Journal of Communication and Computer 12 (2015) 219-227
doi: 10.17265/1548-7709/2015.05.001

Increasing of Reliability of FPGA Implemented Microcontroller Using the Error Self Correcting Techniques

Branislav Dobrucky, Peter Sindler, Jozef Cuntala and Anna Kondelova
Department of Mechatronics and Electronics, University of Zilina, Slovak Republic

Abstract: The paper deals with the concept of extension of self-error corrected approach to the microcontroller structure. The goal is to increase the reliability of the entire microcontroller system implemented in FPGA (field programmable gate array) platform. In such systems of increased reliability, memories with parity-bit or ECC (error-correcting code) memory are used. The paper is focused on safety increasing of the ALU (arithmetic and logic unit), its registers for address and code instruction decoding. By similar way reliability of data transfer unit, interfacing units and further peripheral devices implemented in FPGA can be increased. Tiny program for simulation of single- and double-fault in instruction register, also demonstration of transfer data error and simulation of control unit using Xilinx ISE Design Suite are given in the paper.

Key words: Fault tolerant, microcontroller, software security, reliability, system-on-a-chip.

1. Introduction

Programmable logic devices are situated on important place among microelectronic components thanks to the continuing technological development of semiconductor components, due to new architectures and new approaches to digital systems design [1]. Software design of typical complex digital system design flow is depicted in Fig. 1.

The requirement to achieve the highest reliability of the system is usually interpreted as an effort to reduce failure rates. All values of reliability will be improved by reducing the failure rate. Such a method of reliability increasing is referred to as the fault avoidance. The applicability of this method is limited both because of costs of further reducing of failure rate grows to a certain level disproportionately fast, partly because objective physical obstacles get in the way.

If faults cannot be avoided, other possibilities of improving levels of reliability should be searched. The possible occurrence of faults is taken into account already in the design and implementation of the system so that defects will show only minimal.

This type of reaction to the faults is referred to as fault tolerance. A system which reacts in this way is called fault tolerant system.

When assessing the reliability of fault tolerant systems then it has to be distinguished between component fault and system faults. As a system fault is considered to be only a fault of components that cause unacceptable behavior change so that the ability of system as a whole to perform the desired function is terminated. Topic of the contribution relates to the need for increased reliability of control systems. The reliability is increased by microprocessor design and implementation with reliability increasing elements. Integrating of these elements into the system means that the system achieves the high reliability.

It is known that reliability increasing of micro-controllers is realized via massive Redundancy, i.e., by 2-multiply, 3-multiply of the micro-system [3-6] and others. Our approach to reliability increasing,

Corresponding author: Jozef Cuntala, Ph.D., associate professor, research field: design of electronic systems.

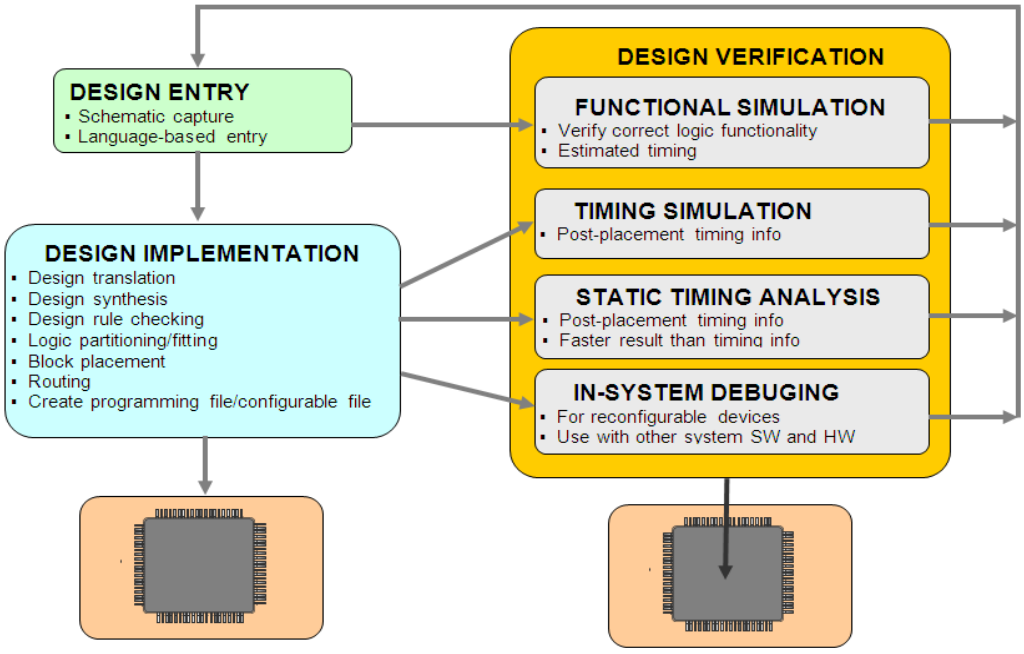

Fig. 1 Typical design flow of PLD (programmable logic devices) [2].

