Czech Technical University in Prague
Faculty of Information Technology
Department of Digital Design

Error Correction Method Based on the Efficient Offline Test

by

Jan Bělohoubek

A Doctoral Study Report submitted to
the Faculty of Information Technology,
Czech Technical University in Prague

PhD programme: Informatics

Prague, May 2016

Supervisor:
doc. Ing. Petr Fišer, Ph.D.
Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Co-Supervisor:
doc. Ing. Jan Schmidt, Ph.D.
Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract

This report deals with error detection and correction methods in digital circuits and systems. This topic is still one of the most important ones in the area of the digital design.

The error detection or even correction ability of the system is paid for the area and/or time domains and for the design time. To find a reasonable trade-off is a difficult task.

Redundancy in both area and time domains was deeply studied separately in the past. Recently, also solutions combining time and area redundancy were presented. Most of them were designed for special architectures such as FPGAs. Some of the solutions combining time and area redundancy may be used in ASICs, but they still have serious disadvantages. Most of them allow only the soft-error handling.

The aim of this report is to present the state-of-the-art in the error detection and correction area and our recent research and also to propose a topic for the doctoral thesis.

A novel method combining time and area redundancy in an efficient way is presented. It is based on the unique combination of an efficient offline test, online error detection and the self-checking logic. The properties of the proposed method were evaluated at the transistor-level by using the benchmark circuits.

The proposed topic is closely related to error detecting and correcting architectures, circuit properties affecting the test length and the accuracy of the fault models.

Keywords:
Domino logic, dynamic logic, error detection, error correction, fault, fault coverage, fault model, offline test, online test, self-checking.

Acknowledgement

This research has been partially supported by the grant GA16-05179S of the Czech Grant Agency, ”Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features” (2016 – 2018) and by CTU grants SGS14/105/OHK3/1T/18, SGS16/121/OHK3/1T/18 and SGS15/119/OHK3/1T/18.

Contents

List of Figures

List of Tables

Abbreviations

ASIC	Application Specific Integrated Circuit
ATPG	Automated Test-Pattern Generator
BICS	Built-In Current Sensor
BIST	Built-In Self-Test
DIVA	Dynamic Implementation Verification Architecture
EMI	Electromagnetic Interference
FPGA	Field Programmable Gate Array
IDDQ	Quiescent I_{dd}, or quiescent power-supply current
NMR	N-Modular Redundancy
RAMS	Reliability, Availability, Maintainability, Safety
SC	Self-Checking
SEU	Single-Event-Upset
TED	Time-Extended Duplex
TMR	Triple Modular Redundancy
TSC	Totally Self-Checking

Chapter 1

Introduction

As digital systems tend to affect our daily lives, the digital systems *dependability* becomes more important. The digital systems dependability is a hot topic today because these control the critical infrastructure or they are built in the devices of our daily use. Thus their malfunction may cause injury or even death of many people.

Number of teams and system designers around the globe deal with their systems dependability and try to create *safe* and *reliable* systems while preserving the *low cost* of the system.

1.1 Motivation

Increasing the system dependability always brings additional costs [22]. The dependability is always conditioned by the *error detection* and *error correction* ability. Thus the system designer should look for balance between the additional costs and the benefits. From the hardware perspective the additional costs are always in two domains. These are the *area* and *time* domain.

The domains of area and time were deeply studied separately. There is a number of widely used approaches in both domains. The individual approaches often enable to tolerate a subset of system faults only or their application is expensive from the area and/or time perspective.

The approaches, where the cost in the domain of time is relatively small (such as *computation repetition*, *reconfiguration*) are able to tolerate temporary faults, which may be caused by *electromagnetic interference* (EMI) or by ionizing radiation, but not permanent faults caused by aging or physical defects. To detect faults caused by aging or by physical defects (in the domain of time), it is required to apply a time consuming test. These approaches may not be used during normal operation.

There are different approaches, which tolerate both temporary and permanent faults (such as *triple modular redundancy* – TMR) with almost no additional cost in the time domain, but their cost from the area point of view is relatively high.

1.2 Problem Statement

Although the area of dependability was deeply studied, there are still open problems. An efficient combination of approaches from the domain of time and the domain of area may potentially reduce the cost of the dependable systems. The approaches combining the area and time *redundancy* presented in the past were often designed for specific applications. For specific applications under limited requirements, this was sufficient, but they have serious problems from the general point of view: the approaches are able to tolerate only a subset of all possible faults, they have a single-point-of-failure, etc.

Up to our knowledge, no efficient general solution combining time and area redundancy with a fault immunity comparable to the area-expensive TMR has been presented. We developed an efficient solution, as it is briefly presented in Section 1.3 and in detail in Chapter 3.

The aim of this report is also to describe approaches currently used for the *error detection* and *correction* and evaluate their properties. This is briefly described together with the theoretical background and brief terminology explanations in Chapter 2.

1.3 Related Work/Previous Results

Our previous work was focused on designing special hardware structures allowing to perform an efficient offline test of the combinational logic [JB.2]. The efficiency of the offline test is given by the test length, which is very short compared to traditional offline-tests with a comparable *fault coverage*. We call this test as a *short-duration offline test* [JB.2].

By combining the offline-testable combinational logic with the area overhead required for error detection, we proposed a system, which has significantly less gates and the fault-immunity comparable to TMR [JB.1], [JB.3].

As our work has been accepted for presentation at [JB.1], the raw evaluation methodology and the used *fault model* were criticized. Thus we adaptated our method to a more accurate transistor-level fault model. We also derived a detailed area/delay gate model based on transistor parameters for a precise technology independent comparison of the different approaches. This work was presented in the paper submitted to the ASYNC conference [JB.5]. Although the paper has been rejected, the reviewers appreciated the detailed evaluation methodology. The reason for the rejection was, that the presented results were not too satisfactory in general. Some of the reviewers also concluded, that the topic of the paper is marginal for the ASYNC conference.

Afterwards we improved the hardware structures used, which increased the usability of our approach while the beneficial properties were preserved. The preliminary results show, that for a particular subset of circuits, significant savings may be achieved by using this improved approach. The last results were submitted to DSD 2016 conference [JB.6] and if accepted, the paper will be presented in August.

1.4 Structure of the Report

The report is organized into 5 chapters as follows:

1. *Introduction*: Describes the motivation behind our efforts together with our goals.
2. *Background and State-of-the-Art*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art. At the end of this chapter, the current methods combining the time and the area redundancy are briefly described.
3. *Overview of Our Approach*: Presents the novel method combining the time and the area redundancy an efficient way.
4. *Preliminary Results*: Contains the experimental evaluation and the used model description. The preliminary results and the results discussion are also present.
5. *Conclusions*: Summarizes our research, suggests possible topics of the doctoral thesis and the further research, and concludes the report.

Chapter 2

Background and State-of-the-Art

As the digital systems tend to affect our lives deeply every day, the area of digital systems *dependability* was closely studied in the past years. In the literature, the dependability is often defined as a combination of the following parameters: Reliability, Availability, Maintainability, Safety (RAMS parameters) and Testability. Optionally, other parameters may be included [25].

The importance of each parameter depends on application. But to achieve any level of dependability, some kind of *redundancy* has to be involved. Redundancy allows the device to handle a malfunction. When an error occurs, the required behaviour depends on the application: just a safe shut-down may be required, degraded minimal function should be preserved, or the device should continue in operation with (almost) no effect even if it contains a faulty part.

The redundancy itself serves to tolerate malfunctions in computational units, storages, or communication channels. Different types of redundancy may be employed, such as information (error detecting and correcting codes), software (N-version programming), *time* (recomputation, offline test), or *area* (concurrent computation in independent units – e.g. N-modular redundancy). Involving any type of redundancy brings additional design, manufacturing and operating costs [22], thus the balance between costs and benefits has to be targeted. From the hardware point of view, all types of redundancy affect the *time* (latency and throughput) and/or the *area* domain.

In a digital system, the additional redundancy offers an information which enables to identify (and repair) an *erroneous* output of the system. To get this kind of information, it is possible to employ number of methods, e.g. perform parallel computations by using independent computational units, perform recomputation using the same unit, or employing offline tests [25].

2.1 Physical Faults

An erroneous system output is caused by a fault at the physical level. In the literature, faults are often categorized according to the duration [25]. The faults, which occur and disappear are denoted as the *transient faults* (sometimes called *soft-errors*). When a transient fault disappears, the function of the affected system is fully restored. Some authors also distinguish *short-duration transient* and *long-duration transient* faults [34]. The example of an transient fault is a change in a memory cell caused by the electromagnetic interference. Sometimes, the transient fault removal may require device reset or re-initialization (correction of a bit-flip in FPGA may require the FPGA reconfiguration [7]). The faults affecting the system permanently are denoted as *permanent faults*. These are often caused by physical defects. Sometimes, *intermittent faults* are mentioned. The intermittent fault never goes away entirely, but it sometimes affects the system function and sometimes is hidden. The intermittent faults often tend to became permanent [25].

