Implementation of decision-making algorithms in redundant systems on FPGA

Mikhail V. Saramud [1,2], Vasiliy V. Losev [1] and Angelina E. Petetskaya [1]

[1]*Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russian Federation*
[2]*Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russian Federation*

Abstract

The paper describes the problem of creating fault-tolerant real-time control systems. The authors describe methods for improving the reliability of control systems for both hardware and software. The features of the system operation in real time are presented. The paper considers various versions of hardware platforms for real-time operation are such as CPU applying RTOS, FPGA, ASIC. The peculiarities of applying FPGA and developing software for it are outlined. The main blocks that make up the FPGA are represented. The hardware platform for prototype implementation is proposed. The authors opt to follow myRIO-1900 based on SoC Xilinx Zynq-7010 as it contains FPGA, CPU with RTOS implementation for it. A new approach is proposed. It combines the implementation of a decision-making block in control systems by means of a real-time operating system running on a CPU and a special FPGA that implements watchdogs and decision-making algorithms in multiversion systems. Watchdog on the FPGA monitors the main decision block in the RTOS. It should provide a response within a specified period of time. If this does not happen, it starts a spare decision block implemented on the FPGA. It is guaranteed to make a decision in one clock cycle of the FPGA. This approach will improve the overall reliability of fault-tolerant control systems without significantly increasing their cost.

Keywords

Software reliability, decision-making, FPGA, multiversion software

1. Introduction

Nowadays, the technology fields are actively developing. They require high reliability and compliance with strict time constraints on the response time (real-time operation). These areas include autonomous vehicles, from autonomous unmanned aerial vehicles to consumer vehicles. Their partial or full autopilot technologies have already been used in mainstream models. Automation and robotization of some industries, the transition to cloud production are also actively going on. In such systems, the reliability of control systems is extremely important, since the cost of a software failure is very high. The implications in production can be restricted (the authors do not take into account the hazardous production with the use of radioactive or toxic substances). However, failures of the vehicle with autopilot function on public roads can lead to tragic consequences.

The approaches that apply redundancy are used to create control systems for such tasks. The increase in cost is insignificant compared to the possible negative consequences of failures. In the case of hardware, duplication, tripling, etc. are used. Such an approach works for hardware [1]. This also applies to the application of FPGA [2]. In the case of software, it makes no sense simply to copy

Proceedings of MIP Computing-V 2022: V International Scientific Workshop on Modeling, Information Processing and Computing, January 25, 2022, Krasnoyarsk, Russia
EMAIL: msaramud@gmail.com (Mikhail Saramud); basilos@mail.ru (Vasiliy Losev); angelina02021999@gmail.com (Angelina Petetskaya)
ORCID: 0000-0003-0344-9842 (Mikhail Saramud);0000-0002-1996-2889 (Vasiliy Losev)

programs or their modules. Also, we may copy all errors of all kinds contained in it. So, multiversion programming is used for this problem solving [3].

The functionally equivalent modules are created in developing versions of software modules for a multiversion system. They are modules that perform the same task but are implemented in the most different ways. Ideally, they are performed by different designers, different development tools, in different languages, etc. Such an approach is necessary to eliminate the most dangerous type of errors, i.e., related faults. They are errors that lead to several versions of an incorrect but identical value in the output. The decision-making process concerning the correct answer selection becomes extremely difficult with related faults. The authors proposed to apply "weights" of versions to solve this problem. A type of version depends on the quality of its work and how it is involved in the calculation of weights class when determining the correct output of the system [4].

2. Operation in real time

A response from a subsystem (for example, the appearance of a value at the output) can lead to no less sad consequences than failures and failures for such fault-tolerant systems. Processes that occur both in the controlled system itself and around it proceed continuously. So, it is necessary to respond in a timely manner to the events taking place. Compliance with time delays is an extremely important characteristic for production and autonomous unmanned systems.

How can these delays in the design of control systems be guaranteed? There exist two main ways. The first one is to apply a platform with a general-purpose CPU and special software, i.e., a real-time operating system (RTOS). The second one is to apply specialized solutions on such integrated circuits and FPGA and ASIC.

Consider the advantages and disadvantages of each solution.

