BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

METHODOLOGY FOR FAULT TOLERANT SYSTEMS DESIGN INTO LIMITED IMPLEMENTATION AREA IN FPGA

METODIKA NÁVRHU SYSTÉMŮ ODOLNÝCH PROTI PORUCHÁM DO OMEZENÉHO IMPLEMENTAČNÍHO PROSTORU NA BÁZI FPGA

PHD THESIS AUTOREFERATE
AUTOREFERÁT DISERTAČNÍ PRÁCE

AUTHOR **Ing. LUKÁŠ MIČULKA**
AUTOR PRÁCE

SUPERVISOR **Doc. Ing. ZDENĚK KOTÁSEK, CSc.**
ŠKOLITEL

BRNO 2017

Abstract

The work presents a methodology of fault tolerant system design into an FPGA with the ability of the transient fault and the permanent fault mitigation. The transient fault mitigation is done by the partial dynamic reconfiguration. The mitigation of a certain number of permanent faults is based on using a specific fault tolerant architecture occupying less resources than the previosly used one and excluding the faulty part of the FPGA from further use. This inovative technique is based on the precompiled configurations stored in an external memory. To reduce the required space for a partial bitstream the relocation technique is used.

Keywords

fault tolerant system design, partial reconfiguration, design methodology, FPGA.

Reference

MIČULKA, Lukáš. *Methodology for Fault Tolerant Systems Design into Limited Implementation Area in FPGA*. Brno, 2017. PhD thesis autoreferate. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. Ing. Zdeněk Kotásek, CSc.

Contents

1. Introduction

In last decades, the huge progress in manufacturing electronic devices mainly stands on shrinking its parts such as chips and transistors. The scaling of transistors to small sizes provides high performance due to higher densities and low power and lower costs per unit but has also very strong drawbacks. These small devices are also very fragile on overstress and other environmental influences during operational lifetime. Additionally, small changes inside fabric caused by these factors can lead to large impact on device performance. It also brings bigger susceptibility to transient upsets. Small nodes use less charge to hold state or data and can be easily altered and upset by noise from outside environment such as radiation. Thus, the dependability of the system becomes the key indicator. System dependability expresses the ability of system to produce the outputs that can justifiably be trusted. To increase the dependability of system, several mechanism can be adopted such as Fault Tolerant (FT) system design which enables a system to continue its intended operation when some part of the system fails.

Nowadays, the FPGA technology became very popular and frequently used. It provides high logic density and possibility to easily upgrade the implemented designs. Another benefit of FPGA design in comparison with custom chips is their relatively short design cycle supported by the possibility of using existing low cost design tools. These benefits together result in low non-recurring engineering costs (NRE) for FPGA design. On the other side, their drawback is their vulnerability to radiation effects [21]. This mainly concerns SRAM-based FPGAs which are becoming increasingly popular for many applications due to their high-throughput capabilities and relatively low cost. The use of fault tolerant system design can be the solution to overcome their higher rate of fault occurrence.

The ability of FPGA to be configured many times also brings new possibilities from the perspective of system fault tolerance. When the system in FPGA is affected by a fault, the reconfiguration can be used to overcome its effects. Partial Dynamic Reconfiguration (PDR) capable to reconfigure only some parts of implemented system while the others can run without interruption and also to change their layout and connections in FPGA can be used to implement the new advanced fault localization and mitigation methods. This flexibility allows the use of the same FPGA for multiple missions without the need of replacement.

1.1 Fault tolerant systems

A fault tolerant system is that one which can perform its function and produce correct outputs even when it is affected by a hardware or software fault. In [7], three condition to state that system is fault tolerant are considered:

- The system computation for given dataset was not interrupted when a fault occured and complete batch of input data was processed.
- The outputs produced by the system are correct.
- The length of computation did not exceed the predefined time limit.

In FT systems, the key goal is to prevent the errors from propagating to observable outputs of computation process. This can be achieved by the adding space, time, information, software or other type of redundancy to original system. In this work, the main focus is put on approaches based on hardware redundancy such as unit replication. One of the most known FT architectures is the system with 3 identical modules and majority voter referred to as Triple Modular Redundancy (TMR) system or the duplex system with 2 modules and checker units. In the systems with replicated modules, the voter or compare logic is the most vulnerable part. Thus, they can be implemented into more resistable fabric or also replicated.

1.2 Fault detection and localization techniques

For fault detection, the capability of systems to mitigate faults which appear during their operation is an important feature.

The fault detection always requires some kind of redundancy. Many techniques based on information redundancy such as parity code, checksum or CRC can be used. When the information redundancy is increased the Hamming distance between the data word and encoded word can be counted and can be used to find the detection parameters of used method such as maximum number of errors which can be detected or repaired. The codes with the ability to detect errors are referred to as Error Detection Codes (EDCs). If they can also repair the error, they are referred to as Error Correction Codes (ECCs).

Fault detection and localization methods can be also based on space redundancy. The simplest form is n-modular redundancy with n replicated modules connected in parallel (e.g. TMR). Their outputs are compared and any difference indicates the presence of fault. The redundancy

of this system is always more then $(n-1)*100\%$. As its benefit, the ability to detect every distinct error can be seen. The problem can arise only when the same error will be produced by all replicated units.

Alternative to module replication is the use of checker unit. This unit is placed in parallel to operational unit and it is computing some function with the same input. Its output can be continuously used to check the correctness of the output of operational unit. When the outputs of the units are not equal, it means there is a fault in the system which caused an error to occur. The drawback of this approach is that we cannot distinguish whether the error is produced by operational unit or its concurrent unit.

Another option can be the use of the unit implementing the inversion of function. This unit can be used to compute the input value from the output of checked unit. This will avoid the possibility of the same fault in both units but the inversion function does not always exist. It also adds some latency to computation.

Off-line fault detection is a widely used technique which is checking the fault occurrence while the application in the FPGA is not running. This method can be based on external testing equipment outside FPGA or the test equipment can be configured into FPGA. The second approach is known as Built-In Self-Test (BIST) which typically uses design consisted of three block - test pattern generator, block under test and output response analyzer which are periodically switched by reconfiguration. Thus, in several steps the entire FPGA can be fully tested. The main drawback is its limitation that it can only detect faults during the test mode when the FPGA is not operating. Thus, some timing-dependent faults or similar may not be detected.

In bitstream readback approach, the controller (typically external) reads the actual FPGA configuration memory contents as well as the contents of flip-flops in CLBs in the form of configuration bitstream. The readback process can be considered as the inversion of FPGA configuration. Bitstream readback is available in two modes. In readback verify mode, the controller reads the configuration of memory cells and compares it with the original bitstream. This mode is mainly used to verify the success of previously done configuration. Readback capture mode also reads configuration memory cells data but in addition to that it also acquires the current states of all internal flip-flops inside CLBs and the state of IOBs. With these gained data from FPGA and the knowledge of data which are expected to be in the configuration memory of FPGA and

other resources in the moment of readback, the diagnosis algorithms can be used to detect and localize faults in FPGA [18] [19].

The techniques based on roving STAR approach are capable to detect and localize faults in FPGA. The approach is based on the dividing the array into tiles with the same number of resources and their structure. Some of these tiles implement the function of system and in some of them, the BIST is implemented. These tiles performing the fault detection and localization are referred to as Self-Testing AReas (STAR). While one STAR is tested off-line the remaining blocks of system which are not utilizing resources from actual STAR continue in run and the aplication in FPGA is not interrupted. Testing is focused on logic blocks and connecting wires. When the testing is finished the partial reconfiguration of FPGA is used to change the layout of design and another tiles previously used to implement the function are configured as BIST and tested. The fault coverage of this approach can be 100% because every tile of FPGA is tested. Hardware overhead of this approach is formed by tiles needed for STARs and the reconfiguration controller logic which is used for roving the STAR through the FPGA.

All presented fault detection methods have both positive and negative features. Table 1.1 is showing the results of detection methods when they are evaluated by several criteria.

Detection method	**Granularity, detect. speed**	**Fault coverage**	**Space overhead**	**Performance overhead**
Unit replication	**coarse fast**	**good** all error occurences	**large** resources for *n-1* modules +S voter needed	**small** voter latency
CED	**coarse fast**	**medium** can be impractical for some functional units	**medium** trade-off with coverage	**small** just latency of checker
Off-line methods	**fine slow**	**very good** can detect also faults not manifested by error	**small** testing controller	**small** just start-up delay
Bitstream readback	**fine slow**	**very good** can detect also faults not manifested by error	**small** readback and testing controller	**small** just start-up delay
Roving star	**fine medium**	**very good** can detect also faults not manifested by error	**medium** resources for STARs + testing controller	**large** switching latency, long critical paths

Table 1.1: The comparison of fault detection methods

1.3 Transient fault mitigation

The use of FPGAs in harsh conditions has significantly risen the number of transient faults mainly caused by ionization radiation. These faults can be mitigated but this requires additional logic.

The susceptibility to these kinds of faults can be lowerred by the special fabrication design to produce radiation-hardened FPGAs. This radiation hardened design is based on protecting the configuration cells at transistor or silicon level. As FPGAs become more and more complex with large number of resources and processing capabilities, the radiation-hardenning becomes excessively expensive in comparison with non-protected ones. Radiation hardened FPGA has slower operating frequency and increased power consumption when compared with its commercial off-the-shelf FPGA counterpart [15].

When a transient fault occurs in FPGA it can be repaired by reconfiguration of affected part of configuration memory. This can be done by complete (static) reconfiguration of FPGA or by PDR of affected Partial Reconfigurable Module (PRM). Static reconfiguration causes the stopping of running design in FPGA and possible loss of current status information of implemented modules. Due to complete reconfiguration of FPGA, this technique does not require the localization of the affected part of FPGA. Nowadays, in most cases the application running in FPGA cannot be stopped during the recovery process and therefore techniques based on PDR are preferred.

Configuration bitstream scrubbing was introduced to correct configuration memory after SEU occurences. This method is based on periodical reconfiguration of PRM by correct Partial Reconfiguration Bitstream (PRB) while the FPGA is in operation. There are two common configuration scrubbing strategies. In blind scrubbing, the periodical reconfiguration of PRM by golden copy of designated partial configuration bitstream is done without knowledge which module is faulty. Another scrubbing methods use bitstream readback to detect if PRM is faulty and must be reconfigured. The reconfiguration can be done by golden copy of bitstream or by the read and corrected one. The scrubbing period should be stated according to failure rate of system. The main drawback of configuration scrubbing method is the need of continual use of the configuration port but techniques such as [5] to overcome this issue exist.

Methods based on PDR are dependent on some kind of detection and localization technique implemented in design which in case of fault detection triggers the process of recovery. Unlike in the configuration scrub-

bing, this process is started only in case of fault detection event. The process of recovery is shown in Figure 1.1. The detection and localization of faulty module is typically done by the design itself implementing techniques such as CED or unit replication with checker units. The error signals are supplied by some kind of PDR controller which will trigger the reconfiguration process using appropriate configuration bitstream downloaded from configuration bitstream storage. PDR controller is typically implemented by some softcore processor in the same FPGA where it performs the PDR or in some external reliable fabric. Many techniques based on this approach were presented [1] [2] [6].

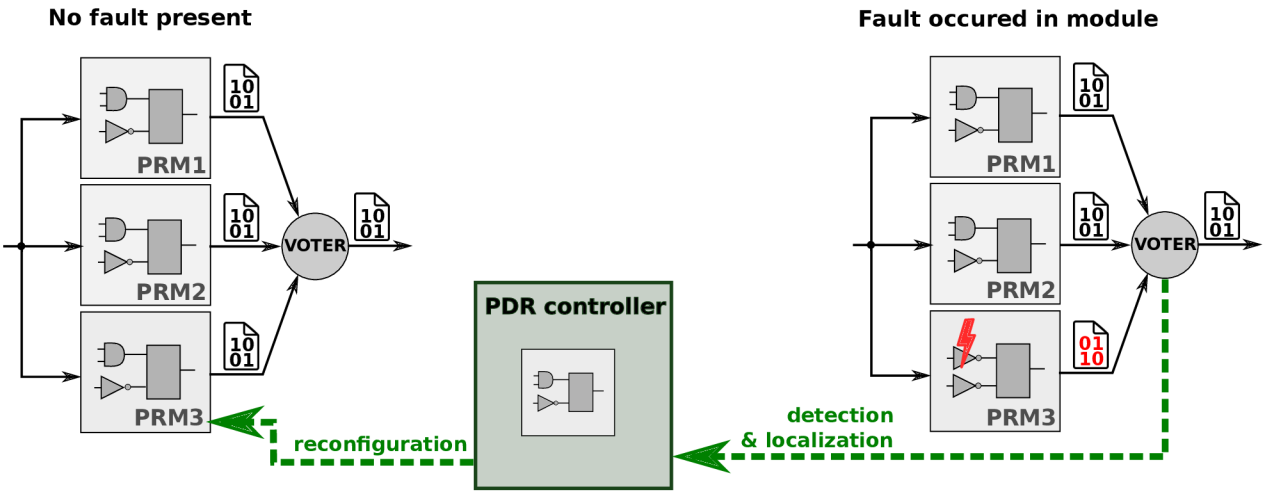


Figure 1.1: Using PDR to recover system after SEU occurence

1.4 Techniques for system recovery after permanent fault occurence

In this work, as the permanent fault is considered, each fault causes a damage of FPGA resource in that way that it cannot be used in FPGA design anymore. This happens mostly by damaging or during the wear-out phase of FPGA or by the impact of harsh environment on FPGA.