Another classification is based on the fault nature. The *stuck faults* and *bridging faults* are those caused by a short; *open faults* are caused by a wire interruption and *delay faults* may be caused by transistor wear-out, voltage drop, or by process variations (capacity or resistance). A high-energy particle may cause a *single-event-upset* (SEU) or a bit-flip.

2.2 Error Detection and Correction

As a result of a physical fault presence, the systems may produce faulty outputs. The methods of error detection serve to identify an incorrect output. The full error detection is conditioned by the *self-checking* property.

The system is *self-checking*, if an occurrence of a fault leads to a faulty output. The important sub-class of the self-checking circuits set are the *totally self-checking* circuits (TSC). The TSC property means, that any fault in the circuit may not cause an undetectable faulty output [33]. Thus any architecture offering full error detection must be TSC to provide full *error coverage*.

At the self-checking circuit output, there must be a *checker* (or a *voter*), which is able to recognize an invalid output.

Error correcting methods offer an additional value compared to the error detecting methods. If the system is error correcting, it may continue in operation even if a part of such a system is faulty.

The error correcting and detecting methods may be classified 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. From the *physical faults* point of view, the area redundancy-based methods are well suitable for mitigation of errors caused by both *transient* and *permanent* faults. Time redundancy is often used for mitigating errors caused by *transient* faults (computation repetition).

2.2.1 N-Modular Redundancy

A well-adopted approach in the area domain for error detecting and correcting systems is the *N-modular redundancy* (NMR). The redundancy of NMR is in the area domain and the error detection (and correction) is performed online, while only small delay penalty is caused by the voter circuit.

Duplex system

The simplest way to achieve *online error detection* is by duplicating the original module and thus creating the *duplex*. The joint output of two identical modules allows to distinguish a correct (outputs match – both outputs are correct) and an erroneous output (outputs are different – one of the outputs is faulty) in case, that at least one of the duplex parts is fault-free. This characteristic corresponds to the TSC property. The duplex voter does the error detection job (employs the TSC property) and if a fault is present, it reports the fault presence. Duplex is the simplest example of a self-checking system.

Triple modular redundancy

In the NMR family, the *online error correction* can be achieved by (at least) triplicating the original module. This is called a *Triple modular redundancy (TMR)* – see Figure 2.1a. TMR is able to produce correct output, if at least two of three identical modules are fault-free.

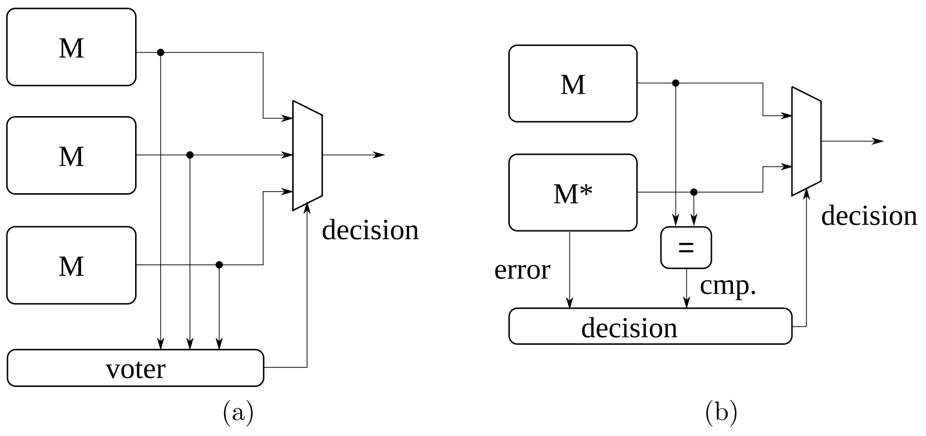


Figure 2.1: Conceptual schemes of an error-correcting (a) TMR and (b) duplex system with a self-checking module `M*`

2.2.2 Self-Checking Modules

From another point of view, self-checking (error detecting) modules in general can be used to construct an error-correcting system – see Figure 2.1b. A simple example is the *bi-duplex system*. It is an error-correcting system consisted of two self-checking duplex modules [25].

The disadvantage of these self-checking circuits is their size. In fact, the self-checking circuit size is typically close to the size of duplex [35], [25].

Asynchronous Self-Checking Circuits

The duplex system is not the only example of a self-checking circuit. Self-checking circuits are the subject of deep research in the area of asynchronous circuits [37]. The asynchronous approach is based on *dual-rail* logic [16], [35], [17]. Here each signal is provided in its direct and complemented form [37]. As every signal is provided in complementary forms, incorrect states are detectable. These circuits contain a *completion detection* parts. Some faults may cause, that such a circuit stops instead of producing erroneous outputs – completion detection never *fires* [16] – these faults are watchdog-detectable. Some other faults may cause invalid outputs – these faults are detectable by the additional hardware [16].

2.2.3 Time Domain Methods

Time domain methods can be used to eliminate programming errors and the transient faults.

N-Version Programming and Code Verification

The *N-version programming* offers immunity to errors caused by developers. Here the code – software or hardware description and test – is developed N-times by different teams. The created software can be run simultaneously in production, while the hardware description and test are used for verification. It is supposed, that the different teams do not make the same mistakes [13]

Computation Repetition

The *computation repetition* offers immunity to (some of the) transient faults but it is unable to overcome permanent faults. If the computation is repeated several times, both error detection and correction may be provided [25].

2.3 Fault Detection

Errors are caused by physical defects, which are modelled by faults, thus error detecting (and correcting) methods are also fault detecting methods. Most of the other fault-detecting methods are offline methods. Thus the system shutdown is required and the system under test may not continue in operation while the test is in progress.

The test methods employ two important parameters: the *observability* and the *controllability*. The observability is the ability to propagate a *fault symptom* from the fault location to the circuit output, where it can be observed. The controllability is the ability to control any signal in the circuit by applying input vectors.

The important quantitative property of every test is the *fault coverage*, which is defined as follows:

$$c = \frac{\#\ faults\ detected}{\#\ faults\ modeled} \tag{2.1}$$

As the fault coverage depends on a number of modeled faults, an accurate fault model is required. There is a number of different fault models used for digital systems. These differ in the coverage of physical defects. The coverage of the real physical defects is connected with the fault model precision and also with its complexity. It has been stated, that most defects are caused by *bridging faults* [21], [27]. It is relatively hard to model bridging faults, thus the time and input invariant models are widely used [1]. In both academy [25] and industry [14], the *stuck-at-fault* model is still widely used.

2.3.1 Fault Models

Stuck-At-Fault Model

The stuck-at-fault model is a *gate-level* model. It comes from the deep history of digital systems, when the main source of faults was the interconnection between logic gates [1]. This model considers two types of permanent faults – permanent logic one (s@1) and permanent logic zero (s@0) at the gate input or output. Today, the stuck-at-fault coverage is still widely used as a metric for the test quality even in industry [14].

It has been shown, that the stuck-at-fault model may be used to detect some of the bridging faults [29], [30], and that a high stuck-at-fault coverage implies high bridging fault coverage [26].

Stuck-Open/Stuck-On Model

A more detailed model is the stuck-open/stuck-on fault model. It is a *transistor-level* model. It considers two types of faults – one corresponds to a permanently open transistor and the second corresponds to a permanently closed transistor. This model may be also defined as an extension of the stuck-at-fault model, where s@1 and s@0 are considered at every transistor gate [1], [10]. Thus the set of modelled faults includes all faults from the stuck-at-fault model and it models number of additional faults [1].

Wired-AND and Wired-OR Model for Bridging Faults

The most commonly used fault model for the bridging faults is the Wired-AND/Wired-OR fault model [1]. A single fault is represented as the AND/OR logic function. As the Wired-AND/Wired-OR fault model [29] does not reflect the behaviour of all types of bridging faults, a number of models reflecting this behaviour has been proposed [18], [26].

Voting Model for Bridging Faults

In [1], the voting model related to the *Byzantine generals problem* for bridging faults has been presented. It is based on the transistor-level comparison of pull-up and pull-down paths conductivity.

2.3.2 Offline Test Methods

The offline methods suspend the system, so the system under test cannot continue in operation while the test is performed. The offline methods are used to detect permanent or long-duration transient faults.