contrary to those, leads to thoroughgoing utilization of self-correcting code for all functional units that are in the microcontroller.

2. Specification of Faults in Electronic Logic Circuit

Condition of an electronic logic circuit is determined by a behavior function, where phenomena which defy the desired behavior occur. In real devices can arise:

- No fault—the device performs all required functions according to the technical conditions.
- Defect (physical fault, failure)—physical imperfection in the logic system caused by improper design, manufacture, or combination of them, and other manufacturing processes that caused eviations from the designed specifications.
- Fault—defined as a phenomenon consisting in the termination of the ability of logical systems to perform a required function according to the technical conditions.
- Error—an in correct response in the behavior of the logic system -a manifestation of a fault/defect.

Faults vary in nature such as:

- Hidden fault—a fault that is hidden and cannot be detected, thus no longer manifests the fault.
- Detectable fault—a fault that leads to a different logical function of a circuit or a system.
- Temporary detectable fault—change in physical parameters, which still does not change the logic function, but the boundaries of given technical parameters are already exceeded.
- Single fault—the presence of only one fault at a given time.
- Multiple faults—the presence of more than one fault at the same time.
- Equivalent faults—faults that manifestation is the same but cannot be distinguished.

The various categories of faults can be divided according to the fault manifestation on:

- Stuck-at;
- Bridging faults, shorts;
- Opens;
- Pattern sensitive faults;
- Parametric faults-one of the most significant parameter is the time – delay faults, timing faults.

Model of permanent faults:

The diagnostics of logic circuits in static mode with a low number of defects for different types of physical faults of semiconductors was enough up to now. Industrial standard of these faults has been created since1959.

It is assumed a fault that causes logical 0 or 1 permanently held on input or output (stuck-at 0, stuck-at 1, sa_0, sa_1, SAF).

3. Fault Models of High Level Logical Systems

Micro processors and microcomputers are highly complex sequential logic systems. To increase the rate of fault detection, for that purpose were created high level functional or behavioral models of faults and fault conditions. Such a model can be considered as good (acceptable) if the test generated on the model basis is revealing the most of “parasitic connection” faults (stuck_at) or other physical defects.

The main idea of the high-level modeling is obtaining of incorrect version of a logical system from its high level description through implementing a fault (fault condition) to this description. This approach is called “[after] disturbance of model.” The model can be disturbed in several ways, for example by modification of micro-operations or changing the truth table. Such methods are often used for microprocessors testing.

High level model maybe explicit, or implicit. An explicit model is able to identify individually each fault. Any fault in this model becomes a target for the generation of a test. In contrast, the implicit model identifies a class of faults with “similar” characteristics, where all faults belonging to one and the same class can be identified by similar procedures. The advantage of the implicit model compared to the explicit one is that there is no need to number explicitly the faults within a single class.

Most of the faults stated here belong to the so-called address faults that model works with n-bit strings that map 2^n address space.

At this point are presented models for different units of data processing section and control section of microprocessors. Faults affecting the microprocessor function can be divided into the following classes:

- Addressing faults affecting the function of address decoding register;
- Addressing faults affecting instruction decoding and sequence of instructions;
- Faults of data memory function;
- Faults of data transmission function;
- Faults of data handling function.

If a fault occurs on the multiplexer, then one of possible events arises for each of the specified source addresses:

- Any address has not been selected (data source);
- An incorrect address has been selected;
- More than one address has been selected and the output of the multiplexer is either the AND function, or OR function of the selected sources, depending on what is the used technology.

For demultiplexers in fault, these possibilities arise for target address:

- Any target has not been selected (address);
- Else, if the correct target has been selected, the other one is selected too.

