Error Correction Method Based On The Short-Duration Offline Test

Jan Bělohoubek, Petr Fišer, Jan Schmidt
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
{jan.belohoubek, petr.fiser, jan.schmidt}@fit.cvut.cz

***Abstract*—The method proposed in this paper allows to construct error-correcting systems by combining time and area redundancy. In such a system, error detection is performed online, while error correction uses a short-duration offline test. The time penalty caused by the offline test applies only when an error is detected. The error-correcting ability in such a system is comparable with TMR, the area overhead is smaller for a class of circuits, and the delay penalty caused by the offline test remains reasonably small. The short-duration offline test is possible only when extensive design-for-test practices are used. Therefore, a novel gate structure is presented, which allows to construct combinational circuits testable by a short-duration offline test. The proposed test offers complete fault coverage with respect to the stuck-on and stuck-open fault model.**

I. Introduction

In applications, where *dependability* is required, some kind of *redundancy* has to be involved. In most cases, the *time* or *area* redundancy is considered. The additional redundancy offers an information which enables to identify and/or repair an *erroneous* output of the system. To get this kind of information, it is possible to perform parallel computations by using independent computational units, perform recomputation using the same unit, or use offline testing [1].

The erroneous system output is caused by a fault at the physical level. From the *physical faults* point of view, area redundancy-based methods are well suitable for mitigation of errors caused by both *transient* and *permanent* faults. Computation repetition (i.e., time redundancy) can be efficiently used for mitigating errors caused by *transient* faults.

Offline testing can be used to identify *long-duration transient* or *permanent fault* presence inside the system under test [1]. Additionally, offline testing can be efficiently used to repair errors only if the test has significant and realistic *fault-coverage*. If the offline test passes, the output of the system may be correct or not, depending on the test coverage. On the other side, if the test does not pass, it is clear, that for the set of input vectors, the system produces an erroneous output (but it can still produce correct outputs for another set of input vectors).

We can divide the error correcting and detecting methods by the impact to the system performance to *online* and *offline* methods. Online methods do not affect the system latency significantly, on the other hand, offline methods suspend the system.

The *online error-correction* can be achieved by triplicating the original module. This is called a *Triple modular redundancy (TMR)* – see Figure 1a. TMR is able to produce correct output, if at least two of three identical modules are fault-free.

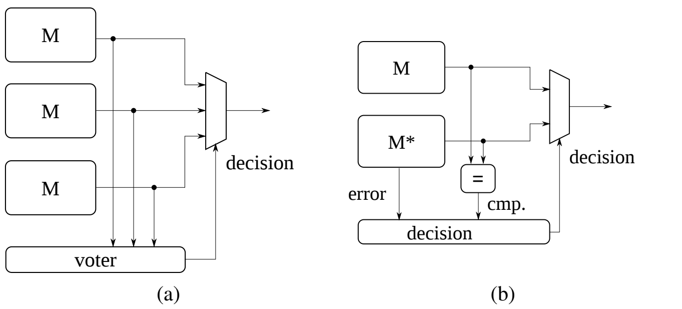

Fig. 1. Conceptual schemes of an error-correcting (a) TMR and (b) duplex system with a self-checking module M*

Online error-correction functionality can also be obtained by using *self-checking modules* [2], [3] in a *duplex system*. A duplex system contains two modules providing the same function. At least one of the two modules must be self-checking (M*) to provide *online error-detection* ability. The self-checking module is used for error localization, and consequently for error correction. This approach is presented in Figure 1b.

Providing online error-detection typically means introducing some area overhead [3]. A simple example of a self-checking module is a duplex system itself [1]. This is why the area of a duplex system with one self-checking module (a duplex too) is close to the area of TMR.

From a conceptual point of view, the *offline test* can be used to provide the same information as the self-checking module – see Figure 2. The main functional difference is that the decision may be delayed. Additionally, TMR detects errors, while tests detect faults. Hence, the fault model used must be realistic and the test's *fault-coverage* must be complete. Unfortunately, such a test has typically number of disadvantages: the test must be generated by an *ATPG* (Automatic Test Pattern Generator), it must be stored in an on-chip memory and the testing itself is time-consuming [1].

It would be advantageous, if the test vectors and the test responses were easy to produce and check in hardware, while the test length would be in orders of tens computational (clock) cycles only and the fault-coverage of 100% with respect to the used fault model. We call such a test a *short-duration test*.

This paper presents a novel method combining offline testing with functional module duplication. Computation is performed in parallel by two independent modules. If the outputs differ, the offline test is executed. The test confirms the fault presence in one of the modules. The block containing a fault may produce faulty outputs, so it is marked as faulty and the other module as the correct one. The offline testable module is denoted as $\mathtt{M}^{**}$ – see Figure 2. We call the system, where one `M` and one $\mathtt{M}^{**}$ module is used as a *Time-Extended Duplex* (TED).

