output for a TMR $(1-P_f)^n$, while for the ITDMR it is $(1-P_f)^{n-k-1}$ TMR is better than IDMR at a low value of P_f.

Based on the probabilistic analysis in Table XVII, the correct probability increases by increasing the value of k (at a constant P_f value); in Fig.10, the PSNR decreases with P_f (it still increases with k for both ITDMR and IDMR). These results are in agreement to show the effectiveness of the proposed approximate schemes.

7.1.2 Value-wise noise

Also in this case (with a normally distributed noise), the PSNR is established for an 8-bit image. Fig.11 shows the error-free image as well as the results at σ=1 for TMR, at k=1 for IDMR and ITDMR.

These results are plotted in Fig.12 versus the variance. From the simulation results, several conclusions can be drawn.

- At the same k, if the variance increases, the PSNR decreases, because the error probability increases.
- ITDMR and IDMR have higher PSNRs compared to TMR (the PSNR of the TMR is nearly constant, so nearly independent of the variance). This occurs, because a value-wise error impacts in most cases the lower bits. The operation of a TMR is static, because it can only establish the strict majority; however, the proposed schemes can tolerate small differences in values, while still producing a voted output, thus achieving impressive improvements over the TMR scheme.
- At a small value of variance compared to the tolerance threshold of ITDMR, the PSNR of ITDMR increases by decreasing k. For example, when σ=1, the PSNRs for k=1, 2, 3 and 4 are 48.41dB, 47.73dB, 47.69dB and 47.69dB. At k=1, this scheme can tolerate most errors for σ=1; as a smaller value of k implies more exact bits, a three-module arrangement such as ITDMR is better at higher values of k than IDMR.

A comparison between the results of Fig.12 and Table XVIII yields the following additional conclusions.

- In Table XVIII, the correct probability increases by increasing the value of k (at the same variance). Fig.12 shows a similar trend.
- If the variance is smaller than the tolerance threshold, the correct probability is close to 1; in this case, the PSNR value is nearly constant, so showing modest improvements with respect to k.
- In all cases, the PSNRs decrease when the variance increases; therefore, an increasing variance makes the final result more inexact for an ap-

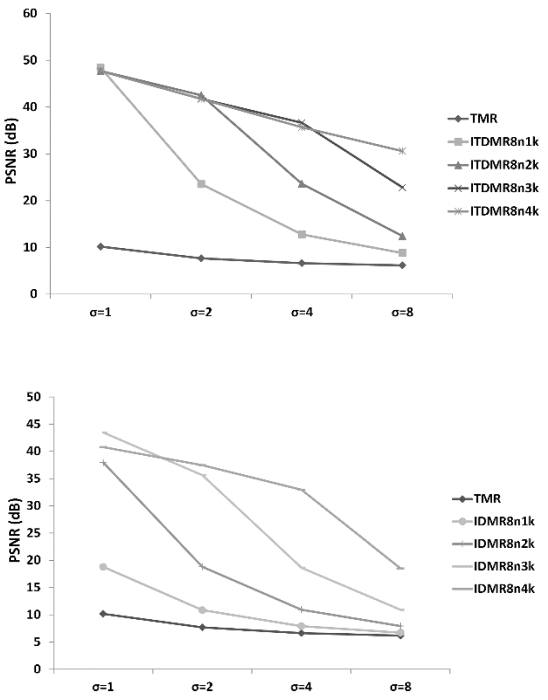

Fig. 12. PSNR of different approximate voting schemes (ITDMR and IDMR) with value-wise noise vs variance and variable k (8 bit image)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2653780, IEEE Transactions on Computers

proximate voting scheme, such as ITDMR.

A 16-bit image has also been considered. Tables XIX-XX show the results and confirm that at a higher number of bits in an image, the improvements in PSNR by the proposed approximate voting schemes are more pronounced. The scalability of approximate voting is excellent, because performance is improved with n for all cases dealt in this manuscript.

