Error Masking 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 article allows to construct error-masking fail-operational systems by combining time and area redundancy. In such a system, error detection is performed online, while error masking is achieved by a short-duration offline test. The time penalty caused by the offline test applies only when an error is detected. The error-masking ability in such a system is very close to TMR, the area overhead is smaller for a well defined 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. The proposed solutions are combined and a comprehensive description of the overall error-masking architecture is provided.**

I. Introduction

In applications, where *dependability* is required, some kind of *redundancy* has to be involved. In most cases, the *time* (temporal) or *area* (spatial) redundancy is considered. The redundancy offers an information which enables to identify and/or repair an *erroneous* output of the system. To obtain 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 in the system under test [1]. Additionally, offline testing can be efficiently used to correct 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, while offline methods suspend the system.

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

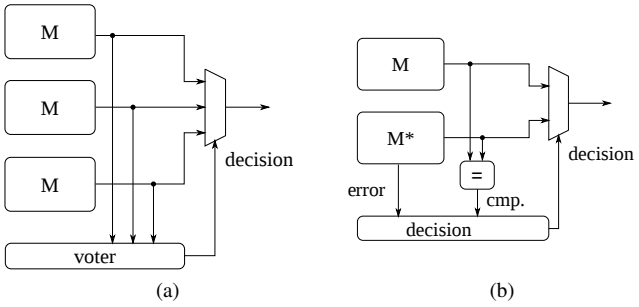


Fig. 1. Conceptual schemes of an error-correcting (a) TMR and (b) duplex system with a self-checking module $\mathtt{M}^{\star}$

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 ($\mathtt{M}^{\star}$) 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, an *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 *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].

To circumvent the problem of expensive test vectors generation and storing in memory, we propose a method, where the test vectors and the test responses are easy to produce and check in hardware, while the test length is in orders of tens computational (clock) cycles only. We call such a test a *short-duration test*. If the fault coverage is 100% with respect to given fault model, we call the test a *complete short-duration test*.

This article presents a novel method combining offline testing with functional module duplication. Computation is performed in parallel by two independent modules and their outputs are concurrently compared. If the outputs differ, the short-duration offline test is executed for one module. The test confirms or disproves the fault presence in the module under test. From this information, the potentially faulty module can be identified. This module may produce erroneous output, so it is marked as faulty and the other module as the correct one. The offline testable module is denoted as M^{**} – see Figure 2. We call the system, where one M and one M^{**} module is used, as a *Time-Extended Duplex* (TED).

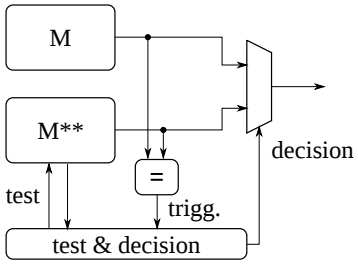


Fig. 2. Conceptual scheme of an error-correcting duplex with module M^{**} (offline testable) – the *Time-Extended Duplex*

The rest of the article is structured as follows: the state-of-the-art is described in Section II, then the basic principles of the Time-Extended Duplex structure are explained in Section III. The approaches allowing to construct a short-duration offline testable combinational logic are presented in Sections IV and V. The detailed description of the TED architecture is provided in Section VI. The complete short-duration test is described in Section VII. The experimental evaluation is presented in Section VIII and the article is concluded by Discussion and Conclusions (Sections IX and X).

II. State-of-the-Art

Other works combining time and area redundancy often deal only with transient or *soft faults* like *single-event upsets* (SEU), e.g., [4]. Some methods presented in the past rely on parts, which are not backed up and are considered to be reliable enough, while the unreliable part of the system is reconfigurable and thus allows the fault-recovery, e.g., [5].

In FPGAs, a kind of duplex system can be used to detect errors caused by *bit-flips*. Reconfiguration is then employed to repair the faulty parts [6], [7].

Approaches employing functionally equivalent units and backup units were also presented [8], [9]. Here, the error detection is significantly delayed [9] and the unit output is not checked in every cycle, thus the fault needs not to be identified, and an incorrect output is produced.

To introduce some level of reliability into high-performance chips, the problem with additional delay caused by checkers was studied years ago [10]. Although fast checkers are used, some delay is still introduced and additionally the area overhead caused by high-performance checkers is large. To allow to use lower performance checkers and mask the introduced delay, pipeline *micro rollback* was introduced in [10]. Similar approaches were presented later, e.g., [11], [12] or [13]. The presented pipeline rollback-based approaches are suitable for handling soft-errors only.

The approach called *dynamic implementation verification architecture* (DIVA) presented in [12] is similar to the pipeline rollback. It is based on concatenation of two pipelines. The first pipeline is more complicated and performs a speculative computation. It is implemented to be as fast as possible, and thus it is less reliable. The second pipeline checks the results of the first pipeline. Because in the second pipeline there are no slow inter-instruction dependencies, it is fast enough, although it is implemented in a robust technology.

A. Previous Work

This paper extends our previous work, while its main principle is preserved. The key part of our method is the use of the complete short-duration test of the combinational module M^{**}. In [14] we propose *C-element-based* gates allowing a test with a complete *stuck-at-fault* coverage at the gate level. This is achieved by circuit monotonicity, symmetry, and increased controllability of the proposed gates, and also because the C-element is state-holding [15].

These properties allow applying very simple test vectors at the circuit inputs – these are just *all-zero* and *all-one* vectors. The circuit is level-by-level flooded by a single value (one or zero). These values are propagated to the circuit primary outputs by using additional control signals.

In a fault-free circuit, an all-zero output is the response to the all-zero input and an all-one output is the response for the all-one input. If there is a fault in the circuit, the opposite logic value is propagated from the fault location up to the circuit outputs. In other words: the circuit is flooded by zeroes and subsequently by ones and any fault blocks the value propagation from the circuit inputs to the outputs. Such test vectors and test responses are easy to produce and check in hardware and the test controller is a simple state machine.

B. Fault Model

The ability of error correction in TED is determined 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 [16]. The faults correspond to permanently closed or open transistors.

The stuck-open/stuck-on fault model includes all gate-level stuck-at-faults and extends the stuck-at-fault model by assuming faults at every wire branch – this corresponds to the permanently open/closed transistor [17]. This model covers more physical defects [17], while remaining reasonably simple to evaluate.

Note, that the bridging faults are not fully covered in both stuck-at and stuck-open/stuck-on fault models, even though the high stuck-at or stuck-open/stuck-on fault coverage implies high bridging fault coverage [18].

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

C. Contribution of the Paper

This article presents a *design for test* (DFT) method allowing to construct fast offline-testable combinational logic modules (M**) and provides a detailed description and evaluation of all prerequisites, such as monotonic circuit construction and simple but accurate gate model. By using the M** blocks, the time-extended duplex system can be constructed. The proposed time-extended duplex has its error-correcting ability very close to TMR and can be smaller at the same time. Even though the time-extended duplex reduces the total size of combinational logic, an additional logic for test is required. Experimental data on the total overhead and comparison with TMR are also provided.