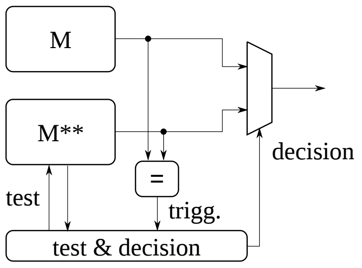

Fig. 2. Conceptual scheme of an error-correcting duplex with module $\mathtt{M}^{**}$ (short-duration testable) – the *Time-Extended Duplex*

This paper extends our previous work [4], while its main principle is preserved. The key part of the presented method is the short-duration test of the combinational module $\mathtt{M}^{**}$. Originally, we used *C-element-based* gates, which allowed to perform a test with a complete *stuck-at fault* coverage at the gate level. This was achieved by circuit monotonicity, symmetry and controllability of the used gate and also because the C-element is state-holding. These properties allowed to apply very simple test vectors – *all-zero* and *all-one*. These values were then propagated to the outputs. In a fault-free circuit, an all-zero output was the response to the all-zero input and an all-one output was the response for the all-one input. If there was a fault in the circuit, the opposite logic value was propagated from the fault location up to the circuit outputs. In other words: the circuit was flooded by zeroes or ones and any fault blocked the value propagation from the circuit inputs to the outputs. Such test vectors and test responses are easy to produce and check in hardware.

A. Fault Model

The ability of error-correction in TED is influenced by the fault-coverage and the accuracy of the selected fault-model. In industry and also in academia, the gate-level stuck-at-fault model is widely used because of its simplicity. We moved to a more accurate transistor-level *stuck-open/stuck-on* fault model [5]. The faults correspond to permanently closed or open transistors. This model includes all gate-level stuck-at-faults and it models many physical defects, while remaining reasonably simple to evaluate.

The test presented in this paper has 100% fault-coverage with respect to the *stuck-open/stuck-on* fault model.

B. Contribution of the Paper

Other works combining time and area redundancy often deal only with transient or *soft faults* like *single-event upsets* (SEU), e.g., [6]. Some of the methods presented in the past require recomputation after reconfiguration, e.g., [7], more instances of the secured unit may be required, or the unit must be divided into equivalent sub-units [8]. Exhaustive testing using equivalent functional and backup units may be required or error detection is delayed [9].

In contrast, our approach is more general – erroneous output is detected immediately, it is possible to secure any combinational circuit with no limitations, and it detects both transient and permanent faults. No functional block reconfiguration is required. The error-correction ability is comparable with TMR and the resulting circuit may be smaller at the same time. As the time redundancy is employed only when a fault is detected, some kind of *handshaking* is required – the system is *globally asynchronous*.

The rest of the paper is structured as follows: In Section II, the Time-Extended Duplex is described. The novel transistor structure is presented in Section III and the short-duration test is described in Section IV. The transistor-level model used to extract circuit parameters is described in V. The experimental evaluation is presented in Section VI. The paper is concluded by Discussion (Section VII) and Conclusions (Section VIII).

II. Time-Extended Duplex

The basic principle of the Time-Extended Duplex has been presented in the previous text. The detailed scheme of the TED is shown in Figure 4. The detailed scheme of the TMR system is shown in Figure 3 for comparison.

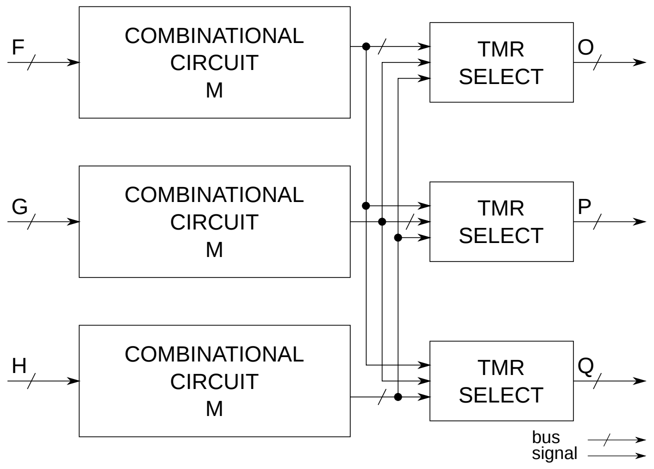

Fig. 3. Detailed scheme of the TMR system

The TMR system is composed of three identical modules (`M`) implementing the given logic function and three voters. The voters' decision is based on online comparison of three output vectors.

A. The Time-Extended Duplex Structure

The TED computational part contains one original module (`M`) and one functionally equivalent to `M`, but offline testable module – ($\mathtt{M}^{**}$). The voter (`OUTPUT SELECT`) decision is based on the offline-test result stored in the `SYSTEM STATE REGISTER` and the outputs of both combinational modules.

