UDC 004.3

V. KHARCHENKO, J. PROKHOROVA, S. OSTROUMOV, V. KULANOV

National Aerospace University Named After M.E. Zhukovsky, Ukraine

FAULT-TOLERANT SOPC-BASED APPROACHES WITH MULTI-VERSION IIP

In this paper different approaches of Infrastructure Intellectual Property (IIP) implementation for System-on-Programmable-Chip (SoPC) are discussed. Several diversity-oriented SoPC approaches and different techniques of checking and reconfiguration for fault-tolerant SoPC FPGA-based projects are proposed. It is described features of two-version IIP development for application, in particular Ice Protection System (IPS).

Fault-tolerance, Infrastructure IP (IIP), Multiversity, System on Programmable Chip (SoPC)

Introduction

Modern semiconductor technology allows to create complete systems in one chip, in particularly SoPC (system-on-programmable-chip). The SoPC technology possibilities grow extremely fast, introducing more and more sophisticated applications, especially for advanced data communications and wireless products. This evolution leads to complex problems in terms of design and more specifically in terms of manufacturing test. To achieve SoPC with high performances, advanced processes technology is used. But the actual process technology is becoming more fault-intolerant, which may slow down yield reliability [1].

To overcome this limitation, design as well as process constraints must be taken into account in the early phases of development. To achieve this close relation between design and process in order to optimize yield, the semiconductor industry has adopted a solution based on embedding a special type of blocks fulfilled different macro functions in a chip [2]. These blocks are called Intellectual Property (IP) cores and their integration is called Infrastructure IP (IIP) [3, 4]. The last phase of implementing IIP is the integration FPGA chip.

The use of IP cores and IIP provides high-performance, high reliability, low power, smaller weight and dimension, and run-time flexibility. [5]. It is important for aerospace systems (especially for central control systems [6]), business-critical systems and others applications.

Many applications require processor unit (soft-processor), which is used for control of different functional IP-cores versions and as a handler of tasks [7]. Soft-processor IP-core represents as a computer-based architecture, which can be used for the processing of complicated equations.

The ability of different IP-cores implementation enables increasing system fault-tolerance using a few variants or extension functionality as well as performance of SoPC [8]. Thus, there are some possible variants for fault-tolerance support: a) the reservation of IP-cores, b) adaptive fault-tolerant architecture for IP-cores, c) multi-version IP-cores development. Also, multi-version technique is used for detecting and tolerating design faults which can arise onstream. Especially, it concerns safety-critical applications such as NPP I&C (Nuclear power plant informational and control systems), for which diversity requirements are part of standards [9, 10].

High reliability achievement has been possible because of different fault tolerance methodologies including diversity approach [5]. Diversity approach has been required by requirements specification in system under consideration. The using version redundancy for real-time systems with rigorous requirements in reliability is one of the most important methods of common mode failure (CMF) risk reduction. A system or system part which consists of two or more IP cores versions is called multi-version IIP (MIIP) [8].

Therefore, the purpose of the paper is the development and implementation of fault-tolerance multi-version SoPC decisions by using IIP and soft-processor technologies. It is described the checking and reconfiguration technique for MIIP-based decisions of SoPC and elements of IPS design.

1. Multi-version IIP Approach

1.1. Architecture

As it is described above most applications require processing unit.

Therefore, there are a few IP-cores in such systems: soft-processor core and additional core which can serve as functional IP-core or supporting reliability core (fig. 1) [8].

The first variant (fig. 1, a) has one soft-processor core which is used as microprocessor and as switching control unit between additional IP-cores.

The second variant (fig. 1, b) has similar structure as the first one, but there FPGA chip includes a few soft-processor cores, which can be used for increase system performance, and one additional IP-core, which can be used for additional functionality.

The variant (Fig. 1, c) has a few soft-processors and a few additional IP-cores.

This one is the most complex, but has all possibilities to support reliability and additional functions simultaneously.

1.2. Checking and Reconfiguration Techniques

The system with MIIP includes checking and reconfiguration block to provide detection and tolerating of different faults.