Naturally, as the time-extended duplex employs time redundancy only when a fault is detected, some kind of *handshaking* is required. The system containing the time-extended duplex is *globally asynchronous*.

Our approach is more general than the works presented in the past – erroneous output is detected immediately, it is possible to secure any combinational circuit with no limitations, and it is possible to detect both transient and permanent faults.

III. High-Level Description of the Time-Extended Duplex

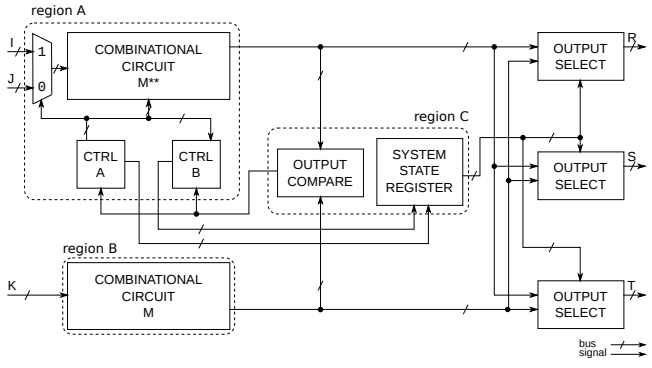


Fig. 3. A high-level scheme of the Time-Extended Duplex

Similarly to the TMR system (Figure 5), the TED structure shown in Figure 3 processes three equivalent inputs (I, J and K) and offers three equivalent outputs (OUTPUT SELECTs produce R, S and T). However, it is composed of only two functionally equivalent combinational logic blocks (M and M**). This arrangement ensures: 1) that TED tolerates the errors in the predecessing logic by comparing three equivalent inputs; and 2) that the following logic is able to select correct TED output in case of an error in the (triplicated) output logic.

The internal duplex arrangement allows error detection, not error masking. The error masking ability is allowed by the short-duration offline test. The offline test is triggered, when the OUTPUT COMPARE block signalizes a mismatch of combinational logic outputs (M and M**). The test is able to detect any (modeled) permanent (or a long-duration transient) fault in M**. The rest of the logic in *region A* (test controller CTRL and input processing) is duplicated, thus the error detection is ensured in the rest of the *region A* – see Figure 3.

If the offline test discloses a fault in M**, or a malfunction is detected (duplicated parts outputs are different) in the rest of the *region A*, the output of M** is assumed to be invalid and the output of M as valid. If no malfunction or fault is detected in *region A*, the output of M** is marked valid and the output of M as invalid. Note, that an TED error caused by a fault located in *region C* cannot cause an erroneous output if both *region A* and *region B* are fault-free.

To be able to tolerate transient faults, which may also cause output mismatch, the TED uses the *recomputation*. A transient fault will trigger the offline test, but the offline test will be (with a high probability) not influenced by that fault. The offline test will always mark M as faulty independently of the transient fault location (because *region A* outputs are marked valid, if no malfunction or fault is detected). Because – in case of the transient fault – it is not possible to state, which output is correct, the outputs must be recomputed after the offline test is performed. Unfortunately, it is not possible to distinguish permanent and transient fault, thus the recomputation must be performed always.

To reduce the massive delay overhead introduced by a permanent fault causing the continuous mismatch in M and M** outputs, the test result memory represented by the SYSTEM STATE REGISTER is introduced. The SYSTEM STATE REGISTER holds the results of the last performed offline test. The content of the register is used for correct output selection, instead of performing the offline test (which is time consuming). The offline test is performed only, if the SYSTEM STATE REGISTER is empty.

Because transient faults may also cause output errors, the SYSTEM STATE REGISTER must be cleared periodically to recover from transient faults. The clearing period must be chosen to reflect the expected transient fault rate (the period must be much lower).

The arrangement with the SYSTEM STATE REGISTER ensures, that the performance degradation is bounded by the SYSTEM STATE REGISTER clear period – this represents the worst case, as not every input of combinational logic necessarily reveals the actual permanent fault.

A. Sequential Logic

The TED is a redundant combinational logic structure, however it can be naturally used as a part of sequential logic. The usage of the TED is straightforward – the way to implement sequential logic using the TED is equal to using any other area-redundancy-based error-masking structure.

The only difference is, that the output of the TED system may be delayed and thus the register write enable must be connected to the TED ready signal (TED signalizes the correct output).

The example of the sequential logic is shown in Figure 4.

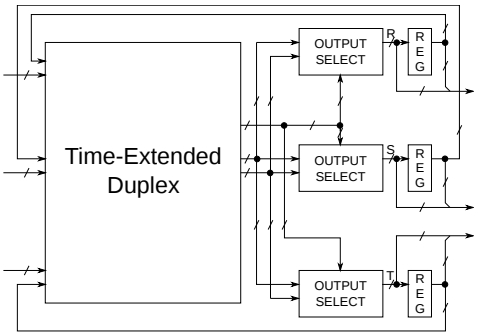


Fig. 4. A high-level example of a sequential logic including the TED

B. Comparison with TMR

The TED system is somehow similar to the TMR system – some of the TMR blocks are equivalent to parts of the TED system – even the interface is very similar – thus the TED description provided in this section is partially based on comparison with the TMR system. The TMR system is shown in Figure 5.

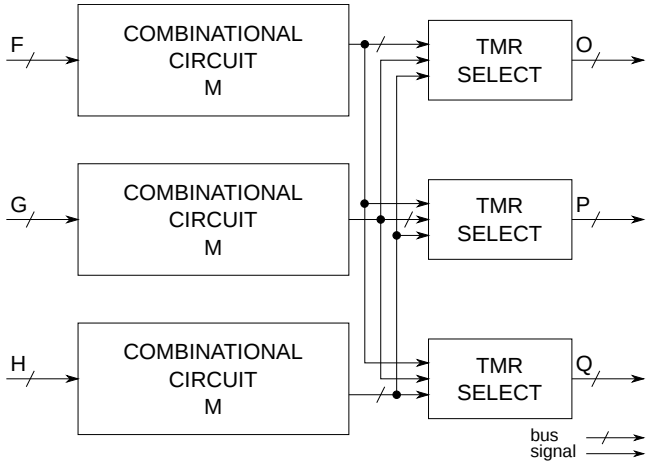


Fig. 5. Detailed scheme of the TMR system

The TED is comparable with the TMR in terms of delay and area only if: 1) the area overhead of the additional logic in TED is less than the area overhead caused by the third combinational logic module and 2) the delay introduced by the offline test is sufficiently small; 3) the offline test has high fault coverage.

In the following sections, we describe how to implement an area-efficient and quickly offline-testable combinational logic block M^{**}, which is the key part of the TED architecture.

The detailed description of the overall TED architecture including all details is provided later in Section VI.

IV. Monotonic Combinational Logic

In [14], we have shown, that monotonicity is required for a short-duration offline test, as it ensures, that the fault symptoms are not flipped (one to zero or zero to one) during the propagation to the circuit outputs, and thus simplifies the overall test.

