Redundant Hardware Components for ASIC. RTL Model and Synthesys

Valentin Rozanov, Yuriy Sheynin, Elena Suvorova
Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russian Federation
{suvorova, sheynin}@aanet.ru, valentin.rozanov@guap.ru

Abstract—**Redundancy is a very popular and effective method to increase fault tolerance of the system. Fault tolerance in modern embedded systems is important feature due to accelerating aging and manufacturing defects, which diagnosis during the chip testing at fabric is impossible. In addition, different ways of system using may need different degree of fault protection. From hardware design point of view (ASIC design especially) redundancy means area and power increasing. It is very important to see the correlation between the components hardware description and its synthesized equivalent. The article considers several variants of synthesized redundant components that show the effect on area and power regarding to their architecture. The main goal of presented research is to describe RTL and Synthesis correlation and additional efforts that need to be done during hardware design flow to get redundant component with fault tolerant mechanism.**

I. INTRODUCTION.

In the hardware (HW) development process, component is an IP block or its part, which is further used as in the System-on-Chip (SoC) performing various functions in the place of application. In the article are considered parts of transport layer controller's IP block. It is more complex in structure and function, than IP blocks of the lower layers of the data transfer protocol, have a higher probability of failure than the Besides that using of thin design rules for SoC allow to place a lot of different components on one chip. Therefore, the functionality of embedded systems grows dramatically. However, using of thin design rules is accompanied with accelerated aging and manufacturing defects that can not be diagnosed during the chip testing at the fabric [1]. Therefore manufactured by thin design rules SoC should include fault mitigation mechanisms [2, 3, 4]. Thus, the task of mitigating the faults that occur in such an IP block is very urgent.[15]

The most common fault effect is a single event effect (SEE). Three common types of SEE are known: single event upset (SEU), single event transient (SET) and single event "latch up" (SEL). A single event upset causes the change of state in a storage element. It affects the memory cells and sequential logic. A single effect transient causes a short impulse at the combinational logic output. The wrong logic state will propagate in case that it appears during the active clock edge. On the other hand, a single event "latch up" causes the excessive current flow through a parasitic bipolar structure in CMOS circuits ([5], [6], [7], [8]). We can clearly separate the known SEU, SET, and SEL fault-tolerant techniques into the two categories: circuit level techniques [9] (hardened-cell design [10], triple modular redundancy (TMR) [11], double modular redundancy (DMR) [12], and error detection and correction for memories [13]) and layout level techniques [14].

The most used method of fault tolerant devices construction is using spatial redundancy. This method means that in case of SEU and SET appear there is additional element with the same functioning which begins to perform the function of fault element.

For most components whose inputs and outputs are bit vectors decomposition on self-similar sub-components may be used. These self-similar components will can process parts of input/output vectors. Therefore, we can say that one component that have N input/output bits may be divided into M sub-components that will have N/M input/output bits [16]. In this case partial redundancy means using additional sub-components, which helps to reduce using area.

Design of redundant hardware (HW) components is a complex process that that forms the HW design flow. Its stages and features are described in section 2. On the stage of RTL design decomposition need to be made to define what approaches designer need to use to make them fault tolerant. Typical composition of controller is presented in section 3.

Synthesis is one of the most important stages of HW design flow. This process is described in section 4. RTL architecture and its synthesis are very bound. Depending on the architecture different variants of synthesized component parameters may be received. Main parameters for the future chip that will be made from RTL are area, power and timing. These parameters may be received during synthesis. The description of these parameters and there changes regarding the RTL architecture are considered in section 5.

Synthesis of redundant RTL architectures, particularly partial redundant, may give different area and power parameters on the same component presented in different variants of sub-components decomposition. To consider this fact several structures were designed and synthesized. The results are presented in section 6.

II. HW COMPONENTS DESIGN FLOW

The design of embedded systems is a complex and dynamic process that involves different tasks at different levels of abstraction [1]. In general design flow may be divided into four main stages: Specification, Description, Synthesis, and Fabrication. Each of the stages contains many steps and details

to take in account. In this paper, first three of the stages are considered.