One possible approach is to use the FPGA fabric which is designed and manufactured with spare resources which can be utilized in case of fatal fault occurence in currently used set of resources. This hardware level approach can be easily used in array based resources (i.e. CLBs) by using multiplexers or other switching logic at the ends of lines of cells. This allows the remapping of a row or a column with damaged component into some spare row or column [4]. For hardenning the interconnection of

CLBs and other hard blocks in FPGA, the fine-grain redundancy in the interconnect blocks can be introduced [22]. The benefit of this approach for permanent fault recovery is its transparentness to the configuration. On the other hand, including the spare hardware resources to device is expensive and it is firmly limited by the number of spare rows or columns.

Another approach is based on the recovery on the configuration level. When the permanent fault occurs, the modification of current design configuration by incremental re-mapping and re-routing is performed to exclude the affected resources when a fault occurs. This approach can in theory utilize all spare resources which are currently not used by implemented application for logic or routing affected by faults. The drawback of this method is the need of adaption of FPGA mapping, placement and routing tools to operate autonomously with considering existing faults in implementation area. The incremental change of design requires not negligible time for processing, it can increase power consumption and area overhead. Many techniques based on this approach were presented [14] [16] [17].

As the opposite to online design modification, some other approaches (see [13] [23]) are trying to prepare the possible solutions before the fault appears, in the design phase. Then, the implementation space of FPGA is typically divided into several tiles and the desired design is splitted into modules which are configured into different tiles leaving one or more tiles unused. The configurations with these alternative implementations are precompiled and created partial configuration bitstreams are stored in some type of memory. When a fault is detected and localized in some tile, the reconfiguration of the entire design is performed with this precompiled configuration which does not utilize the resources from this tile. Since each configuration of the design contains the implementation of the same function and the interface between the entire reconfigurable area and the rest of design is fixed and the same in all cases, all partial bitstreams are interchangeable and can be configured to this partial reconfigurable region. With this approach, a fault in logic block and in local interconnections can be handled. This technique minimizes the recovery time since the process consists of alternative configuration selection and PDR with its precompiled bitstream. The drawbacks of this approach can be seen in its poor area efficiency and complicated mitigation of mutliple faults but the main one is the requirement of external storage for precompiled partial configuration bitstreams. This can be reduced by some techniques such as bitstream compression but there is always a trade-off with increased time and complexity of recovery process.

The ability of modern FPGAs to be reconfigurated dynamically can be used by evolutionary methods such as [3]. They can recover the system correct operation through evolution when faults occur. These methods offer a large degree of flexibility in the number and distribution of faults which can be mitigated. There is no need to preciselly localize the fault. Evolutionary methods attempt to facilitate repair through the reuse of damaged resources. The fitness function of implemented Genetic Algorithm (GA) is able to internally evaluate the residual functionality of the design in FPGA and assess the fitness value. This value is used for the upcoming selection phase. With this approach, very big flexibility can be achieved and all remaining non-faulty resources can be utilized by the new design. The drawback of this method is the complexity and the flexibility what can result in very time-demanding search of satisfactory design with unpredictable duration and its result. The logic for evolutionary algorithms can cause unnegligble area overhead.

Although many different approaches to system recovery after permanent fault occurence exist, none of them is considered as universally applicable. Table 1.2 is showing the comparison of presented recovery methods from different aspects.

Recovery method	**Recovery speed**	**Resource overhead**	**Performance overhead**	**Flexibility of recovery**
Hardware level	**very fast** just switching lines in hardware	**low** spare physical resources needed, no requirements for impl. space	**low** no design change	**low** entire row/column excluded
Alternative configurations	**medium** configuration selection and reconfig. delay	**high** reconfig. controller and config. selector & storage for for configurations	**low** alternative configuration can be optimized	**medium** trade-off with the number of configurations
Incremental remapping & rerouting	**poor** time demanding remapping and rerouting	**high** design impl. and reconfig. controller	**medium** trade-off with impl. controller complexity	**high** non-faulty resources can be effectively utilized
Evolutionary algorithms	**poor** may take long time to evolve	**high** after bitstream is read, parsed and analysed	**low** can be optimized by setting fitness function	**high** non-faulty resources can be effectively utilized

Table 1.2: The comparison of permanent fault recovery methods

2. Motivation and goals of the research

2.1 Motivation

The scaling of electronic devices and still less robustness of components bring the strong need for more complex securing against the occurence of faults. The use of electronic devices in new rough and noisy environment is also another source of problems. For example, in the aerospace industry there are requirements on electronic devices for their resilience against radiation and on hardenning them against negative effects of material aging during long term missions.

In recent decades, new possibilities and new challenges in the area of system design appeared. Programmable electronic devices such as CPLDs and FPGAs allowed rapid prototyping and started the era of reconfigurable computing. Faulty design can be easily fixed after the first deployment and the same hardware can be also used to perform various tasks during the lifetime where some of these can be unforeseen. The FPGAs came up with new possibilities in the field of fault tolerant hardware design. The dynamic reconfiguration can be now used for changing the mapping and routing inside FPGA in order to mitigate the faults which have occurred. The new challenges with fault tolerance in FPGAs are connected with their configuration saving. Very often the FPGAs which have configuration stored in SRAM memory are used [9]. They are popular because of their lower price and easy use they offer. Higher susceptibity to SEU faults in comparison with other FPGA types can be seen as their drawback.

Many approaches for making digital systems more dependable were presented. Fault tolerant system design offers the possibility to overcome the impact of fault occurence while the use of detection and localization methods together with fault mitigation based on PDR can offer to restore the fully operational state of system. This can be done autonomously without the need of user intervention and without stopping system operation. Nowadays, the utilization of FPGAs is not only in rapid prototyping but they are used frequently also in long term missions. Thus, the study of system dependability has to focus also on permanent faults which occur more likely with the increasing age of FPGA. Many techniques for mitigation of SEU effects in FPGA and also several mitigation techniques

for permanent damage of resources in FPGA are available. None of them is universally applicable due to their high demands on memory (e.g. precompiled alternative configurations), time-demanding fault recovery (e.g. evolutionary algorithms), area overhead (e.g. incremental change of design), etc. Thus, it makes sense to focus on optimization of these techniques and creating such methodology which will describe how to create design with effective fault recovery ability.

2.2 Goals of the research

The effort to develop a methodology for fault tolerant systems design was driven by the goal to satisfy the following aspects.

- The localization of the FPGA part (PRM) affected by fault.
- The determination of the fault type and its classification according to considered fault model.
- The driving of repair proccess to return the system
  - to the exactly same state as there was before - in case of transient fault,
  - to the state when the functions of system are producing correct outputs - in case of permanent fault.
- Keeping the design running during the reconfiguration process if it is possible.
- Enabling the support for synchronization process after reconfiguration is completed.
- The effort to shrink the number of the FPGA resources needed as hardware overhead because of the system design according to proposed methology.

The goal of this thesis is to combine the existing well known techniques together with new approaches. As an example, the CED technique together with online checkers can be used not only to ensure the fault tolerance in system but also to localize the module affected by a fault in FPGA if it is possible. This localization information will point at specific reconfigurable module of FPGA which is faulty. Then some reconfiguration controller will use this information to process fault mitigation in

it. The PDR together with FT design will be used to ensure the correct operation even during the fault recovery process.

The goals of the research can be summarized in the following way:

1. To propose the methodology for the FT design of digital system into FPGA with the ability to recover after transient and permanent fault occurence which satisfies these conditions:

   - The designed architecture of system is operating in limited implementation area which means that it can only utilize the resources from the area of FPGA which was designated for the system at the begining of its lifetime.
   - The occured transient fault in one system module is mitigated while the rest of modules in FPGA are not affected by it.
   - If the architecture of implemented system has to be modified to recover after permanent fault occurence, the new one has to keep producing correct outputs and it should remain fault tolerant if it is possible.

2. To design the reconfiguration controller which will control the mitigation process in FPGA after fault occurence done by PDR. It supplies the information about the detection and localization of fault, it determines its type and controls the reconfiguration process. Alternatively, it can also trigger the sychronization process when it is needed.

3. To create test platform which will enable the evaluation of methods and procedures described by the proposed methodology. For the FT architectures designed by means of methodology principles, the ability to survive will be tested by fault injection.

The proposed methodology covering these points is described in the following chapter.

3. Methodology for FT system design into limited implementation area in FPGA

In this chapter the principles of the proposed methodology which aims at securing system by implementing its parts as fault tolerant systems into the limited implementation area in FPGA are described.

The limited implementation area from the perspective of this research means the set of FPGA resources assigned for implementation of some system parts which are important from the dependability point of view. This implementation area is specified during the design phase of system implementation and it cannot be modified during system lifetime. This assessment limits the fault mitigation technique during permanent fault recovery process.

3.1 Methodology basic principles

The proposed methodology defines the process of securing digital system designed and implemented in FPGA. In other words, it can be understood as the recipe how to redesign the given architecture of a system in FPGA and how to prepare the system for recovery after fault attack and thus make its lifetime longer. Such methodologies have their justification e.g. in long term missions where the implementation area becomes smaller after every permanent fault which occurs in the design.

The detection and localization process is based on the comparison of replicated functional units in FT architectures and on other CED techniques. No specific methods are intended. The mitigation technique requires the localization on the PRM level. When the faulty PRM is localized, it must be determined to which type of fault defined by fault model this particular fault belongs. Mitigation process is different for both types of fault - transient and permanent. Both of them are driven by developed controller unit - Generic Partial Dynamic Reconfiguration Controller (GPDRC). This unit has a crucial role in the system because it is responsible for the task of fault mitigation and is able to control the reconfiguration performed through ICAP interface (see Section 3.3).

In the developed methodology, the design is protected by means of FT architecture to guarantee the resilience against both independently occurring transient faults and given number of permanent faults which affect the FT system correct operation. The methodology suggests to divide the implementation into certain number of PRMs. This set of PRMs put together is called *a configuration* in the following text. Each unit of FT system is placed in one PRM and it is assigned to Partial Reconfiguration Region (PRR). PRMs are designed in a uniform way which means that relative position of all sources, connections and proxy logic inside it is identical for the particular type of the unit. This is required for relocation process.

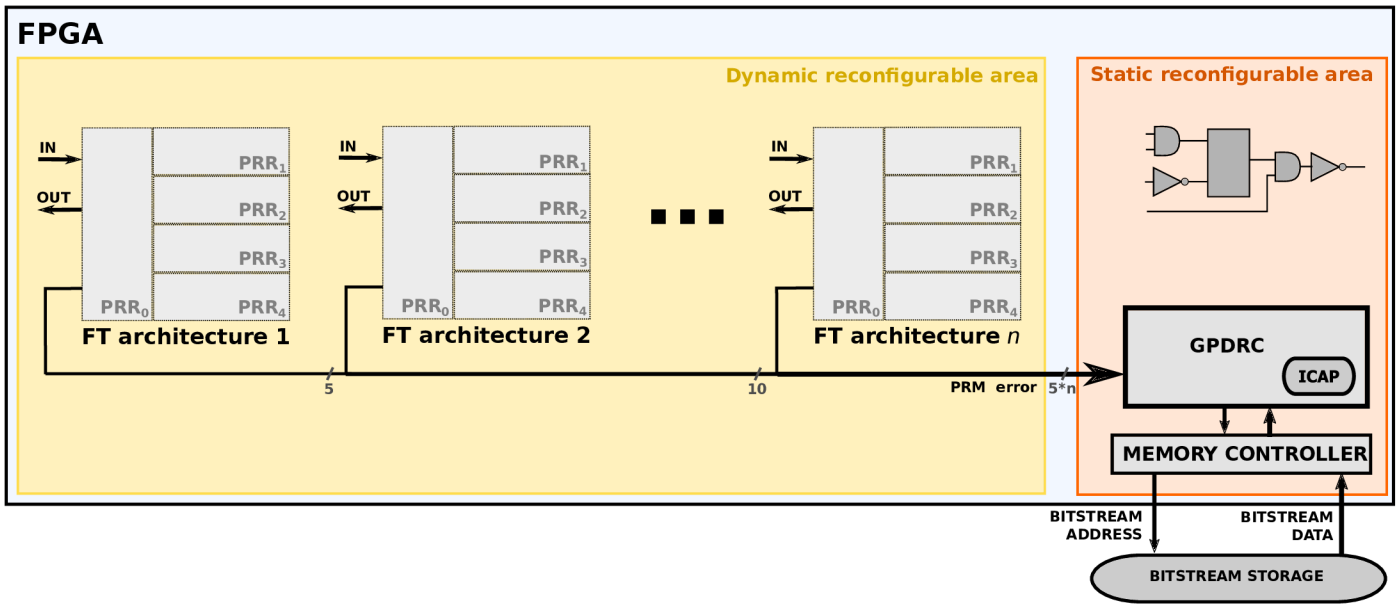


Figure 3.1: The main structure of the proposed methodology

In Figure 3.1, an example of the complete FT system design in FPGA based on the principles of methodology is shown. It consists of dynamic part in which FT architectures are placed and static part which contains GPDRC. The GPDRC utilizes the information about detection and localization of faults from the CED logic units of FT architectures. The set of error signals from PRMs (assigned in PRR1 - PRR4) are the inputs to GPDRC. Splitting FT architecture into several PRMs gives the possibility to exclude from the implementation one or several PRMs when they are affected by permanent fault. The interconnection signals between modules and the connections between the particular module and the rest of FPGA pass through single PRM assigned to PRR0 which is neighbouring with all other PRRs. The other 4 PRRs can be assigned by PRMs

of different units of the selected FT architecture. The number of these uniformly sized and structured PRRs can vary.