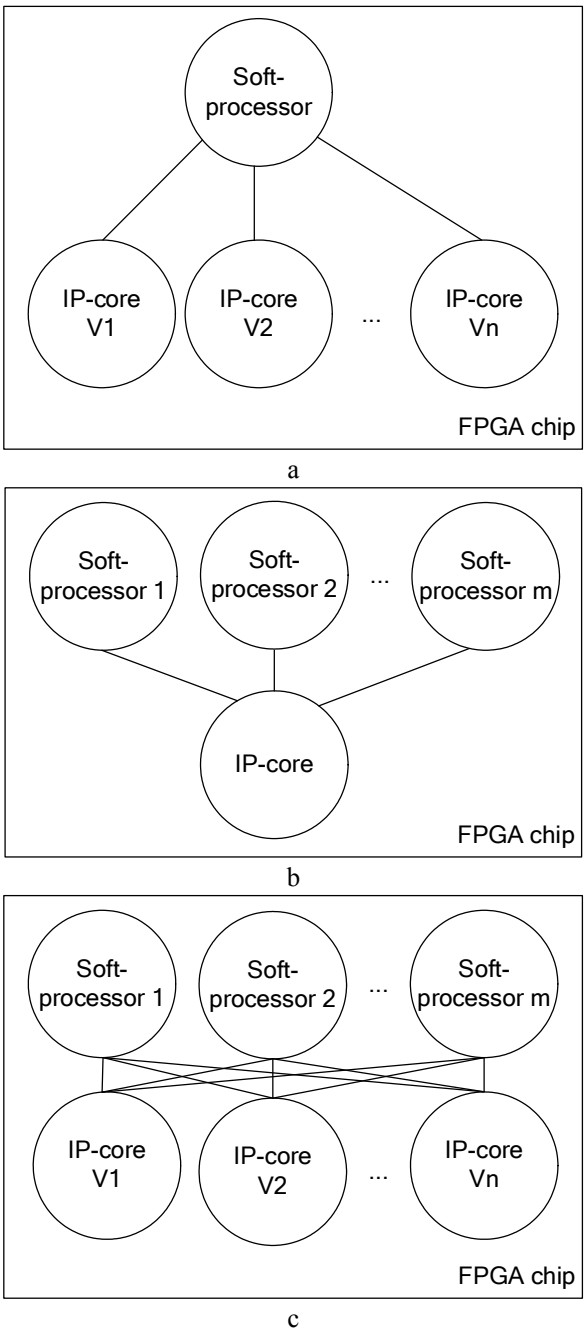

Fig. 1. SoPC architecture variants

The several architectures are represented bellow:

1) *double-channel system* where both versions of project and their checking means are embedded in one chip, it is shown in Fig. 2, where V1 – first version,

V2 – second version, CRB – checking and reconfiguration block;

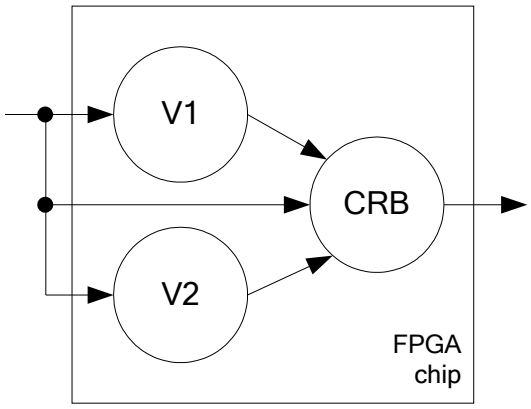

Fig. 2. Two-version system with IP cores embedded in one FPGA

2) *double-channel system* where either of the two diverse projects is embedded in separate chip and checking and reconfiguration scheme is distributed between two chips (fig. 3). This structure is proposed in [5];