Addressing faults and sequence of instructions:

The instruction can be seen as a sequence of micro-instructions. Each microinstruction consists of a set of micro-operations, which are performed in parallel. Micro-operations represent elementary data transfer and elementary data operations. Faults in addressing affect the execution of instructions and can create the following fault conditions:

- One or more micro-operations have not been activated (they has not been performed);
- Other group of micro-operations has been activated or redundant micro-operations have been activated.

A very important conclusion implies from this model. It is a real threat to carry out the instruction

which does not belong to the instruction set of the microprocessor.

Faults of data memory function:

Memory functions are always executed in a certain physical memory. These fault conditions may occur within the array of memory cells:

- One or more cells have permanent 0 or 1;
- One or more cells spontaneously over write 0 to 1 or 1 to 0;
- Two cells, or more pairs of cells can be “coupled”, i.e., that, if one cell changes its status, it will change the second cell of the pair without prejudice to be addressed.

Faults of data transmission functions:

These functions include all data transfer over the buses and between registers and the microprocessor units. Fault conditions can be:

- One or more lines may be permanently in the 0 or 1;
- One or more lines can create AND or OR function by reasons of short circuits or faulty connections.

Faults of data handling functions:

These faults generally tend to have very specific (individual) models by reason of very wide range of issues for which they are designed. Therefore, common models are not generated for them, but quite specific. This is perhaps the main shortcoming of the proposed approach, because microprocessors are affected only, but not the logical systems that use them.

4. Main Topic — Our Approach

FPGA programmable devices can also comprise a microprocessor structure [1]. For increasing reliability of the microstructure in such systems the memories with parity-bit or ECC are used.

Besides the safety memories (FLASH program memory, S-RAM data memory) is needful to increase the reliability also of other functional parts of central processing unit (Fig. 2).

It deals namely with following items:

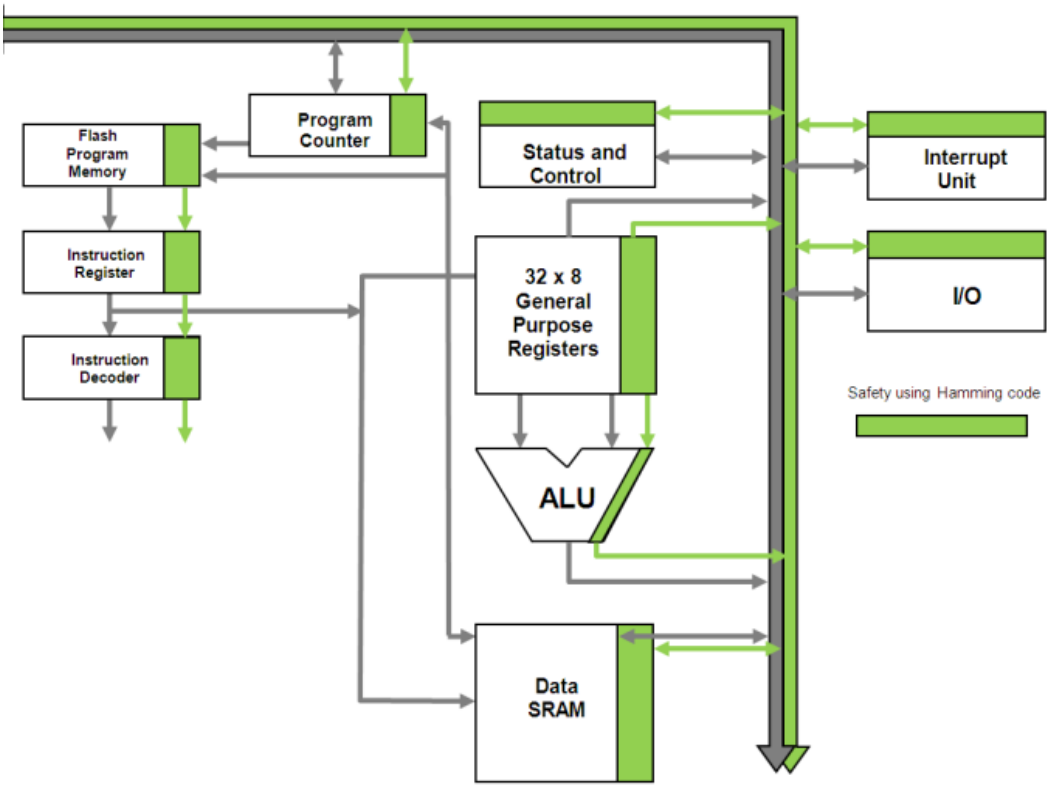

Fig. 2 Block diagram of fault tolerant central processing unit (s-CPU).