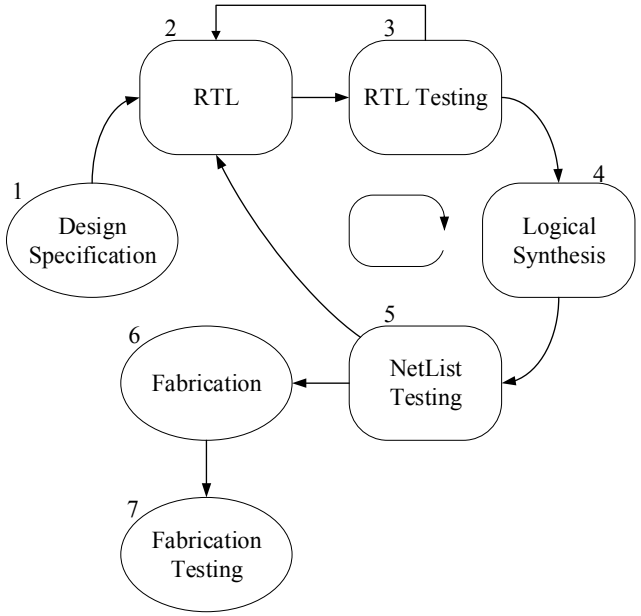

Fig. 1. HW component design flow

The design Specification usually describes the detailed functional requirements from the IC, the minimum operating speed, the maximum power and area for the IC and other robustness and reliability requirements. Before RTL modeling stage, specification is tested to check if there are any logical conflicts. It may be done with high level programing languages or with a special software.

RTL Design is the stage when specified functions are described using a hardware description language (HDL) such as Verilog or VHDL. The whole design is tested and verified, including test coverage analysis. Therefore RTL is an iterational process that include design and testing Before the synthesis design need to be clean of code redundancy and errors. However, synthesis tool removes all redundant and not used elements during the process.

Logic Synthesis is the process of conversion the RTL code into gate-level "netlist". The RTL design code once verified to be sufficiently bug-free and functioning as per the requirement, it is given to a logic synthesizer CAD tool which converts the code into a list of ports, standard cells, their pins and the interconnections between them (gate-level "netlist"). Any pre-synthesized or custom designed macros are also included with the RTL code.

III. Typical controller components.

Looking on the transport layer protocols hardware implementation, can be said that they consist of certain number of components. These components are:

1) counters
2) state machines
3) CRC counters
4) memory
5) other logic

Counters are used to count incoming and outgoing words, addresses in memory, timers etc. State machine defines the current state of data processing: waiting of header, reading address or other data, transferring data to the upper levels or to the network. CRC counter is needed to check if there are any errors in data. Memory for buffers, FIFOs, storing configuration information is used. Other logic consists of flags, switches, connection of components depending on configuration or flags, or state of state machine. Combining these elements almost any hardware controller may be constructed.

Designing any HW controller with redundancy need analysis and decomposition. Decomposition helps to choose and design the most effective combination of redundant components for the implementing HW controller.

On the stage of RTL design two methods for fault tolerance increasing may be used:

1) Using code algorithms to check data changes and correct errors, such as Hamming codes, Gray Codes, Parity bits etc.
2) Using redundancy in components to switch components (or their parts) off to the working copies.

Fault tolerant RTL model means additional testing of a tolerance function of the IP-block or its components. In this situation, simple test-benches are not effective, because they change input signals and check the output, while it is necessary to imitate changes inside the components. For these purposes special subcomponents are needed, that may be ruled by the test bench. These subcomponents design and testing take more time to develop.

IV. Logical Synthesis and its' "ingredients".

Logical synthesis is the process when abstraction of RTL model takes form of real HW device with signal transition delays, power supply that is needed for the controller, area of the chip. To make synthesis of the component there are three main thing that developer needs: RTL description, libraries of logical elements and constraints description. Combining these "ingredients" in synthesis CAD tool developer gets net list, which can be moved to the fabric to provide a chip. Lets consider each of three elements.