Application of Test Vectors

The representative of the offline methods is the test vector application and response observation. It can be used to identify the *long-duration transient* or *permanent fault* presence inside the system under test [25]. Thus it is widely used after fabrication or during the maintenance. It may also be used for systems or system parts, where longer shut-down is not critical or which are regularly down. The test execution time is typically significant, thus performing the test during normal operation is not possible. The test generation is one-time, but also the time consuming task. The test is generated by an *ATPG* (Automatic Test Pattern Generator). If the system is regularly offline, it is possible to perform the offline test (*Built-in self-test* – BIST), but the test patterns must be stored in an on-chip memory. Although the test compression is possible [24], [6], additional overhead is introduced.

IDDQ Testing

The other widely used offline test method, which is based on the fault-current measurement is the *IDDQ testing* (quiescent I_{dd}, or quiescent power-supply current) [14]. Some of the physical faults may cause shorts, and thus an increased power consumption. The advantage of the IDDQ testing is, that just the controllability property of the test vectors is important – ATPG and alternatively the memory for the test patterns is required, as for the test vector application method. The observability is achieved thought the measured current. The fault-current measurement is widely used in practice [39], [41]. IDDQ testing is suitable for bridging faults detection.

2.3.3 Online Test Methods

The faults may also be detected during normal operation. The error detection methods (Section 2.2) have the ability to detect fault presence. Another representative of the online fault detection methods are the Built-In Current Sensors.

Built-In Current Sensors

As mentioned above, some faults may cause increased power consumption. The fault-current is typically measured offline [39], [41], but in the past years, much work has been done also in the Built-In Current Sensors (BICS) area, starting from [3] in 1996, where the first BICS for deep sub-micron technologies has been presented. Recently, BICSs were proposed also for transient faults detection [32].

One BICS is able to monitor only a limited number of power rails due to a limited resolution and current load capacity. This implies that using multiple parallel BICSs for the whole circuit [3] may be required, especially for bigger circuits. As the principle of BICS is the same as for the IDDQ testing, it is also suitable for the bridging faults detection.

2.3.4 Fault Detection-Based Error Correction

The problem of the offline test is that although the system test passes, the system may be still faulty. Typically the test applies just a small amount of possible system inputs or their combinations. In fact, if the test passes, the information provided is: the system works correctly for the given test inputs in the given order. The certainty provided by the test depends mainly on the used *fault model* and the test *fault coverage*. Thus offline testing can be efficiently used for error correction only if the test has a significant and realistic *fault coverage*.

2.4 Combining Time and Area Redundancy for Error Correction

The online methods increase the costs in the area domain and the offline methods in the time domain. Sometimes, it may be efficient to combine both approaches. As applying the offline methods disables the system function for some time, it is efficient to use an online method for error detection and when an error is detected, then use an offline method for error correction. Thus the online error detection is a trigger for the offline error correction.

The works combining time and area redundancy often deal only with transient or *soft faults* like *single-event upsets* (SEU), e.g., [36]. The usability of presented solutions, where not only transient faults are considered is problematical due to a delayed fault detection [23], [5].

Today, according to the best of our knowledge, no general approach combining time and area redundancy in an efficient way with the fault-immunity comparable to TMR has been presented by someone else. The following approaches combining time and area redundancy were presented in the past:

2.4.1 Handling SEUs in FPGAs

This approach is used for handling transient *bit-flips* in FPGAs. A kind of duplex system is used to detect errors online and then reconfiguration is performed to repair the faulty parts [19], [7] – see Figure 2.2.

2.4.2 Using Dependable Parts

The approach presented in [15] relies 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

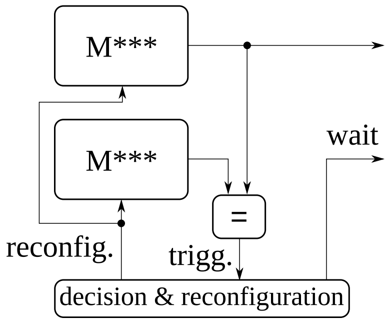


Figure 2.2: Conceptual scheme of an error-correcting duplex employing the reconfigurable modules `M`***

allows the fault-recovery. The system is described at the high level. It was designed as the radiation tolerant. For the given application, the approach is efficient. But this approach deals only with transient faults in the reprogrammable part. Additionally it is not general and the system contains parts, which may be denoted as a single point of failure.

2.4.3 Backup Units

This approach employs backup units. The backup units are used for periodical checking of the functional units. In case of a fault, the functional unit is replaced by a backup unit [23], [5]. The significant disadvantages of this approach are, that the fault detection is delayed significantly and that the unit output is not checked in every cycle. Some of the input vectors may disclose present faults but other vectors may produce correct results. Thus the fault need not to be identified while the incorrect output is produced. This is why the practical usage of this approach is problematical.

2.4.4 Pipeline Rollback

To introduce some level of reliability into high-performance chips, the problem with additional delay caused by checkers was studied years ago [38]. Although fast checkers are used, some delay is still introduced and additionally the area 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 [38]. Similar approaches were presented later, e.g. [40], [4] or [11]. The presented pipeline rollback-based approaches are suitable for handling soft-errors.

2.4.5 Dynamic Implementation Verification Architecture

The approach called *dynamic implementation verification architecture* (DIVA) presented in [4] is similar to the pipeline rollback. It is based on the 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.

Chapter 3

Overview of Our Approach

As described in 2.4, the current works combining the time and the area redundancy often deal only with transient or *soft faults* like *single-event upsets* (SEU), e.g., [36]. The other methods presented in the past require recomputation after reconfiguration, e.g., [15], or their practical usage is problematical [23], [5].

Our approach is more general – an erroneous output is detected with a small delay only, it is possible to secure any combinational circuit with no limitations, and it detects both transient and permanent faults. The error correction ability is equivalent to the TMR system and the resulting circuit can 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.*

Our method combines offline testing with the functional module duplication. Computation is performed in parallel by two independent modules. If the outputs differ, an 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 second module as the correct one. The offline testable module is denoted as `M**` – see Figure 3.1. We call the system, where one `M` and one `M**` module is used as a *Time-Extended Duplex* (TED).

From a conceptual point of view, the *offline test* can be used to provide the same information as the self-checking module from Figure 2.1b. The main difference is that the decision may be delayed. Additionally, this approach can be efficiently used, only when the offline test has a complete *error-coverage.* As any error is caused by a fault presence, a test with a complete *fault coverage* with respect to a realistic fault model has to be used. 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 [20], [12].

Our approach requires a special test. The test vectors and the test responses should be easy to produce and check in hardware. The test length must be in orders of tens computational (clock) cycles only and the required fault coverage is 100% with respect to the used fault model. We call such a test a *short-duration test.*

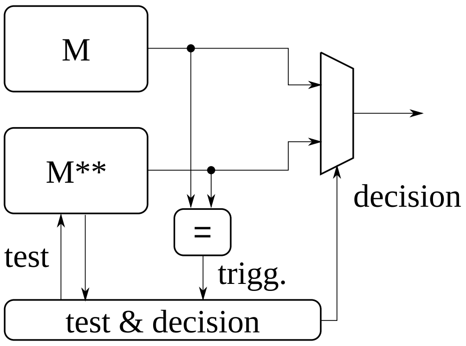


Figure 3.1: Conceptual scheme of an error-correcting duplex with module `M**` (short-duration testable) – the *Time-Extended Duplex*

3.1 Fundamental Ideas

The fundamental ideas behind the proposed method are shortly described here. These are discussed in detail in the following Sections.

Stuck-At-Fault Propagation

We started our research in the area of *monotonic circuits*. These are circuits containing no inverters. The $0 \rightarrow 1$ change at the circuit inputs, may only cause $0 \rightarrow 1$ changes in the circuit and the $1 \rightarrow 0$ change at the circuit inputs, may only cause $1 \rightarrow 0$ changes in the circuit. An example of a monotonic circuit is the dual-rail (AND/OR based) circuit. We observed, that if every gate output in such a circuit is connected to at least one OR gate and to at least one AND gate, the test with 100% fault coverage with respect to the stuck-at-fault model requires only two test vectors. These are *all-zero* and *all-one* vectors. It is intuitively clear, that such two-vector-testable circuits are rare.

Circuit Transformation in the General Case

The circuits with the property enabling the two-vector-test are rare, but it is possible to modify any circuit to meet the described requirements. The circuit transformation is done in two steps. At first the original circuit is transformed to satisfy monotonicity. Dual-rail logic is a straight-forward transformation. The dual-rail circuit must be then modified to increase its observability – every gate output in such a circuit will be connected to at least one OR gate and to at least one AND gate.