- Status and control registers;
- Address and code instruction registers;
- General purpose registers;
- ALU unit;
- Program counter;
- Interrupt unit;
- I/O module;

The reason for which are mentioned measures proposed is to provide safety and highly reliable operation of the micro-structure in FPGA devices. The goal is to eliminate:

- Failures of addressing affecting instruction decoding;
- Failures of function affecting data memory;
- Failures of function affecting data transfer;
- Failures of function affecting data processing;
- Failures of interrupt function;
- Failures of peripheral microprocessor devices.

Besides that the self-correction of single failure with indication or its program processing via interrupt is needful. And the indication of failure occur place is requested.

For illustration of data transfer malfunction the demonstration of error processing is shown in Fig. 3.

Hamming self-corrected code has been chosen for demonstration (with correction of single-error, and indication of double-error).

For easier understanding, there is in the next also shown the short program for the simulation of single and double-error in instruction register as an example:

5. Model Behaviour Simulation for Verification

The verification is very important part of the PLD design and occurs at all its phases. Designer applies more tools to verify the design function. Functional simulation is performed in conjunction with design entry, but before placing and routing, to verify correct logic functionality. Timing simulation is performed after place and route. At that time the software back-annotates the logic and routing delays to component interconnection (called a net list) for simulation. Static timing calculator provides faster results than timing simulation [1, 2].

In earlier days, “breadboards” were constructed to verify the design of VLSI chips. These were printed circuit (or wire-wrap) boards with discrete components that implemented the chip’s function. Today, much of the scaled models and breadboards have been replaced by computer simulations.

Thus, the high complexity of formal methods allows their use only at the higher behavior level. In spite of the incompleteness, simulation provides a better check on the manufacturability of the design. An ideal system of design verification should combine the behavior-level formal verification with the logic and circuit-level simulation.

The process of realizing an electronic system begins with its specification, which describes the input/output electrical behavior (logical, analog, and timing) and

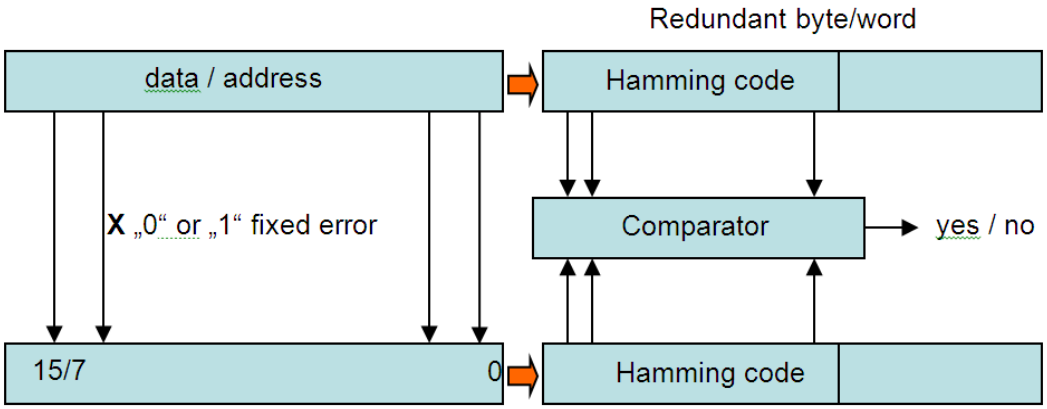

Fig. 3 Demonstration of processing data transfer function error.

```
prog_1: ldi R16, 255
out DDRB, R16
dd: inc R16
out PORTB, R16
rjmpdd
```

Fig. 4 Short program for the simulation of single- and double-error in instruction register signal timing is shown in simulation section.

other characteristics (physical, environmental, etc.) (Fig. 5) [7].

The specification, shown as a shaded block in the figure, is the starting point for the design activity. The process of synthesis produces an above mentioned net list. The design is verified by a true-value simulator. True-value means that the simulator will compute the response for given input stimuli without injecting any faults in the design. The input stimuli are also based on the specification. Typically, these stimuli correspond to those input and output specifications that are either critical or considered risky by the synthesis procedures. A frequently used strategy is to exercise all functions with only safety critical data patterns.