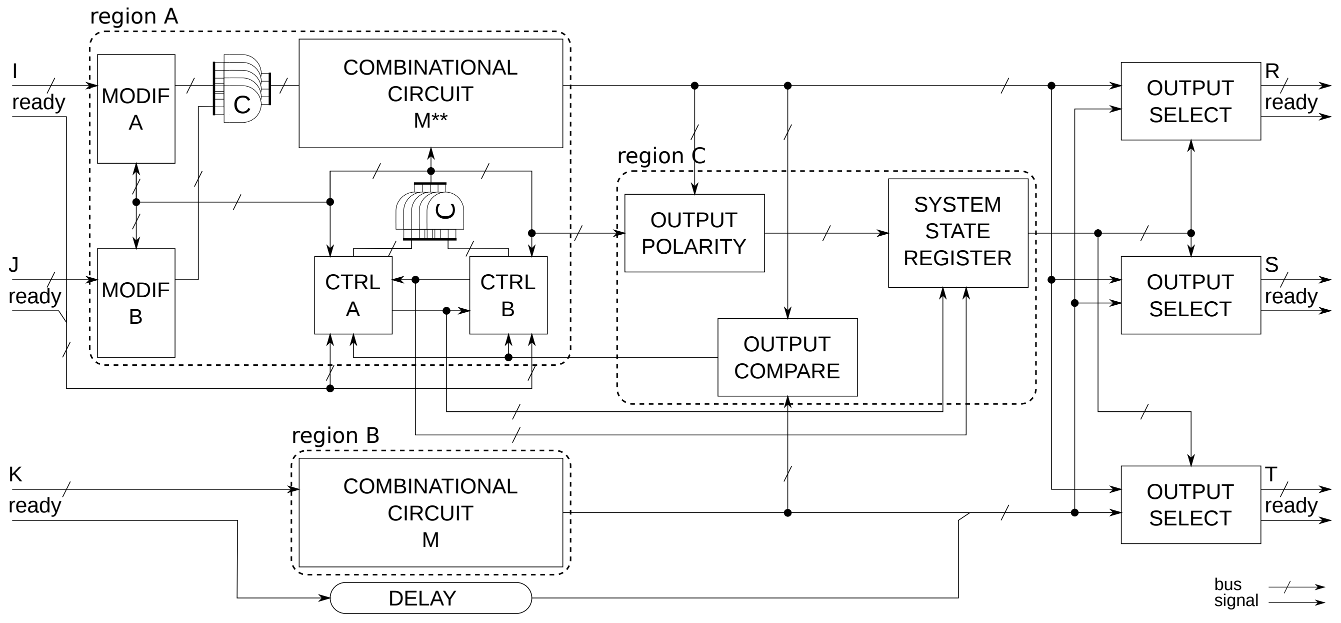

Fig. 4. Detailed scheme of the Time-Extended Duplex

The TED output may be delayed depending on the offline test – the `ready` signal is used to signalize data validity.

The offline test is launched only if outputs of `M` and `M`** differ and `SYSTEM STATE REGISTER` contains no result of the previous test. The output difference is signalized by `OUTPUT COMPARE` module.

The test controller is designed as a *self-checking* circuit. It is composed of two identical and independent controllers (`CTRL A` and `CTRL B`) and an array of C-elements. The C-element serves as a two-input comparator. If both inputs of C-elements match, the output changes to this value, otherwise the original output value is conserved [10]. The C-elements outputs are used to control the test and these are also driven back to each controller to compare with the controller's output. An error is detected at least by one of the controllers.

The offline test responses are checked by the `OUTPUT POLARITY` module. This module produces an error/ok signal for every bit of `M`** output. The information about detected faults is stored in the `SYSTEM STATE REGISTER`. The faults detected by `CTRL A` and `CTRL B` and by `OUTPUT POLARITY` are stored separately, but for simplicity, the storage is represented by a single block.

The input checker and the offline-test generator is composed of `MODIF A` and `MODIF B` modules and an array of C-elements. Depending on the control signals, the output of `MODIF A` and `MODIF B` is an all-zero or all-one vector or a conversion of the single-rail input to dual-rail one (see next sections). During the normal operation, `MODIF A` and `MODIF B` are in the conversion mode – C-elements serve as online checkers.

B. *Error Detection and Correction*

The TED design and the short-duration test of `M`** allows a clear error-source localization in *region A* or *region B*. An error caused by a fault located in *region C* may not cause an erroneous output if both *region A* and *region B* are fault-free.

In case of a single fault, correct operation is guaranteed (the TED output is correct). To secure the output, the TED offers three equivalent outputs, the same number as in the TMR system. Thus faults in one `OUTPUT SELECT` are tolerated.

For *region A*: the offline test finds any fault in `M`**. This test is also able to identify an error at the `M`** input, thus the C-element array errors are detected; `MODIF A` and `MODIF B` input inequality and output errors for a given input are also detected during the offline test. The controller is self-checking.

For *region B*: If the outputs of `M`** and `M` do not match and no fault was detected during the offline test, the `M`** output is fault-free and the `M` output is faulty.

The offline test is followed by recomputation – it guarantees the correctness of the output in case of an transient fault. In case of a permanent fault, it is not necessary to perform the test in every clock cycle – the stored test results are used to correct the output in subsequent computational cycles and the period between two tests may be configured by an external reset signal. This is introduced to reduce the load caused by permanent faults and to recover from the transient faults.

III. Proposed Transistor-Level Structure

The short-duration test of `M`** requires a special gate design. The gate has to be monotonic and has to allow propagation of all possible fault symptoms (one and zero). We propose a novel structure similar to the dynamic *domino* *logic*, which was deeply studied and is used even in industry. Domino logic gates are *monotonically rising* [11].