Unfortunately an arbitrary combinational logic function cannot be implemented by AND/OR gates only, inverting function must be present. Because inverter breaks monotonicity, no inverter must be used inside the combinational logic block. To introduce inversion, preserve combinational logic monotonicity, and reduce the area overhead, we investigated several different ways.

A. Fully-Monotonic Design

The simplest solution to transform any logic function to a monotonic function is to adopt a fully monotonic logic design, the *Dual-rail* logic [19], [15]. In dual-rail logic, an inverter is represented as a wire-swap only and every signal is represented by a value on the complementary wires. Unfortunately the area of this approach is approximately double compared to the original single-rail circuit.

B. Isolated Monotonic Logic Blocks

In our case, we require monotonicity for testing. To reduce the area (and power), we allow inverters (single-rail to dual-rail converters) at the physical inputs of the M^{**} module. This allows to move from the dual-rail design to a structure we call *isolated monotonic* combinational logic blocks – see Figure 6.

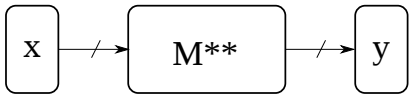


Fig. 6. Isolated monotonic logic block (M^{**}) contains AND/OR gates only. The arrays of inverters at its inputs/outputs are denoted x and y

In an isolated monotonic logic block, inverters are placed at the input or output of the monotonic logic block. When the monotonic logic is a dual-rail circuit, the input inverters transform the single-rail signals to dual-rail signals without disrupting the monotonicity of the isolated block – see blocks x and M^{**} in Figure 6.

For our method, a single-rail output of the module M^{**} 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 from the dual-rail implementation (only the positive outputs remain). Circuit parts feeding only the removed outputs should also be removed – see Figure 8. Then the dual signals (originating from the dual-rail implementation) serve as inverters replacements only. The number of outputs in such a circuit is equal to the number of outputs in the original single-rail circuit (module denoted M), and the number of the M^{**} inputs varies between once and twice the number of the M inputs.

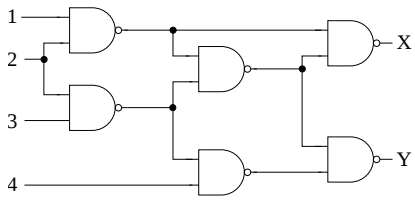


Fig. 7. Original NAND-based circuit

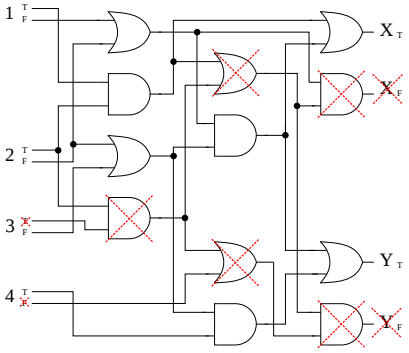


Fig. 8. Dual-rail logic circuit derived from the circuit in Figure 7 – every NAND gate was replaced by an AND and OR gate pair. The crossed-out gates, inputs, and outputs are removed by the reduction (M**)

After reduction, the resulting circuit in Figure 8 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 7 – both polarities of inputs 1 and 2 are required to compute outputs.

The reduction presented in [14] results in circuits having about 60% of the area of the dual-rail circuits on average. The resulting area for all benchmark circuits was between 50% and 100% of the dual-rail circuit area. The extreme values were achieved for smaller circuits only. Large circuits (those from the *IWLS 2005* benchmark set [20]) were close to the average.

If inverted outputs are allowed – see the block of output inverters denoted *y* in Figure 6 – both polarities can be selected during reduction. Here, the reduction success depends not only on the circuit structure, but also on the output polarity selection.

We developed five simple ways to achieve the highest reduction. Two simplest approaches take just the set of positive (as described above) or just the set of negative outputs. Another three approaches are greedy heuristics. All greedy heuristics start with the first output pair and continue with the other pairs. From each pair of the dual-rail circuit outputs, the output with smaller additional cost is selected (e.g., selecting one polarity implies adding less gates than selecting the other polarity). The heuristics differ just in the cost function. The cost functions are: number of gates, circuit size (gate sizes may be different – see Section VIII-A), and delay.

We applied all the developed approaches to benchmark circuits. When the best result for every circuit was selected, we achieved only 3% improvement on average compared to the approach taking just the set of positive outputs.

For the set of benchmark circuits, we additionally compared the heuristics with results of the *Monte-Carlo* method taking random output selections. We achieved no improvement compared to the best result given by one of the heuristics. Thus it can be concluded, that all of the greedy heuristics give results close to optimum.

The isolated monotonic logic blocks can also be created in a different way. It is possible to start from a single-rail circuit containing inverters and apply transformations preserving the logic function by moving inverters to circuit primary inputs/outputs, as shown in Figure 9. If some of the gate outputs are present in both forms – direct and inverted, the gate is duplicated – the first duplicate produces only the direct and the other only the inverted form. This heuristic produces also an internally monotonic circuit with inverters at circuit primary inputs/outputs and the number of circuit inputs is (usually) also greater than in the original single-rail circuit.

We performed a comparison of the heuristic we developed to perform the described transformations with the heuristics reducing the dual-rail circuits and no improvement has been achieved. The heuristic transforming the single-rail circuit directly gave always worse, or (in the best case) the same results as the best heuristic used for the dual-rail reduction.

The overall algorithm for constructing the smallest isolated combinational logic block is thus the following: take the minimized single-rail circuit (Figure 7) and create its dual-rail equivalent (Figure 8). Then apply all heuristics proposed for reduction (Figure 8) and select the best result.

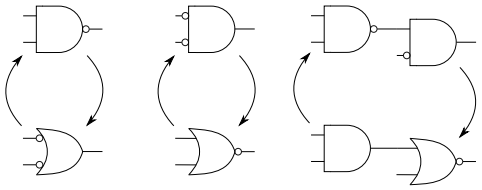


Fig. 9. An example of gate transformations allowing to move inverters to the circuit primary inputs/outputs

V. Proposed Offline-Testable Solution

The monotonicity itself is not enough to ensure, that all possible fault symptoms (one and zero) will be propagated up to the circuit primary outputs. It ensures, that these are without any change, but they may be still masked and thus not observable.

The short-duration test of M** requires a special gate design. The gate has to allow propagation of all possible fault symptoms (one and zero) up to the circuit primary outputs without any change and with no masking. We propose a novel reconfigurable gate structure allowing propagation of both fault symptoms (zero and one), which is similar to the dynamic *domino logic*.

The proposed gate can be configured to: 1) propagate fault symptom one (OR gate); 2) propagate fault symptom zero (AND gate); 3) set its output to 1 or 0; 4) work as a one-bit capacitance-based memory. First, we describe the domino logic in Section V-A, as it is the basis for our design, and then the transistor-level structure of the gate itself in Section V-B.

A. Domino Logic

Domino logic is a logic from the dynamic logic family [19]. The gates in dynamic logic work in two alternating phases: *precharge* and *evaluation*. In the first phase, the gates are forced to a defined state (by a dedicated control) and in the second phase, the gate inputs are evaluated.