TABLE XIX
PSNR OF VOTERS WITH BIT-WISE NOISE (16-BIT IMAGE)

	P_F=0.01%	P_F=0.1 %	P_F =1%	P_F =5%	P_F =10%
Word TMR	55.21 dB	38.13 dB	21.88 dB	10.00 dB	6.32 dB
ITDMR k=1	58.00 dB	39.06 dB	24.53 dB	12.02 dB	7.81 dB
ITDMR k=2	76.90 dB	41.77 dB	26.20 dB	13.42 dB	8.92 dB
ITDMR k=3	82.31 dB	43.36 dB	28.21 dB	15.12 dB	10.37 dB
ITDMR k=4	89.79 dB	48.44 dB	30.71 dB	17.62 dB	12.58 dB
IDMR k=1	29.77 dB	20.83 dB	13.22 dB	7.22 dB	5.34 dB
IDMR k=2	30.59 dB	21.44 dB	13.99 dB	7.80 dB	5.77 dB
IDMR k=3	31.86 dB	22.01 dB	14.87 dB	8.59 dB	6.39 dB
IDMR k=4	32.69 dB	23.32 dB	16.09 dB	9.66 dB	7.26 dB

TABLE XX
PSNR OF VOTERS WITH VALUE-WISE NOISE (16-BIT IMAGE)

	σ=1	σ=2	σ=4	σ=8
Word TMR	8.49 dB	6.06 dB	5.03 dB	4.63 dB
ITDMR k=1	48.54 dB	22.58 dB	11.61 dB	7.56 dB
ITDMR k=2	47.91 dB	42.33 dB	22.75 dB	11.82 dB
ITDMR k=3	47.86 dB	41.95 dB	36.84 dB	22.62 dB
ITDMR k=4	47.84 dB	41.92 dB	36.03 dB	30.96 dB
IDMR k=1	17.36 dB	9.42 dB	6.29 dB	5.03 dB
IDMR k=2	37.21 dB	17.50 dB	9.63 dB	6.44 dB
IDMR k=3	43.76 dB	35.70 dB	17.71 dB	9.83 dB
IDMR k=4	41.36 dB	38.06 dB	32.78 dB	18.15 dB

7.1.3 Uneven noise

In this section, uneven noise is considered under the following three scenarios for the three modules of the model of Fig.9.

- Modules A and B are error-free; module C has a large noise.
- Module A is error-free, module B has a small noise and module C has a large noise.
- Modules A and B have small noise; module C has a large noise.

TABLE XXI
PSNR OF VOTERS WITH VALUE-WISE UNEVEN NOISE

	$\sigma_A = 0$ $\sigma_B = 0$ $\sigma_C = 8$	$\sigma_A = 0$ $\sigma_B = 1$ $\sigma_C = 8$	$\sigma_A = 1$ $\sigma_B = 1$ $\sigma_C = 8$
Word TMR	Infinity	6.56 dB	5.82dB
ITDMR k=1	Infinity	28.75dB	15.53dB
ITDMR k=2	Infinity	48.31dB	47.40dB
ITDMR k=3	Infinity	44.45dB	43.60dB.
ITDMR k=4	Infinity	42.05dB	41.92dB

Note that for an error-free module, σ=0, while for a large (small) noise σ=8 (σ=1) for value-wise voting and P_f = 0, 1 and 15 % for error-free, small and large noise respectively in bit-wise voting. Tables XXI and XXII show the simulation results for the PSNRs of bit-wise and value-wise voting schemes.

TABLE XXII
PSNR OF VOTERS WITH BIT-WISE UNEVEN NOISE

	$P_{fA} = 0$ $P_{fB} = 0$ $P_{fC} = 15\%$	$P_{fA} = 0$ $P_{fB} = 1\%$ $P_{fC} = 15\%$	$P_{fA} = 1\%$ $P_{fB} = 1\%$ $P_{fC} = 15\%$
Word TMR	Infinity	16.40 dB	13.60dB
ITDMR k=1	Infinity	18.53dB	16.98dB
ITDMR k=2	Infinity	19.96dB	18.32dB
ITDMR k=3	Infinity	21.82dB	20.00dB
ITDMR k=4	Infinity	24.39dB	23.41dB

As three modules are considered in the model of Fig.9, IDMR is not evaluated for these three uneven noise cases. The following conclusions can then be drawn.