To illustrate the application of methodology for securing a real system, several FT architectures were proposed which can be used in degradation strategy of some system unit (see Figure 3.2). The first, most robust FT architecture, is TMR architecture with doubled voter which enables the detection of errors also in the voter. The next TMR architecture uses just simple non-protected voter unit. The last architecture is based on Duplex system with a comparator. This architecture is not fault tolerant since there is no possibility to distinguish which output of two replicated units is incorrect. But this system can run correctly until the first fault occurs and then it is detected by compare unit.

Each replicated functional unit is implemented in single PRM referred to as PRM_FU, complex voter unit is implemented in its own PRM referred to as PRM_VOTER and the routing between replicated units and the FT architecture external interface is constrained into PRM referred to as PRM_ROUTE.

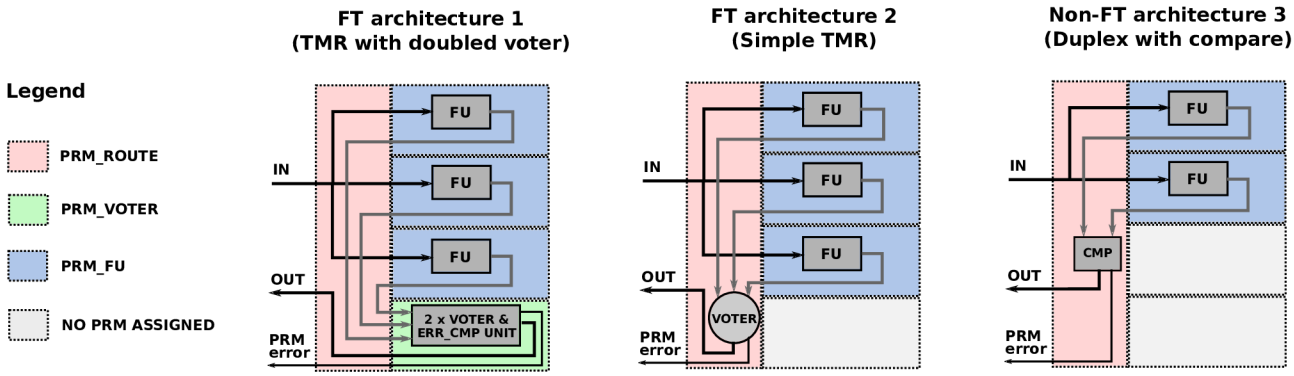


Figure 3.2: The assignment of PRRs by different PRMs

3.2 Generations of alternative FT architecture configurations

The methodology is based on the existence of precompiled configurations of an FT design which are applied when a permanent fault occurs. These configurations are divided into several generations. Configurations from one generation contain the same FT architecture but with different PRM placement. The enumeration of all possible generations for such FT architectures is shown in Figure 3.3.

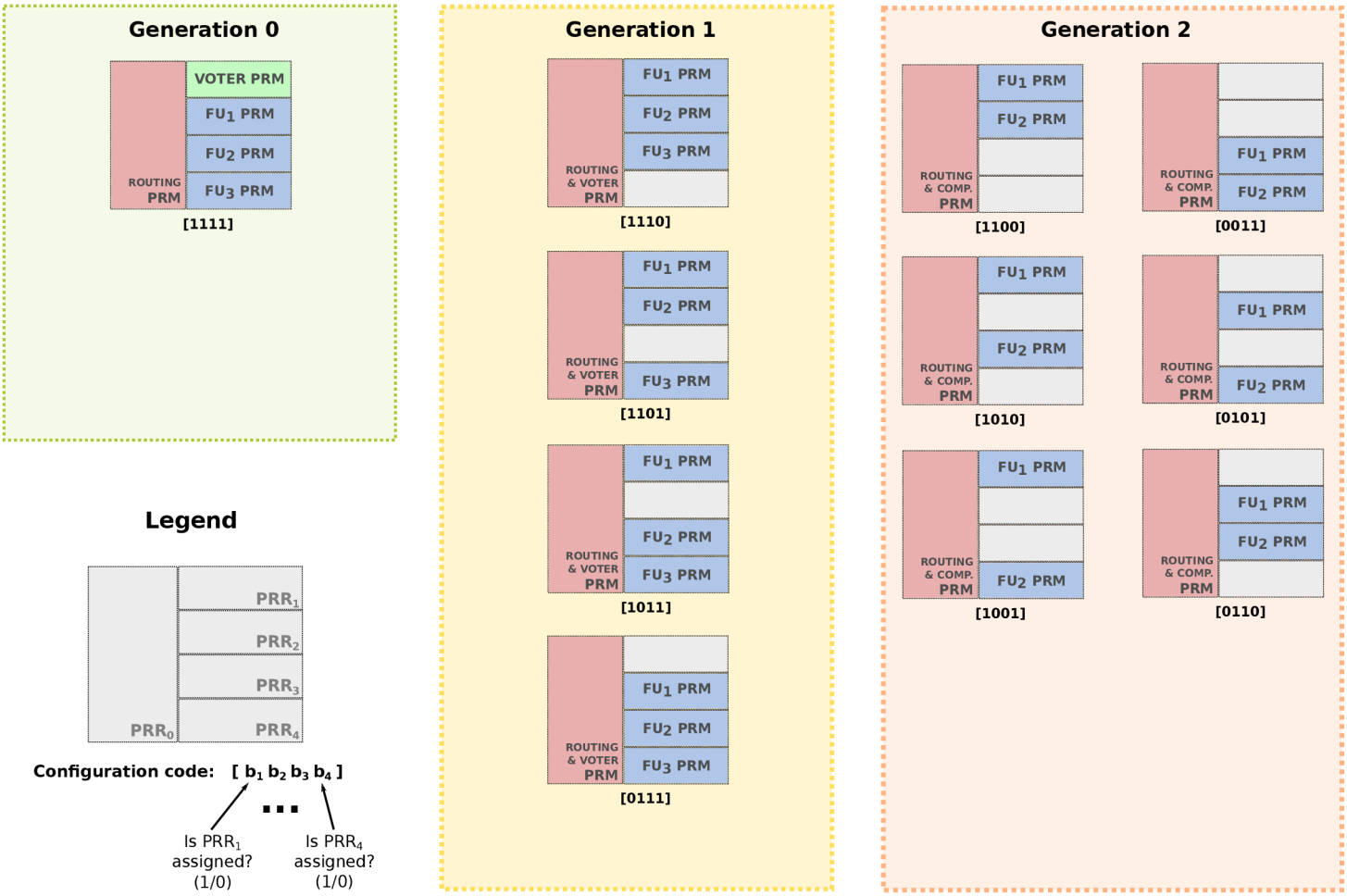


Figure 3.3: The generations of FT architectures and their alternative configurations

The number of unused PRMs (PRMs excluded from use) in configurations of each generation reflects the generation number. The code of configuration is assembled from flags indicating if the corresponding PRR is assigned by PRM (see legend in Figure 3.3). The configuration with code 1111 from generation 0 represents the starting configuration for this system part. After the first permanent fault is detected and affected PRM is localized, the new configuration excluding the faulty PRM from the next generation is chosen to be used for system implementation. This principle is applied again when a new fault affects another PRM. The number of possible variants of configurations is rising with the number of PRMs affected by fault. To reduce the memory requirements for the configuration, bitstream relocation method is used to avoid the existence of several copies of PRM containing the same type of unit. Only one copy of PRM bitstream for each type of PRM except PRM_ROUTE is

needed. Only the bitstream designated as PRM_ROUTE is stored for each configuration in the memory.

Due to the specifics of design and implementation flow adopted by Xilinx tools, the generated partial configuration bistream of PRM cannot be assigned to different PRR than it was originally designated to. One PRB has to be generated for each PRR where the PRM will be configured. Thus, if there is a need to apply N PRMs of different types to any of M PRRs, $N * M$ PRBs have to be produced and stored in external memory for run-time partial reconfiguration. With the adoption of bitstream relocation technique, the number of generated PRBs is reduced to N. These PRBs can be used then for reconfiguration of all PRRs satisfying the conditions for the application of relocation technique. These conditions are applied in the design phase and the implementation phase. One of the main limitations of this technique is the need to have all PRRs with identical FPGA resources. In common, this technique always starts by generating the PRBs for all types of PRMs in one chosen location of PRR. Before the run-time reconfiguration, the bitstream manipulation modifying the information related to its location to apply it into other different PRR is needed.

3.3 Generic partial dynamic reconfiguration controller

The concept of the first GPDRC for transient fault mitigation was presented in [20]. The first implementation within system with counter and SEU injection was presented in [8]. Previous GPDRC design has been extended to be able to perform reconfiguration of entire FT system (several PRMs) when the permanent fault occurs in its PRM. New issues such as choosing the proper configuration from the next generation of configurations, performing the relocation process on loaded PRBs and the synchronization of the complete FT system were solved and implemented into controller. The GPDRC for transient and permanent fault mitigation was presented in [12].

Before the development of GPDRC, several design goals to be achieved were defined:

- The resource utilization of new controller has to be lower than the standard controller units implemented by universal softcore processors. It must be built in generic way to be able to perform PDR in the systems with the different number of PRMs.

- The controller should be autonomously able to determine the type of fault which occured in a PRM, whether it is a transient or a permanent one - for this purpose the information on whether the fault occurred during n successive reconfiguration cycles (the reconfiguration cycle consists of faulty PRM detection, PRM reconfiguration, PRM synchronization) can be used. If the fault occurrence is equal or lower than n, the fault is seen as a transient one, otherwise it is concluded that the fault is a permanent one.
- The PDR will be done via internal reconfiguration interface (ICAP in Xilinx FPGAs) and utilize its full speed (up to 100MHz).
- To reduce the number of needed precompiled PRBs the controller has to implement the technique to use the same PRB for the PDR of several PRMs where it is possible (e.g. the same type of PRM but different physical assignment to PRR).
- The controller should allow the synchronization of reconfigured PRMs.
- The controller should support different external memory devices.

The detailed architecture of GPDRC can be seen in Figure 3.4. Its interface contains an error vector of FT architectures as input. Its width depends on the number of FT architectures and the available number of PRMs for each of them. The next interface signals such as bitstream address and data are designated to communication with external bitstream storage when bitstream is transported through ICAP interface of FPGA. The *sync done* and *rec done* signals are intended for controlling the synchronization of reconfigured PRMs in FT architectures, the *fatal* signal announces the situation when the FT architecture cannot be repaired by GPDRC because the number of available PRMs has fallen below the required minimum.

3.4 Fault mitigation procedure

In Figure 3.5 the behavior of the system after a fault is detected in PRM is shown in flow diagram. The fault is detected by the FT architecture. The FT architecture generates a set of error signals which identify the faulty PRM (step 0). This is possible due to the fact that the functional units and voters are implemented into separate PRMs and the relation between the units and PRMs where they are placed is known.