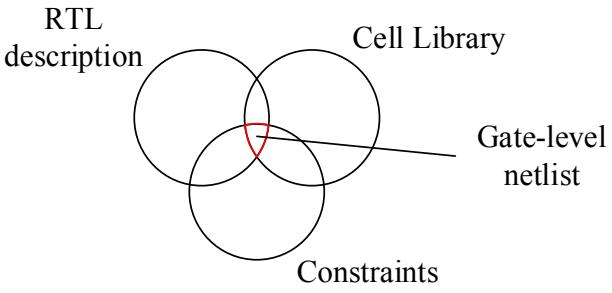

Fig 2. Elements for synthesis

RTL model describes the connection between functional blocks, and logic that is needed according to specification.

The main idea of the synthesis process is to convert the description made with HDL to the number of logical elements, MUXes, flip-flops connected to each other. Basic building blocks of a digital design such as combinational logic gates (NOT, AND, OR, NAND, AND-OR-INVERT, OR-AND-INVERT, multiplexers, adders), sequential cells (Flip-flops, Scan flip-flops, latches) and special cells (tie cells, delay cells, filler cells) are custom designed (like analog design), verified, characterized and later used for the design of larger circuits. Let us call them "elements" These elements may have two and more inputs: for example 4 input NAND. All these elements are described in a special document, called "technology library" or "standard cell library". This document contains full description of elements, such as size, number of inputs/outputs, power that is needed to work, resistance, etc. Technology libraries are not only the abstraction of the elements, but the description of their physical equivalents constructed with some number of transistors. There are also dimensions of each logical element. Using them, the information about the area of the controller or its components may be obtained. Electrical characteristics of the elements are also described in the libraries.

All elements are connected with the wires. All of them have different length, and element have different time to react on signal changing (because of there difference in number of transistors to construct, number of input ports, etc.), as the result – different time of signal transition from one point of scheme to the other. To be sure that signal will pass through the certain time it is necessary to constrain the period of this time. Usually it is the period of clock signal, that is used in designing controller. Also we need to constrain informational signals to there clock signals to be sure that all parts of the controller will work synchronous with the chosen clock signal. As signal transit time of the elements is described in technology libraries, during the synthesis timing information may be obtained.

There are situations, when the chain of the elements between two flip-flops is too long, and signal needs more time than one clock period to pass it. This situation leads to negative timing slack and HW designer have to change design to remove long chains.

One of the most useful features of synthesis is optimization. If there are variant to make chains shorter, or there are too much logic in the RTL, or it is redundant, that may be reorganized – synthesis tool will optimize it. However, this feature is useful until the necessary redundancy will be optimized.

V. AREA, POWER AND TIMING.

The most important parameters of the designed controller are area that is needed to allocate all elements on the silicon, and timing that need to be correct to the chosen value of clock period. Redundant scheme design needs additional area. Total area that may be used for logic and wires placement is limited by the physical size of the chose chip-case and fabrication technology. Information about power, that will be used by the scheme, is important for understanding what power supply we need to use. These three parameters are closely related with the design. Each specification may be designed with several different constructions of logic. The same is justly for the redundant designs.

Let's consider the example: specification needs to design a duplicated CRC counter. There are two variants it may be implemented. "Variant 1" is presented on fig. 3.

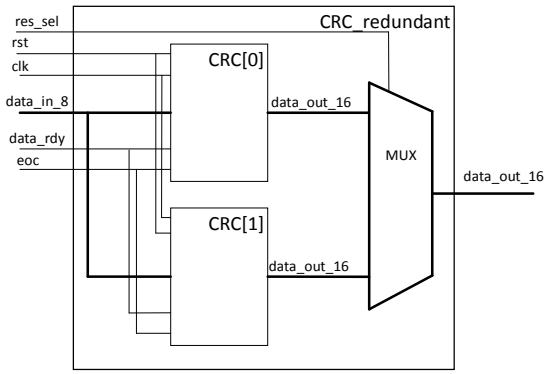

Fig 3. Structure of synthesized redundant CRC counter, "Variant 1"

Top component CRC_redundant for "Variant 1" has:

- input ports – input data vector (data_in (8b)), valid data flag (data_rdy(1b)), end of counting (eoc(1b)) and reset (rst) and clock (clk) signals.
- output ports- output of CRC result (data_out (16b)).

Two CRC components work together, The result of calculations is chosen by the multiplexor (MUX) ruled by some voter or similar mechanism.

Top component CRC_redundant for "Variant 2" has the same number of input and output ports (fig.4).