The use of a general-purpose processor makes the system more universal and flexible. It is quite simple and quick to develop software in high-level languages for it. Through this approach the authors suppose to apply the existing redundant methods to increase software reliability, including multiversion programming. There exist some compilers and development tools for various programming languages for most hardware platforms (x86, ARM, RISV-V) and RTOS (FreeRTOS [5], LinuxRT [6]. It will implement the necessary diversification in a full way when developing versions of multiversion software. Built-in RTOS tools help provide the necessary delays, prioritize various tasks and ensure secure interaction among them. It also makes development faster and easier.

A significant advantage is the wide distribution of ready-made solutions, not only of the processors themselves, but also of ready-made boards containing memory and all the necessary peripherals. Such solutions today have a low cost. They are available with rather high performance. Another significant advantage is the ability to apply ready-made libraries and solutions, as for example the OpenCV library [7] for machine vision tasks.

The disadvantages are the higher power consumption of the general-purpose CPU and the fact that the RTOS does not guarantee the execution of particular operations for the given period of time, but only monitors it. Moreover, in this approach, a software failure in one of the modules can affect the entire system and lead to its failure, or lead to a lack of response of the system for a certain time. It amounts to a failure for the given period of time.

What is FPGA? FPGA is based on a set of configurable logic blocks Configurable Logic Block (CLB). This is the main element. It can perform one of the basic logic functions or store calculation results in registers (triggers). The function performed by the CLB depends on its commutation, which is set by the cells of the configuration memory Static RAM. In addition to The CLB FPGA also contains such memory blocks as random-access memory (BRAM) and Digital Signal Processor (DSP) blocks. DSP blocks are capable of multiplying and adding 18-bit numbers every clock cycle.

The volume of the circuit depends on the number of CLB blocks in the FPGA, i.e., the complexity of the "program" that it is capable of executing. The number of DSP blocks and BRAM memory blocks also affects the possibility of implementing a particular scheme.

The FPGA development is quite laborious. However, new tools make the development process much easier. There are several types of FPGA development, for example, low level using specialized Verilog or VHDL languages that operate at the register transfer level (RTL). There exist compilers or translators

with high-level languages like C and C ++ to the RTL level like Vivado HLS. There are also development tools such as LabVIEW from National Instruments for developing in the graphical "G" programming language. There is a LabVIEW FPGA module that helps make developments for FPGA both in the graphical "G" language and in the Hardware Description Language (HDL).

A specific peculiarity is the fact that the FPGA structure help implement a deep neural network circuit that will operate at the clock frequency of the board. This can be extremely useful for tasks such as pattern recognition and other classification tasks.

FPGAs are significantly more energy efficient than CPUs, second only to ASICs. However, it is quite expensive to produce ASICs in single copies. It makes sense either for mass production or with very strict requirements for power consumption. The ASIC scheme tested on the FPGA can be ordered for mass production.

FPGAs retain the advantage of being configurable compared to ASICs.

As a result, the authors can conclude that the FPGA platform is the best fit for the tasks at hand; it can be applied, in particular, to create flight controllers for drones.

3. Development on FPGA

It is necessary to understand how the development is carried out since the authors are focusing on FPGA in our choice. In fact, a developer creates, not programs the algorithm, because it is not the program code on the central processor that runs, but the logic circuit in the FPGA. A developer creates the FPGA firmware. It is the switching of the necessary blocks that performs all the operations.

Nowadays there are two main manufacturers of FPGAs; they are Xilinx [8] and Intel [9]. The authors consider the solution of Xilinx, since it is a system on a chip (SoC) that includes both an ARM-based CPU and the FPGA itself.

When developing a prototype that implements the proposed approach, it is planned to apply the National Instruments myRIO-1900 controller [10] based on SoC Xilinx Zynq-7010 [11]. This SoC contains a CPU with two CortexA9 cores running at 667MHz and FPGA is operates on 28,000-cell. The peculiarity of this platform is also the ability to develop in the LabVIEW environment. Moreover, the software part will be executed within NI Linux. It can be developed both in the graphical "G" language, and using code inserts of more traditional programming languages within LabVIEW. Also, plug-in libraries can be applied. They can be developed in any programming language applying any development tool. The main point is that it is possible to compile the library for the platform we need, namely Linux ARM.

4. Combined approach

Taking into account the information described earlier, the authors conclude that each platform has its own advantages and disadvantages. The development of classic software for execution on general-purpose CPUs is much faster and cheaper. A lot of ready-made solutions, templates, libraries are used while the development. However, it is impossible to guarantee 100% neither the absence of possible errors inherent in the developed software, nor the exact response time of the system, even when using RTOS. FPGAs, in turn, are devoid of this drawback, since they are guaranteed to execute the entire logic circuit in a clock cycle and have a stable response time.