- As expected, if only two modules are error-free, the final output is still error-free.
- For bit-wise noise, if the value of k increases, the PSNR increases; in this case, TMR can be regarded as k=0. A larger k means more errors can be tolerated leading to a higher PSNR.
- Consider the relation between k and the variance σ for value-wise noise; a small value for k means that a smaller error can be tolerated and more accuracy is achieved (provided the variance is within the tolerable value). Hence, ITDMR has the largest PSNR at k=2; this occurs, because the variance is out of bound for the error that ITDMR with k=1 can tolerate. For k=3 and 4 the variance is within the tolerable value, but at higher values of k, more errors appear.

7.2 FIR Filter

In signal processing, a finite impulse response (FIR) filter is defined as a filter whose impulse response is of a finite duration. The output y of a linear time invariant system is determined by convolution of its input signal x with its impulse response b. For a discrete-time FIR filter, the output is a weighted sum of the current value and a finite number of previous values of the input. Therefore, a FIR filter implements the following equation:

$$y[n]=\sum_{i=0}^{N-1} x[n-i]\,b[i] \qquad (12)$$

where x[n] is the input signal, y[n] is the output, and b[i] is the filter coefficient. An implementation in block form of a FIR filter is shown in Fig.13.

In this manuscript, the FIR filter is designed using the FDA tool in Matlab [20]. Initially, a 10th order transposed low-pass filter is considered.

To evaluate the effectiveness of the proposed voting schemes, the input signal is randomly generated in the

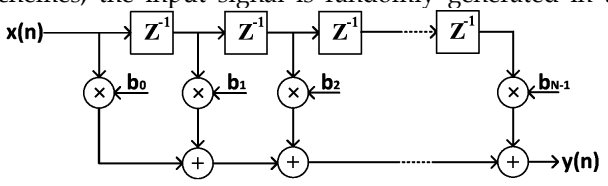

Fig. 13. FIR filter implementation using the transposed of the direct form

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2017.2653780, IEEE Transactions on Computers

range [-0.5, 0.5] [13]; the input data, the filter coefficients and the output of the filters are quantized in terms of 16 and 32 bits.

Similar to the images considered previously, bit-wise and value-wise noises are inserted in the inputs. Bit-wise noise is defined as the noise affecting each bit of the inputs to the voter with the same probability to change (flip) its value. The flip probability is denoted by P_f and each bit is independent. Value-wise noise is defined as the normally distributed noise, the variance of the noise is denoted by σ. Noise is inserted starting at a sufficiently low rate and an error-free state is assumed prior to inserting each error. The PSNR between the error-free output of the filter and the output of the voter is measured over 1000 simulation runs.

TABLE XXIII
AVERAGE PSNR OF VOTERS WITH BIT-WISE NOISE FOR THE FIR FILTER (16BIT)

	P_F=0.01%	P_F=0.1 %	P_F =1%	P_F =5%	P_F =10%
Word TMR	81.52 dB	77.64 dB	68.71 dB	59.05 dB	55.51 dB
ITDMR k=1	74.33 dB	70.79 dB	62.65 dB	56.61 dB	54.79 dB
ITDMR k=2	74.88 dB	71.31 dB	63.11 dB	56.93 dB	55.03 dB
ITDMR k=3	75.34 dB	71.76 dB	63.50 dB	57.31 dB	55.29 dB
ITDMR k=4	76.03 dB	72.41 dB	64.08 dB	57.81 dB	55.66 dB
IDMR k=1	82.63 dB	78.69 dB	69.64 dB	60.54 dB	56.61 dB
IDMR k=2	82.79 dB	78.85 dB	69.78 dB	61.27 dB	57.15 dB
IDMR k=3	83.24 dB	79.28 dB	70.16 dB	62.22 dB	57.85 dB
IDMR k=4	83.79 dB	79.80 dB	70.62 dB	63.09 dB	58.66 dB

TABLE XXIV
AVERAGE PSNR OF VOTERS WITH VALUE-WISE NOISE FOR THE FIR FILTER (16BIT)

