The Components Spatial Redundancy Method Based on Design Space Exploration

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*—The fault mitigation for modern embedded systems developed by thin design rules (40 nm and less) is necessary feature due to accelerating aging and manufacturing defects, for which diagnosis during the chip testing at fabric is impossible. Different ways of spatial redundancy are used for fault mitigation in the SoC. They provide different achievable mean time between failures (MTBF). For various embedded systems a different lifetime is planned, therefore fault probability is required. Realization of these methods has different hardware cost (additional area on the chip). The area is one of most critical parameters for SoC in embedded systems and is strongly constrained. We propose the method for development of components' spatial redundancy. Method is based on design space exploration (DSE). It allows to select design spatial redundancy with considering area constraints and fault probability requirements.**

I. Introduction

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].

Different approaches of components spatial redundancy are used for SoCs [5], [6], [7], [8]. All approaches of spatial redundancy lead to increasing of SoC's area. Area of spare components and the fault probability could be various for different approaches and depend on way of spare components placing. The required fault probability depends on planned embedded system lifetime.

In many cases the smallest area overheads lead to smallest fault probability. However, dependency between these parameters is very complex. It is determined by scheme of spare components integration into the system, by the size (area) of additional multiplexers that are used. These multiplexers themselves do not have redundancy, fault mitigation for them is not implemented, thus they could decrease achievable fault probability.

Therefore the operating parameters of a developed embedded system strongly depends on selected approach for spatial redundancy in it.

To solve the problem we propose a method of spatial redundancy selection that is based on design space exploration (DSE). Based on design space exploration methods are widely used for modern SoC development [9], [10], [11], [12]. The design space exploration of NoCs is commonly formulated as a constrained optimization problem [13]. This approach is used for different tasks (such as buffer size selection, arbitration rules selection and many others) that have high computation complexity.

The N-dimensional design space is formed in the frame of this approach. Number of dimensions is equal to quantity of system parameters, for which values should be specified or constraint in the system design.

In the 2nd section we consider the approaches to spatial redundancy for NoC components. In the 3rd section we represent proposed spatial redundancy method based on design space exploration. In the 4th section we describe obtaining scheme for the coordinate of the point on the area overheads axis. . In the 5th section we describe obtaining scheme for the coordinate of the point on the AXIS of fault probability. In the 6th section we represent the ways of component's redundancy.

There are various approaches of building component spatial redundancy. We consider two most common ways of spatial redundancy as use cases: the whole component redundancy and slicing redundancy of subcomponents [1].

Some spare components are included into the system, when the main component functionality is critical for the SoC (fig. 1). Quantity of spare components is equal to number of errors, against which the system should be tolerant. Using of one spare component when the main component is failed allows to support the SoC functionality without degradation.

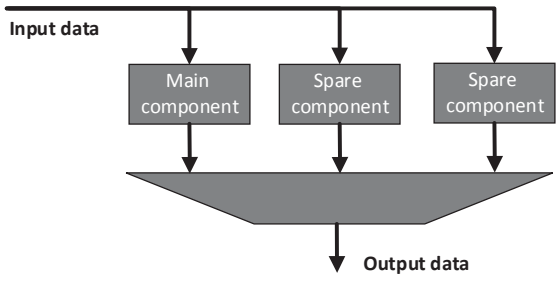


Fig. 1. The example of whole redundant structure

ISSN 2305-7254

This approach essentially increases the SoC hardware cost (area), while the strong area constraints are typical for many embedded systems. Therefore, it is often impossible to provide redundancy for many components, which functionality is critical for the SoC. It significantly reduces operating parameters of the SoC when this approach is used, and essentially limits its scope.

Also we consider another way of spatial redundancy in this paper. Realization of this approach would require less area. It is based on decomposition of a basic component onto subcomponents that are self-similar to the basic component.

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. However implementation of self-similar sub-components involves additional overheads (with increasing of area and timing constraints). In NoC can be identified quite a number of types of components overhead for implementation of which as a group of components are not big.

For example an arithmetic logic unit (ALU) component operates witch N-width operands can be constructed of M ALU subcomponents with N/M-width operands (fig. 2). Hardware cost of the ALU component realized "as whole" are equal to hardware cost of the ALU component constructed of subcomponents.