5.1. Configuration Time Model Synthesis

Based on time duration single design stage of configuration (reconfiguration, partial reconfiguration) the general time dependences for configuration time T_{CONF}, reconfiguration time T_{RECONF} and partial reconfiguration time $T_{PARCONF}$ can be defined [1].

If we denote the architecture arch, size circuit type, kind of configuration interface mod, size of reconfigured partition rp and frequency of configuration clock f_{CCLK} as independent variables, we can define following relations:

$$T_{CONF} = T_{BS}(arch, type, mod, f_{CCLK}) \quad (1)$$

$$T_{RECONF} = T_{BS}(arch, rp, mod, f_{CCLK}) \quad (2)$$

$$T_{PARCONF} = T_{BS}(arch, rp, mod, f_{CCLK}) \quad (3)$$

Where, $arch, type, rp, mod, f_{CCLK}$ as mentioned above, T_{BS} is time duration uploading of bit file.

The total time of configuration TTCONF consists of three time phases:

$$T_{TCONF} = T_{INIT}(arch, type) + T_{BS}(arch, type, mod, f_{CCLK}) + T_{SU}(f_{CCLK}) \quad (4)$$

Where, T_{INIT} is time of initialization, T_{SU} time of start-up sequence—circuit activation.

Those general function dependences worked-out based on analyses of configuration (reconfiguration, partial reconfiguration) of single architectures and circuit types were implicated in the model created in Matlab environment. The time analyses and synthesis have been done for pre FPGA architecture of Virtex-4, Spartan-3, Spartan-6, Virtex-5 and Virtex-6.

6. Implementation into Field Programmable Gate Array Xilinx —Virtex4

6.1 Stage#1— Simulation Using VHDL Language

Next two examples show the simulation of the short program using VHDL language [8, 9] without and with self-correction in Xilinx ISE Design Suite environment (Fig. 6a and Fig. 6b).

Fig. 7a and Fig. 6b are viewed the fragments of simulation of computation in instruction register for the case machine computing of short programme in

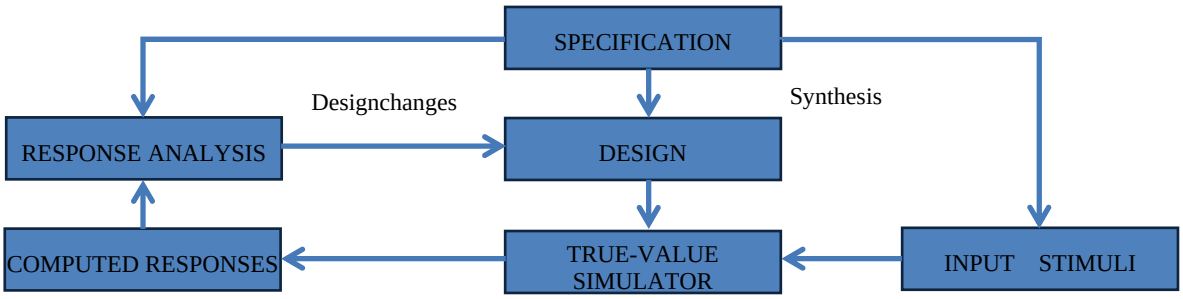

Fig. 5 Simulation for design verification [7].