Assume that the control signal, called *clock*, forces the gate output to 1 during precharge. In the evaluation phase,

the output remains 1 or switches to 0, depending on the input values, as shown in Figure 10a). This design style significantly reduces the load at the gate inputs and also the gate size compared to static CMOS, because the gate inputs drive the NMOS transistors only.

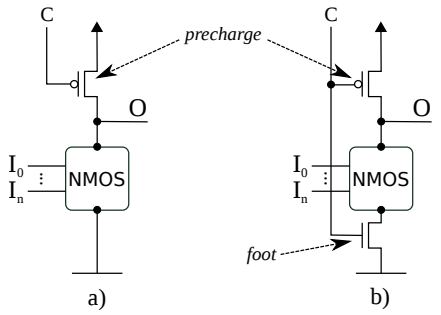


Fig. 10. a) dynamic-logic gate and b) dynamic-logic gate with foot

The precharge function can be realized by a single PMOS transistor only. If it is not guaranteed, that the gate inputs are always 0 during precharge, it may be necessary to add an additional NMOS transistor, which is called *foot* [19] – see Figure 10b).

The main issue with the domino gates described above is, that they require *monotonically rising* inputs during evaluation. The outputs of gates described above are *monotonically falling* (during evaluation) – this implies, that those gates cannot be simply concatenated to form deeper circuits. This problem can be solved by inserting a static CMOS inverter at the dynamic gate output – this design style is called domino logic. Domino logic gate outputs are *monotonically rising* during evaluation [19] – see Figure 11.

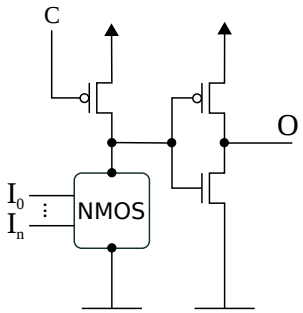


Fig. 11. Domino-logic gate

The disadvantage of the dynamic domino logic is, that it employs a high fan-out clock signal. This disadvantage is much lower, than one would expect, because:

- the clock controls only one (a single PMOS for unfooted) or two (one PMOS and one NMOS for footed gates) transistors per gate,
- clock-controlled transistors may be relatively small, because the design tolerates longer rising delays (up to half of the computational cycle for 50% clock duty cycle),

thus the load caused by transistor gates is relatively low. The main issue is, that there is the need for additional (balanced) metal wires to distribute the clock signal.

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 [19]. When the dynamic domino AND and OR gates with precharge to zero are used, the number of PMOS transistors is reduced significantly, compared to the number of NMOS transistors.

Domino logic thus represents a trade-off by providing the faster and smaller gates with reduced static power and increased dynamic power. Additionally, the domino logic was deeply studied and was also adopted by industry to develop high-performance chips [19].

B. Proposed Transistor-Level Structure

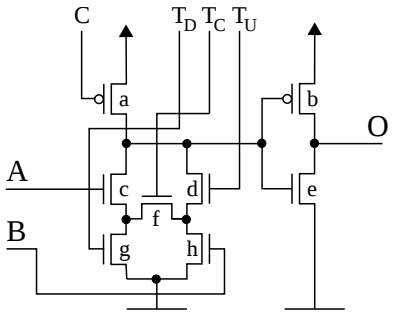


Fig. 12. Proposed transistor-level structure

The proposed structure based on domino-logic is shown in Figure 12 – this structure can realize both logic functions (AND/OR) depending on the control signals T_U, T_C and T_D. The proposed structure is still a domino logic gate. The novelty is in increased controllability of the gate, which is used for testability – during the test, the other functions of this structure are used.

As the described structure is domino-logic-like, it 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, where the output value is switched to 0 (↓) during precharge – depending only on the control (clock) signals. During evaluation, it preserves its value or is switched to 1 (↕) depending on both the gate inputs and control (clock) signals.

TABLE I. CONTROL SIGNALS FOR AND

step	C	T_U	T_C	T_D	O
precharge	0	0	0	0	↓
evaluation	1	0	1	0	↕

TABLE II. CONTROL SIGNALS FOR OR

step	C	T_U	T_C	T_D	O
precharge	0	0	0	0	↓
evaluation	1	1	0	1	↕

The additional *clock* signals, are used for mode selection, during the test, and as the foot control. The load at these additional clock signals is significantly smaller than at the

default domino-logic clock signal, because these signals control the smaller NMOS transistors only. Additionally, for AND function, only T_C is switched during computation and T_U and T_D are permanently closed – the same applies for the OR gate function.

Other combinations of the control signals are used during the offline test to set specified signals to a desired value, to preserve a logic value for a small amount of time between few clock cycles, or to raise a fault symptom, when a specific fault is present in the gate. An example of other control signals combinations is setting all control signals to 1, which causes that the gate output is switched to 0; when all control signals are set to 0, then the output is switched to 1. A one bit capacitance memory is realized by isolating the internal-node capacitance – T_C, T_U and T_D are set to 1 and C is set to 0.

According to the best of our knowledge, no similar structure has been proposed before.

VI. Time-Extended Duplex Structure Details

In this section, we detail the structure described in Section III by incorporating principles described in Sections IV and V. We also provide some implementation details and notes making the design flow clear.

The detailed scheme of the TED is shown in Figure 13. Please, refer to the higher-level TED scheme in Figure 3 and the TMR scheme in Figure 5 for comparison.

As the M** module is the *isolated monotonic logic block* (see IV-B), the OUTPUT SELECT modules aggregate the output inverters coming from this block. Compared to the TMR SELECT (see Figure 5), the OUTPUT SELECT decision is based on two inputs and (optionally) on the performed offline test result. This module is optimized for size and delay.

Complementary modules denoted as MODIF A and MODIF B aggregate input inverters coming from M** module as described in Section IV-B. These modules serve also as an offline-test generator. Depending on common 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 output. The internal structure and an example of the MODIF block is in Figure 14.

Outputs of both modules (MODIF A and MODIF B) are driven into an array of C-elements. C-elements serve as two-input comparators. This arrangement allows to detect faults at the TED inputs and in both combinational circuits. If inputs I an J differ, only bits, which are the same in both MODIF A and MODIF B outputs are guaranteed to be propagated thru C-elements. Different bits propagate depending on the previous state of affected C-elements. This implies, that C-elements sometimes mask input errors and sometimes do not mask them.

If the error is not masked by an array of C-elements (and also by the following combinational logic), the test is triggered by OUTPUT COMPARE, which signals *output mismatch*. The state-holding property of C-elements together with the MODIF A and MODIF B ability to set their output to all-one or all-zero vector allows to emulate a two-input multiplexer in just two steps (Figure 15), and thus the difference of inputs may be detected during the offline test. Additionally, C-elements allow to perform a concurrent test of MODIF A and MODIF B.