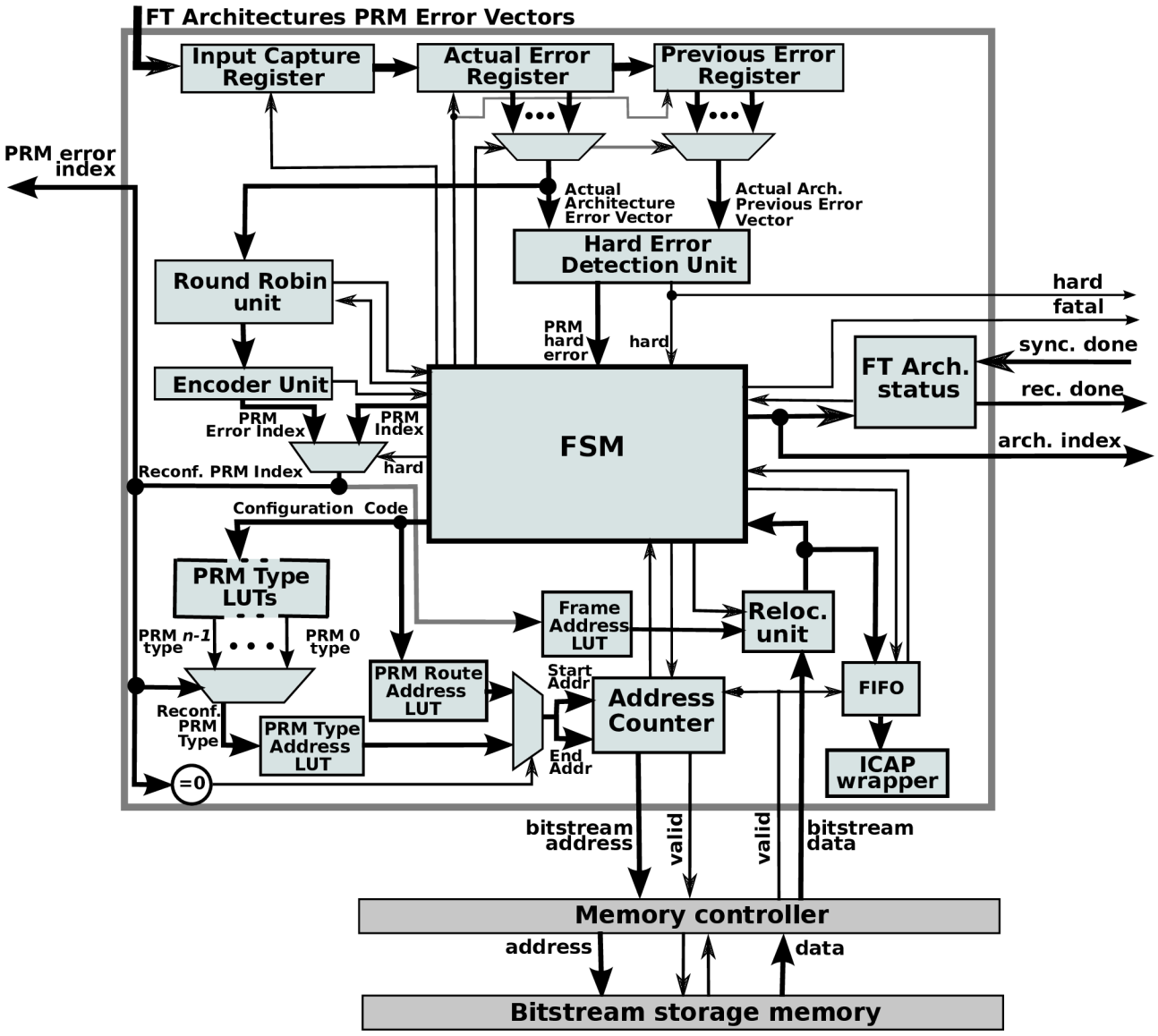


Figure 3.4: Fault tolerant system structure for SRAM based FPGA

When the faulty PRM is localized, the GPDRC determines, if the occured fault will be considered as transient or permanent one. The solution used in the case of transient fault occurence is denoted as the option A further in this text. If the fault is seen as a permanent one, then the subsequent steps depend on whether the current configuration comes from the final generation (Generation 2 in this case). The GPDRC stores the configuration code of actual configuration so it is able to identify that it is from final generation. If it is from final generation, there is no additional option to continue in mitigation of this new permanent fault and the FT architecture will indicate this to GPDRC unit. Then, the intervention from outside is needed (e.g. physical placement of configuration is moved

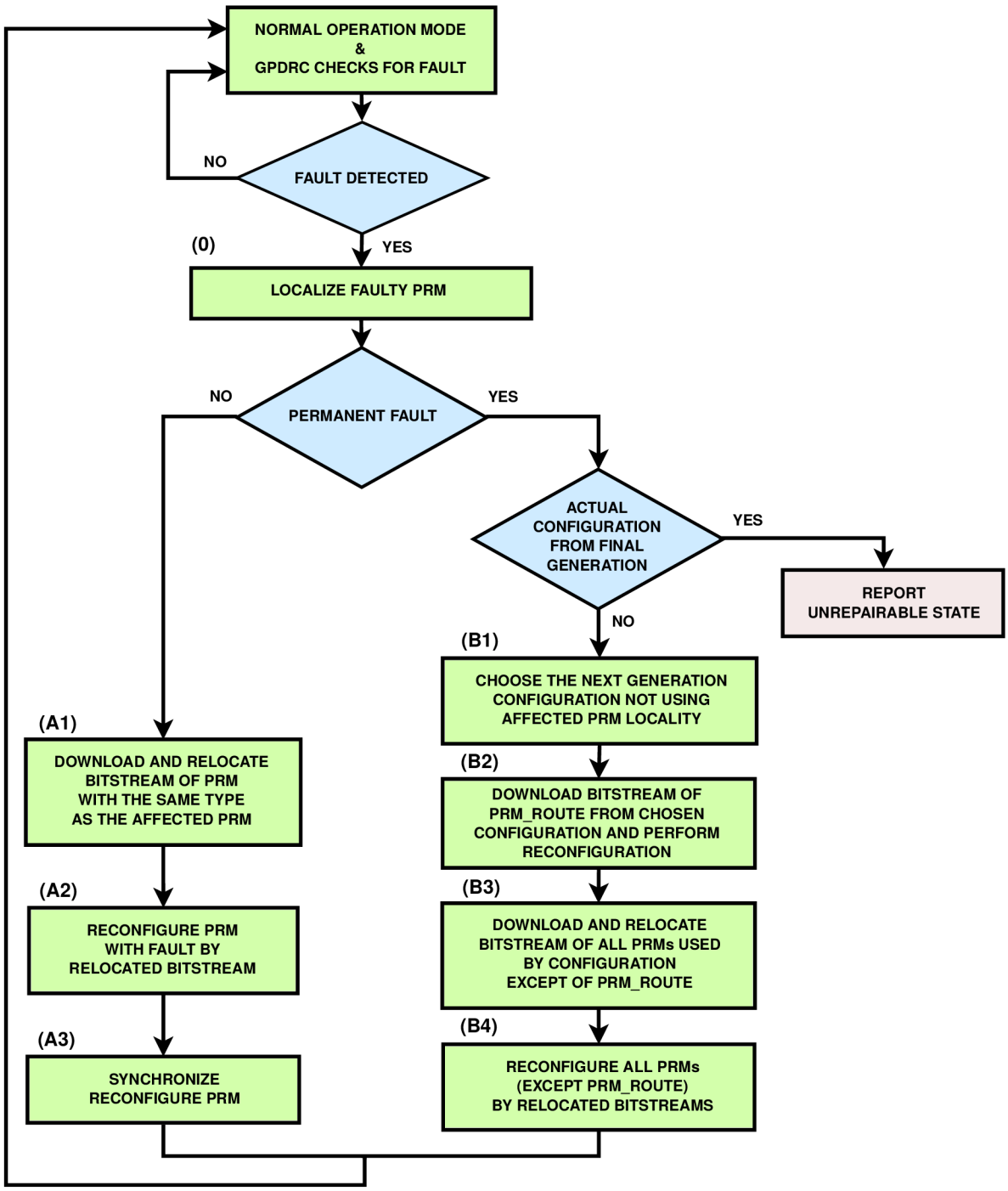


Figure 3.5: Reconfiguration flow diagram

to another locality of FPGA or the FPGA is replaced with a new one). In the situation when actual configuration is not from final generation, it is possible to mitigate the occured fault and the solution is denoted as the option B.

Option A - recovery from a transient fault: After a transient fault is detected, GPDRC reads from external memory the PRB which re-

sponds to the type (PRM_FU, PRM_VOTER, PRM_CHECKER, etc.) of the identified faulty PRM. The type of the unit is known because the GPRDC knows which configuration is configured actually and the distribution of PRMs in it. The downloaded PRB is originally designated to the first suitable PRR (typically to PRR1). Therefore, the next step of mitigation process (step A1) will be the relocation of this bitstream in such way that it can be used for reconfiguration of the affected PRM. The reconfiguration process of this PRM with the relocated PRB is driven by GPDRC (step A2).

After the reconfiguration is finished, in some cases the PRM must be synchronized with other components of FT architecture. The synchronization can be also controlled by GPDRC (step A3).

Option B - recovery from a permanent fault: After a permanent fault is detected in PRM and the actual configuration does not belong to the final generation, new configuration from the following generation is selected. This configuration will not use the faulty PRM. The GPDRC will choose configuration according to configuration code which will respond to bitwise negation of the vector of error signals from FT architecture (B1 step).

The PRB for PRM_ROUTE (PRM with the interconnections) of selected configuration is stored in the external bitstream storage. This bitstream is designated to reconfigure resources of PRR0 (the only PRR of FPGA where this bitstream of PRM_ROUTE can be assigned). This implies that there is no need to relocate this PRB (step B2).

The downloading of PRB copies implementing all remaining PRMs will be the next action. The number of needed bitstream copies and their type (if it is implementing PRM_FU, PRM_CHECKER or PRM_VOTER) is determined by the selected configuration. PRBs of all PRM types are downloaded from the same destination, as in the case of reconfiguration after transient fault. Each of these downloaded PRBs will go through relocation process which will make them suitable for appropriate PRRs (step B3).

The downloaded and relocated PRBs are used for the reconfiguration of PRMs, which are used in the configuration (step B4). After completion of the reconfiguration, local reset of units in newly configured PRMs is performed. Also some kind of synchronization (state recovery of all units in affected PRMs) can be performed in this step.

4. Design of FT architecture by means of developed methodology principles

The process of FT architecture system design to meet requirements defined by the proposed methodology is described in this chapter.

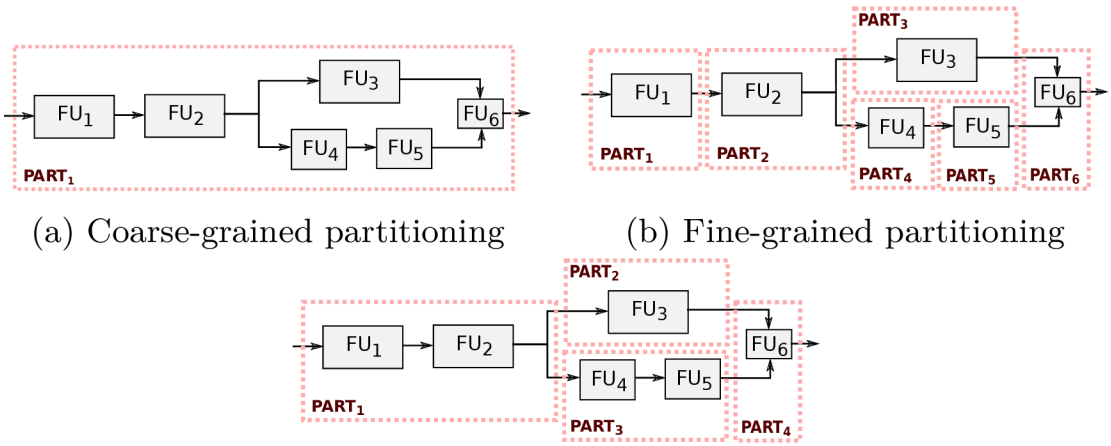
4.1 Fault tolerant architectures design

The application of the methodology requires the specific process of system design. When this design is adopted, it is ensured that faults appearing subsequently in functional modules or other FT modules (containing voters, checkers, etc.) of design can be mitigated.

The original system design delivered from a designer for securing has to be divided into important parts in terms of required dependability and the remaining parts which may remain unsecured (from the methodology point of view) or they are secured in some other way. From the chosen important system parts every single part will be secured as single FT architecture with fault mitigation capability according to the methodology. The process of partitioning has to be driven by designer knowledge of importance of each system part. This can be gained as the result of modelling reliability of system parts and the impacts of faults occured in specific system part to entire system. The partitioning can be done with different granularity (see also Figure 4.1)

- Coarse-grained partitioning - The complete system is just one part.
- Fine-grained partitioning - The system is divided into more smaller parts. This reduces the overall size of all needed bitstreams but the GPDRC size is increased and it brings more complexity to fault mitigation process.
- Mixed partitioning - The combination of two previous approaches can be done by grouping several small system parts into several groups and implement each of them as single FT architecture.

The next step is the selection of degradation strategy for each chosen important part according to their stated level of importance. Permanent



(a) Coarse-grained partitioning

(b) Fine-grained partitioning

(c) Mixed partitioning

Figure 4.1: Design partitioning with different granularity

fault occurence in system is mitigated by downgrading the FT architecture from the robust one to less robust one. This step is required every time a permament fault occurs in currently occupied PRR containing the PRM of FT architecture (see Figure 4.2). The less robust FT architecture will exclude this PRR from the further use. The number of PRRs which can be excluded at the same time then specifies the number of permanent faults which can be handled by this secured part of the system.