It is possible to satisfy the two-vector-testability requirement by adding AND and OR gates into the monotonic circuit. Such a circuit will have more outputs (additional outputs serve just to increase observability) and more gates. Thus the resulting circuit would be definitely bigger than the original dual-rail circuit (and naturally bigger than the original single-rail one). The evaluation of this approach will be done in a future work.

The other option is to preserve the existing circuit structure and replace traditional gates by reconfigurable gates. The reconfigurable gate should behave as an AND or OR gate depending on control signals. This approach does not increase the number of gates. The gate fan-out is also preserved. Additionally it allows the fault localization – the fault symptoms are propagated to potentially fault-affected outputs. This property increases the locality of the error detection. The disadvantage is, that testing of the reconfiguration and the novel gate design is required – the reconfigurable gate is a bigger and more complicated structure compared to the traditional gate structures.

Reducing Area Overhead

The dual-rail logic introduces approximately double area compared to the single-rail implementation. If no dual-rail output is required, it is possible to reduce the area by removing half of the dual-rail outputs. Thus only the single-rail output remains and up to half of gates can be removed with the removed outputs – these are gates feeding only the removed outputs. After the reduction, the additional logic serves as the inverter replacement only. This allows to transform any single-rail circuit to the monotonic single-rail circuit with smaller area overhead.

Reconfigurable Gates and the Stuck-At-Fault Model

In [JB.1], we presented the novel design of the reconfigurable gates. We proposed the combinational logic test with 100% stuck-at-fault coverage and we proposed the conceptual scheme of the Time-Extended Duplex.

Reconfigurable Gates and the Stuck-On/Stuck-Open Faults

As the stuck-at-fault model is not very realistic, in [JB.6], we propose the novel design of the reconfigurable gates and the combinational logic test with 100% stuck-on/stuck-open transistor-level fault coverage.

3.2 Time-Extended Duplex

The basic principle of the Time-Extended Duplex has been presented in the previous text. The detailed scheme of the TED system is shown in Figure 3.3. The detailed scheme of the TMR system is shown in Figure 3.2 for comparison.

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

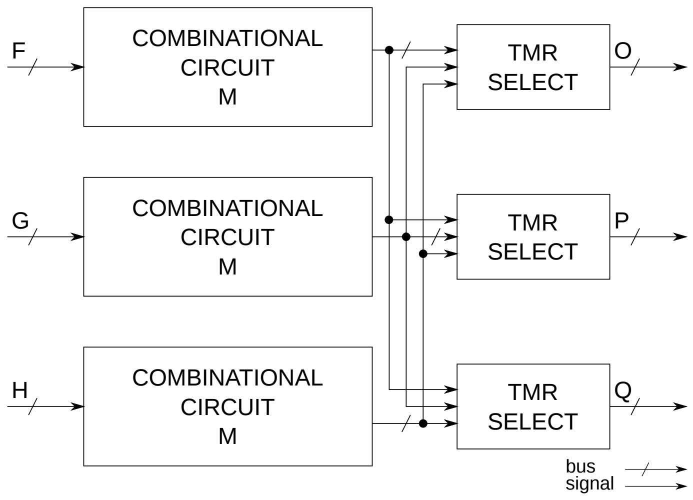


Figure 3.2: Detailed scheme of the TMR system

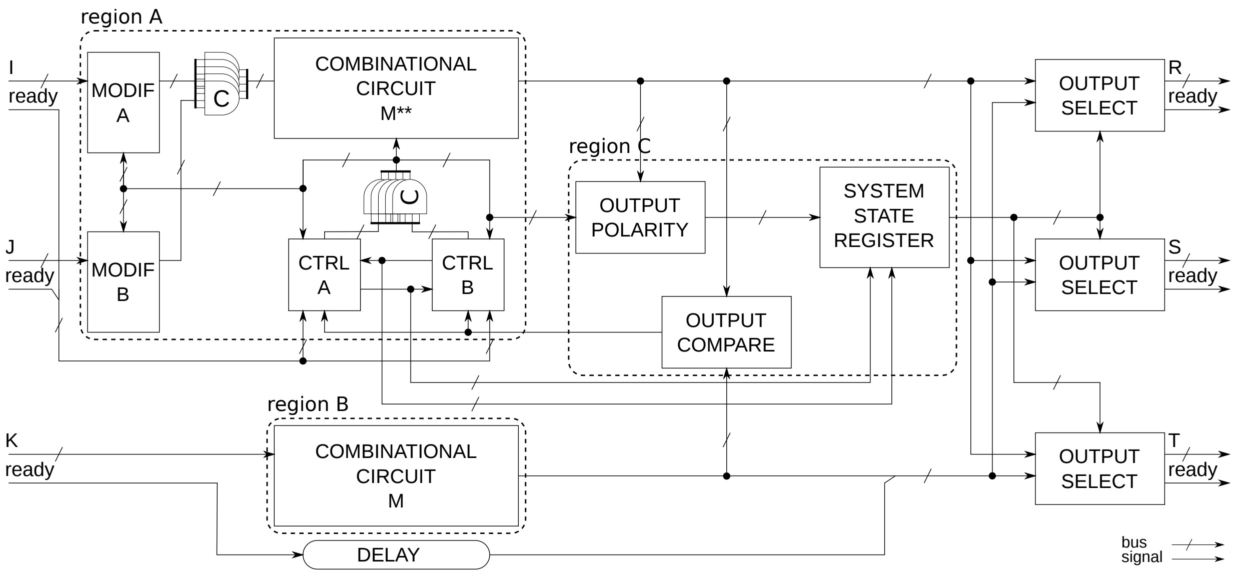


Figure 3.3: Detailed scheme of the TED system

3.2.1 The Time-Extended Duplex Structure

The TED computational part contains one original module (`M`) and one functionally equivalent to `M`, but offline testable module – (`M**`). The voter (`OUTPUT SELECT`) decision is based on the offline-test result and the outputs of both combinational modules. 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. The output difference is signalled by the `OUTPUT COMPARE` module.

The test controller is designed as a *self-checking* circuit (see TSC in Section 2.2). 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 [37]. The C-elements outputs are used to control the test and these are also driven back to each controller to compare with the controllers' 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. This arrangement allows to detect faults at the TED input. `MODIF A` and `MODIF B` serve also as the test-input generators. Depending on the control signals, the output of `MODIF A`, `MODIF B` is an all-zero or all-one vector or a modification of the input vector, which is characteristic for every `M**` module (see next sections). The state-holding property of C-elements together with the `MODIF A` and `MODIF B` ability to set its output to all-one or all-zero vector allows to emulate a multiplexer in just two steps. Using C-elements in place of a multiplexer allows to test `MODIF A` and `MODIF B` concurrently.

3.2.2 Error Detection and Correction

The TED system 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 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.

3.3 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* [41].

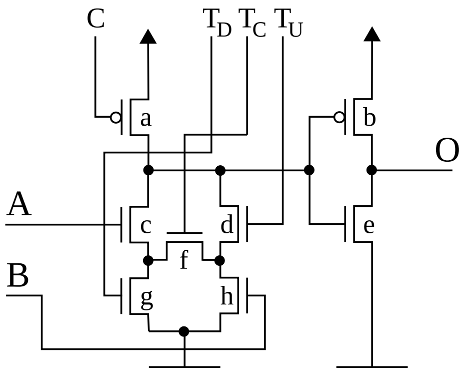


Figure 3.4: Proposed transistor-level structure

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 [41]. 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 3.4 – this structure can realize both logic functions depending on the control signals $\mathtt{T}_U$, $\mathtt{T}_C$ and $\mathtt{T}_D$. 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 used 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 3.1 and 3.2.

step	C	$\mathtt{T}_U$	$\mathtt{T}_C$	$\mathtt{T}_D$	O
precharge	0	0	0	0	$\downarrow$
evaluation	1	0	1	0	$\updownarrow$

Table 3.1: Control signals for AND

step	C	$\mathtt{T}_U$	$\mathtt{T}_C$	$\mathtt{T}_D$	O
precharge	0	0	0	0	$\downarrow$
evaluation	1	1	0	1	$\updownarrow$

Table 3.2: Control signals for OR

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 [41], [37]. In dual-rail logic, an inverter is represented as a wire-swap only. From the gate-level point of view, the circuit is monotonic.

3.4 Reducing Dual-Rail Logic

Dual-rail logic 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 3.5 and 3.6. 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*` [JB.1].