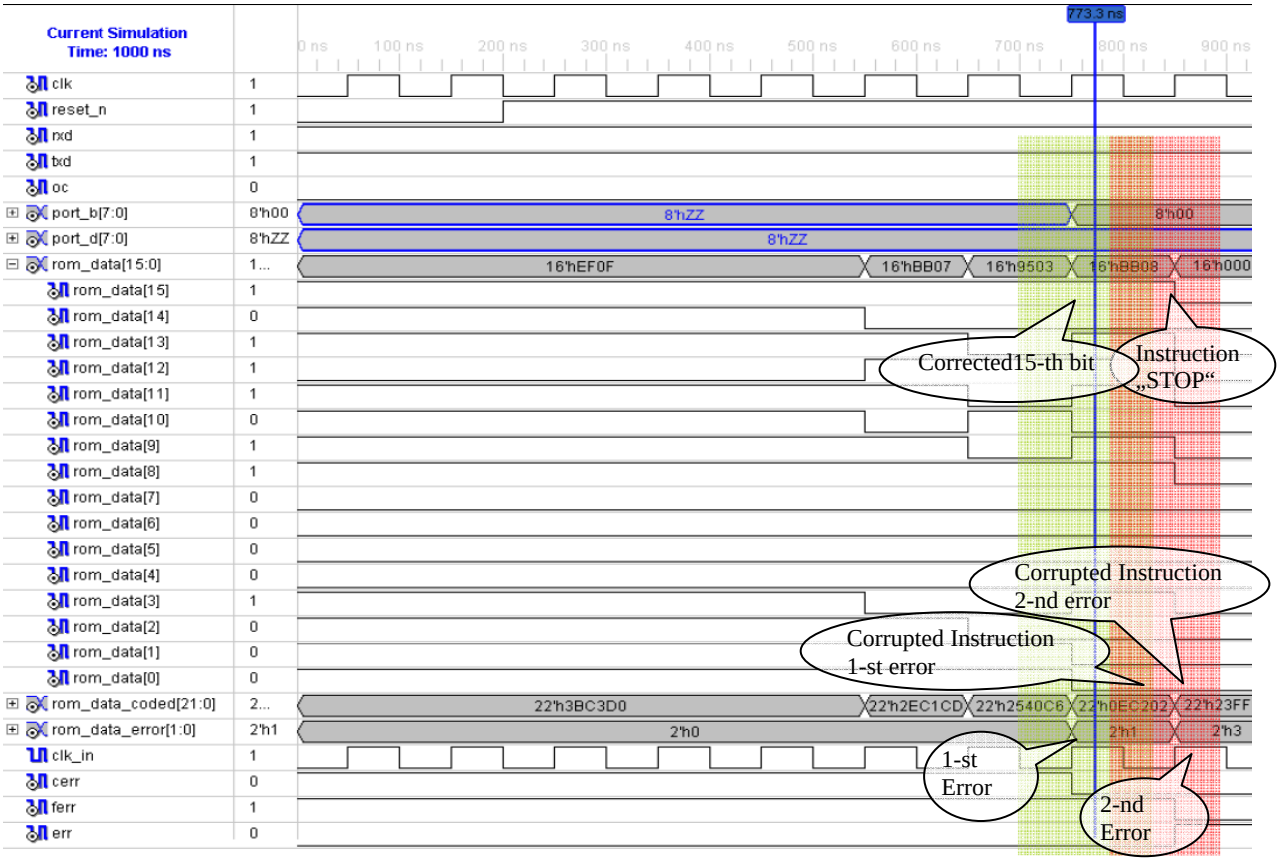

Fig. 6a Simulation of the short program running in the instruction register with self-correction.