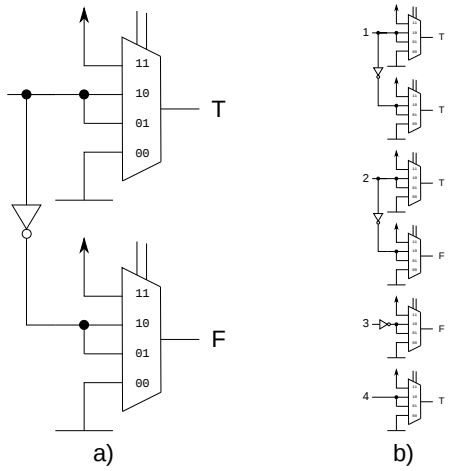


Fig. 14. The MODIF block is composed of pairs of multiplexers (a) with semi-dual outputs. Any but one of both multiplexers can be removed depending on the combinational circuit structure. An example of a complete MODIF block for the circuit in Figure 8 is in subfigure (b)

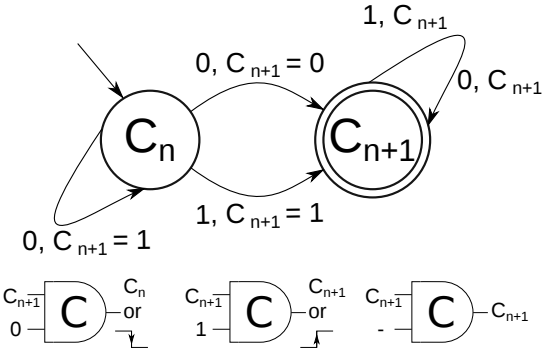


Fig. 15. C-element input to output switching behavior is given by the C-element state-holding property: the value C_{n+1} is propagated to C-element output in two steps – by setting the second input to 0 and subsequently to 1 – independently of the previous output state value (C_n)

The test controller is designed as a *self-checking* duplex circuit. It is composed of two identical and independent Moore-type controllers (CTRL A and CTRL B) and an array of C-elements. The C-elements serve also as two-input comparators. If both inputs of C-elements match, the output changes to the input value, otherwise the original output value is conserved [15]. 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 thus detected at least by one of the controllers.

Alternatively, the C-elements may be skipped and the self-checking controller may be designed as a master-slave, where one of the controllers produces the true control signals and the second just checks their correctness. Although this approach will reduce the area overhead, the arrangement with C-elements allows – in case of malfunction of one of controllers – to inhibit any transition on control signals: if the correctly working controller performs no output transitions, the C-elements outputs are stable.

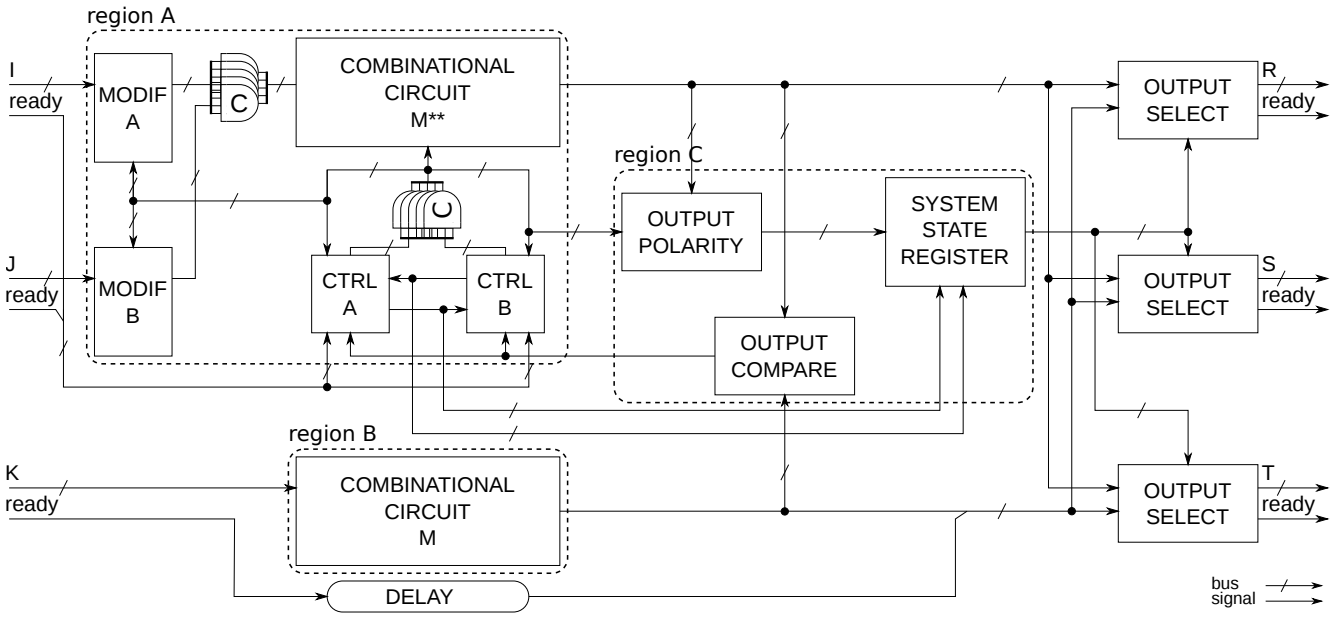


Fig. 13. Detailed scheme of the Time-Extended Duplex

The OUTPUT COMPARE module (a XOR tree) signals the output mismatch as a single-bit information to the test controllers. The size and delay of this part are influenced significantly by the number of combinational logic outputs.

The OUTPUT POLARITY module checks the offline test responses and the information about detected faults is stored into the SYSTEM STATE REGISTER.

VII. 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 VII-B and VII-C) or errors at the outputs of other modules (Section VII-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, erroneous 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 control signals of gates with the same gate level are joined to form a single control signal, driven by the test control logic. 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 IV), one circuit input is represented by one or two signals (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 excite and propagate the fault symptoms. This is the core idea of the short-duration test.

For example, if the gate preset to 0 is performed, then a stuck-open in an NMOS transistor of a gate at the first level will inhibit transition to 1, and thus 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. Every sub-test is described in a dedicated table (Tables III, IV and V) as the sequence of iterations over the circuit levels. For every step of each sub-test, the values of control signals C, T_U, T_C and T_D are defined for each circuit level. The value of the gate output (signal O) is defined in the last column – arrows are used for transitions caused by the control signal setting ($0 \rightarrow 1$ or $1 \rightarrow 0$) in case of fault-free behavior.

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 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 IV) to detect stuck-on faults. Additionally, the tests are able to detect some faults of the other type 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. A detailed example of sub-test 1 for a fault-free circuit is in Figure 16 and for a faulty circuit in Figure 17.

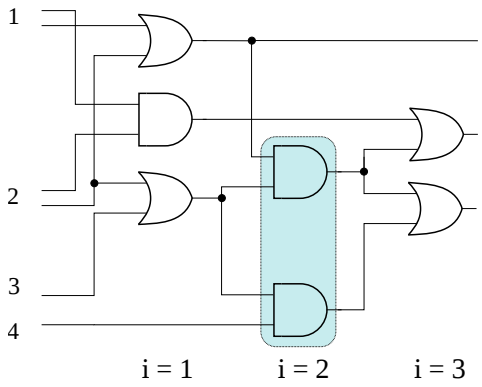
Table VI shows, which sub-test detects a stuck-open/stuck-on fault for a given transistor (see transistor labels in Figure 12).