	σ=2	σ=4	σ=8	σ=16
Word TMR	56.60 dB	54.99 dB	54.24 dB	53.89 dB
IDMR k=1	61.99 dB	57.00 dB	55.08 dB	54.28 dB
IDMR k=2	75.14 dB	61.95 dB	56.99 dB	55.09 dB
IDMR k=3	138.12 dB	75.43 dB	61.98 dB	57.00 dB
IDMR k=4	135.69 dB	132.26 dB	75.46 dB	61.94 dB
ITDMR k=1	141.55 dB	135.86 dB	71.27 dB	60.49 dB
ITDMR k=2	141.35 dB	108.67 dB	64.95 dB	58.32 dB
ITDMR k=3	141.01 dB	135.47 dB	105.17 dB	65.01 dB
ITDMR k=4	141.02 dB	135.14 dB	129.45 dB	100.95 dB

TABLE XXV
AVERAGE PSNR OF VOTERS WITH BIT-WISE NOISE FOR THE FIR FILTER (32BIT)

	P_F=0.01%	P_F=0.1 %	P_F =1%	P_F =5%	P_F =10%
Word TMR	79.45 dB	75.66 dB	66.96 dB	58.25 dB	55.02 dB
ITDMR k=1	73.25 dB	69.77 dB	61.74 dB	55.95 dB	54.36 dB
ITDMR k=2	73.27 dB	69.78 dB	61.75 dB	55.94 dB	54.36 dB
ITDMR k=3	73.30 dB	69.81 dB	61.78 dB	56.04 dB	54.43 dB
ITDMR k=4	73.66 dB	70.15 dB	62.08 dB	56.12 dB	54.54 dB
IDMR k=1	91.57 dB	87.21 dB	77.18 dB	59.02 dB	55.63 dB
IDMR k=2	91.62 dB	87.26 dB	77.22 dB	59.01 dB	55.62 dB
IDMR k=3	91.74 dB	87.37 dB	77.32 dB	59.30 dB	55.80 dB
IDMR k=4	92.00 dB	87.62 dB	77.54 dB	59.52 dB	55.92 dB

The average PSNRs of the different voting schemes for this application are given in Tables XXIII through XXVI.

TABLE XXVI
AVERAGE PSNR OF VOTERS WITH VALUE-WISE NOISE FOR THE FIR FILTER (32BIT)

	σ=2	σ=4	σ=8	σ=16
Word TMR	56.58 dB	54.99 dB	54.24 dB	53.89 dB
IDMR k=1	63.62 dB	60.58 dB	57.73 dB	55.59 dB
IDMR k=2	74.89 dB	61.97 dB	57.02 dB	55.09 dB
IDMR k=3	234.44 dB	75.02 dB	61.97 dB	57.02 dB
IDMR k=4	232.06 dB	228.63 dB	76.30 dB	61.97 dB
ITDMR k=1	153.21 dB	64.84 dB	58.35 dB	55.73 dB
ITDMR k=2	237.65 dB	147.95 dB	64.92 dB	58.30 dB
ITDMR k=3	237.37 dB	231.79 dB	136.58 dB	64.94 dB
ITDMR k=4	237.33 dB	231.44 dB	225.77 dB	148.83 dB

From these results, the following conclusions can be drawn.

- When the value of k increases, the PSNR increases for IDMR or ITDMR, because the probability of each module to generate a valid result increases.
- Also in this case, ITDMR always outperforms TMR for bit-wise noise; the IDMR scheme sometimes performs worse than the TMR scheme, especially for bit-wise noise.
- For the value-wise noise, IDMR and ITDMR perform better than the TMR scheme because a value-wise error impacts in most cases the lower bits. TMR can only establish the strict majority, while the proposed schemes can tolerate differences in values.