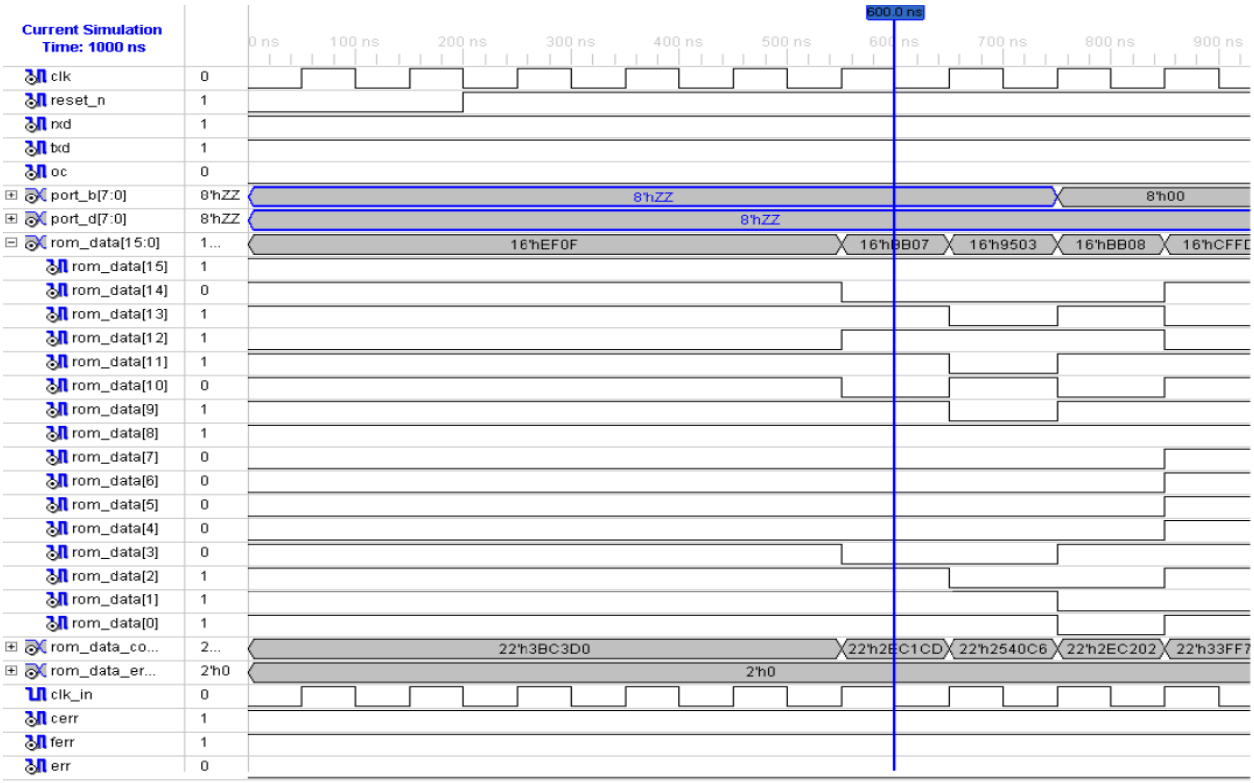

Fig. 6b Simulation of the short program running in the instruction register without self-correcting.

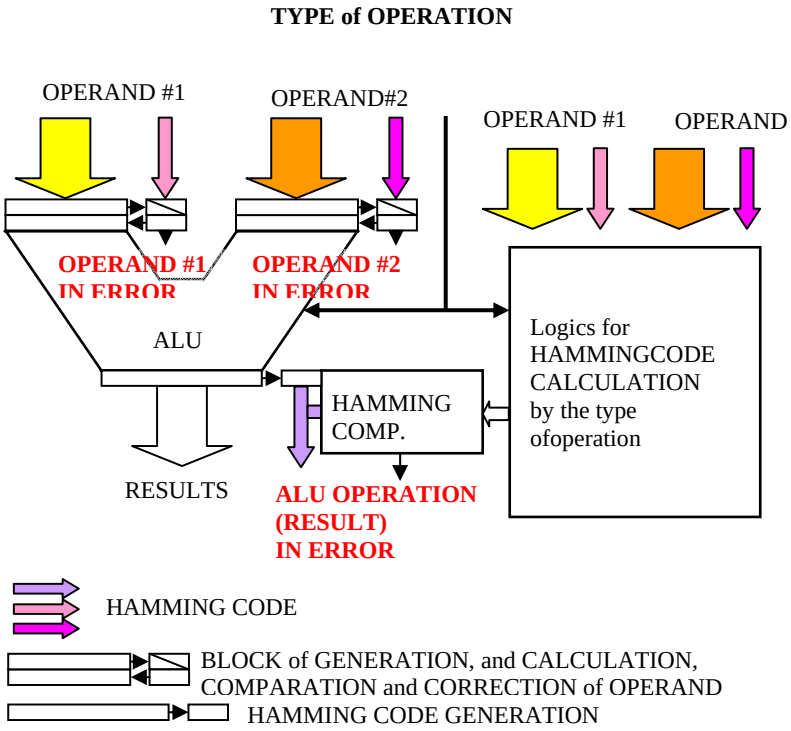

Fig. 7 Block diagram of fault tolerant control processor ALU unit.

Fig. 4 without errors (Fig. 6a) and with one error resp. two errors on the Fig. 6b.

The cerr and ferr are output logical variables of logical circuits, which detects one error and/or two error computed machine code instruction.

6.2 Stage #2—Implementation

Further logical step after simulation is implementation. Nowadays, we are actually working on 8-bit ALU unit safety increasing using Hamming code (in VHDL) (Fig. 7).

Mode of operation of this circuit:

(1) All operands are checked [even corrected] for their correctness at the input position. When error [recoverable or not] occurred it is signaled via appropriate interrupt;

(2) When no error occurred (or was successfully corrected) ALU continues processing;

(3) At the same time all needed operands enter an additional unit which predicts appropriate redundant bits using different algorithm than ALU operates. The initial processing is, in general, the same that works at the ALU inputs;

(4) An additional ALU unit computes the correct redundant bits for the result (of operation performed];

(5) Both these redundancy bit group are compared. If they didn't pass then an appropriate signal is generated and processed. There can be more different ways or result post processing defined according to the level of reliability required, e.g. the result may be accepted, conditionally accepted or rejected.