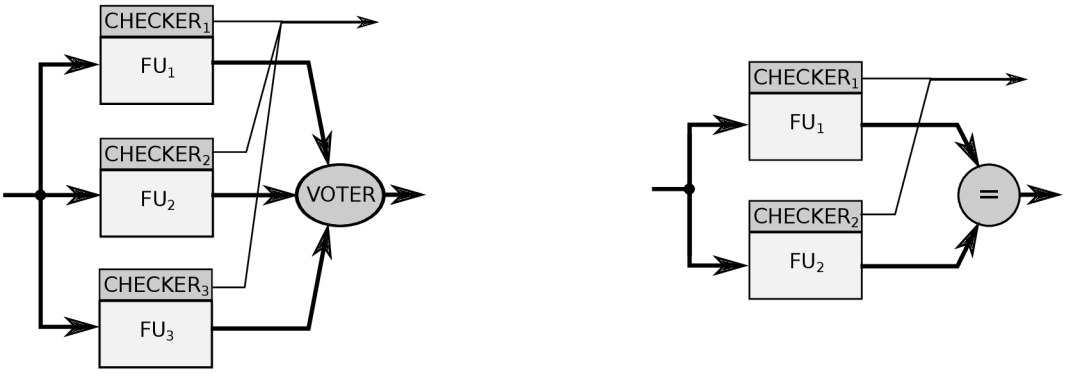


Figure 4.2: The set of FT architectures as a sample of degradation strategy

In the next step, the implementation area in FPGA for each system part implemented as FT architecture has to be stated. This area is allocated in dynamic area. It enables the modification of assigned PRMs by PDR. The remaining parts are placed in static area. For each chosen important part of system, several PRRs will be created. To these PRRs, the PRMs of currently used FT architecture will be assigned according to

stated procedure (see Figure 4.3). The location and the size of PRRs for implementing one system part must respect this conditions:

- The number of PRRs is the same or bigger than the number of PRMs of the starting (the most robust) FT architecture for given system part.
- The set of created PRRs will contain one specific PRR for PRM with routing (PRM_ROUTE). This PRR has to be located in the neighbourhood of all other PRRs.
- Every PRR from the set of created PRRs (except of the PRR designated to be configured by PRM with routing) has to have the same size, the same structure and the same local placement of the FPGA resources.
- The placement of PRR and also the size of the smallest possible PRR (PRR_{min}) is limited by the fact that reconfiguration is done per configuration frames.As the configuration frame is modifying the configuration of specified number of resources at once, the location and the size of PRR has to respect these principles and can only allocate resources corresponding to one PRR_{min} or its multiples.

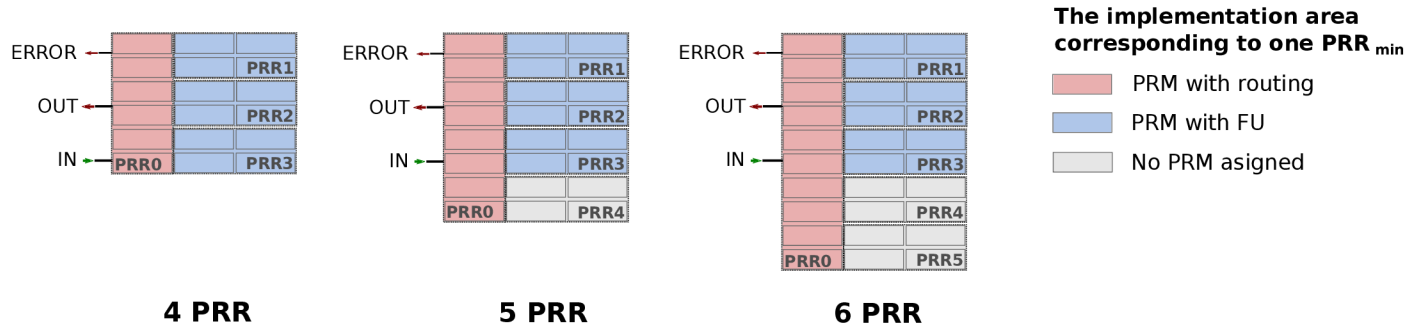


Figure 4.3: Several possibilities with area allocation for simple TMR architecture

4.2 The implementation of generated FT architectures

The complete process starting with the entry of unsecured system design to the final step of configuration of FPGA with the equipment to tolerate the fault impacts and their mitigation consists of several steps:

1. Design entry - the designer enters VHDL source codes.

2. The specification of the system parts to be secured - the designer chooses the parts and the degradation stategy for each of these parts.

3. The generation of FT architectures for the use in degradation strategies - the developed tool is used for the generation of FT architectures for each system part.

4. The creation of secured FT system - the original system design is modified by replacement of selected parts by their implementations as FT architectures. This can be done without much effort because the interface of original part (unit) is a subset of the interface of the generated FT architecture. Further, the GPDRC instance has to be added and the error signals from all FT architectures have to be gathered and connected to its error input. The controller for some external memory device (e.g. the developed SD card controller) has to be added, too. This unit is needed to provide the configuration bitstream data for GPDRC. Alternatively, the synchronization controller and logic to perform synchronization of the modules of FT architectures can be added as well in this step.

5. The implementation of static design with the starting configuration - for the complete (static) reconfiguration of FPGA, the system design where all chosen important parts are secured with most robust FT archictures from generation 0 is used. This implementation run is also used for generating partial bitstreams for all PRMs utilized by FT architectures in generation 0. From these partial bitstreams, one from each PRM type is chosen as golden copy to be stored in external memory storage. These bitstreams can be later relocated and used during fault mitigation process.

6. The implementation of all partial configuration bitstreams - to create partial bitstreams which can be used by GPDRC for recovery from permanent fault, PRBs for each PRM with routing for all possible alternative configurations in each FT architecture is created.

5. Implementation and experimental results

This chapter describes the implementation results of systems where the methodology was applied and several experiments simulating transient and permanent fault occurences and their mitigation

5.1 The implementation of GPDRC

In the secured system design, a very important role is designated to GPDRC unit. The reason for its development as the alternative to controllers implemented into softcore processor is its smaller size and lower reconfiguration latency due to its specialization. Its size (the number of utilized FPGA resources) is mainly affected by the number of PRMs into which the system is implemented.

For the evaluation of GPDRC resource utilization results for different system partitioning approaches, a design with counters, registers, decoders and other logic was created. The complexity of this implemented system does not play any role in the evaluation of GPDRC size. It is mainly influenced by the overall number of PRMs and other attributes mentioned in above paragraph. Thus, the entire design in FPGA was divided into several FT architectures and they were divided into the same number of PRMs. The experiments were done for 3 to 6 PRMs. The size of GPDRC for various numbers of FT architectures and the number of PRMs is presented in Figure 5.1.

The size of GPDRC and its units together with the comparison with the size of MicroBlaze IP core used as PDR controller is shown in Table 5.1. These results are valid for 32 FT architectures with 6 PRMs per each controlled by the GPDRC. The meaning of the columns is as follows: the name of unit (column 1), the size of unit in slices (2), the number of occupied LUTs (3) and FlipFlops (4) and the size of TMR alternative (5).

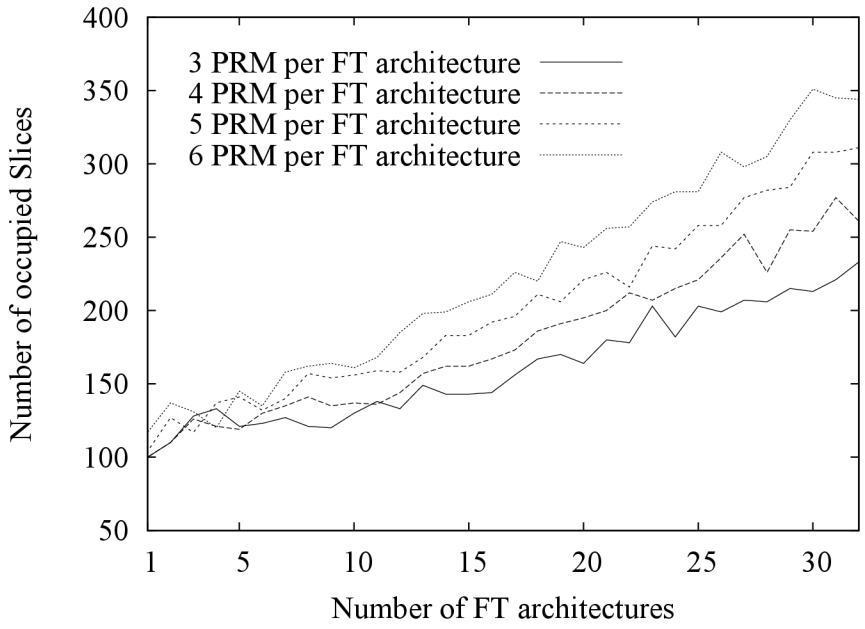


Figure 5.1: GPDRC size vs. the number of FT architectures for various numbers of PRMs per FT architecture

ML506 - Virtex5 192 PRMs	Size [slices]	LUTs [#]	F/Fs [#]	TMR [slices]
Input Capture Register	49 (0,6%)	97	192	127 (2,6x)
Actual Error Register	48 (0,6%)	101	101	124 (2,6x)
Previous Error Register	48 (0,6%)	192	192	124 (2,6x)
Hard Error Unit	3 (0,1%)	4	0	9 (3,0x)
Round Robin Unit	5 (0,1%)	6	6	14 (2,9x)
Error Encoder	3 (0,1%)	3	0	6 (2,0x)
Relocation Unit	7 (0,1%)	16	1	20 (2,9x)
Architecture Status Unit	2 (0,1%)	49	32	6 (3,0x)
Address Counter	22 (0,3%)	52	21	56 (2,5x)
FSM	22 (0,3%)	48	17	59 (2,7x)
Others (LUTs, MUXs...)	135 (1,7%)	317	186	414 (3,1x)
GPDRC total	**344 (4,2%)**	**885**	**748**	**959 (2,8x)**
MicroBlaze	**628 (7,7%)**	**1414**	**1491**	**1664 (2,8x)**

Table 5.1: The numbers of FPGA resources for GPDRC (32 FT architectures, 6 PRM per FT architecture)

5.2 FT architectures developed to secure a given part of system

This section presents the basic features of FT architectures which were developed as a model architectures for each generation (0, 1 and 2). Dif-

ferent FT architectures which have the ability to detect and localize faults on PRM level can be used. The proposed FT architectures utilize 5 PRMs and thus 5 error signals can be identified on the output of PRM_ROUTE block. These signals are connected to the inputs of GPDRC where they indicate the occurrence of a fault.

The initial FT architecture of Generation 0 is based on TMR scheme in which the outputs of all FUs are checked by the majority element (voter). This architecture consists of 5 PRMs (3 PRM_FUs, PRM_VOTER and PRM_ROUTE). Figure 5.2 presents the proposed structure of this architecture. Each FU of the architecture is implemented as a standalone PRM without any additional diagnostic logic. The outputs of all PRM FUs are connected into PRM_VOTER block which is implemented as a duplex architecture because of the need to detect fault occurrence in its structure.

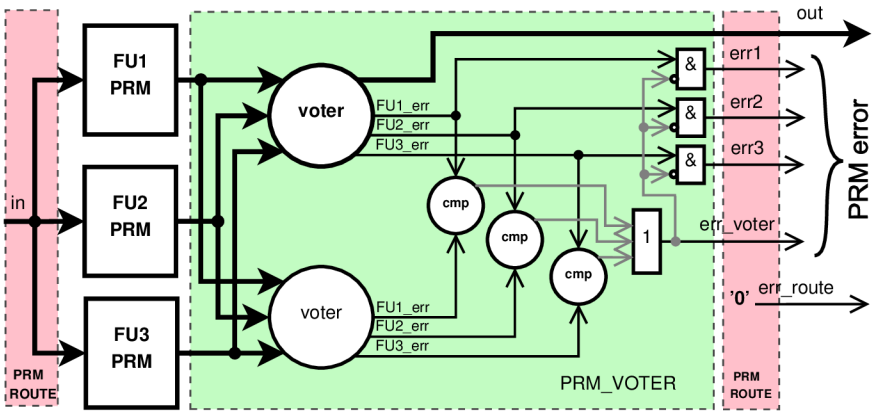


Figure 5.2: The FT Architecture of Generation 0 based on TMR

The FT architecture of Generation 1 is based on a duplex scheme with the addition of one PRM with CHECKER unit (PRM_CHECKER). As can be seen in Figure 5.3, this architecture consists of four PRMs (2 PRM_FU, PRM_CHECKER and PRM_ROUTE). Each FU of the architecture is implemented as a single PRM and their outputs are switched by output multiplexor which is controlled by error signal from diagnostic logic.

In order to detect any fault in PRM_ROUTE block, this block is supposed to be implemented as duplex architecture with comparator. The alternative of FT architecture of Generation 1 can be seen in Figure 5.4. The comparator output is connected to error signal *err_route*, the occurrence of logical one value on error signal will cause the start of PDR process.