TABLE VI. SUB-TESTS COVERING THE FAULTS

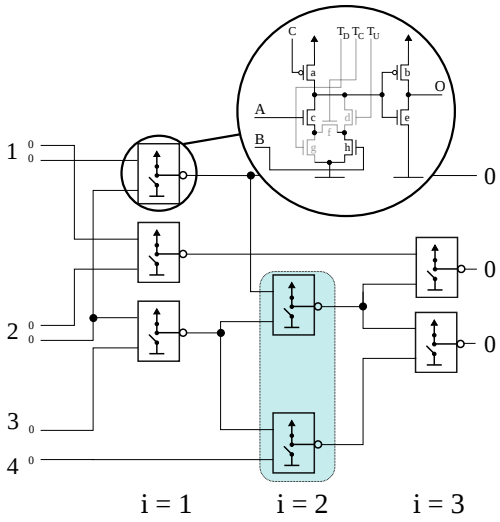
transistor	tests covering faults	
	stuck-on (short)	stuck-open
a	3	2
b	2*	1, 3
c	2	3
d	2	1
e	1*, 3*	2
f	2	1, 3
g	2	1
h	2	3

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.

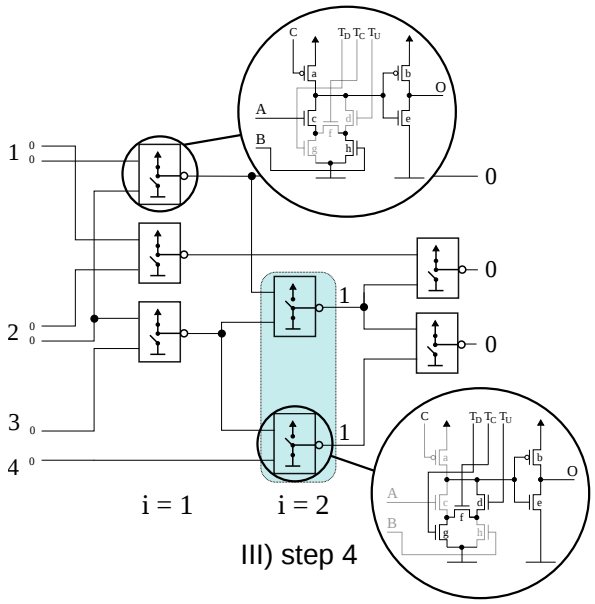
In other tests, the output is tested only once at the end of each sub-test. The tests principle is that the value at the faulty gate output is flipped even if it should stay constant during the test. The value flip in the lower level causes that a pull-down path in the following level becomes conductive 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.



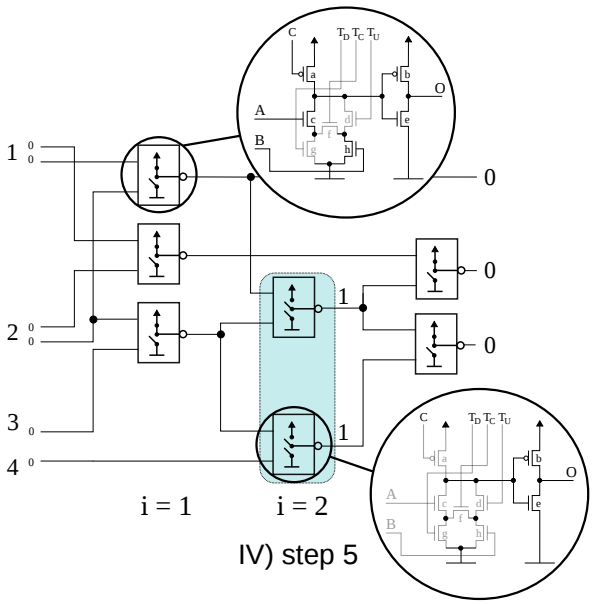
I) an example circuit



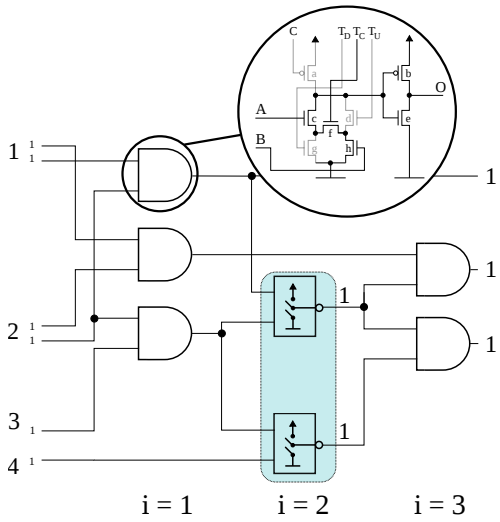
II) steps 1 + 2



III) step 4



IV) step 5



V) steps 6 + 7 + 8

Fig. 16. An iteration of the sub-test 1 for a fault-free circuit for level 2 (i = 2). Compare with the column “step” in Table III.