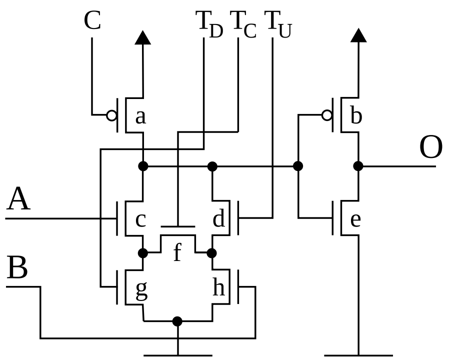

Fig. 5. Proposed transistor-level structure

The overall advantage of domino logic is the gate size and speed. The *mobility ratio* for *holes/electrons* is 2 – 3. This causes that PMOS transistors have to be bigger than the NMOS ones to achieve the same conductivity [11]. Therefore, by using the dynamic domino AND and OR gates with precharge to zero, the number of PMOS transistors is reduced.

The proposed dual AND/OR structure is shown in Figure 5 – this structure can realize both logic functions depending on the control signals T_U, T_C and T_D. From the functional point of view, the proposed structure is still a domino logic gate. The novelty is in increased controllability of the gate, which is used for testability. According to the best of our knowledge, no similar structure has been proposed before.

The described structure operates in two phases: *precharge* and *evaluation*. The operation mode and the gate function (AND/OR) is set by control signals, as shown in Tables I and II.

TABLE I. CONTROL SIGNALS FOR AND

step	C	T_U	T_C	T_D	O
precharge	0	0	0	0	$\downarrow$
evaluation	1	0	1	0	$\updownarrow$

TABLE II. CONTROL SIGNALS FOR OR

step	C	T_U	T_C	T_D	O
precharge	0	0	0	0	$\downarrow$
evaluation	1	1	0	1	$\updownarrow$

This behavior causes that both gates will run the *falling domino* effect in the whole circuit during the evaluation phase.

Of course, an arbitrary logic function cannot be implemented by AND/OR gates only – inverting function must be present. The offline test requires monotonicity, and thus no inverter must be used. To introduce inversion, *dual-rail* logic has to be employed as in traditional domino logic [11], [10]. In dual-rail logic, an inverter is represented as a wire-swap only. From the gate-level point of view, the circuit is monotonic.

Dual-rail is used to ensure monotonicity of the circuit only – this is required for testing. The inverters may be present at the physical input of the M^{**} module only (in MODIF A and MODIF B). The inverters transform the single-rail to the dual-rail signals and do not disrupt the monotonicity of M^{**} itself. The single-rail output of the module is sufficient, thus only those internal signals should remain, which are required to compute the single-rail output. Therefore, we can remove half of the dual-rail circuit outputs (only the positive outputs remain). Circuit parts feeding only the removed outputs should also be removed – see Figures 6 and 7. Then the dual signals serve as inverters replacements only, the number of outputs is equal to M, and the number of the M^{**} inputs varies between one and twice the number of the M inputs. The number of gates in such a circuit is then usually much lower compared to the implementation of M^{*} [4].

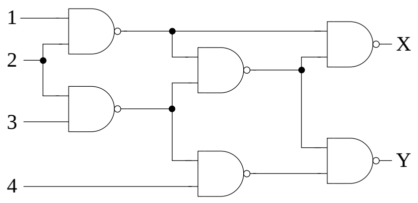

Fig. 6. Original NAND-based circuit

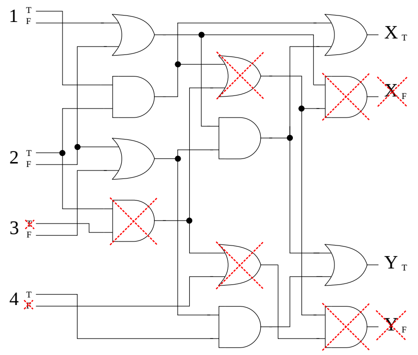

Fig. 7. Dual-rail logic circuit derived from the circuit in Figure 6 – every NAND gate was replaced by an AND and OR gate pair. The crossed-out gates, inputs, and outputs are removed by the reduction (M^{**})

After the reduction, the resulting circuit example in Figure 7 is smaller but it still has more gates than the original single-rail one, the number of outputs is the same, and the number of inputs is increased – it has 6 primary inputs instead of 4 in the original single-rail implementation in Figure 6 – both polarities of inputs 1 and 2 are required to compute the outputs.

The reduction success depends on the circuit structure. We also made number of experiments with the reduction. Allowing inverters also at the output sometimes allows to remove additional gates. During these experiments, we applied number of simple greedy heuristics and no significant improvement has been achieved (no improvement for most of the circuits and 3% on average). Moreover, testing of the output inverters requires additional logic. If the output inverters are to be eliminated, the following logic must be modified to accept inverted inputs.

IV. PROPOSED OFFLINE TEST

The offline test of M^{**} is composed of several sub-tests. Each sub-test is designed to cover a set of faults in M^{**} (Sections IV-B and IV-C) or errors at the outputs of other modules (Section IV-A).