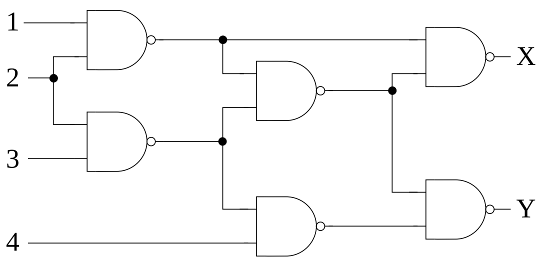


Figure 3.5: Original `NAND`-based circuit

After the reduction, the resulting circuit example in Figure 3.6 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 3.5 – 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 especially for bigger circuits (no improvement for most of the circuits and 3% on average). For some of the smaller circuits, the improvement may be significant – see Appendix A. Unfortunately the output inverters bring additional problems: testing of the output inverters requires additional

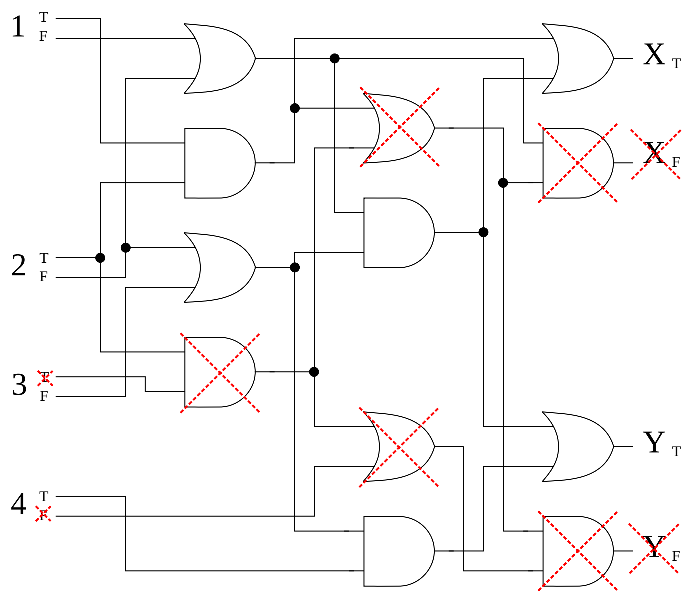


Figure 3.6: Dual-rail logic circuit derived from the circuit in Figure 3.5 – every `NAND` gate was replaced by an `AND` and `OR` gate pair. The crossed-out gates, inputs, and outputs are removed by the reduction (`M**`)

logic. If the output inverters are to be eliminated, the following logic must be modified to accept inverted inputs. Thus it seems to be advantageous not to allow output inverters.

3.5 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 3.5.2 and 3.5.3) or errors at the outputs of other modules (Section 3.5.1).

3.5.1 `M**` Inputs Test

At the beginning of the test, the *inputs sub-test* is performed. The inputs sub-test is executed in the following steps: at first, the `MODIF B` is used to propagate the output of `MODIF A` thru the C-elements and the output of `M**` is computed by using the `MODIF A` output only. The output is then compared with the `M` output. The same is repeated for the `MODIF B` outputs. If one of the two results matches with the `M` output, the output of the second `MODIF` is erroneous. If no output matches with the `M` output, the test continues to the next sub-test.

3.5.2 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 3.4), one circuit input may be represented by two signals – two primary inputs.

The test of M** is inspired by ideas described at the beginning of this chapter and in [JB.1] and [JB.6] – 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.

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	$\downarrow$
4	in level i:				0
	1	1	1	1	$\uparrow$
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				$\uparrow$
8	Check if the circuit output is *all-one*				1
9	if (++i $\leq$ depth) then goto 3				1

Table 3.3: The test sequence of the *sub-test 1*

The sub-test 1 (Table 3.3) and the sub-test 3 (Table 3.5) were designed to detect stuck-open faults and the sub-test 2 (Table 3.4) 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.

step	C	T_U	T_C	T_D	O
1	set circuit primary inputs to 0				
2	1	1	1	1	$\uparrow$
3	0	0	0	0	$\downarrow$
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 3.4: The test sequence of the *sub-test 2*

step	C	T_U	T_C	T_D	O
1	0	0	0	0	$\downarrow$
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	$\uparrow$
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 3.5: The test sequence of the *sub-test 3*

Table 3.6 shows, which sub-test detects the stuck-open/stuck-on fault for a given transistor (see transistor labels in Figure 3.4).

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 although 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 although it should be closed (for sub-test 2) or

	tests covering faults	
transistor	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

Table 3.6: Sub-tests covering the faults

vice-versa (for sub-test 3). In this way, a possible fault syndrome is propagated up to the primary outputs.

3.5.3 Inverter Stuck-On Faults

Several tests in Table 3.6 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 – recall [3] or [32] and see details in Section 2.3.3.

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 [3]. 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.

3.5.4 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:

$$t_{tot} = t_{input} + t_1 + t_2 + t_3 \; [+ \; 2 \cdot t_{BICS}] \tag{3.1}$$

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:

$$t_{tot} \leq 2 \cdot t_e \; + (d \cdot t_e) + (t_e + d \cdot t_e) + (t_e + d \cdot t_e) \; [+ \; 2 \cdot t_{BICS}] \tag{3.2}$$

$$t_{tot} \leq (3d + 4) \cdot t_e \; [+ \; 2 \cdot t_{BICS}] \tag{3.3}$$

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.

Chapter 4

Preliminary Results

In this chapter, the evaluation of our method is presented. The combinational parts comparison is provided – comparison of `M` and `M**` in Section 4.2. Subsequently, the complete comparison of the TED and TMR systems is provided in Section 4.3. For a fair comparison we use a transistor-level technology independent model described in Section 4.1.

4.1 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 [41].

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 capacitances. 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* [41].

Based on the transistor-level properties, a model for every logic gate is created. The gate model has the following parameters: *size*; *precharge delay* expressing the time required to charge the 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 other 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 3.6). 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 3.3, 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.

By using this model, understanding basic gate properties is simple, but the model usability for comparison of delay of different circuits is limited. For gates with high fan-out, the modeled delay may be too pessimistic – in reality optimized cells would be used. Thus we provide the delay comparison for similar structures only. For different structures, we use the circuit depth.

gate	in capacitance	out 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

Table 4.1: Gate properties

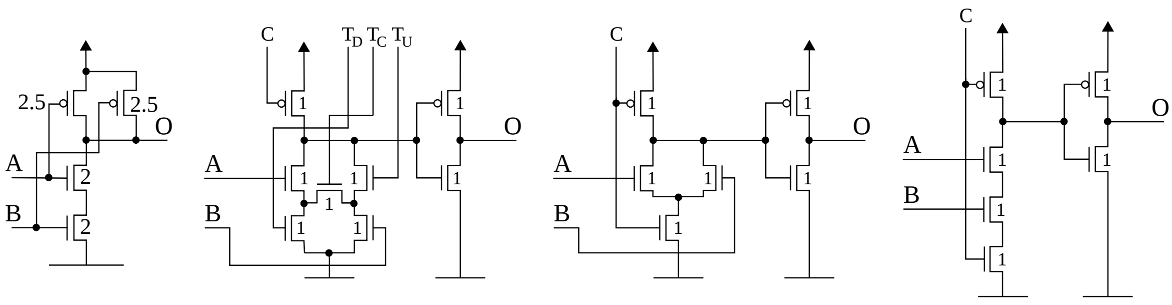


Figure 4.1: Transistor-level schematics and sizes of gate structures from Table 4.1

For static NAND we have chosen a symmetric conductivity, which is usual [41]. 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 4.1 and Figure 4.1.

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:

$$l = f \cdot 4.5 \tag{4.1}$$

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

$$l = f \cdot 1 \tag{4.2}$$

4.2 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* [42], *LGSynth'93* [28], *ISCAS'85* [9], *ISCAS'89* [8], and *IWLS 2005* [2].

The flow was as follows: the benchmark circuits were pre-processed by the *ABC* [31] 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 the *AIG* (And-Inverter Graph) format. The *AIG* was then used to construct the reduced dual-rail circuits as described in Section 3.3. Circuit characteristics were extracted using the gate model from Section 4.1.

The reduction step reduces the area of the original dual-rail circuit to 60% on average. The number of gates in the TED system after the reduction is less than or equal to the number of gates in the TMR system, as shown in Figure 4.2 (66% represents the number of gates in the duplex system and 100% represents the number of gates in the TMR system).