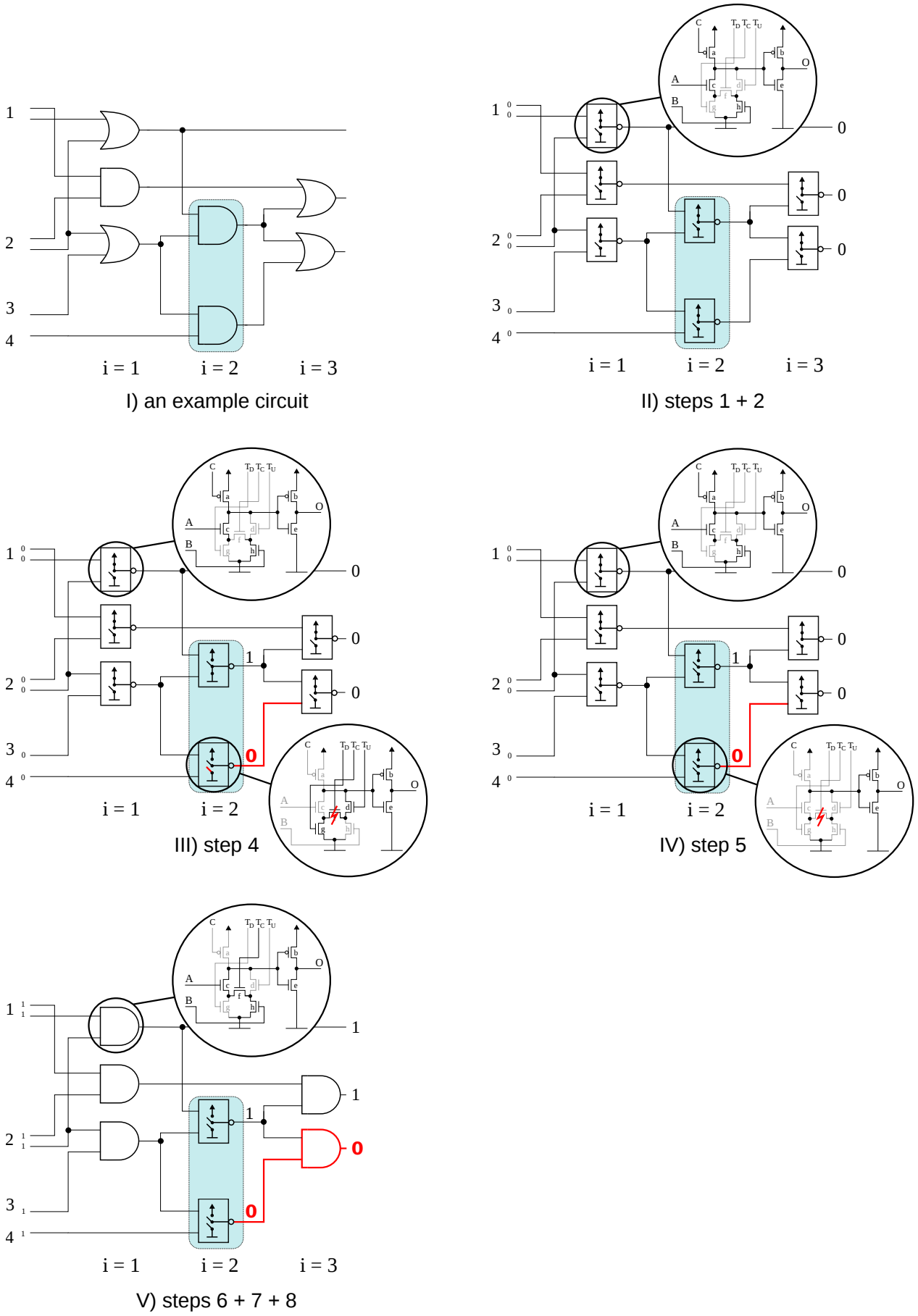


Fig. 17. An iteration of the sub-test 1 for a faulty circuit for level 2 (i = 2). Compare with the column "step" in Table III. A behavior for a stuck-open fault in transistor f is shown. Stuck-open faults in transistors g, d or b in the same gate expose equivalent fault symptom.

C. Uncovered Stuck-On Faults

Several tests in Table VI are marked by an asterisk. The stuck-on faults at transistors 'b' and 'e' need not 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 [16], [19], but in the past years, much work has been done also in the Built-In Current Sensors (BICS) area, starting from [21] in 1996, where the first BICS for deep sub-micron technologies has been presented. Recently, BICSs were proposed also for transient faults detection [22].

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 [21]. 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 performing 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 the test time).

As the used fault model does not fully reflect the bridging faults, it is advantageous to use BICS not only for uncovered stuck-on faults, but also for the online detection of bridging faults located at the gate outputs.

D. The Overall Test

The overall offline test is composed by concatenating all the sub-tests. Sub-tests 1 – 3 form the *short-duration offline test* of `M**`. If these tests are interleaved by *fault-current* measurement, as described above, they form the *complete short-duration offline test*. The offline test used in TED includes also the functional `M**` inputs test.

The offline test length is variable. If an erroneous input is identified during the inputs sub-test, the test is terminated, with 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.

VIII. Experimental Evaluation

In this section, we provide a comparison of the proposed solution with the standard static and dynamic domino logic implementations based on standard benchmark sets.

In section VIII-B, we provide a comparison of the proposed combinational-logic style (used to implement `M**`) with the static NAND-based and dynamic domino logic styles (used to implement `M`).

The complete comparison of the TED (Figure 13) and the TMR (Figure 5) systems is provided in Section VIII-C.

The comparison is based on a simplified and technology-independent, but accurate gate model described in Section VIII-A.

A. 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 [19].

We consider that the conductivity of an NMOS transistor is 2.5-times higher than the conductivity of a PMOS transistor. The same assumption as for the conductivities is made for the transistor gate capacitances. Thus, the load caused by the PMOS transistor of the same conductivity is 2.5-times higher than that of the NMOS one. This allows to approximate the load at the output of each gate and thus the technology-independent estimation of delay and dynamic power consumption is possible. More detailed and precise description of gate delay/size models can be found in literature under the term of *logical effort* [19].

Based on the transistor-level properties, the simplified 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 capacitance during precharge; *internal delay* expressing the time required to charge internal gate capacitance during evaluation; *input capacitance* expressing the capacitance 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 capacitance is not affected.

As described in Section V-B, 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 capacitance	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
$\text{AND}_{proposed}$	1.0	0.4	5.0	6.0	8.0
$\text{OR}_{proposed}$	1.0	0.4	5.0	4.0	8.0

For static NAND we have chosen a symmetric conductivity, which is usual [19]. 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 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:

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

B. Combinational Parts Comparison

A 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* [23], *LGSynth'93* [24], *ISCAS'85* [25], *ISCAS'89* [26], *IWLS 2005* [20] and *QUIP 2005* [27] and *EPFL 2016* [28].

The flow was as follows: the benchmark circuits were pre-processed by the *ABC* [29] 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 an 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 V-B. Circuit characteristics were extracted using the gate model from Section VIII-A.

The quantitative results of the comparison based on all circuits from the set are shown in Table VIII. M_{stat} represents the static CMOS NAND implementation and M_{domino} the dynamic domino logic implementation of M. M^{**} represents the proposed implementation. Table VIII shows size and delay comparison of the proposed (M^{**}) and the standard (M) module implementations – if the number in Table VIII is less than 100%, the proposed implementation (M^{**}) is better than the standard implementation (M).

TABLE VIII. M^{**} AND M COMPARISON

	min	max	median	avg
area: $\text{M}^{**}/\text{M}_{stat}$	52%	147%	84%	88%
delay: $\text{M}^{**}/\text{M}_{stat}$	38%	266%	92%	94%
area: $\text{M}^{**}/\text{M}_{domino}$	133%	133%	133%	133%
delay: $\text{M}^{**}/\text{M}_{domino}$	100%	100%	100%	100%

On average, the proposed structure is better than the static implementation of M in both area and delay (M^{**} compared to M_{stat}) and it has equivalent delay and 133% of the standard domino logic implementation area (M^{**} compared to M_{domino}) – for additional 33% of area, the proposed solution offers 100% fault coverage with respect to the selected fault model.

C. Time-Extended Duplex and TMR System Comparison

The comparison of combinational logic provides just partial information about the proposed method usability. Thus, a comparison of the complete TED structure (Figure 13) and the dynamic domino logic-based TMR system (Figure 5) is provided. TED is based on the domino logic variant (M^{**}), and it is better than the static CMOS in both area and delay at the same time, thus comparison with domino logic-based TMR system only makes sense.