A. M** *Inputs Test*

At the beginning of the test, the *inputs sub-test* is performed. MODIF B is used to propagate the output of MODIF A thru the C-elements. This is performed in two steps: the output of MODIF B is set to all-one – this propagates all ones from the MODIF A output to the C-elements output. Then the output of MODIF B is set to all-zero – this propagates zeroes from the MODIF A output to the C-elements output. After that, the output of M** is computed by using the MODIF A output only. The output is then compared with the M output.

The same steps are then repeated for the MODIF B output and the result is also compared with the output of M. If one of the two M** outputs matches the M output and the other does not match, errorneous MODIF has been detected. If no output matches with the M output, the test continues to the next sub-test.

B. M** *Test*

The main part of the test is a short-duration test of the module M**. The following sub-tests are performed level by level. The *gate level* is defined as the maximal path length (number of gates) from the circuit primary inputs. The *circuit depth* is the maximum of the gate levels. The primary inputs are at level 0.

The term *primary input* is used in all sub-tests and refers to physical, not the logical circuit inputs. In the reduced dual-rail logic (Section III), one circuit input may be represented by two signals – two primary inputs.

The test of M** is inspired by ideas described in Section I – the circuit is periodically flooded by a single value (1 and 0 alternate), and the flood propagation can be disrupted by faults. As this happens level-by-level, a fault in a lower level will cause the same fault symptom at higher levels. During the test, the control signals are used to discover faults and propagate the fault symptoms. This is the core idea of the short-duration test.

For example, a stuck-open in an NMOS transistor of a gate at the first level will cause that a zero value will occur at an input of a gate (configured as AND) at level two. This value – the fault symptom – is propagated up to the circuit outputs.

The proposed short-duration test of M** itself is divided into 3 sub-tests.

The sub-test 1 (Table III) and the sub-test 3 (Table V) were designed to detect stuck-open faults and the sub-test 2 (Table IV) to detect stuck-on faults. Additionally, the tests are able to detect some of the second-type faults as a side-effect. Stuck-opens are generally relatively simple to detect because the gate is unable to change the output (the gate output retains its previous value). Every sub-test contains a cycle with the number of iterations equal to the circuit depth.

Table VI shows, which sub-test detects the stuck-open/stuck-on fault for a given transistor (see transistor labels in Figure 5).

In sub-test 1, the output is checked in every iteration because the precharge function of gates in the targeted level is tested – the level-by-level fault-symptom propagation is not possible. In this case, the function of gates in the targeted level is checked and the other gates are configured to propagate fault symptoms up to the circuit outputs.

In other tests, the output is tested only once at the end of each sub-test. The tests principle is that the value at the faulty gate output is flipped even if it should stay constant during the test. The value flip in the lower level causes that a pull-down path in the following level becomes conductive even if it should be closed (for sub-test 2) or vice-versa (for sub-test 3). In this way, a possible fault syndrome is propagated up to the primary outputs.

TABLE III. THE TEST SEQUENCE OF THE *sub-test 1*

step	C	T_U	T_C	T_D	*O*
1	set circuit primary inputs to 0				
2	start in level i = 1				
3	in all levels:				
	0	0	0	0	↓
4	in level i:				0
	1	1	1	1	↑
5	in level i:				1
	1	0	0	0	1
6	in levels other than i:				0
	1	0	1	0	0
7	set circuit primary inputs to 1				↑
8	Check if the circuit output is *all-one*				1
9	if (++i $\leq$ depth) then goto 3				1

TABLE IV. THE TEST SEQUENCE OF THE *sub-test 2*

step	C	T_U	T_C	T_D	*O*
1	set circuit primary inputs to 0				
2	1	1	1	1	↑
3	0	0	0	0	↓
4	start in level i = 1				
5	in all levels:				0
	1	0	0	0	0
6	in level i:				0
	1	0	1	1	0
7	in level i:				0
	1	1	1	0	0
8	in level i:				0
	1	1	0	1	0
9	if (++i $\leq$ depth) then goto 5				0
10	Check if the circuit output is *all-zero*				0

TABLE V. THE TEST SEQUENCE OF THE *sub-test 3*

step	C	T_U	T_C	T_D	*O*
1	0	0	0	0	↓
2	set circuit primary inputs to 1				0
3	1	0	0	0	0
4	start in level i = 1				
5	in level i:				0
	1	0	1	0	↑
6	in level i:				1
	1	0	0	0	1
7	if (++i $\leq$ depth) then goto 5				1
8	Check if the circuit output is *all-one*				1

TABLE VI. SUB-TESTS COVERING THE FAULTS

transistor	tests covering faults stuck-on (short)	stuck-open
a	3	2
b	2*	1, 3
c	2	3
d	2	1
e	1*, 3*	2
f	2	1, 3
g	2	1
h	2	3

C. Inverter Stuck-On Faults

Several tests in Table VI are marked by an asterisk. The stuck-on faults at transistors 'b' and 'e' cannot be necessarily detected by the presented sub-tests. The detectability of these faults depends on the fault nature. From the functional point of view, a fault causing an error at the gate output should be detectable by the presented tests. But in reality, it can behave as a transient fault if a short in the transistor causes, that the output voltage is close to the next gate input threshold. Such a fault can cause errors on a random basis and may or may not be detected.