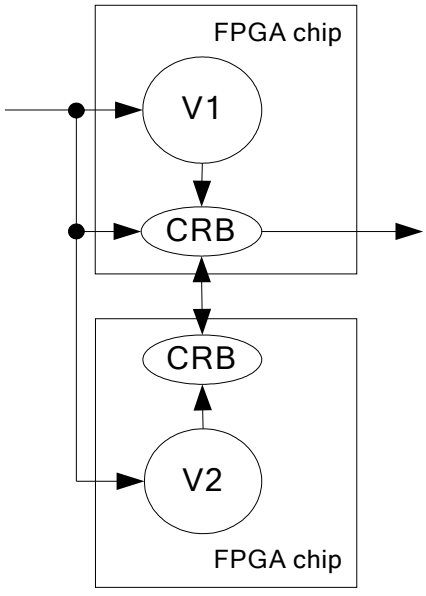

Fig. 3. Two-version system with distributed architecture

3) *four-channel system* which consists of channels allocated in two subsystems including first and second versions in each subsystem. Either of the two subsystems is embedded in a single chip. CRB is embedded in a separate chip too (fig. 4). Functional models and schemas of multi-channel computer systems reconfiguration means [11] can be used for CRB design.

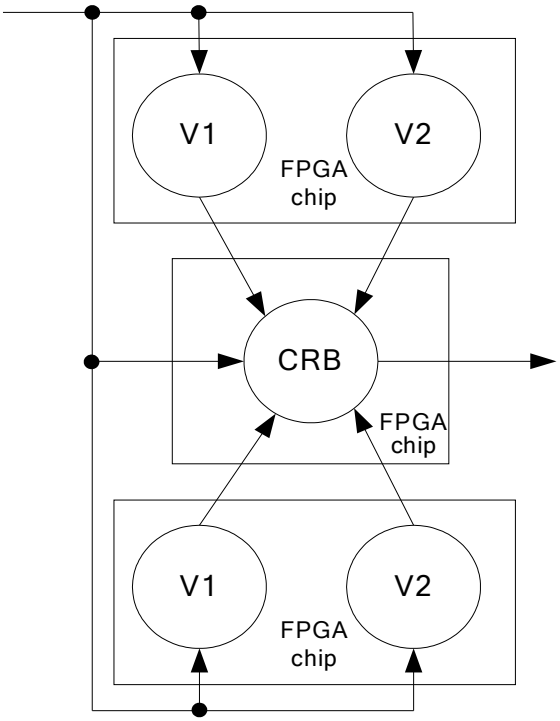

Fig. 4. Two-version system with four channels

Besides, it is possible designing majority three-version IIP (based on one- or three-chip realization).

The application of MIIP allows to decrease probability of CMF and reliability as a whole. Probabilities of up state for one-version and two-version two-channel IIP (fig. 2) are calculated according to following formula:

$$P_1 = (2p_p - p_p^2)p_d p_{c1},$$

$$P_2 = (2p_p p_d - p_p^2 p_d^2)p_{c2},$$

where $p_p = 1 - q_p$, $p_d = 1 - q_d$, q_p, q_d – probabilities of version failures because of physical and design faults; p_{c1}, p_{c2} – probabilities of checking and reconfiguration means up states for one- and two-version structures respectively.

Increasing reliability due MIIP using can be estimated by index

$$\delta P = P_2 / P_1 .$$

If $p_{c1} = p_{c2}$, then

$$\delta P = \frac{2 - p_p p_d}{2 - p_p} .$$

As MIIP-based system is designed from high-reliable components, probabilities of components down states can sum up with inessential inaccuracy. Therefore,

$$q_1 \approx q_{c1} + q_p^2 + q_d ; \quad q_2 \approx q_{c2} + (q_p + q_d)^2 ;$$

$$\Delta q \approx q_d - q_d (2q_p - q_d) .$$

If $q_p = q_d$ then

$$\Delta q \approx q_d - q_d^2 .$$

Thus, two-version structure allows essentially to decrease probability of down state of IIP-decision.

1.3. IP Cores Synchronization

Reconfiguration technique takes into account features of different version synchronization and ability of embedded checking means. SoPC operate with multiple asynchronous clocks at very high frequencies. SoPC systems have multiple interfaces, some using standards with very different clock frequencies. IP blocks consisting of SoPC can operate with both the one clock signal and independent operating frequency. Therefore SoPC design contemplates developing of synchronization subsystem. It is provide high performance of SoPC and absence of faults in interaction process [12].

The Checking and reconfiguration block should receive synchronous data from IP cores.