The area of TMR SELECT (see Figure 5) 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 and outputs). From the empirical observations of relations between sizes of modules in the TED and TMR systems, we deduced the following expressions (|A| represents the area of A):

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

For a circuit where the expression (6) holds, TED is (likely) better than the TMR system from the area perspective (additional logic in TED occupies smaller area than it was obtained by using a duplex instead of a triplex structure).

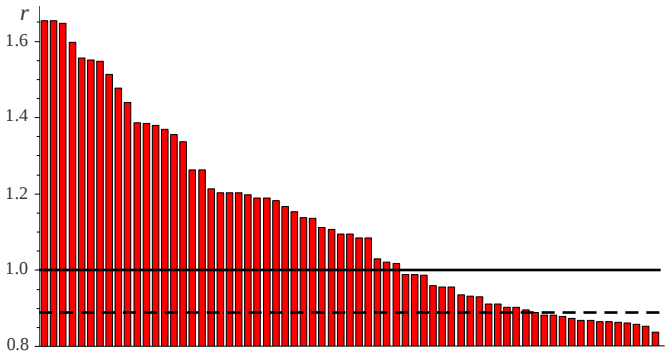


Fig. 18. The comparison of the TED and TMR systems for 67 selected circuits. The following ratio: $r = \frac{|\texttt{TED}|}{|\texttt{TMR}|}$ is shown. The circuits are shown descending ordered by r. The TED takes less area than TMR for circuits under the solid line. The equation (6) holds for circuits under the dashed line

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) 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. Note that combinational parts were extracted from sequential benchmark circuits, therefore such large numbers of inputs and outputs may appear for larger circuits.

The solid line in Figure 18 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. Because the TED brings additional delay, the dashed line symbolizes the border from where the TED is better than TMR (for most of the circuits), if the area and delay are equally important. Detailed data for some circuits from Figure 18 are in Table IX.

IX. Discussion

In the TED system, there is a number of additional modules compared to the TMR system. TED is smaller compared to the TMR system when the area saved in combinational logic is bigger than the area occupied by the additional modules in the TED system. The size of additional modules is almost constant (`CTRL A` and `CTRL B`) or depends on the number of combinational logic inputs and 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 impact of additional delay may be reduced by using the pipeline and speculative execution techniques. Speculative execution is often used with branch prediction to speed-up modern processors [30]. In the TED system context the speculation means, that the execution continues, although the correctness of the result is not known yet. As the additional delay in the TED system is smaller than half of the total delay of the TED system (for the proper set of circuits), the duration of such speculation is one cycle only. Thus, only one pipeline stage rollback must be performed if a fault is detected.

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

X. Conclusions

A new method for a design of error-mitigating circuits was presented and a comprehensive description of the overall error-masking architecture was provided. Our 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. We developed an efficient algorithm to create the isolated monotonic combinational logic blocks.

The error-mitigating ability of the proposed Time-Extended Duplex is very close to TMR, thus a comparison of TMR and TED was presented.

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 SGS17/213/OHK3/3T/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] 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.

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

[6] P. Fiser, P. Kubalik, and H. Kubatova, "An efficient multiple-parity generator design for on-line testing on fpga," in *11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools, 2008, DSD'08*, Sept 2008, pp. 96–99.

[7] J. Borecký, "Dependable systems design methods for fpgas," Ph.D. dissertation, the Faculty of Information Technology, Czech Technical University in Prague, 8 2015.

TABLE IX. THE COMPARISON OF THE TED AND TMR SYSTEMS FOR A SOME OF THE SELECTED CIRCUITS

Circuit name	# inputs	# outputs	M** area	M area	TMR area	TED area	r [%]
ac97_ctrl__part0 [20]	2167	2115	84 k	63 k	344 k	524 k	165,49%
ethernet__part0 [20]	10522	10400	359 k	269 k	1565 k	2405 k	164,80%
pci_bridge32 [20]	3376	3171	157 k	118 k	597 k	855 k	154,86%
oc_cfft_1024x12__part0 [27]	419	416	33 k	25 k	105 k	146 k	147,76%
bigkey [23]	452	229	37 k	28 k	100 k	126 k	135,55%
spi [20]	240	237	33 k	25 k	91 k	110 k	126,38%
des [23]	256	245	40 k	30 k	108 k	124 k	120,33%
bar [28]	135	128	24 k	18 k	62 k	72 k	120,33%
oc_aes_core_inv [27]	1056	667	113 k	85 k	331 k	364 k	116,65%
oc_aes_core [27]	659	529	99 k	74 k	271 k	286 k	110,64%
tv80 [20]	367	353	78 k	59 k	201 k	212 k	109,49%
des_area [20]	367	192	62 k	47 k	154 k	160 k	108,55%
des_perf__part0 [20]	5643	1572	961 k	721 k	2569 k	2422 k	98,79%
clma [23]	94	101	42 k	31 k	102 k	98 k	98,65%
table_out_0_52 [28]	100	52	40 k	30 k	95 k	89 k	95,63%
arbiter [28]	256	129	95 k	71 k	224 k	204 k	93,51%
table_out_0_42 [28]	103	43	45 k	34 k	105 k	96 k	93,19%
table_out_0_54 [28]	106	55	50 k	37 k	115 k	105 k	93,03%
aes_core [20]	657	529	275 k	206 k	666 k	592 k	91,07%
table_out_0_68 [28]	87	69	71 k	53 k	164 k	143 k	88,30%
table_out_0_103 [28]	106	104	110 k	82 k	255 k	218 k	86,45%
table_out_0_93 [28]	106	94	120 k	90 k	277 k	235 k	85,40%

[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] Y. Tamir and M. Tremblay, "High-performance fault-tolerant VLSI systems using micro rollback," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 548–554, Apr 1990.

[11] C. Weaver and T. Austin, "A fault tolerant approach to microprocessor design," in *International Conference on Dependable Systems and Networks, DSN 2001, 2001*, July 2001, pp. 411–420.

[12] T. M. Austin, "DIVA: a reliable substrate for deep submicron microarchitecture design," in *Microarchitecture, 1999. MICRO-32. Proceedings. 32nd Annual International Symposium on*, 1999, pp. 196–207.

[13] K. A. Campbell, P. Vissa, D. Z. Pan, and D. Chen, "High-level synthesis of error detecting cores through low-cost modulo-3 shadow datapaths," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[14] 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.

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

[16] 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.

[17] J. M. Acken and S. D. Millman, "Fault model evolution for diagnosis: Accuracy vs precision," in *Custom Integrated Circuits Conference, 1992., Proceedings of the IEEE 1992*, May 1992, pp. 13.4.1–13.4.4.

[18] S. Ma, I. Shaik, and R. S. Fetherston, "A comparison of bridging fault simulation methods," in *Proceedings of International Test Conference, 1999.*, 1999, pp. 587–595.

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

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

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

[27] Altera, "Benchmark Designs for the Quartus University Interface Program (QUIP)," 2005.

[28] Integrated Systems Laboratory LSI, École Polytechnique Fédérale De Lausanne, "The EPFL Combinational Benchmark Suite," http://lsi.epfl.ch/benchmarks, 2016.

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

[30] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach, Fifth Edition*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.