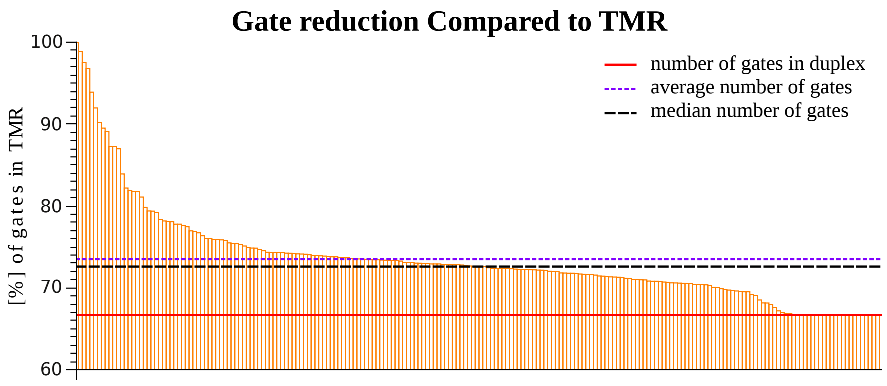


Figure 4.2: Number of gates in the combinational part of the time-extended duplex compared to the number of gates in the combinational part of NAND-based TMR. Each vertical line represents one circuit. The circuits are sorted in descending order by the area-overhead.

The quantitative results of the comparison based on all circuits from the set are shown in the Table 4.2. $\texttt{M}_{stat}$ represents the static `NAND` implementation and $\texttt{M}_{domino}$ the dynamic domino logic implementation of `M`. $\mathtt{M}_{stat,del}$ represents the scaled static implementation, where the area of `M` has been scaled by the delay ratio between the static and the proposed solution ($\mathtt{M}^{**}$) to provide fair comparison. If the number in Table 4.2 is lower than 100%, the proposed implementation ($\mathtt{M}^{**}$) is better than the given `M`.

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

Table 4.2: $\mathtt{M}^{**}$ and `M` comparison

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

4.3 Time-Extended Duplex and TMR System Comparison

The comparison of combinational logic provides just a partial information about proposed method usability. Thus, a comparison of the complete TED structure (Figure 3.3) and the dynamic domino logic-based TMR system (Figure 3.2) is provided. The TED is based on the domino logic variant ($\mathtt{M}^{**}$), thus comparison with domino logic-based TMR system only makes sense.

The area of `TMR SELECT` (see Figure 3.2) 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`):

$$if\ (\#inputs \approx \#outputs)\ and\ (\#outputs > 50):$$

$$\begin{aligned}
|\mathtt{TMR}| - 3\cdot|\mathtt{M}| &\approx \frac{|\mathtt{TED}| - |\mathtt{M}| - |\mathtt{M}^{**}|}{4} \\
12\cdot|\mathtt{TMR\ SELECT}| &\approx |\mathtt{TED}| - |\mathtt{M}| - |\mathtt{M}^{**}| \\
12\cdot|\mathtt{TMR\ SELECT}| &< 2\cdot|\mathtt{M}| - |\mathtt{M}^{**}| \\
12\cdot|\mathtt{TMR\ SELECT}| &< 0.66\cdot|\mathtt{M}|
\end{aligned}$$

It may be simplified to:

$$if\ (\#inputs \approx \#outputs)\ and\ (\#outputs > 50):$$

$$18 \cdot |\mathtt{TMR\ SELECT}| < |\mathtt{M}| \tag{4.3}$$

The area of `TMR SELECT` is proportional to the number of circuit outputs; the area of the additional logic in the TED is also proportional to the number of circuit outputs (and inputs). For a circuit where the expression (4.3) holds, the TED is likely better than the TMR system from the area perspective.

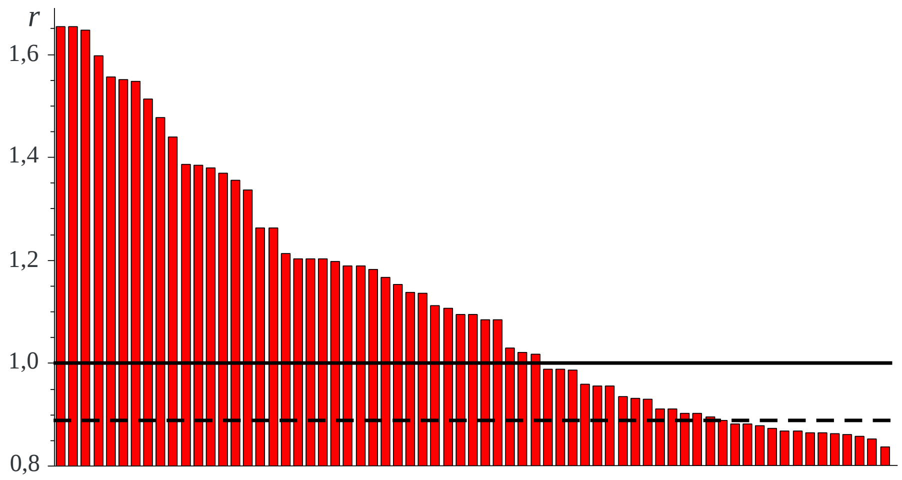


Figure 4.3: The comparison of the TED and TMR systems for 67 selected circuits. The following ratio: $r = \frac{|\mathtt{TED}|}{|\mathtt{TMR}|}$ is shown. The circuits are shown descending ordered by r

The heuristic based on the expression (4.3) has been verified by using 67 circuits selected from the original benchmark set. These circuits were selected to satisfy the expression (4.3) 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 4.3 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 of 67 circuits. The dashed line is the border from where the expression (4.3) has predicted, that the TED will be area-efficient compared to the TMR system.

Under the dashed line, there are 14 of 67 circuits – these are circuits with at least 11% improvement. Thus, according to the experimental evaluation, the heuristic based on the expression (4.3) is pessimistic.

4.4 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 it 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.

We characterized the set of circuits, where it is advantageous to use the TED system in place of the TMR system from the area point of view – expression (4.3). These are the circuits having relatively large combinational parts and a relatively small number of outputs. Still, there is the delay penalty compared to TMR.

The impact of additional delay may be reduced by using the pipeline and the speculative execution techniques. Speculative execution is often used with branch prediction to speed-up the modern processors [22]. 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. Compared to approaches described in Section 2.4.4, our approach handles both soft and permanent faults.

4.5 Summary

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 the TMR system.

We identified the class of circuits, where our approach is advantageous from the area point of view. According to the expression (4.3), the method is beneficial for relatively large combinational circuits.

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

Chapter 5

Conclusions

The digital systems dependability remains a hot topic of the digital design area. Although the error detection and/or correction in the separate time and the area domains is well studied (see Chapter 2), an efficient combination of the time and the area redundancy is still an open problem. The approaches described in Section 2.4 are limited to soft errors or their usage is problematical in general.

The presented approach combining time and area redundancy is general and removes some of the similar solutions disadvantages. The main disadvantage of our solution is that the set of circuits, where it can be used conveniently, is limited. Additionally some problems were disclosed during the previous work: the approach employing the offline test requires high certainty – every possible system error must correspond to a detectable fault – although the fault models were studied in the past, their accuracy still requires a deeper study; the relation between the circuit structure and the test length requires also further study.

These tasks are not exclusively related to the presented approach. Some of them are related to offline testing in general. The further research will include design-for-test methods, test generation methods and monotonic circuit properties.

5.1 Proposed Doctoral Thesis

Title of the thesis:
Error Recovery Techniques Based on Fault Detection and Localization

The author of the report suggests to present the following:

5.1.1 Adding Gates To Satisfy the Two-Vector-Testability

We propose to investigate the properties of the monotonic circuit transformation allowing 100% stuck-at-fault coverage, which is based on additional gates, as described in Section 3.1. The area overhead based on benchmark experiments is an important parameter.

The stuck-at-fault coverage will be 100%, but investigation of the fault coverage for more realistic fault-models is also important.

5.1.2 Dual-Rail to Singe-Rail Circuit Transformation Preserving Monotonicity

Although number of greedy heuristics have been applied to reduce the number of gates in the dual-rail circuit, no significant improvement has been observed. Randomized methods may bring better results. We started a cooperation with Ing. Zdeněk Vašíček, Ph.D. from Brno University of Technology. He applied the CGP genetic algorithm, but the current results are not too satisfactory, thus a further investigation is required.

5.1.3 Short-Duration Test Conditions

During our research, we developed the short-duration test for the dynamic logic with respect to the stuck-at-fault model and the stuck-open/stuck-on fault model respectively. Thus we described a set of conditions allowing the short-duration test in dynamic logic. It would be advantageous to deduce the *sufficient conditions* for the short-duration test with respect to different fault models. Deeper study of fault models properties is required.

The information about the fault presence may be *local* – the output may be faulty, or *global* – the circuit may be faulty. The previous research was concentrated to the fault localization because it makes the decisions more local and thus the voter and the additional delay is smaller. Because the methods allowing the voter delay masking are known (section 2.4.4), making the information globally need not to be extremely time consuming. This opens a new direction of the research: to sacrifice the locality to achieve the smallest possible test length.