This can be solved by applying *fault-current* measurement. The fault-current is normally measured externally [5], [11], but in the past years, much work has been done also in the Built-In Current Sensors (BICS) area, starting from [12] in 1996, where the first BICS for deep sub-micron technologies has been presented. Recently, BICSs were proposed also for transient faults detection [13].

One BICS is able to monitor only a limited number of power rails due to a limited resolution and current load capacity. This implies using more parallel BICSs for the whole circuit [12]. We propose to use BICSs just for fault detection at the output inverting stage of the proposed gate. Just one power rail has to be measured using BICS. Based on the previous sentences, this reduces the area overhead caused by using parallel BICSs. Additionally, the increased controllability of the circuit allows to perform the required test by applying two test-vectors only – one vector to force the value `1` at the output of all gates and the second for the value `0`. The mentioned stuck-on faults are detectable using BICS at the end of *sub-test 2* and *sub-test 3*, therefore, no additional test cycles are required (although BICS tends to be slow and thus increase test time). Additionally, BICS can be used for the online detection of bridging faults located at the gate output.

D. The Overall Test

The overall offline test is composed by concatenating all the sub-tests. Additionally, sub-tests 1 – 3 may be interleaved by *fault-current* measurement, as described above. The test length is variable. If an erroneous input is identified during the inputs sub-test, the test is terminated, with the indication of a fault presence. If not, the test continues with the next three sub-tests. If no fault is detected during sub-tests 1 – 3, the output of module `M` is marked as faulty.

The total test length is given by the following equations:

$$t_{tot} = t_{input} + t_1 + t_2 + t_3 \; [+ \; 2 \cdot t_{BICS}] \qquad (1)$$

assuming that d is the circuit depth and t_e is the upper estimation of the time required for signals setting during the sub-tests, we can substitute:

$$t_{tot} \leq 2{\cdot}t_e + (d{\cdot}t_e) + (t_e + d{\cdot}t_e) + (t_e + d{\cdot}t_e) \; [+ \; 2{\cdot}t_{BICS}] \qquad (2)$$

$$t_{tot} \leq (3d + 4) \cdot t_e \; [+ \; 2 \cdot t_{BICS}] \qquad (3)$$

This implies that the resultant test length depends on gate sizes and the circuit depth only.

The parameter d depends on the circuit structure. In real circuits, d is often smaller than 10. In general, t_e is the time of few computational cycles only (clock cycles for clocked circuits). Thus, the total test length remains in orders of tens of computational cycles only.

V. USED GATE MODEL

To compare properties of circuits designed by using the proposed gate structure with static CMOS `NAND` gates and with standard dynamic domino logic, we use a transistor-level model. Our model considers just the transistor *channel width* and *length*. For comparison, static CMOS `NAND` has been chosen because of its area-efficiency and domino logic gates because of delay equivalence to the proposed gate structure [11].

We consider that the conductivity of an N-type transistor is 2.5-times higher than the conductivity of a P-type transistor. The same assumption as for the conductivities is made for the transistor gate capacities. Thus, the load caused by the P-type transistor of the same conductivity is 2.5-times higher than that of the N-type one. This corresponds to the *logical effort* [11].

Based on the transistor-level properties, the model for every logic gate is created. The gate model has the following parameters: *size*; *precharge delay* expressing the time required to charge internal gate capacity during precharge; *internal delay* expressing the time required to charge internal gate capacity during evaluation; *input capacity* expressing the capacity at the gate input; *output current* expressing the minimal current delivered by the gate output.

If the delay of the `NAND` gate is to be minimized, its size must be increased, but this affects the input capacitance of the gate inputs and thus increases the gate input load. It may imply that subsequent gates should be resized as well.

For the proposed (and also a standard domino) gate, the inverter at the output partially shadows the outputs from the inputs – the output current is affected by the size of the transistor 'b' (Figure VI). If the transistor size is doubled, the output load charging is two times faster. Naturally, doubling the size of 'b' will increase the internal gate delay but the input capacity is not affected.

As described in Section III, the proposed `AND` and `OR` gate structures are equivalent in general. The only difference is in internal delay during the evaluation caused by the different number of transistors in series (2 for OR and 3 for AND), which is the same as for equivalent standard domino gates.

For gates with high fan-out, the modeled delay may be too pessimistic with our model. Thus, for circuits with similar structure, we compare delay. For circuits with dissimilar structures, we compare circuit depth.

TABLE VII. GATE PROPERTIES

gate	input capacity	output current	precharge delay	internal delay	area
NAND$_{static}$	4.5	1	-	-	9
inverter$_{static}$	3.5	1	-	-	3.5
AND$_{domino}$	1.0	0.4	5.0	6.0	6.0
OR$_{domino}$	1.0	0.4	5.0	4.0	6.0
AND$_{proposed}$	1.0	0.4	5.0	6.0	8.0
OR$_{proposed}$	1.0	0.4	5.0	4.0	8.0

For static NAND we have chosen a symmetric conductivity, which is usual [11]. For dynamic gates we have chosen the smallest possible sizes because they are faster compared to static NAND gates. For all model parameters of used gates, see Table VII.