In the implementation of a FIR filter, the filter coefficients are stored in registers and soft errors may affect the contents of these registers and therefore, the output value

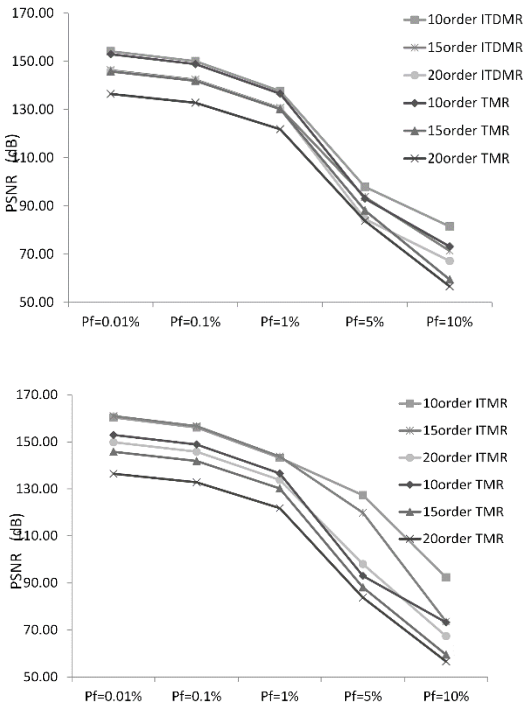

Fig. 12. Average PSNR for voting schemes (n=16) of different orders: (a) bit-wise error of ITDMR and TMR; (b) bit-wise error of ITMR and TMR.

(similar to a bit-wise error as defined previously) may change. Hence, this possible scenario is also evaluated. The 10, 15 and 20th order FIR filters are considered; the FIR filter coefficients are always given by a word length of 16 bits. Among these registers, each bit has a flip probability given by P_f . A bit-wise error is introduced for each filter coefficient; the error free input signal for each FIR filter is randomly generated in the range [-0.5, 0.5], while the quantification of the input signal is specified by the coefficient.

Fig.14 shows the average PSNR values of ITDMR, ITMR and TMR at n=16 and k=4 for different FIR orders and bit-wise errors in the filter coefficients. Performance is affected at high values of P_f and with increasing filter order (i.e. the PSNR decreases). However, in all cases, an approximate voting scheme is still better than a TMR.

The PSNR value of a voter decreases when the order of the FIR filter increases; however, at a higher order, the coefficient registers have a larger probability to be affected by a soft error, hence the decrease of the PSNR at an increase of the FIR order.

7 Conclusion

This paper has presented the analysis and design of novel voting schemes whose operations are based on approximate computing. Approximate computing relaxes the strict static voting and masking that modular redundancy schemes utilize to generate a correct output. The first proposed scheme is referred to as inexact double modular redundancy (IDMR) while the second scheme (ITDMR) combines IDMR with TMR. Both these schemes utilize approximate criteria when comparing and assessing the outputs of at least two modules for reliable computing. IDMR and ITDMR voters have been designed at nanometric features sizes and simulated using Hspice to assess different figures of merit, such as delay, power dissipation, circuit complexity, process variability and critical supply voltage. TMR has the least delay, but consumes more power and its process variability is worse than the proposed schemes. Image processing and FIR filters have been analyzed as possible applications of the proposed approximate voters; the PSNR has been measured and in most case, the proposed schemes perform better than TMR.

However, the proposed scheme still has the limitation. In control flow dominated applications, taking average of input may lead to chaos. In the data flow applications, the proposed scheme has better performance.

In conclusion, approximate voting by IDMR and ITDMR shows the following advantages over a TMR with static masking and exact voting.

- Except the delay, approximate voting hardware for both IDMR and ITDMR improves over all circuit-level figures of merit, such as power dissipation and tolerance to variations.
- Probabilistic measures based on bit-wise and word-wise voting confirm the viability of the proposed schemes to reach a voted output in the presence of differences in the output values of the modules.
- For the considered applications (image processing and FIR filters), the proposed schemes show higher PSNR values, thus consistently and significantly improving on an exact voting scheme such as TMR.

References

[1] R. Baumann, ''Soft errors in advanced computer systems,'' IEEE Design Test of Computers., vol. 22, no. 3, pp. 258–266, May–Jun. 2005.

[2] M. Nicolaidis, ''Design for soft error mitigation,'' IEEE Trans. Device Mater. Reliabil., vol. 5, no. 3, pp. 405–418, Sep. 2005.