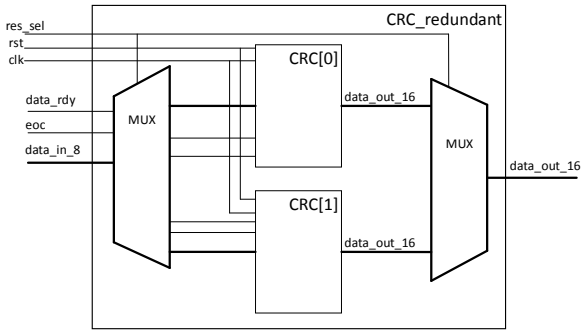

Fig 4. Structure of synthesized redundant CRC counter, "Variant 2"

The main difference is that CRC component work separately – only one of them at the same time. In this variant two multiplexors will be needed – on the input, to switch incoming data to the active CRC counter, and one after the CRC component to get the result of calculations. In real synthesis process the number of multiplexors depend on elements that cell library contains. Need to be noticed, that for long vectors that need to be multiplexed are used as many

MUXes as bits in vector. Fixed maximum and minimum size of the elements allows not very wide variety of MUXes inputs and outputs.

Table 1 contains the output of area and power for the presented variants of redundancy design. Cell and cell area columns contains information about used cells for instances construction and their area. The synthesis tool took these cells from the "technology library". Area is measured with micrometers.

TABLE I. OUTPUT OF AREA PARAMETERS DURING SYNTHESIS FOR REDUNDANT CRC.

Var. Num.	Instance	Cells	Cell Area	Leakage Power(nW)	Dynamic Power(nW)
1	crc_comb	154	7065	29.747	3237211.887
2		194	8146	35.388	2519175.992
1	crc_redund[1]	69	3244	13.370	1419375.331
2		69	3244	13.370	1170423.129
1	crc_redund[0]	69	3244	13.370	1511188.337
2		69	3244	13.370	877449.634

Instances crc_redund_1 and crc_redund_0 have the same area, because there construction was not changed. Differs only the area of the top instance, that implements connection of their CRC counters. It is obviously because of given descriptions and pictures. The most interesting in this table is power columns. Having the same leakage power CRC counters have very different dynamic one for variants 1 and 2. Total dynamic power for the top instance "crc_comb" is bigger for the variant with smaller area. The reason is in the connection of the CRC instances. In "Variant 2" one of them is not working, and as the result – lower power consumption.

This example shows the problem of HW designer. It is necessary to choose the proper variant of components construction for the best parameters of power and area.

Timing for this two variant are rather same. However it need to be said that for "Variant 1" it is 3649, and for the "Variant 2" – 3651. The difference is very small and appears because of using different library element parameters by the synthesis tool. It should be noted that synthesis tool also has effect on the results because of using different optimization algorithms and using different library elements and their parameters.

VI. RESULTS OF SYNTHESIZED REDUNDANT STRUCTURE

Let us consider the fault tolerant component design. This component implements the partial redundancy architecture. For the example, simple arithmetical component was taken. The idea of the architecture is presented on figure 5. Decomposition gives an opportunity to use spare sub-component instead of full copy of component. Spare sub-component will protect the component against one failure. Quantity of spare sub-components should be equal to quantity of mitigated failures. IN-MUX divides input data between sub-components. Spare MUX receives data from both lines and switches data to spare sub-component selected for operation. In this situation, spare sub-component can operate data instead of sub-component 1 or 2. OUT-MUX combine data from operating sub-components, and may use data from spare sub-component if it is needed. All MUXes are applied by one line from Fault Detector. This method of redundancy is called sliding redundancy. [16].

Component constructed of sub-components uses spare sub-component to mitigate one fault. However it is unclear how many subcomponents is too much or enough, and when the overheads for this redundancy construction will become most ineffective.

The main parameter to understand the efficiency is area of the synthesized component. Further research was divided into two parts:

1) Variable number of subcomponents in one decomposed component with the fixed input/output vectors size synthesize
2) Variable number of spare subcomponents for the chosen variant of components decomposition

First step will help to understand if there is difference between numbers of sub-components, and how many they are different. The second step may give the information about number of spare components that makes area bigger that duplicating full component.