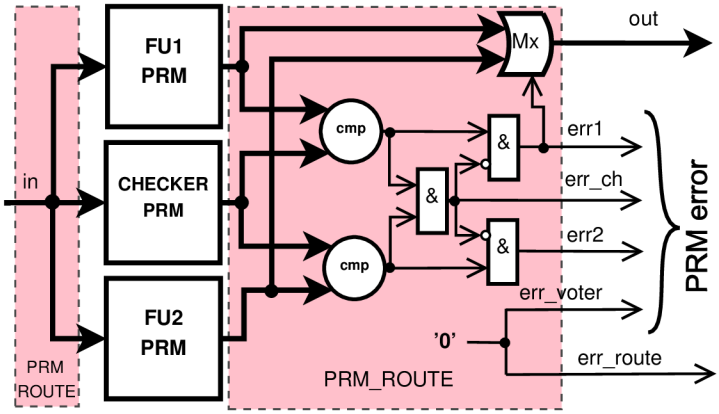


Figure 5.3: The FT Architecture of Generation 1 based on Duplex with checker

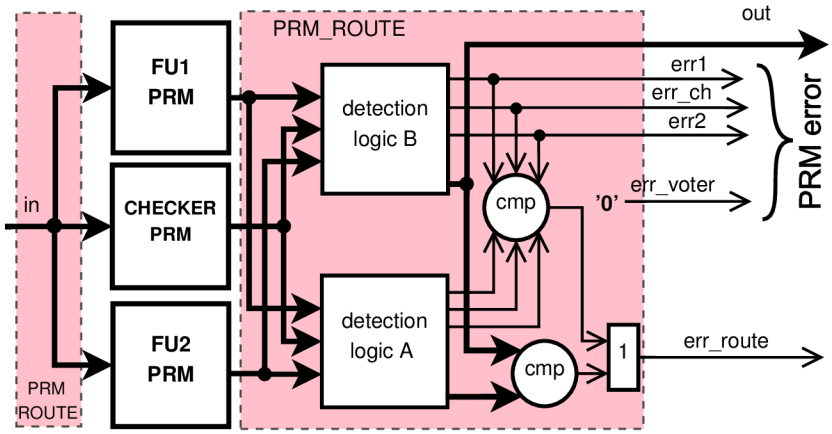


Figure 5.4: The alternative FT Architecture of Generation 1

The final architecture of Generation 2 is based on classical duplex scheme (see Figure 5.5). This architecture is not FT and it has only 3 PRMs (2 PRM_FU and PRM_ROUTE). In this architecture, PRM_ROUTE block contains additional diagnostic logic for fault detection. Because it is not known which one of the two PRM implementing FUs is faulty, the reconfiguration process is applied to both of FU PRMs. This is final architecture, no recovery from permanent fault is possible.

5.3 Evaluation of resource overhead

The sizes of FT architecture components which cause hardware overhead in FPGA are shown in Table 5.2. In this table, the overhead of only those units which were utilized and extended by our methodology when compared to the standard use of these units are taken into account. For

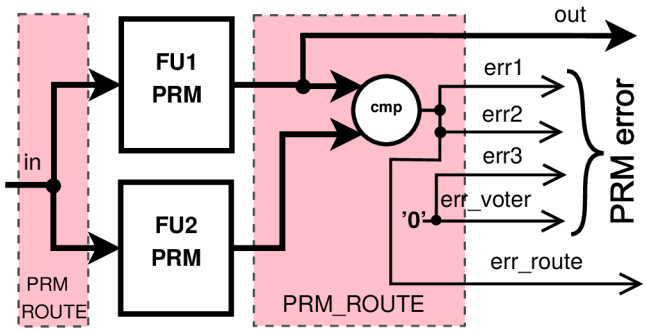


Figure 5.5: The architecture of Generation 2 based on Duplex

the generation 0, the overhead includes the size of PRM_ROUTE and PRM_VOTER units. The sizes of any of three FUs were not considered into overhead as they are present also in the standard TMR scheme. The size of PRM_VOTER unit was decremented by the size of standard majority voter unit without the ability of faulty unit localization to get only the overhead caused by the use of our methodology. For both types of the generation 1 and for the generation 2, the overhead includes only PRM_ROUTE unit for the same reasons as for the generation 0. The meaning of the columns is as follows: column 1 - the width of each FU output in bits; column 2 to 5 - the overhead of FT architecture from the specified generation in slices.

XC5VSX50T data width [bits]	Generation 0 [slices]	Generation 1 [slices]	Generation 1-variant [slices]	Generation 2 [slices]
2	12	5	12	1
4	22	11	24	2
8	36	17	39	3
16	68	31	68	7
32	126	57	122	12
64	206	111	210	23

Table 5.2: The overheads of Generations in slices

5.4 Implementation results of different approaches to the partitioning of original system

The key step in design process of securing a given system is its partitioning into parts which will be implemented as standalone FT architectures. To examine the properties of a secured system such as hardware overhead or the size of PRBs used for the reconfiguration after fault occurence for different types of its partitioning, the test design of system with MB-LITE softcore processor (see [10]) was developed. Altough the top-level design contains only two main units - the instance of MB-LITE and Wishbone adapter, the processor can be further divided into 4 functional units - IF, ID, EX and MEM.

From the implementation results revealed the fact that the unit performing the execute stage of pipeline (MB-LITE - EX) utilizes much more slices than other units which is caused by a big number of used LUTs. Therefore, the following possibilities for partitioning based on different granularity were proposed:

- 1 FT architecture - all functional units are grouped together and replicated (coarse-grained partitioning), see Figure 5.6.
- 2 FT architectures - EX unit of MB-LITE processor is implemented as one FT architecture, the remaining units are grouped together and implemented as the second FT architecture.
- 5 FT architectures - each unit mentioned in the above provided table is implemented in a single FT architecture (fine-grained partitioning).

Table 5.3 shows the implementations of all variants of the secured system. They were compared by their resource utilization (column 1), hardware overhead in comparison to original design (column 2) and the sizes of their PRBs (column 3).

The results summarized in the table show that the HW overhead is slightly lower for the variant with 1 FT architecture. This is caused by the smaller GPDRC unit due to lower number of PRMs. On the other side, this is degraded by bigger PRB size which causes longer reconfiguration time. The table shows that there is a tradeoff between HW overhead and the overall size of all PRBs (or the time of reconfiguration).

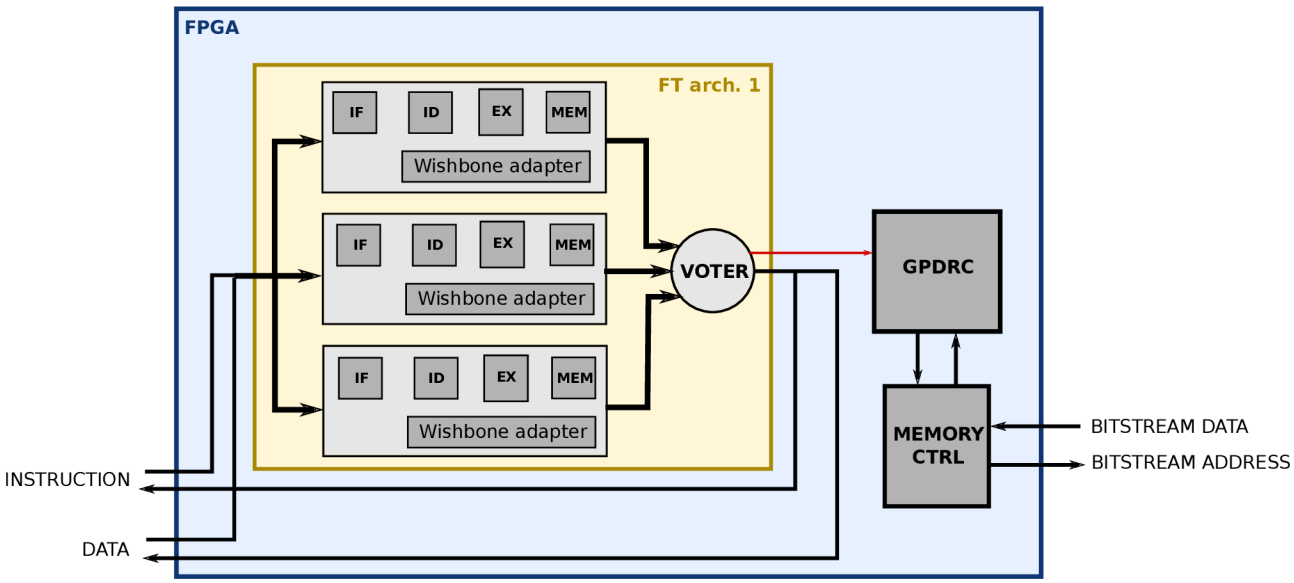


Figure 5.6: All units in one FT architecture

XC5VSX50T The variant of secured system	PRMs [#]	Slices [#]	HW overhead [%]	Bitstream sizes *PRM_ROUTE*: [kB] *PRM_FU*: [kB]
Original design	0	723	0	-
TMR design (w/o PDR)	0	2287	216	-
1 FT arch.	5	2421	235	6,6 408
2 FT arch.	10	2484	244	6,6; 6,6 92,4; 39,6
5 FT arch.	25	2572	256	6,6; 6,6; 6,6; 6,6; 6,6 6,6; 19,8; 92,4; 13,2; 6,6

Table 5.3: The comparison of resource utilization and hardware overhead for different implementations of the given system

5.5 SEU testing platform for the evaluation of FT system design by means of methodology principles

To evaluate the quality of secured FPGA to cope with transient and permanent fault occurence, the special test platform was developed. The testing was based on fault injection into configuration bitstream to simulate an SEU fault occurence. The platform allows to observe the behaviour of entire secured system implemented in FPGA when a fault occurs. The test platform contains several parts which are creating together the necessary test and evaluation equipment (see Figure 5.7). The FPGA is

configured by the implementation of system secured by the means of the methodology. The remaining parts of the test platform are implemented and run on PC.

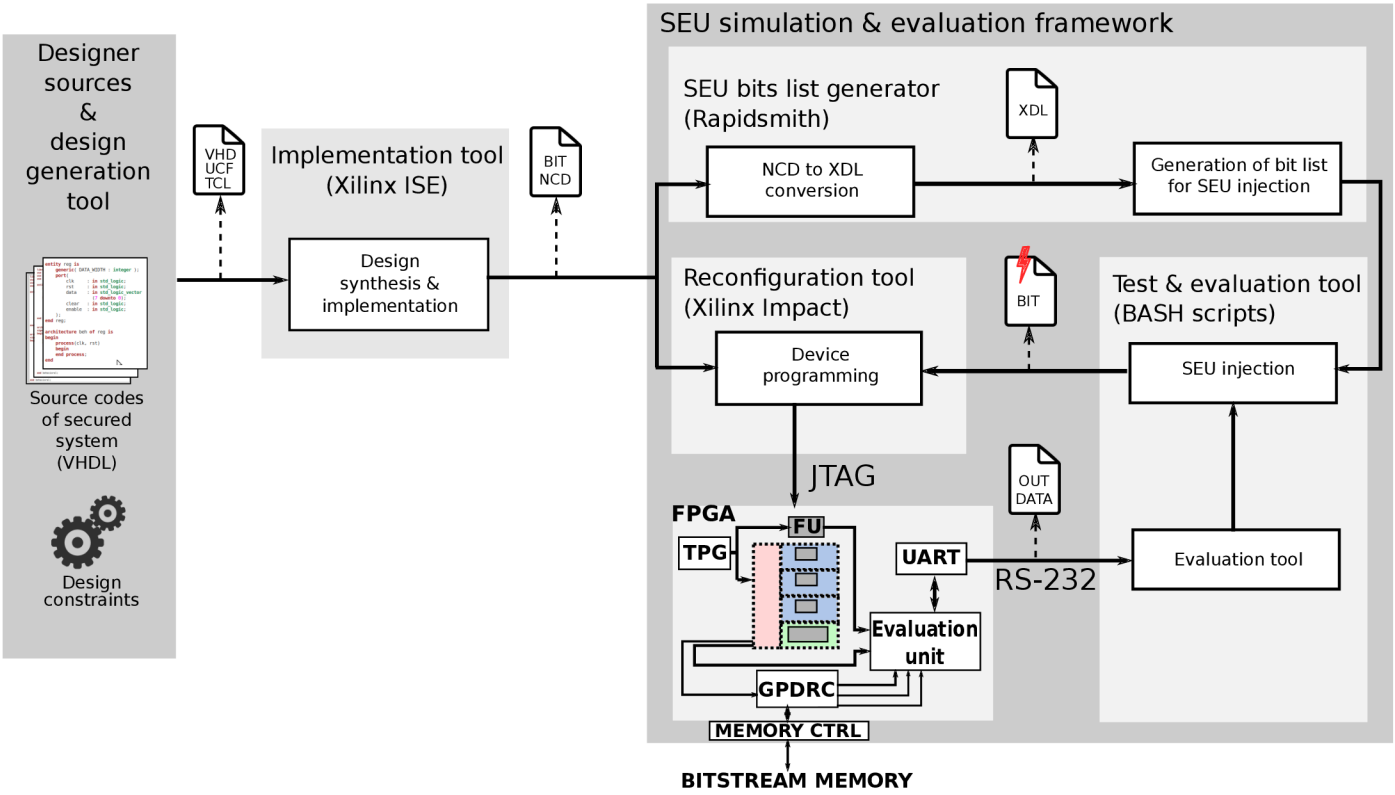


Figure 5.7: Dependability evaluation platform with SEU injection test platform

For the SEU injection into configuration memory, the external SEU injector presented in [11] was used. It uses PDR to simulate the radiation-induced upsets by artificially changing the contents of the configuration memory. This injector is written as TCL script and is run on PC. It accesses the JTAG external reconfiguration interface of FPGA. It uses the ChipScope library function of Xilinx ISE toolkit to perform the upset in configuration memory by toggling some bit value in the configuration bitstream.