The technique of synchronization is following. When CRB receives data from first IP core (one version) and doesn't receive data from second IP core (other version), this block provides latency as long as data from the second IP core will be received into CRB similar as first input data from the second IP core. After the data from both IP cores arrived to CRB checking and reconfiguration are carried out.

2. Implementation of MIIP

2.1. SOPC-based IPS

The main purpose of IPS is the heaters control. These heaters are located on the empennage, on the wing and on propellers (fig. 5).

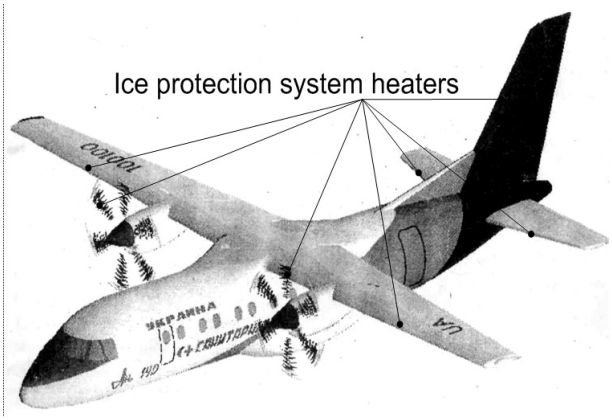

Fig. 5. IPS heaters

IPS consists of few parts. They are: heaters and its control block. The control block has the next constituents: input logic, calculation block (soft-processor) and switching device. Input logic and calculation block are developed on the one FPGA chip. Switching device is a powerful switch for commutation nearly thirty-forty amperes for heaters. Thus, it is an additional block.

Input logic receives and handles discrete signals. These signals influence heaters' on time period and power up time.

Soft-processor core is needed for the heater status tracing because it is necessary to measure voltage and current then to evaluate heater resistance and to sum this information as diagnostic signals. Interval of heater resistance is known thus, it is possible to define a short circuit or an interruption.

According to the description above it is possible next variant (fig. 6) which consists of a few versions of functional IP-cores and a few versions of soft-processor-cores and CRB.

There are two versions of soft-processor-core and functional IP-core which can be implemented by using a few hardware definition languages (HDL). According to IPS, VHDL is used for one channel, JHDL (Java HDL) [13-15] – for second channel, and schematic design project – for the simplest logic and for CRB. Quartus-II is used for VHDL design version and JHDL CAD

Tools – for JHDL design implementation.

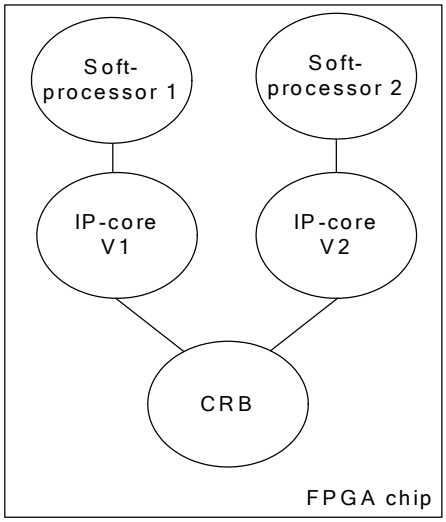

Fig. 6. IPS variants based on SoPC decisions

All these languages can be imported to any modern CAD Tools (e.g. Quartus II, Xilinx ISE, etc.).

2.2. JHDL Version Implementation

To develop second IPS version implementation JHDL CAD Tools are used. JHDL is a set of FPGA CAD tools developed at BYU that allows the user to design the structure and layout of a circuit, debug the circuit in simulation, net list and interface with back-end tools for synthesis [14]. One of the main advantages of using JHDL is Java-based object-oriented design approach [15].

IPS development process of JHDL version consists of the following stages:

1) project specification;

2) general (coarse-grain) structural description of the system (system behavioral model);

3) project fragmentation;

4) structural and schematic system blocks implementation (fine-grain approach - block behavioral models);

5) Java-class implementation of each system blocks taken separately (*.java, *.class);

6) Java IPS implementation (*.jar);

7) project verification;

8) project deployment.