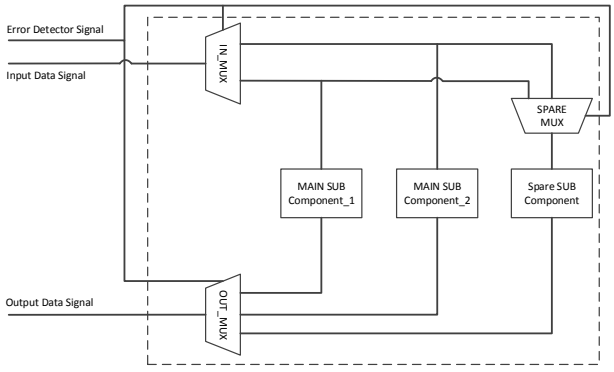

Fig 5. Component constructed with tow subcomponents and one spare sub-component

For the first step component with two input vector 128 bit width for each (as operands) was chosen. Component also has command code input (3 bits) and flags to active subcomponents choosing. It's width depends on number of subcomponents.

For these variants, synthesis was made. The results of components area and there overheads are presented in Table II.

TABLE II. OUTPUT OF AREA PARAMETERS FOR "COMPLEX" COMPONENTS CONSTRUCTION.

Number of elements	Sub_component "width"	Components total area (um)	Overheads area (um)
2	64	1699102	59541
4	32	740054	55968
8	16	363034	58573
16	8	199359	62222
32	4	146198	63499

Synthesis tool presents area of instances that are used in RTL model, or that are the result of tool itself recombination of these instances. Not to loose the structure of the sub-components (because it is important for redundancy) special flag was used, that preserve recombination of instances.

In contrast to the total area of the component, overhead area is calculated, using formula (1)

$$S_O = (S_T - S_C) + \left(S_C - \sum_{1}^{N} S_B\right) \qquad (1)$$

where:

S_O – area of overheads

S_T – total area of the component

S_C – area of complex component that combines subcomponents

S_B – area of base subcomponents used in the architecture

N– Number of subcomponents (including spare subcomponent).

Need to be noticed that subcomponents area was different in one synthesized component. It happens because of using different elements from the technology library by the synthesis tool.

Fig 6 presents graph for the received areas. Easy to see that area decreases when increases the number of subcomponents. Decrease slows down after 8 sub-components. Difference in area between 8, 16 and 32 sub-components is not very big.

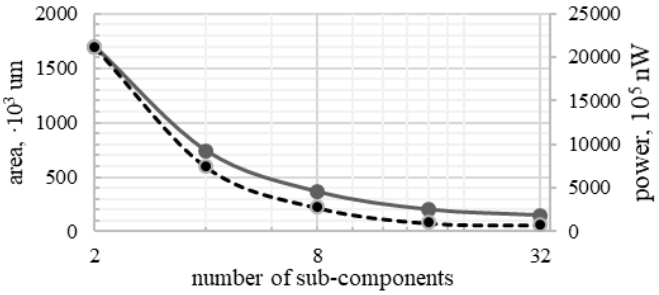

Fig 6. Total area and power changes

Power changing is also presented on fig 6. Dotted line shows decreasing of power in a linear dependency with the area. However, as being shown in section five, power depend not only on the area of the component but also on its architecture. Presented components have the same internal architecture.

Overhead area graph presented on the fig 7 is also very informational. The smallest overhead area has the component with four subcomponents, and the biggest one is variant with 32 sub-components. The last one result is more expectable than others are, because additional MUXes usage to connect elements.

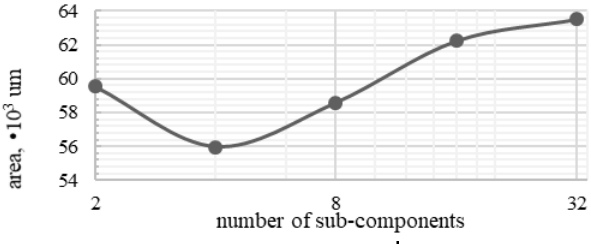

Fig 7. Overhead area changes

However, the difference in the overhead areas is not so big. Presumably, the reason is in using different logical elements from the technology library and their combinations. Libraries contains many elements with different parameters, and synthesis tool uses element according to the scheme trying to get the most effective result. That is why the synthesis process takes a lot of time.