One simulation step for testing a secured design consists of one SEU injection into one of FT architecture PRMs and checking its output for error. When the fault is detected by the compare logic in the evaluation unit or by detection and localization logic implemented in FT architecture, the status message is sent via RS-232 to the evaluation tool in PC. If the reconfiguration is performed, the GPDRC status is observed and after *rec. done* signal is set, the next status message is sent to the evaluation tool.

Experimental results of GPDRC transient fault mitigation process

The test platform described in the above section was implemented and tested with Virtex 5 FPGA (XC5VSX50T) on an ML506 development board. To implement the system design for FPGA and for bitstream generation, the Xilinx ISE 14.7 toolkit was used. The FU contains several 8-bit counters, decoders and multiplexers, the data width of input was 6 bits and the data width of output was 16 bits. The design contains one FT architecture with 5 PRMs and the FT architectures described in Section 5.2 was used in the degradation strategy. The size of a PRB for PRMs with FU, PRM with doubled voter and PRM with checker unit was 6632 bytes, the sizes of PRBs for each PRM with routing were 26582 bytes.

XC5VSX50T PRM type	PRM utiliz. %	SEU detected by FT arch. #	FT arch. output errors #	SEU missed by FT. arch #	GPDRC reconf. #
PRM_FU (all generations)	45%	7806	552	25	7826
PRM_VOTER (generation 0)	30%	3345	1571	105	3345
PRM_CHECKER (generation 1)	45%	6901	552	11	6900
PRM_ROUTE (generation 0)	1%	0	21	21	0
PRM_ROUTE (generation 1)	12%	3542	1243	234	3541
PRM_ROUTE (generation 2)	6%	2432	1056	351	2430

Table 5.4: The number of detected SEUs in FUs of the architecture

From the results summarized in Table 5.4, it can be seen that the FT architecture of generation 0 and 1 can detect and repair more than 97% SEUs in PRM with FU, voter or checker unit. Except the FT architecture from generation 0 which do not have ability to detect faults in PRM with routing, this PRM type in the FT architectures from other generations was able to detect faults in more than 85% cases. Almost all detected faults have triggered the mitigation process done by GPDRC. In all cases, the PRM with routing was able to survive most of the SEU faults injected inside it due to very low utilization of FPGA resources.

Testing and evaluating recovery from permanent fault occurence

According to the results of SEU injection campaign during transient fault simulation process only some configuration bits of FT architecture PRMs were used for permanent fault simulation. A specific bit was chosen for permanent fault injection campaign, if the fault that it creates in the unit implemented in PRM was detected by the detection logic of FT architecture or it has been manifested as an error on FT architecture output during the transient fault simulation campaign.

The experimental results for a permanent fault injection campaign to the same FT architectures as in previous experiment are shown in Table 5.5. The meaning of the columns in the table is as follows: column 1 - the type of FT architecture and which generation it belongs to; column 2 - the number of injected SEU faults; column 3 - the number of incorrect data on the outputs of FT architecture; column 4 - the number of permanent faults detected; column 5 - the number of performed permanent fault recoveries by the reconfiguration to a different FT architecture; column 6 - the mean time to repair the system to the correctly operating state.

XC5VSX50T Generation	Injected faults #	FT arch. output data errors #	Perm. fault detected #	Recovery done #	MTTR [ms]
Generation 0 (TMR-2xVOTER)	12768	2144	12515	12480	512
Generation 1 (TMR-simple)	12575	1796	12287	9802	351
Generation 2 (Duplex)	7987	708	156	0	-

Table 5.5: The number of successfully detected permanent faults and the MTTR for permanent fault recovery

The results summarized in Table 5.5 show that the number of detected faults is decreasing with the number of PRMs which are used by FT architecture. This is caused mainly by masking the faults which are injected into excluded PRMs. The repair process is shorter for less robust FT architectures due to the fact that the reconfiguration of fewer PRM is performed.

6. Conclusions

In this work, the methodology of FT system design with the ability to mitigate transient faults caused by SEUs and to recover from several permanent fault occurences was proposed as the alternative to existing methods or methodologies. This methodology benefits from the ability to PDR in modern FPGAs which can be used for the run-time repair or the change of current FPGA configuration. The production of correct outputs from the system implemented in FPGA even during its PDR is ensured by its designing as an FT architecture. For the transient and permanent fault mitigation techniques, the set of relocatable PRBs to create the sets of alternative configurations is created and used for PDR. The final system implementation is then ready to survive many transient and several permanent fault occurences.

6.1 Benefits of this research

As the main benefit of this research, the proposal of alternative methodology for the FT system design with the ability of fault mitigation can be mentioned. This methodology brings some new features such as the use of dedicated reconfiguration controller or the application of the relocation technique in transient fault mitigation and also in the recovery process from permanent fault occurence. This greatly suppresses the main disadvantage of the use of precompiled configurations to mitigate faults which is space demanding storing of many configuration bitstreams. All benefits are summarized in the following points.

- The exact procedure of transformation the system design entered by designer to secured system where selected important parts are implemented as FT architectures with the mechanism of transient fault mitigation and recovery from permanent fault was described.
- The dedicated reconfiguration controller (GPDRC) with the ability to determine the type of fault and perform the correct mitigation procedure was developed. It was designed with the effort to reduce the necessary area and performance overhead.
- There is a possibility to define the level of importance for every system part by specifying the degradation strategy. This strategy

is used when the recovery after permanent fault occurence is performed.

- The final system design can be extended with synchronization mechanism for reconfigured units. The GPDRC is designed to cooperate with the synchronization controller.

6.2 Possible enhancements of methodology

This complex methodology is based on many principles and incorporates many methods which can be further enhanced to achieve better performance of final secured system, to lower the neccessary area overhead in FPGA or to lower the necessary capacity of external bitstream storage.

One of possible approaches to reduce the space for storing the data is data compression. It can be incorporated into final FT system designed by means of proposed methodology. Due to the fact that the GPDRC unit does not implement the direct read from the memory but it uses external memory controller unit, the decompressor unit for processing the compressed bitstream can be inserted to the data path between these units.

Current methodology uses the standard PR design flow defined by Xilinx to implement the final FT system and generate the static design bitstream and the set of partial configuration bistreams. Another flow such as IDF can be adopted to make the system more secure by reducing the possibility of fault occurence which affects more than one PRM at once. This can be achieved by thorough isolation of PRMs. It can also simplify the relocation of PRBs designated to these isolated PRMs.

Several enhancements would be also possible in the GPDRC unit. One of the current issues in this unit is its permanent occupation of ICAP. If the system entered by designer would like to use the PDR ability of FPGA for its reconfiguration it will not be possible because only one instance of this unit can be used. This can be solved by excluding the ICAP instance from the GPDRC unit and using it externally. Then some multiplexing logic can be added and this one ICAP instance can be shared by the original system and by the GPDRC. Because the final secured system (designed by means of the methodology) is based on the set of FT architectures and thus the fault can be masked by them the instant fault mitigation is not necessary. The GPDRC unit can wait until the ICAP instance is not used and then finally perform the mitigation process.

Bibliography

[1] Cristiana Bolchini, Antonio Miele, and Marco D. Santambrogio. Tmr and partial dynamic reconfiguration to mitigate seu faults in fpgas. In *22nd International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT '07)*, pages 87–95, Washington, DC, USA, 2007. IEEE CS.

[2] Cristiana Bolchini, Antonio Miele, and Marco D. Santambrogio. Tmr and partial dynamic reconfiguration to mitigate seu faults in fpgas. In *22nd International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT '07)*, pages 87–95, Washington, DC, USA, 2007. IEEE CS.

[3] R. F. DeMara and Kening Zhang. Autonomous fpga fault handling through competitive runtime reconfiguration. In *2005 NASA/DoD Conference on Evolvable Hardware (EH'05)*, pages 109–116, June 2005.

[4] F. Hatori, T. Sakurai, K. Nogami, K. Sawada, M. Takahashi, M. Ichida, M. Uchida, I. Yoshii, Y. Kawahara, T. Hibi, Y. Saeki, H. Muroga, A. Tanaka, and K. Kanzaki. Introducing redundancy in field programmable gate arrays. In *Custom Integrated Circuits Conference, 1993., Proceedings of the IEEE 1993*, pages 7.1.1–7.1.4, May 1993.

[5] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb. Fpga partial reconfiguration via configuration scrubbing. In *Field Programmable Logic and Applications (FPL '09)*, pages 99–104, Washington, USA, 2009. IEEE CS.

[6] Yoshihiro Ichinomiya, Shiro Tanoue, Motoki Amagasaki, Masahiro Iida, Morihiro Kuga, and Toshinori Sueyoshi. Improving the

robustness of a softcore processor against seus by using tmr and partial reconfiguration. In *Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, FCCM '10, pages 47–54, Washington, DC, USA, 2010. IEEE Computer Society.

[7] Hlavicka J. *Cislicove systemy odolne proti porucham.* CVUT, 1992.

[8] Straka M., Miculka L., Kastil J. and Kotasek Z. Test platform for fault tolerant systems design qualities verification. In *15th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 336–341. IEEE Computer Society, 2012.

[9] F. Kastensmidt, L. Carro, and R. Reis. Designing fault tolerant systems into sram-based fpgas. In *Design Automation Conference, 2003. Proceedings*, pages 650–655, June 2003.

[10] T. Kranenburg and R. van Leuken. Mb-lite: A robust, light-weight soft-core implementation of the microblaze architecture. In *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, pages 997–1000, March 2010.

[11] Kastil J., Straka M., Miculka L. and Kotasek Z. Dependability analysis of fault tolerant systems based on partial dynamic reconfiguration implemented into fpga. In *15th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 250–257. IEEE Computer Society, 2012.

[12] Miculka L. and Kotasek Z. Generic partial dynamic reconfiguration controller for transient and permanent fault mitigation in fault tolerant systems implemented into fpga. In *17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 171–174. IEEE Computer Society, 2014.

[13] J. Lach, W. H. Mangione-Smith, and M. Potkonjak. Enhanced fpga reliability through efficient run-time fault reconfiguration. *IEEE Transactions on Reliability*, 49(3):296–304, Sep 2000.

[14] Vijay Lakamraju and Russell Tessier. Tolerating operational faults in cluster-based fpgas. In *Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays*, FPGA '00, pages 187–194, New York, NY, USA, 2000. ACM.

[15] Tyler M. Lovelly and Alan D. George. Comparative analysis of present and future space processors with device metrics. *Journal of Aerospace Information Systems*, 14(3):184–197, 2017.

[16] J. Narasimham, K. Nakajima, C. S. Rim, and A. T. Dahbura. Yield enhancement of programmable asic arrays by reconfiguration of circuit placements. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(8):976–986, Aug 1994.

[17] M. M. Pereira, L. Braun, M. Hübner, J. Becker, and L. Carro. Run-time resource instantiation for fault tolerance in fpgas. In *2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 88–95, June 2011.

[18] Aiwu Ruan, Bairui Jie, Li Wan, Junhao Yang, Chuanyin Xiang, Zujian Zhu, and Yu Wang. A bitstream readback-based automatic functional test and diagnosis method for xilinx fpgas. *Microelectronics Reliability*, 54(8):1627–1635, 2014.

[19] T. D. A. W. Ruan and P. L. B. R. Jie. A bitstream readback based fpga test and diagnosis system. In *2014 International Symposium on Integrated Circuits (ISIC)*, pages 592–595, Dec 2014.

[20] M. Straka, J. Kastil, and Z. Kotasek. Generic partial dynamic reconfiguration controller for fault tolerant designs based on fpga. In *NORCHIP '10*, pages 1–4, Washington, DC, USA, 2010. IEEE CS.

[21] M. Wirthlin, E. Johnson, N. Rollins, M. Caffrey, and P. Graham. The reliability of fpga circuit designs in the presence of radiation induced configuration upsets. In *Field-Programmable Custom Computing Machines, 2003. FCCM 2003. 11th Annual IEEE Symposium on*, pages 133–142, April 2003.