Proposed technique feature of JHDL-version implementation is behavioral model usage.

System behavioral model allows tracking system pathology. The output signals of the IPS are compared with the behavioral model ones in a real-time mode (fig 7).

The IPS behavioral model and it blocks ensure system verification on different stages of project design.

To be convinced of the behavioral model correctness work the whole system in a visual mode is performed (fig. 8).

The source code fragment of a frequency divider behavioral model and its final implementation is on fig. 9 and 10 respectively.

The Control System Implementation can be designed not only in JHDL (Java), but also one can choose another CAD Tools (e.g. Quartus II, Xilinx ISE). JHDL allows converting various projects (different designers, CAD Tools, etc.) from EDIF files format to Java class, supplying multi-version approach as well.

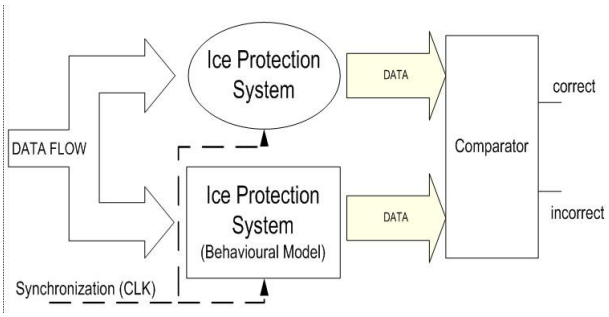

Fig. 7 Project verification model

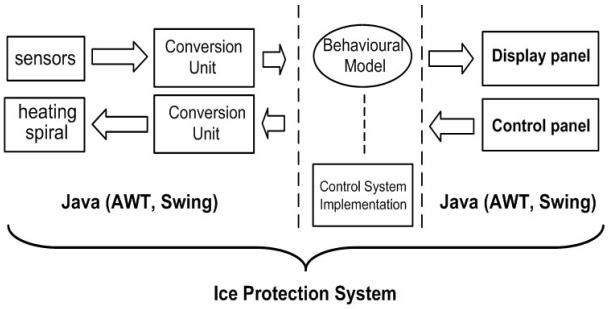

Fig. 8. Ice protection system – visual implementation

```
//tick = SYSCLKFREQ/(2*CLKOUT)
public void clock() {
   if (tick == count) put_one = true;
   count++;
   if (put_one) freq.put(this, 1);
   else freq.put(this, 0);
   if (count == 2*tick) this.reset();
}

public void reset() {
    put_one = false;
    count = 0;
}
```

Fig. 9. Frequency divider behavioral model source code

```
int reg_width = (int) Math.ceil(Math.log
              (SYSCLKFREQ/CLKOUT) / Math.log(2));
Wire countHz = wire(reg_width);
Wire tickHz = nor(xor(countHz, constant
              (reg_width, SYSCLKFREQ/(2*CLKOUT)-1)));
new upcnt(this, vcc(),tickHz,
              constant(reg_width,0), countHz);
regce_o(not(freq), tickHz, freq);
```

Fig. 10. Frequency divider JHDL source code

2.3. Checking and Reconfiguration Block Architecture

As shown in Fig. 11 system has two different versions of IP cores (V1, V2).

The Input Data enter V1 and V2. Some part of the data is stored in the Block of the Reference Valuation Generation (BRVG), where the reference valuation is generated. The reference valuation is used for comparison of output data from V1 and V2. Moreover, results from V1 and V2 compare too.

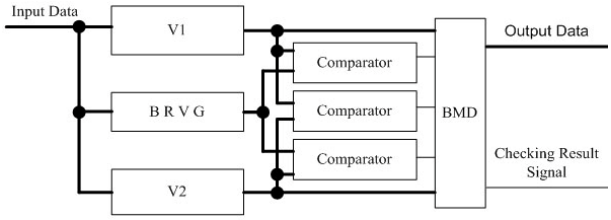

Fig. 11. Checking and reconfiguration technique for diversity project

Results of comparison come into the Block of Making Decision (BMD) where release version is chosen and formed the Output Data and Checking Result Signal. If the reference valuation and output data from V1 are equal then V1 is the release version and if the reference valuation and output data from V2 are equal then V2 is the release version too.