5.1.4 Fault-Attack-Resistant Microarchitecture

I became the part of the team of solvers of the Project *Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features* [JB.11]. This project deals with fault-tolerant and attack resistant architectures.

The fault-attack-resistant architecture should not offer the information about fault presence by any *side-channel*, including power consumption or delay. It would be advantageous to propose a *microarchitecture*, maybe develop novel techniques at the transistor level, which will be able to mask faults with a minimal delay. The design should be done with respect to the number of side-channels. As we have gained experiences with the transistor level design and evaluation, this would be also a proper research task.

Bibliography

[1] 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*, pages 13.4.1–13.4.4, May 1992.

[2] C. Albrecht. IWLS 2005 Benchmarks. Technical report, June 2005.

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

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

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

[6] J. Balcárek, P. Fišer, and J. Schmidt. Simulation and SAT based ATPG for compressed test generation. In *Euromicro Conference on Digital System Design (DSD), 2013*, pages 445–452, Sept 2013.

[7] J. Borecký. *Dependable Systems Design Methods for FPGAs*. PhD thesis, the Faculty of Information Technology, Czech Technical University in Prague, 8 2015.

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

[9] 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)*, pages 677–692. IEEE Press, Piscataway, N.J., 1985.

[10] J.A. Brzozowski and K. Raahemifar. Testing C-elements is not elementary. In *Proceedings of Second Working Conference on Asynchronous Design Methodologies, 1995*, pages 150–159, May 1995.

[11] 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)*, pages 1–6, June 2015.

[12] Huan Chen and J. Marques-Silva. A two-variable model for SAT-based ATPG. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(12):1943–1956, Dec 2013.

[13] Liming Chen and A. Avizienis. N-version programming: A fault-tolerance approach to reliability of software operation. In *The Twenty-Fifth International Symposium on Fault-Tolerant Computing, Highlights from Twenty-Five Years, 1995*, pages 113–, Jun 1995.

[14] AEC Technical Committee. Fault simulation and test grading, 2007.

[15] D.R. Czajkowski, P.K. Samudrala, and M.P. Pagey. SEU mitigation for reconfigurable FPGAs. In *Aerospace Conference, 2006 IEEE*, pages 7 pp.–, 2006.

[16] Ilana David, Ran Ginosar, and Michael Yoeli. Self-timed is self-checking. *Journal of Electronic Testing*, 6(2):219–228, 1995.

[17] Al Davis and Steven M Nowick. An introduction to asynchronous circuit design. Technical report, Technical Report UUCS-97-013, September 1997.

[18] John M Emmert, Charles E Stroud, and James R Bailey. A new bridging fault model for more accurate fault behavior. In *Proc. of the Design Automation Conf*, pages 481–485, 2000.

[19] 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*, pages 96–99, Sept 2008.

[20] H. Fujiwara and T. Shimono. On the acceleration of test generation algorithms. *IEEE Transactions on Computers*, C-32(12):1137–1144, Dec 1983.

[21] J. Galiay, Y. Crouzet, and M. Vergniault. Physical versus logical fault models MOS LSI circuits: Impact on their testability. *IEEE Transactions on Computers*, C-29(6):527–531, June 1980.

[22] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach, Fifth Edition.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011.

[23] T. Koal, M. Scholzel, and H.T. 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*, pages 153–158, April 2014.

[24] Bernd Könemann. LFSR-coded test patterns for scan designs. In *Proc. IEE European Test Conference*, pages 237–242, 1991.

[25] Israel Koren and C. Mani Krishna. *Fault-Tolerant Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[26] S. Ma, I. Shaik, and R. S. Fetherston. A comparison of bridging fault simulation methods. In *Proceedings of International Test Conference, 1999.*, pages 587–595, 1999.

[27] Edward J. McCluskey and Chao-Wen Tseng. Stuck-fault tests vs. actual defects. In *Proceedings of the 2000 IEEE International Test Conference*, ITC '00, pages 336–, Washington, DC, USA, 2000. IEEE Computer Society.

[28] K. McElvain. IWLS'93 Benchmark Set: Version 4.0. Technical report, May 1993.

[29] K. C. Y. Mei. Bridging and stuck-at faults. *IEEE Transactions on Computers*, C-23(7):720–727, July 1974.

[30] S. D. Millman and E. J. McCluskey. Detecting bridging faults with stuck-at test sets. In *Proceedings the 1988 International Test Conference, New Frontiers in Testing*, pages 773–783, Sep 1988.

[31] A Mishchenko et al. ABC: A system for sequential synthesis and verification. `http://www.eecs.berkeley.edu/~alanmi/abc`, 2012.

[32] 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 *5th European Workshop on CMOS Variability (VARI), 2014*, pages 1–6, Sept 2014.

[33] D. K. Pradhan and J. J. Stiffler. Error-correcting codes and self-checking circuits. *Computer*, 13(3):27–37, March 1980.

[34] Marc Renaudin, P. Bastos, Rodrigo, G Sicard, and F. Kastensmidt. Asynchronous circuits as alternative for mitigation of long-duration transient faults in deep-submicron technologies. *Microelectronics Reliability*, 50(9–11):1241–1246, 09 2010. 21st European Symposium on the Reliability of Electron Devices, Failure Physics and Analysis.

[35] Jian Ruan, Zhiying Wang, Kui Dai, and Yong Li. Design and test of self-checking asynchronous control circuit. In *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, volume 4644 of *Lecture Notes in Computer Science*, pages 320–329. Springer Berlin Heidelberg, 2007.

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

[37] Jens Sparsø and Steve Furber. *Principles of Asynchronous Circuit Design: A Systems Perspective*. Kluwer Academic Publishers, Boston, 1st edition, 2001.

[38] Y. Tamir and M. Tremblay. High-performance fault-tolerant VLSI systems using micro rollback. *IEEE Transactions on Computers*, 39(4):548–554, Apr 1990.

[39] Laung-Terng Wang, Cheng-Wen Wu, and Xiaoqing Wen. *VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

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

[41] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[42] S. Yang. Logic synthesis and optimization benchmarks user guide: Version 3.0. Technical report, MCNC Technical Report, January 1991.

Publications of the Author

Reviewed Relevant Publications

[JB.1] J. Bělohoubek, P. Fišer and J. Schmidt *Novel C-Element Based Error Detection and Correction Method Combining Time and Area Redundancy.* Euromicro Conference on Digital System Design (DSD), 2015, Funchal, Madeira – Portugal, 2015.

Other Relevant Publications

[JB.2] J. Bělohoubek *Novel gate design method for short-duration test.* POSTER 2015, Prague, Czech Republic, 2015.

[JB.3] J. Bělohoubek *Novel Error Detection and Correction Method Combining Time and Area Redundancy.* Počítačové architektury a diagnostika 2015, Zlín, Czech Republic, 2015.

Other Publications

[JB.4] J. Bělohoubek *Smart re-use of hardware peripherals for better software UART.* The 3rd Prague Embedded Systems Workshop, Roztoky u Prahy, Czech Republic, 2015.

Rejected Relevant Publications

[JB.5] J. Bělohoubek, P. Fišer and J. Schmidt *Design for Short-Duration Test Based on Dynamic Logic.* 22nd IEEE International Symposium on Asynchronous Circuits and Systems, Porto Alegre, Brazil, 2016.

Submitted Relevant Publications

[JB.6] J. Bělohoubek, P. Fišer and J. Schmidt *Error Correction Method Based On The Short-Duration Offline Test.* Euromicro Conference on Digital System Design (DSD), 2016, Limassol, Cyprus, 2016.

Other Results

[JB.7] J. Bělohoubek and J. Schmidt *Fully asynchronous QDI implementation of DES in FPGA*. Cryptographic architectures embedded in reconfigurable devices, Annency, France, 2014 (unpublished lecture).