Testing methods needed for VLSI and microprocessor structure validation are widely documented in works [5, 10-17].

7. Conclusions

The method for on-line error detection and correction of all fundamental parts of high-performance micro-controller using Hamming code with two errors and one error correction is described in the paper. The basic concept of that approach is to recover the execution errors promptly for each instruction cycle.

Main advantage of proposed solution is to use the

same frame of security resistance and robustness against logical failure that makes the design easier and more formally transparent.

Some parts of the system have been verified using one-and two-bits failures injection. The FPGA VIRTEX 4 XILINX as well as XILINX ISE Design Suite environments were used for implementation.

Acknowledgments

The authors wish to thank for the financial support to Slovak Grant Agency VEGA by the grant No. 1/0928/15and R&D operational program Centre of excellence of power electronics systems and materials for their components No. OPVaV-2008/2.1/01-SORO, ITMS 26220120046 funded by ERDF (European regional development fund).

References

[1] Kondelova, A. 2013. “Synthesis of Reconfigurable Numeric Control Systems and Their Implementation into Complex Integrated Circuits (in Slovak).” Ph.D. thesis, Faculty of Electrical Engineering, University of Zilina.

[2] Cuntala, J., Sindler, P., and Kondelova, A. 2004. “Progress in Programmable Logic.” *Advances in Electrical and Electronic Engineering* 3: 219-22.

[3] Yilmaz, M., Meixner, A., Ozev, S., and Sorin, D. J. 2007. “Lazy Error Detection for Microprocessor Functional Units.” In *Proceedings of the IEEE Int’l Symp. on Defect and Fault Tolerance in VLSI Systems*.

[4] Holmberg, P. 2004. “Domain-Specific Platform FPGA.” *FPGA and Programmable Logic Journal* 2 (Jan.) 103-15.

[5] Bolchini, C., Salice, F., and Sciuto, D. 2002. “Designing Self-checking FPGAs through Error Detection Codes.” Presented at the 17th IEEE Int’l Symposium on Defect and Fault Tolerance in VLSI Systems (DFT’02), Canada.

[6] Spanik, P., Hock, O., and Frivaldsky, M. 2012. “Utilization of the New Trends of FPGA Circuit Simulations in Power Converter Applications.” In *Proceedings of 2011 Int’l Conf. on Applied Electronics, Pilsen (CZ)*, 279-82.

[7] Baleani, M., Ferrari, A., Mangeruca, L., Sangiovanni, A., Vincentelli, Peri, M., and Pezzini, S. 2003. “Fault-Tolerant Platforms for Automotive Safety-Critical Applications.” In *Proceedings of CASES’03 Int’l Conf. on Compilers, Architectures and Synthesis for Embedded Systems, San Jose (CA, USA).*

[8] Maliniak, D. 2003. “Tool up for Alternative Standard ASICs”. *Electronic Design* (Sep.).

[9] Skahill, K. 1996. *VHDL for Programmable Logic*. Menlo Park: Addison-Wesley Publishing.

[10] Bushnell, M. L., and Agrawal, V. D. 2000. *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*. Kluwer Academic Publisher.

[11] Novak, O., and Gramatova, E. 2005. *Handbook of Testing Electronic Systems (in Czech)*. Prague: CVUT Publishing Centre.

[12] Yarmolik, V. N. 1990. *Fault Diagnosis of Digital Circuits*. New York: John Wiley & Sons.

[13] Wilkins, B. R. 1990. *Testing Digital Circuits —An Introduction*. London (UK): Chapman and Hall.

[14] Hlavicka, J. 1982. *Diagnostics of Digital Electronic Circuits (in Czech)*. Prague (CZ): SNTL/ALFA Publisher.

[15] Eidukas, D., and Stupak, V. 2011. “Efficiency of Electronic Devices for Unsteady Flow of Failures.” *Elektronika Ir Elektrotechnika* 110: 57-60.

[16] XAPP 151 (V1.5), Virtex Series Configuration Architecture User Guide. XAPP 151.

[17] Saggese, G. P. 2005. “Microprocessor Sensitivity to Failures: Control vs. Execution and Combi-national vs. Sequential Logic.” Presented at the Intl. Conf. on Dependable Systems and Networks.