If output data from one of two versions and the reference valuation aren't equal this version doesn't use for the Output Data forming.

Checking and reconfiguration block allows to increase fault tolerance of complicated system detect and tolerate faults of IP core due to design faults or hardware physical faults.

Conclusion

There are different SoC decisions for creating sophisticated fault-tolerant systems. Described approach and proposed decisions are one of possible directions of the self-repairing chips development.

Implementation of multiversity increases dependability in comparing with one-version structure. In case of four-channel systems reliability measure is reduced but trustworthiness is extended.

Features of modern FPGA and their tools are like that implementation of version redundancy is normal process. There are many different languages, models of implementation on chip and testing methods.

Considered MIIP-based decisions, checking and reconfiguration means are used in two-version project (on HDL and Java HDL) for aircraft Ice Protection System.

JHDL CAD Tool gives a great flexibility in ICS design process. This approach allows to implement ICS system behavioral model and to compare the state of the target system with JHDL hardware description in a real time mode.

In this paper some design stages of ICS JHDL version implementation and the example of behavioral and hardware models for frequency divider (as part of IPS) were considered.

References

1. Barkalov A., Wegrzyn M. Design of control units with programmable logic. – University of Zielona Gora Press, 2006. – 150 p.

2. Zorian Y. What is an Infrastructure IP? // IEEE Design & Test of Computers. – 2002. – Vol. 19, no. 3. – P. 5-7.

3. Forli L., Portal J.M., Nee D., Borot B. Infrastructure IP for back-end yield improvement // ITC International Test Conference. – 2003. –P. 1129-1134.

4. Tabatabaei S., Ivanov A. Embedded timing analysis: A SoC infrastructure // IEEE Design & Test of Computers. – May-June 2002. – 19(3). –P. 22–34.

5. Kharchenko V.S., Tarasenko V.V., Ushakov A.A. Fault tolerant embedded digital FPGA systems. – KhAI, Kharkiv, Ukraine, 2004. – 210 p.

6. Mikrin E.A. On-board spacecraft control complexes and software development for them. – M.: MSTU, 2003. – 336 p.

7. Gould J. Designing flexible, high-performance embedded systems // X cell journal, 58, third quarter 2006. – P. 66-70.

8. Ostroumov S.B. , Kharchenko V.S. , Usha-kov A.A. Fault-tolerant infrastructure IP-cores for SoC: basic variants and realizations // IEEE East-West Design & Test Workshop, Sochi, Russia, 2006. – P. 194-197.

9. IAEA NS-G-1.3 (International standards).

10. NP 306.5.02/3.035-2000 (National standards of Ukraine).

11. Kharchenko V.S., Prokhorova J.N. Fault tolerant systems with FPGA-based reconfiguration devices // Proceedings of IEEE East-West Design & Test Workshop, Sochi, Russia, September 15-19, 2006. – P. 190-193.

12. Chernikov V., Viksne P., Shelukhin A., Panfilov A. Synchronization subsystem of 1879bm3 system on chip for high speed mixed signal processing // Information technologies in science, education, telecommunication and business, 2005. – P. 335-336.

13. Kulanov V.A. Analysis of the Digital systems developing by using JHDL design tools // ICTM-2006 Thesis of reports, Kharkiv, Ukraine, 2006. – P. 297.

14. Hutchings B.L., Bellows P., Hawkins J., Hemmert S., Nelson B., Rytting M. A CAD suite for high performance FPGA design // J. M. Arnold, K. L. Pocek, editors, Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, CA,. IEEE Computer Society, IEEE Computer Society Press, April 1999. – P. 12–24.

15. Bellows P., Hutchings B.L. JHDL – an HDL for reconfigurable systems // Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, J. M. Arnold, K. L. Pocek, Eds., Napa, CA, IEEE Computer Society, IEEE Computer Society Press, April 1998. – P. 175–184.

Поступила в редакцию 12.01.2007

Рецензент: д-р техн. наук, проф. А.М. Романкевич, Национальный технический университет Украины «КПИ», Киев.