Finished Projects

[JB.8] member of team of solvers SGS14/105/OHK3/1T/18, Czech Technical University in Prague

[JB.9] member of team of solvers SGS15/119/OHK3/1T/18, Czech Technical University in Prague

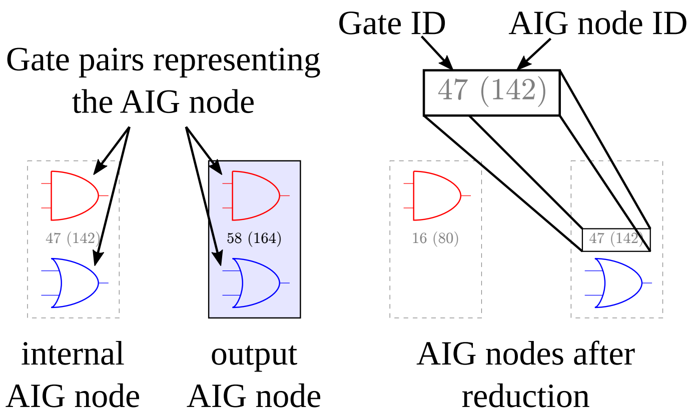
Current Projects

[JB.10] member of team of solvers SGS16/121/OHK3/1T/18, Czech Technical University in Prague

[JB.11] member of team of solvers GA16-05179S of the Czech Grant Agency: *Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features (2016 – 2018)*

Appendix A

Dual-Rail Logic Reduction Example: the Cordic Benchmark



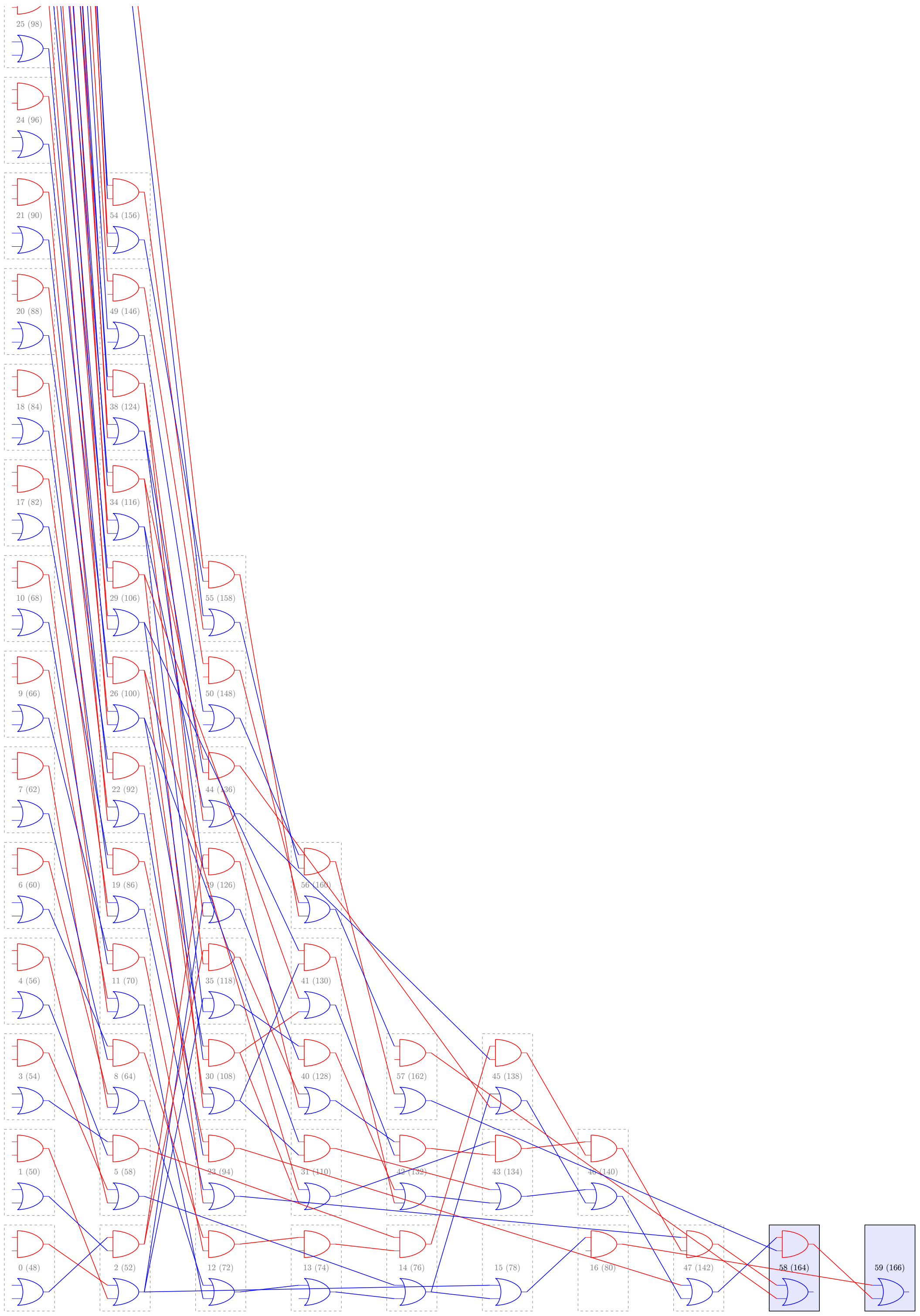


Figure A.1: Part of the greedy heuristic dual-rail logic reduction output for the *cordic* circuit from the benchmark set – output inverters are not allowed; reduction is unsuccessful

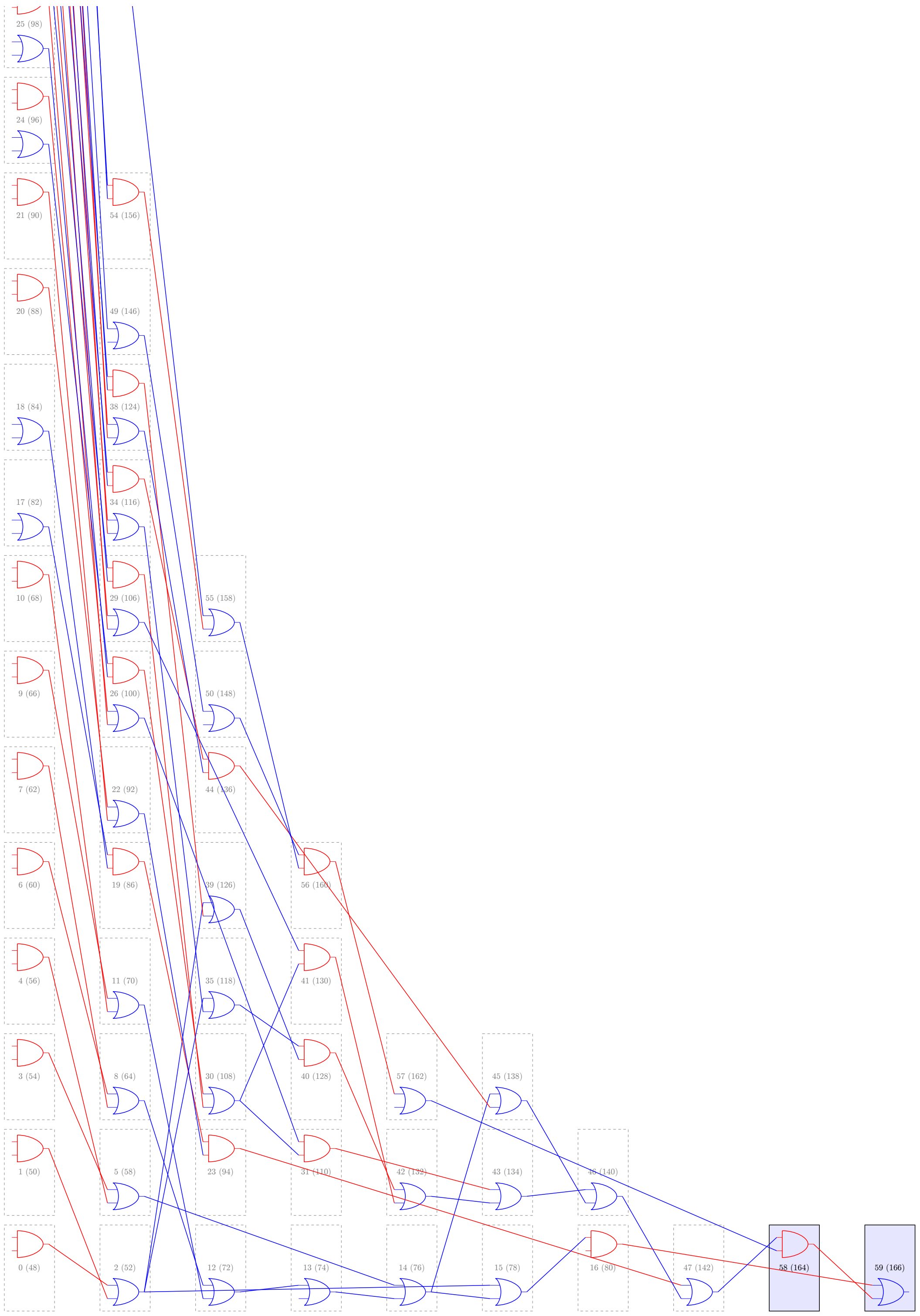


Figure A.2: Part of the greedy heuristic dual-rail logic reduction output for the *cordic* circuit from the benchmark set – output inverters are allowed; reduction is successful