Analyzing areas (total and overhead), for presented variants of the component the best is number eight, and the worst is number two.

The second step is to define which number of spare sub-components is optimal from the area and power points of view. Relying on the results of different component decomposition from step one was chosen the variant with eight subcomponents.

Result of the area and power are presented in Table III. Synthesis was made for eight variants, incrementing the number of spare subcomponent.

TABLE III OUTPUT OF AREA PARAMETERS FOR COMPLEX COMPONENTS WITH 1-8 SPARE SUBCOMPONENTS

Number of elements	Components total area (um)	Sub-component area range (□10³um)	Dynamic Power(nW)
1	363034	32-38	270945739
2	417669		297057426
3	475778	44-32	328501623
4	553481	48-33	337851237
5	613140	49-32	358526247
6	695502	52-32	392399004
7	755621	54-32	414260421
8	873906	58-32	459296937

Graph on fig 8 illustrates changes of the area. Can be seen that area increases very slow. The horizontal line show the area that is equal two areas of the component.

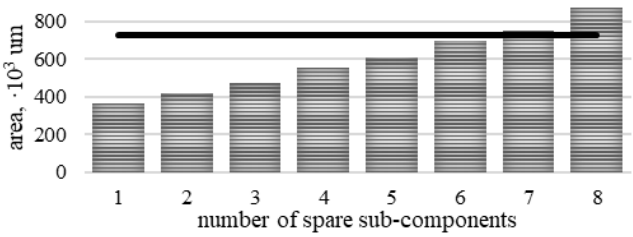

Fig 8. Area of component changing depending on number of spare subcomponents

Area of the component with six sub-components is almost equal to the fully duplicated component. However, six spare sub-components means six faults mitigation instead of one in case of full duplication.

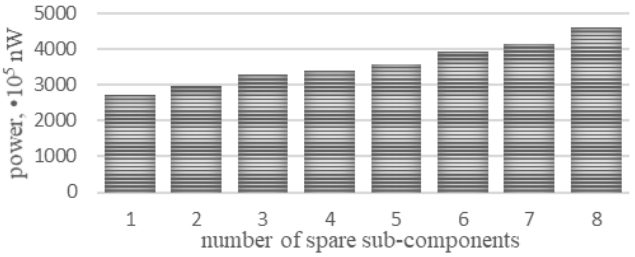

Fig 9. Overhead area changes

Graph on fig 9 illustrates the increasing power. It has linear dependence with the area in this situation. It could be seen that beside variants have small difference in power. Comparing variant one and six the difference is near $1200 \cdot 10^5$ nW. Looking on Table I it can be said that using fully duplicated component will lead to the duplication of the power supply (if components operate together at the same time). For the considering components with partial redundancy the increasing of power is near 7% from variant with one spare subcomponent.

VII. Synthesis and RTL interaction.

One more thing that was not described in this article yet is interaction between HW description on HDL and synthesis tool. To automate this process special scripts on Tool Command Language (tcl) are used. This scripts describes all stages that need to be passed to get "netlist". They are:

- Reading vhdl/verilog files of the project
- Declearing constraints for input and output signals
- Setting up parameters of the synthesis for optimization algorithms, restructuring algorithms, computing power using, etc.

To get the results for sections five and six also TCL script was used. During work with Cadence synthesis tool (Genus) some simple rules were introduced:

1) Everything that may be assigned as a variable need to be assigned. It may be clock signal names, clock periods, paths to the project, names of output files, etc. (TCL)
2) If it is possible to combine input and output signal physically (in vectors instead separate signals), or by meaning (using same parts in names of the signals) it is better to do this (HDL)
3) Every input and output need to be constrained, if it's could not be constrained – set parameter of the signal delay (TCL)
4) If number 2 of this list is done use cycles to make constraints, code becomes easy to red, and saves time to write and debug (TCL)
5) Everything that may by set during reset signal need to be set during it (HDL)

In brackets place of rule usage is signed – HDL or TCL. Effect of these rules is not so noticeable in small projects, but they helps to make synthesis more convenient in a huge projects which synthesis takes several hours to do. However, time to synthesis also depends on architecture. Getting the results for this paper the longest tame took variants with increasing number of spare components – from 10 to 20 minutes for each. The shortest one was making synthesis of full redundant CRC counters that took not more than 5 minutes for each variant.