The proposed approach is to implement the bulk of the algorithmically complex software in traditional ways, but applying a multiversion approach. The traditional scheme of RTOS operation with the ability to run multi-version software is presented in Figure 1:.

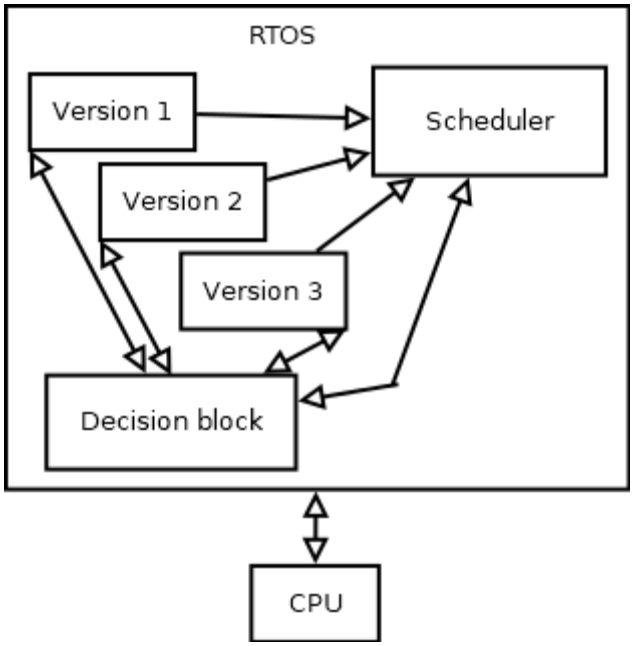

Figure 1: Schematic diagram of the multiversion runtime in RTOS

In this system, a decision block has a great influence on the overall reliability of the system. It is responsible for what value will be recognized as correct and will go to the output of the system, as well as for what period of time this will happen. It is necessary to apply the most stable decision-making algorithms in the decision-making block. Also, it is necessary to implement a mechanism that will select a correct answer in the given period of time, even if one or several versions are "late". The decision block is also the place for problems with time delays, since it is influenced by both the time it takes for versions to provide a response and how correctly the operating system's scheduler will work. A scheduler needs to close versions that did not complete their work in the allotted time and transfer control to the decision block so that it can select the correct version from the response pool. However, errors and malfunctions are possible in the work of the decision-making unit itself. The situations are possible when, for a certain set of input data, the decision-making algorithms embedded in the block fail. Other types of errors are also possible, i.e., problems with version data with memory used for the operation of the block, etc.

In order to avoid these problems and "insure" a decision block, implemented as an RTOS stream; a watchdog timer and its own decision block are implemented on the FPGA. The timer continually checks whether the main decision-making unit has provided the required data on time. Suppose it is necessary to provide a system response time of no more than 200ms for a certain operation. 1 clock cycle of FPGA operation takes 20ms. Therefore, if the watchdog timer detects a problem at this clock cycle, the next time it will be possible to start a duplicate decision block on the FPGA itself. In this example, we can set a watchdog timer to 150ms, if after this time the main decision block has not provided the required data, then on the next clock cycle we start the decision block on the FPGA. Thus, it will take no more than 40ms (2 FPGA clock cycles). In this scheme, we guarantee a response time of no more than 190ms. And during normal operation of the system, the main decision block is triggered earlier than 150ms and the watchdog timer does not work. The proposed scheme of work is presented in Figure 2:.