[3] Von Neumann, J., "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," Automata Studies, Ann. Of Math. Studies, no. 34, C. E. Shannon and J. McCarthy, Eds., Princeton University Press, pp. 43-98, 1956.

[4] N. Vaidya and D. Pradhan, "Fault-Tolerant Design Strategies for High Reliability and Safety," IEEE Trans. Computer, vol. 42, no. 10, pp. 1195-1206, Oct. 1993.

[5] W. H. Pierce, Failure-Tolerant Computer Design, Academic Press, 1965.

[6] Samudrala, P.K.; Ramos, J.; Katkoori, S. "Selective Triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs," Nuclear Science, IEEE Transactions on , vol.51, no.5, pp. 2957- 2969, Oct. 2004.

[7] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," Computers, IEEE Transactions on, vol. 62, no. 9, pp. 1760–1771, 2013.

[8] S.-L. Lu, "Speeding up processing with approximation circuits," Computer, vol. 37, no. 3, pp. 67–73, 2004.

[9] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in Design, Automation Test in Europe Conference Exhibition (DATE), 2012, 2012, pp. 1257–1262.

[10] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: imprecise adders for low-power approximate computing," in Low Power Electronics and Design (ISLPED) 2011 International Symposium on. IEEE, 2011, pp. 409–414.

[11] K. Chen, J. Han and F. Lombardi, "An Approximate Voting Scheme for Reliable Computing," Proc. DATE, pp. 293-296, Grenoble, March 2015.

[12] P. Reviriego, Bleakley, C.J.; Maestro, J.A., "Diverse Double Modular Redundancy: A New Direction for Soft-Error Detection and Correction," Design & Test, IEEE , vol.30, no.2, pp.87,95, April 2013

[13] P. Reviriego, C. Bleakley, and J. A. Maestro, "Structural DMR: A technique for implementation of soft error tolerant FIR filters, " IEEE Trans. Circuits Syst. II, vol. 58, no. 8, pp. 512–516, Aug. 2011.

[14] S. Mitra, McCluskey, E.J., "Word-voter: a new voter design for triple modular redundant systems," VLSI Test Symposium, 2000. Proceedings. 18th IEEE , vol., no., pp.465,470, 2000

[15] L. A. Tambara, F. L. Kastensmidt, P. Rech and C. Frost, "Decreasing FIT with diverse triple modular redundancy in SRAM-based FPGAs," 2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Amsterdam, 2014, pp. 153-158.

[16] A. Sanchez-Clemente, L. Entrena and M. Garcia-Valderas, "Partial TMR in FPGAs Using Approximate Logic Circuits," 2015

15th European Conference on Radiation and Its Effects on Components and Systems (RADECS), Moscow, 2015, pp. 1-4.

[17] S. Venkataraman, R. Santos and A. Kumar, "A flexible inexact TMR technique for SRAM-based FPGAs," 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016, pp. 810-813.

[18] S. R. Chowdhury, A. Banerjee, A. Roy, H. Saha, "A high speed 8 transistor full adder design using novel 3 transistor XOR gates", World Academy of Sciences, Vol.22, 2008.

[19] A. Rubio, Figueras Pàmies, J.; Vatajelu, E.; Canal Corretger, R., "Process variability in sub-16nm bulk CMOS technology", Project: Terascale Reliable Adaptive Memory Systems,FP7-INFSO–IST -248789, 2012. 12.

[20] Matlab documentation Center 'Using FDAtool': http://www.mathworks.com/help/signal/ug/opening-fdatool.html

First A. Author All biographies should be limited to one paragraph consisting of the following: sequentially ordered list of degrees, including years achieved; sequentially ordered places of employ concluding with current employment; association with any official journals or conferences; major professional and/or academic achievements, i.e., best paper awards, research grants, etc.; any publication information (number of papers and titles of books published); current research interests; association with any professional associations. Author membership information, e.g., is a member of the IEEE and the IEEE Computer Society, if applicable, is noted at the end of the biography.

Second B. Author Jr. biography appears here.

Third C. Author biography appears here.