The advantage of dynamic logic is obvious. Consider two functionally equivalent circuits: one composed of static NAND gates and the second of the proposed or domino gates. If there is a gate with fan-out *f* in the NAND-based circuit, the load of the gate output is:

$$l = f \cdot 4.5 \tag{4}$$

The output load of the proposed (or the standard domino) gate with an equivalent fan-out is:

$$l = f \cdot 1 \tag{5}$$

VI. EXPERIMENTAL EVALUATION

For better understanding of the proposed approach, we provide comparison of M and M** (Figures 3 and 4) in Section VI-A. Consequently, the complete comparison of the TED and TMR systems is provided in Section VI-B.

A. Combinational Parts Comparison

The comparison of the proposed M** with static NAND-based M and dynamic domino logic-based M is provided.

We synthesized 240 circuits from the following benchmarks: *LGSynth'91* [14], *LGSynth'93* [15], *ISCAS'85* [16], *ISCAS'89* [17], and *IWLS 2005* [18].

The flow was as follows: the benchmark circuits were preprocessed by the *ABC* [19] tool. At first, combinational parts were extracted by the command `comb` and the following script was applied:

```
st; dch; map; mfs; b
```

This script was iterated 20-times. The library of standard two-input gates was used by the `map` command. The result of the preprocessing was optimized combinational part of the benchmark circuit in an *AIG* (And-Inverter Graph) format. The *AIG* was then used to construct the reduced dual-rail circuits as described in Section III. Circuit characteristics were extracted using the gate model from Section V.

The reduction step reduces the area of the original dual-rail circuit to 60% on average.

The quantitative results of the comparison based on all circuits from the set are shown in the Table VIII. M$_{stat}$ represents the static NAND implementation and M$_{domino}$ the dynamic domino logic implementation of M. M$_{stat,del}$ represents the scaled static implementation, where the area of M has been scaled by the delay ratio between the static and the proposed solution (M**) to provide fair comparison. If the number in Table VIII is lower than 100%, the proposed implementation (M**) is better than the given M.

TABLE VIII. M** AND M COMPARISON

	min	max	median	avg
area, M$_{stat}$	52%	147%	84%	88%
delay, M$_{stat}$	38%	266%	92%	94%
area, M$_{stat,del}$	32%	233%	76%	82%
area, M$_{domino}$	133%	133%	133%	133%
delay, M$_{domino}$	100%	100%	100%	100%

On average, the proposed structure is better than the static NAND implementation of M in both area and delay (M** compared to M$_{stat,del}$) and it has equivalent delay and 133% of the standard domino logic implementation area.

B. Time-Extended Duplex and TMR System Comparison

The comparison of combinational logic provides just partial information about proposed method usability. Thus, a comparison of the complete TED structure (Figure 4) and the dynamic domino logic-based TMR system (Figure 3) is provided. The TED is based on the domino logic variant (M**), thus comparison with domino logic-based TMR system only makes sense.

The area of TMR SELECT (see Figure 3) is proportional to the number of TMR system outputs. If the number of inputs and outputs of the combinational part is bigger than 50, then the area of additional TED logic is proportional to the number of inputs/outputs (the synthesis results show, that the constant part of the additional logic is bigger than the variable part for circuits with less than 50 inputs/outputs). From the empirical observations of relations between sizes of modules in the TED and TMR systems, we deduced the following expressions ($|\mathtt{A}|$ represents the area of A):

$$\begin{aligned} &if\ (\#inputs \approx \#outputs)\ and\ (\#outputs > 50): \\ &|\mathtt{TMR}| - 3\cdot|\mathtt{M}| \approx \frac{|\mathtt{TED}| - |\mathtt{M}| - |\mathtt{M}^{**}|}{4} \\ &12\cdot|\mathtt{TMR\ SELECT}| \approx |\mathtt{TED}| - |\mathtt{M}| - |\mathtt{M}^{**}| \\ &12\cdot|\mathtt{TMR\ SELECT}| < 2\cdot|\mathtt{M}| - |\mathtt{M}^{**}| \\ &12\cdot|\mathtt{TMR\ SELECT}| < 0.66\cdot|\mathtt{M}| \end{aligned}$$

It may be simplified to:

$$\begin{aligned} &if\ (\#inputs \approx \#outputs)\ and\ (\#outputs > 50): \\ &18\cdot|\mathtt{TMR\ SELECT}| < |\mathtt{M}| \end{aligned} \tag{6}$$

For a circuit where the expression (6) holds, the TED is likely better than the TMR system from the area perspective.

The heuristic based on the expression (6) has been verified by using 67 circuits selected from the original benchmark set. These circuits were selected to satisfy the expression (6)

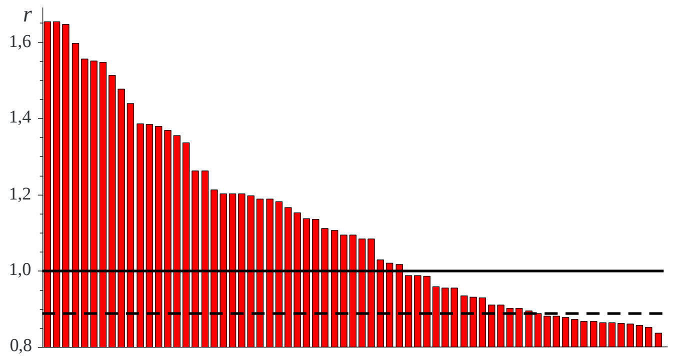

Fig. 8. The comparison of the TED and TMR systems for 67 selected circuits. The following ratio: $r = \frac{|\text{TED}|}{|\text{TMR}|}$ is shown. The circuits are shown descending ordered by r

conditions. Additionally, only circuits with at least 3k of gates were selected and the number of circuit inputs and outputs was limited to 15k for the selected circuits.