Conclusion

The article considers different aspects of HW controllers and components design. Making a fault tolerant HW controllers using redundancy, each step of the design flow need to be expanded with additional operations. On the level of RTL it is components decomposition. On the level of synthesis it is analysis of the results of RTL design to get the most "low cost" variant from area point of view. Also during synthesis it is necessary to be very careful with timing slacks, and define timing constraints corresponding to the resulted structure of the controller.

It is not possible to take already designed soft IP and make it fault tolerant. Except the case when it is used triple times with a voter. Design fault tolerant controller or component needs lots of changes in the RTL model. Testing fault tolerant controllers need special components that will model fault inside the controller.

Fault tolerant HW controllers is a combination of different techniques. Among them are:

- Redundancy approaches (such as double/triple block usage, with voting mechanisms)
- Reconfigurable components and IP-blocks
- Using redundant codes (such as Hamming codes, Gray Codes, Parity bits etc.)

Using only one variant of fault tolerance may be not effective from area and power points of view. However using different methods of tolerance leads to more complex construction and takes more time to design and test. In this case the main goal is to choose the most effective ratio according to the area, power and timing.

Summing up presented on section 6 example can be said that using partial redundancy in hardware components is more effective from area and power points of view than using fully duplicated components. One spare sub-component uses 93% less power and needs 90% less area mitigating the same number of faults.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under the contract RFMEFI57816X0214.

References

[1] International Technology Roadmap for Semiconductors (ITRS), 2013dd
[2] Armin Runge, "FaF NoC: a Fault-tolerant and Buerless Network-on-chip", Procedia Computer Science, vol. 56, 2015, pp. 397–402.

[3] Erica Cota, Alexandre de Morais Amory and Marcelo Soares Lubaszewski, Reliability, Availability and Serviceability of Networks-on-Chip. Springer, 2012.
[4] Synthesis of RTL Descriptions for Architecture Exploration, Haupt Seminar, Yazan BOSHMAF, 2007
[5] R. R. Troutman, Latchup in CMOS technology: The problem and its cure, Kluwer Academic Publishers, Boston, 1986.
[6] S. H. Voldman, Latchup, John Wiley & Sons, Chichester, 2007.
[7] N. H. E. Weste and D. M. Harris, CMOS VLSI design: A circuits and systems perspective, Fourth Edition, Addison-Wesley, Boston, 2011.
[8] L. Ye, G. Xiaohan, X. Weiwei, H. Zhiliang, and D. Killat, "An experimental extracted model for latchup analysis in CMOS process", In Proc. 8th IEEE International Conference on ASIC, Hunan (China) 2009, (pp.1035-1038)
[9] R. C. Lacoe, "Improving integrated circuit performance through the application of hardness-by-design methodology", IEEE Transaction on Nuclear Science, vol.55, no.4, pp.1903-1925, 2008.
[10] M. P. Baze, S. P. Buchner, and D. McMorrow, "A digital CMOS design technique for SEU hardening", IEEE Transaction on Nuclear Science, vol.47, no.6, pp.2603-2608, 2000.
[11] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability", IBM Journal of Research and Development, vol.6, no.2, pp.200-209, 1962.
[12] J. Teifel, "Self-voting dual-modular-redundancy circuits for single-event-transient mitigation", IEEE Transaction on Nuclear Science, vol.55, no.6, pp.3435-3439, 2008.
[13] P. K. Lala, "A single error correcting and double error detecting coding scheme for computer memory systems", In Proc. 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, Cambridge (USA) 2003, (pp.235-241)
[14] G. K. Maki and P.-S. Yeh, "Radiation tolerant ultra-low power CMOS microelectronics: Technology development status", In Proc. NASA Earth Science Technology Conference, Hyattsville (USA) 2003, (pp.1-4)
[15] Structural Redundancy and Design Space Exploration Method for the Hardware Components With Fault Mitigation Design, FRUCT 20
[16] Rozanov V. Suvorova E. Approaches to the SoC IP-blocks' Design with Errors' Mitigation, FRUCT 21