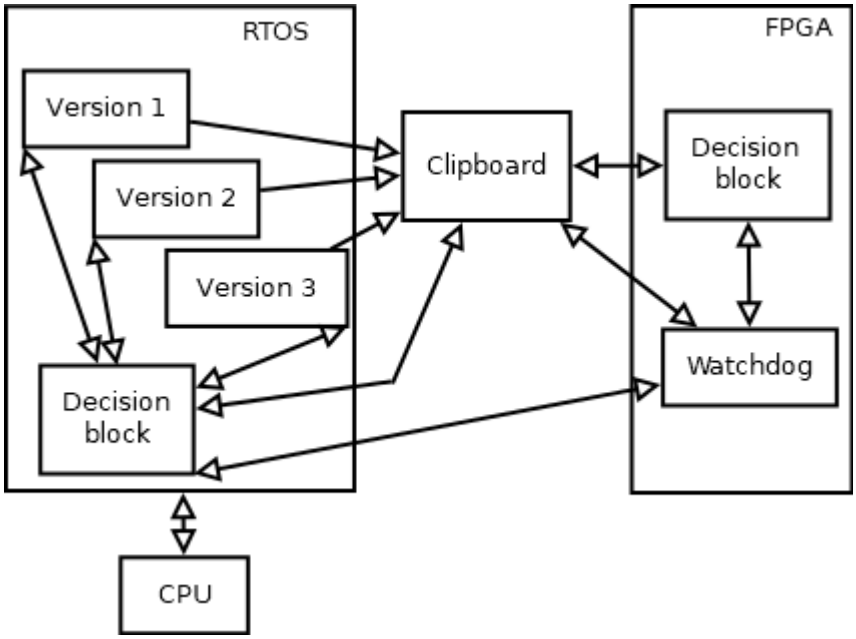

Figure 2: Schematic diagram of the combined approach

The presented diagram reflects the principle of the proposed approach. Versions in the course of their operation place data on the clipboard. Both RTOS and FPGA have access to it. The main decision-making block also places the results of its operation on the clipboard. The main decision-making block also places the results of its operation on the clipboard. When a task arrives, the RTOS scheduler sets a watchdog timer and does the rest of the normal operation. It puts the input data for the version into the appropriate queues, runs them, runs a decision block, etc. A timer monitors whether one of two events has occurred. It also monitors whether the main decision-making block has correctly completed its work on time, or the time has been over and the main decision-making block for some reason did not have time to finish its operation. In the first case, a timer is removed, in the second case it runs a spare decision block implemented on the FPGA.

5. Conclusion

So far, multiversion programming is one of the most effective approaches to increasing the software reliability. One of the places where failures and delays can occur is a decision block required for this approach to operate. A new approach has been proposed for this problem solving. It supposes the combined implementation of a decision block in control systems by means of a real-time operating system running on a CPU and a special FPGA that implements watchdog and decision-making algorithms in multiversion systems. The watchdog on the FPGA makes sure that the main decision-making unit in the RTOS provides an answer within a given time frame. If this does not happen, it runs a spare decision block implemented on the FPGA. It is guaranteed to make a decision in one clock cycle of the FPGA operation.

The proposed approach helps guarantee the specified response time of the system, due to the peculiarities of the FPGA operation. It is important to note that decision-making algorithms can differ significantly in the main block and in the FPGA. The most advanced algorithms should be implemented in the main block. More simple decision-making algorithms can be implemented on the FPGA. The algorithms could be executed in one cycle of its operation. Moreover, various algorithms applied in decision blocks increase the reliability further since the possibility of duplication of an algorithmic error inherent in one of the algorithms is excluded.

6. Acknowledgements

The paper was prepared with the financial support of a grant from the President of the Russian Federation (075-15-2021-060).

7. References

[1] U. Afzaal, J. -A. Lee, "Low-cost Hardware Redundancy for Fault-mitigation in Power-constrained IoT Systems," 2020 International Conference on Information and Communication Technology Convergence (ICTC), 2020, pp. 60-62, doi: 10.1109/ICTC49870.2020.9289420.P. S. Abril, R. Plant, The patent holder's dilemma: Buy, sell, or troll? Communications of the ACM 50 (2007) 36-44. doi:10.1145/1188913.1188915.

[2] P. Mallavarapu, H. N. Upadhyay, G. Rajkumar and V. Elamaran, "Fault-tolerant digital filters on FPGA using hardware redundancy techniques," 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), 2017, pp. 256-259, doi: 10.1109/ICECA.2017.8212811.

[3] Eckhardt, Dave E., Larry D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors." IEEE Transactions on software engineering 12 (1985) 1511-1517.

[4] Igor V. Kovalev, Mikhail V. Saramud, Vasiliy V. Losev, «Simulation environment for the choice of the decision making algorithm in multi-version real-time system» Information and Software Technology 120 (2020). doi:10.1016/j.infsof.2019.106245.

[5] https://www.freertos.org/.

[6] https://rt.wiki.kernel.org/index.php/Main_Page.

[7] https://opencv.org/.

[8] https://www.xilinx.com/products/silicon-devices/fpga.html.

[9] https://www.intel.com/content/www/us/en/products/details/fpga.html.

[10] http://www.ni.com/myrio.

[11] https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.