The solid line in Figure 8 symbolizes the border, where the TED system is better than the TMR system from the area point of view. Under this "success line", there are 28 out of the 67 circuits. The dashed line is the border from where the expression (6) has predicted, that the TED will be area-efficient compared to the TMR system.

Under the dashed line, there are 14 out of the 67 circuits – these are circuits with at least 11% improvement. Thus, according to the experimental evaluation, the heuristic based on the expression (6) is pessimistic.

VII. Discussion

In the TED system, there is a number of additional modules compared to the TMR system. TED is efficient compared to the TMR system when the area saved in combinational logic is bigger than the area occupied by the additional modules. The size of additional modules is almost constant (`CTRL A` and `CTRL B`) or depends on the number of combinational logic inputs/outputs (other modules).

The delay of the TED system is bigger than the delay of TMR system. The additional depth introduced by `MODIF` and the array of C-elements is 4. However, the most critical module from the delay point of view is the `OUTPUT COMPARE` block. The delay/depth (and area) of `OUTPUT COMPARE` grows linearly with the number of combinational circuit outputs.

The proposed method is thus suitable (compared to the TMR system) only for circuits having relatively large combinational parts with a relatively small number of outputs.

VIII. Conclusions

A new method for a design of error-correcting circuits was presented. This method combines time and area redundancy in an efficient way. It employs a novel gate structure and a short-duration offline test to reduce the area, while the time penalty remains reasonable. The error-correcting ability of the proposed Time-Extended Duplex is equal to TMR.

In the experimental part, we identified a class of circuits, where our approach is advantageous from the area point of view. According to the expression (6), the method is beneficial for relatively large combinational circuits.

As a significant portion of the additional delay in the TED system is proportional to the number of circuit outputs, TED is efficient for circuits with a small number of outputs only.

Acknowledgment

This research has been partially supported by the grant GA16-05179S of the Czech Grant Agency and by CTU grant SGS16/121/OHK3/1T/18.

References

[1] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[2] H. Kopetz, "Dependability," in *Real-Time Systems*, ser. Real-Time Systems Series. Springer US, 2011, pp. 135–166.

[3] V. Nelson, "Fault-tolerant computing: fundamental concepts," *Computer*, vol. 23, no. 7, pp. 19–25, July 1990.

[4] J. Bělohoubek, P. Fišer, and J. Schmidt, "Novel C-Element Based Error Detection and Correction Method Combining Time and Area Redundancy," in *Euromicro Conference on Digital System Design (DSD), 2015*, Aug 2015, pp. 280–283.

[5] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.

[6] P. 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*, vol. 51, no. 5, pp. 2957–2969, Oct 2004.

[7] D. Czajkowski, P. Samudrala, and M. Pagey, "SEU mitigation for reconfigurable FPGAs," in *Aerospace Conference, 2006 IEEE*, 2006, pp. 7 pp.–.

[8] T. Koal, M. Scholzel, and H. Vierhaus, "Combining fault tolerance and self repair at minimum cost in power and hardware," in *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems*, April 2014, pp. 153–158.

[9] M. Balaz and S. Kristofik, "Generic self repair architecture with multiple fault handling capability," in *Euromicro Conference on Digital System Design (DSD), 2015*, Aug 2015, pp. 197–204.

[10] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*, 1st ed. Kluwer Academic Publishers, Boston, 2001.

[11] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley Publishing Company, 2010.

[12] S. Athan, D. Landis, and S. Al-Arian, "A novel built-in current sensor for IDDQ testing of deep submicron CMOS ICs," in *Proceedings of 14th VLSI Test Symposium, 1996.*, Apr 1996, pp. 118–123.

[13] R. Possamai Bastos, J.-M. Dutertre, and F. Sill Torres, "Comparison of bulk built-in current sensors in terms of transient-fault detection sensitivity," in *CMOS Variability (VARI), 2014 5th European Workshop on*, Sept 2014, pp. 1–6.

[14] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0," MCNC Technical Report, Tech. Rep., Jan. 1991.

[15] K. McElvain, "IWLS'93 Benchmark Set: Version 4.0," Tech. Rep., May 1993.

[16] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *Proceedings of IEEE International Symposium Circuits and Systems (ISCAS 85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[17] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *IEEE International Symposium on Circuits and Systems, 1989.*, May 1989, pp. 1929–1934 vol.3.

[18] C. Albrecht, "IWLS 2005 Benchmarks," Tech. Rep., Jun. 2005.

[19] A. Mishchenko *et al.*, "ABC: A system for sequential synthesis and verification," http://www.eecs.berkeley.edu/∼alanmi/abc, 2012.