[22] A. J. Yu and G. G. F. Lemieux. Defect-tolerant fpga switch block and connection block with fine-grain redundancy for yield enhancement. In *International Conference on Field Programmable Logic and Applications, 2005.*, pages 255–262, Aug 2005.

[23] Shu-Yi Yu and Edward J. McCluskey. Permanent fault repair for fpgas with limited redundant area. In *DFT '01: Proceedings of the 16th IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pages 125–133, Washington, DC, USA, 2001. IEEE Computer Society.

Author's publications

1. Towards a State Synchronization Methodology for Recovery Process after Partial Reconfiguration of Fault Tolerant Systems. In *4th Prague Embedded Systems Workshop.* Proceedings of 4th PESW, 2016 (20%)

2. Miculka L. and Kotasek Z. Generic partial dynamic reconfiguration controller for transient and permanent fault mitigation in fault tolerant systems implemented into fpga. In *17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 171–174. IEEE Computer Society, 2014 (70%)

3. Szurman K., Miculka L. and Kotasek Z. State synchronization after partial reconfiguration of fault tolerant can bus control system. In *17th Euromicro Conference on Digital Systems Design*, pages 704–707. IEEE Computer Society, 2014 (30%)

4. Szurman K., Miculka L. and Kotasek Z. Towards a state synchronization methodology for recovery process after partial reconfiguration of fault tolerant systems. In *9th IEEE International Conference on Computer Engineering and Systems*, pages 231–236. IEEE Computer Society, 2014 (20%)

5. Miculka L. and Kotasek Z. Synchronization technique for tmr system after dynamic reconfiguration on fpga. In *The Second Workshop on Manufacturable and Dependable Multicore Architectures at Nanoscale (MEDIAN 2013)*, pages 53–56. Politecnico di Milano, 2013 (80%)

6. Miculka L. Metoda návrhu systémů odolných proti poruchám do omezeného implementačního prostoru na bázi FPGA. In *Počítačové architektury & diagnostika 2013*, pages 63–68. University of West Bohemia in Pilsen, 2013 (100%)

7. Miculka L., Straka M. and Kotasek Z. Methodology for fault tolerant system design based on fpga into limited redundant area. In *16th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 227–234. IEEE Computer Society, 2013 (50%)

8. Straka M., Kastil J., Kotasek Z. and Miculka L. Fault tolerant system design and seu injection based testing. *Microprocessors and Microsystems*, 2013(37):155–173, 2013 (10%)

9. Kastil J., Straka M., Miculka L. and Kotasek Z. Dependability analysis of fault tolerant systems based on partial dynamic reconfiguration implemented into fpga. In *15th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 250–257. IEEE Computer Society, 2012 (15%)

10. Miculka L. and Kotasek Z. Design sychronization after partial dynamic reconfiguration of fault tolerant system. In *15th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 20–21. IEEE Computer Society, 2012 (70%)

11. Miculka L. Metoda návrhu systémů odolných proti poruchám do omezeného implementačního prostoru na bázi FPGA. In *Počítačové architektury & diagnostika 2012*, pages 109–115. Faculty of Information Technology, Czech Technical University in Prague, 2012 (100%)

12. Straka M., Miculka L., Kastil J. and Kotasek Z. Test platform for fault tolerant systems design qualities verification. In *15th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 336–341. IEEE Computer Society, 2012 (30%)

13. Miculka L. Metoda návrhu systémů odolných proti poruchám do omezeného implementačního prostoru na bázi FPGA. In *Počítačové architektury & diagnostika 2011*, pages 61–66. Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, 2011 (100%)

Publications cited by other authors

- Miculka L. and Kotasek Z. Generic partial dynamic reconfiguration controller for transient and permanent fault mitigation in fault tolerant systems implemented into fpga. In *17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 171–174. IEEE Computer Society, 2014

  - B. H. Krishna and C. A. Kumar. A novel method of reconfigurable image processing using fpga. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 3784–3789, 2016
  - S. Di Carlo, P. Prinetto, P. Trotta, and J. Andersson. A portable open-source controller for safe dynamic partial reconfiguration on xilinx fpgas. In *Field Programmable Logic and Applications (FPL), 2015 25th International Conference on*, pages 1–4, 2015

- Miculka L. and Kotasek Z. Synchronization technique for tmr system after dynamic reconfiguration on fpga. In *The Second Workshop on Manufacturable and Dependable Multicore Architectures at Nanoscale (MEDIAN 2013)*, pages 53–56. Politecnico di Milano, 2013

  - J. Jiménez, U. Bidarte, C. Cuadrado, E. García, and J. Lázaro. Safesoc: A fault-tolerant-by-redundancy evaluation card for high speed serial communications. In *2016 Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–4, 2016

- Miculka L., Straka M. and Kotasek Z. Methodology for fault tolerant system design based on fpga into limited redundant area. In *16th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 227–234. IEEE Computer Society, 2013

  - A. S. B. Lopes, E. Santos, M. Kreutz, and M. Pereira. A runtime mapping algorithm to tolerate permanent faults in a cgra. In *2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 63–70, 2016
  - R. Backasch, G. Hempel, S. Werner, S. Groppe, and T. Pionteck. Identifying homogenous reconfigurable regions in heterogeneous fpgas for module relocation. In *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, pages 1–6, 2014

- Straka M., Kastil J., Kotasek Z. and Miculka L. Fault tolerant system design and seu injection based testing. *Microprocessors and Microsystems*, 2013(37):155–173, 2013

  - P. H. W. Leong, H. Amano, J. Anderson, K. Bertels, J. M. P. Cardoso, O. Diessel, G. Gogniat, M. Hutton, J. Lee, W. Luk, P. Lysaght, M. Platzner, V. K. Prasanna, T. Rissa, C. Silvano, H. So, and Yu Wang. Significant papers from the first 25 years of the fpl conference. In *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–3, 2015
  - Thomas E. Carney, Richard P. McWilliam, and Alan Purvis. Modelling electronic circuit failures using a xilinx fpga system. *Procedia CIRP*, 38:277 – 282, 2015
  - Deepa Jose, P. Nirmal Kumar, Arfath Hussain, and Prabhu Shanker. Vlsi circuit partitioning using ant colony optimisation to yield fault tolerant testable systems. *Arabian Journal for Science and Engineering*, 39(12):8709–8729, 2014
  - Antonio da Silva, Pablo Parra, Óscar R. Polo, and Sebastián Sánchez. Runtime instrumentation of systemc/tlm2 interfaces for fault tolerance requirements verification in software cosimulation. *Model. Simul. Eng.*, 2014:42:42–42:42, 2014
  - Reza Omidi Gosheblagh and Karim Mohammadi. Article: Dynamic partial based single event upset (seu) injection platform on fpga. *International Journal of Computer Applications*, 76(3):19–24, 2013
  - D. Jose, P. N. Kumar, and A. David Naveen Dhas. Implementation of power optimized vlsi designs for reliable processing using majority circuit. In *2013 Annual IEEE India Conference (INDICON)*, pages 1–6, 2013
  - Reza Omidi Gosheblagh and Karim Mohammadi. New approach to emulate seu faults on sram based fpgas. *Journal of Electronics (China)*, 31(1):68–77, 2014
  - Reza Omidi Gosheblagh and Karim Mohammadi. Seu-secure parity prediction multiplier on sram-based fpgas. *Journal of Circuits, Systems and Computers*, 23(06):1450081, 2014
  - Xiuhai Cui, Haigang Yang, Yu Peng, and Xiyuan Peng. Research on the packing algorithm for anti-seu of fpga based on triple modular redundancy and the numbers of fan-outs of the net. *Journal of Electronics (China)*, 31(4):284–289, 2014

  - M. Psarakis, A. Vavousis, C. Bolchini, and A. Miele. Design and implementation of a self-healing processor on sram-based fpgas. In *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 165–170, 2014
  - Shobana. M and Senthil Murugan. S. Reconfigurable data processing using duplex fault tolerance system. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–5, 2015
  - R. Santos, S. Venkataraman, and A. Kumar. Generic scrubbingbased architecture for custom error correction algorithms. In *2015 International Symposium on Rapid System Prototyping (RSP)*, pages 112–118, 2015
  - I. Villalta, U. Bidarte, J. Gomez-Cornejo, J. Lazaro, and C. Cuadrado. Dependability in fpgas, a review. In *2015 Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6, 2015
  - D. Agiakatsikas, N. T. H. Nguyen, Z. Zhao, T. Wu, E. Cetin, O. Diessel, and L. Gong. Reconfiguration control networks for tmr systems with module-based recovery. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 88–91, 2016
  - M. Vavouras and C. S. Bouganis. Area-driven partial reconfiguration for seu mitigation on sram-based fpgas. In *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–6, 2016

- Kastil J., Straka M., Miculka L. and Kotasek Z. Dependability analysis of fault tolerant systems based on partial dynamic reconfiguration implemented into fpga. In *15th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 250–257. IEEE Computer Society, 2012

  - Khaza Anuarul Hoque, Otmane Ait Mohamed, and Yvon Savaria. Formal analysis of SEU mitigation for early dependability and performability analysis of fpga-based space applications. *Journal of Applied Logic*, 21:–, 2017
  - V. Simek and R. Ruzicka. Reconfigurable platform with polymorphic digital gates and partial reconfiguration feature. In *2014 European Modelling Symposium*, pages 501–506, 2014
  - K.A. Hoque, O.A. Mohamed, Y. Savaria, and C. Thibeault. Probabilistic model checking based dal analysis to optimize a

combined tmr-blind-scrubbing mitigation technique for fpga-based aerospace applications. In *Formal Methods and Models for Codesign (MEMOCODE), 2014 Twelfth ACM/IEEE International Conference on*, pages 175–184, 2014

  - B. Navas, J. Oberg, and I. Sander. The upset-fault-observer: A concept for self-healing adaptive fault tolerance. In *Adaptive Hardware and Systems (AHS), 2014 NASA/ESA Conference on*, pages 89–96, 2014
  - Felix Siegle, Tanya Vladimirova, Jorgen Ilstad, and Omar Emam. Mitigation of radiation effects in sram-based fpgas for space applications. *ACM Comput. Surv.*, 47(2):37:1–37:34, 2015
  - F. Siegle, T. Vladimirova, C. Poivey, and O. Emam. Validation of fdir strategy for spaceborne sram-based fpgas using proton radiation testing. In *2015 15th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, pages 1–8, 2015
  - F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam. Availability analysis for satellite data processing systems based on sram fpgas. *IEEE Transactions on Aerospace and Electronic Systems*, 52(3):977–989, 2016
  - L. Sterpone, L. Boragno, and D. M. Codinachs. Analysis of radiation-induced seus on dynamic reconfigurable systems. In *2016 11th International Symposium on Reconfigurable Communicationcentric Systems-on-Chip (ReCoSoC)*, pages 1–6, 2016

- Straka M., Miculka L., Kastil J. and Kotasek Z. Test platform for fault tolerant systems design qualities verification. In *15th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 336–341. IEEE Computer Society, 2012

  - Felix Siegle, Tanya Vladimirova, Jorgen Ilstad, and Omar Emam. Mitigation of radiation effects in sram-based fpgas for space applications. *ACM Comput. Surv.*, 47(2):37:1–37:34, 2015

Curriculum vitae - Ing. Lukáš Mičulka

- Born on December 15th, 1985 in Uherské Hradiště.
- 2001 - 2005: Secondary school (gymnasium) in Uherské Hradiště.
- 2005 - 2008: Bachelor degree programme Information Technology at the Faculty of Information Technology at the Brno University of Technology. The studies finished by state final examination.
- 2008 - 2010: Master degree programme Computer Systems and Networks at the Faculty of Information Technology at the Brno University of Technology. The studies finished by state final examination.
- From 2010: Doctoral degree programme Computer Science and Engineering at the Faculty of Information Technology at the Brno University of Technology. The state doctoral examination passed in 2012.

Activities

- Teaching the tutorials in Peripheral devices course.
- The supervisor of bachelor thesis (4) and master thesis (1).
- The reviewer of bachelor thesis (3) and master thesis (3).

Research projects

- Zvyšování spolehlivost a provozuschopnosti v obvodech SoC, GAČR, GA102/09/1668, 2009-2011, team member
- Matematické a inženýrské metody pro vývoj spolehlivých a bezpečných paralelních a distribuovaných počítačových systémů, GAČR, GD102/09/H042, 2009-2012, team member
- Bezpečné, spolehlivé a adaptivní počítačové systémy, BUT, FIT-S-10-1, 2010, co-solver
- Manufacturable and Dependable Multicore Architectures at Nanoscale, COST, IC1103, 2011-2015, team member
- Metodiky pro návrh systémů odolných proti poruchám do rekonfigurovatelných architektur - vývoj, implementace a verifikace, MŠMT, LD12036, 2012-2015, team member
- Architektury paralelních a vestavěných počítačových systémů, BUT, FIT-S-14-2297, 2014-2016, team member