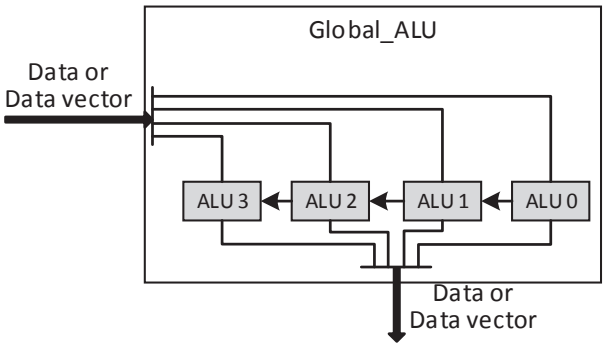


Fig. 2. The example of ALU structure

Also, the channel switch can be realized on base of channel switches with less quantity of ports (fig. 3)

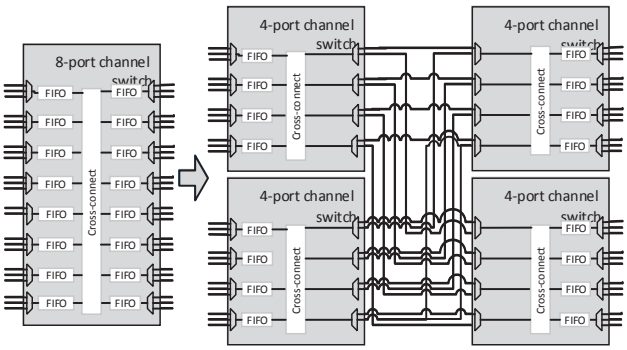


Fig. 3. The example of switch structure

For protection the component against one failure, we don't need to duplicate whole component. We can include one spare subcomponent. Quantity of spare subcomponents should be equal to quantity of mitigated failures.

Let's consider inclusion of spare subcomponents into the component. The inclusion scheme for ALU is represented on fig. 4. The component Global_ALU consists of four subcomponents – ALU0 – 3 in this example.

If we need to mitigate one fault, we should include one spare subcomponent ALU S (we denote modified ALU component - Global_ALU_R). ALU S should be used instead any of subcomponents (ALU0 – 3). Therefore, the Global_ALU_R includes multiplexors and interconnections that allow to transfer input date to inputs of base ALU sub-components and to transfer output date from these components to the global output instead of any base ALU.

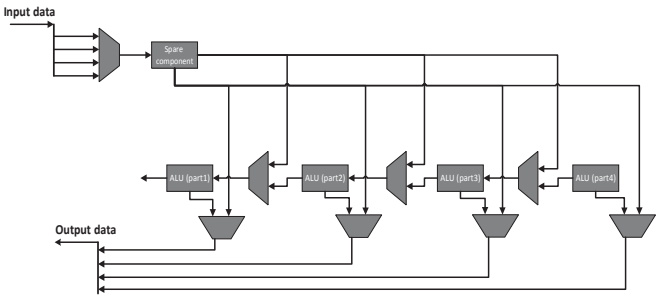


Fig. 4. The spatial redundancy scheme on subcomponent layer

Thus the spare subcomponent ALU_S, some multiplexers and interconnection lines are added in the basic structure of Global_ALU component for realization of this spatial redundancy way.

The schemes of spare subcomponents inclusion for first and second approach are essentially differ, they have different area overheads. Using of these approaches allow to realize components that resist to equal quantity of faults.

The multipliers, used for including of spare components are bottleneck for botch schemes. The considered approaches do not allow to mitigate faults in these multiplexers. Quantity of multiplexers and its area are various for different approaches, therefore fault probability will be different.

In this paper we propose the components spatial redundancy method based on design space exploration. This method allow to choose spatial redundancy way correspondingly the area constraints and required fault probability.

II. THE COMPONENTS SPATIAL REDUNDANCY METHOD BASED ON SPACE EXPLORATION.

The design space for our problem includes two dimensions: fault probability and area overheads.

Our method includes an algorithm used for building of points in the design space. The algorithm includes following main steps:

1) Development of the basic structure (without spare components) and development of the structure with spare components for every considered spatial redundancy way.

2) Evaluation of area for every component included into the structure.

3) Evaluation of the area for the basic structure

4) Evaluation of the area for the structure with spare components

5) Evaluation of the area overheads that arises due spare components – obtaining the coordinate of the point on the area overheads axis

6) Development of the Markov net for the structure with spare components

7) Evaluation of probabilities to fail for the elements of Markov net correspondingly its areas and types.

8) Evaluation of fault probability - obtaining the coordinate of the point on the fault probability axis

III. The coordinates of the point obtained values on the axis of area overheads

We introduce following notations:

Sb – the area of base component

Sc – the area of one subcomponent (self-similar to base component) for second way of spatial redundancy

$$Sc = \frac{Sb}{N},$$

where N – quantity of subcomponents in base component.

Sr – the area of scheme with spatial redundancy, Sr1 – the area when first way is used, Sr2 – the area when second way is

$$Sr1 = (K+1) * Sb + Sm \quad (1)$$

used.

where K – redundancy multiplicity

$$Sr2 = N * Sc + (K+1) * Sc + Sm2 = Sb + (K+1) * Sc + Sm2 \quad (2)$$

Sm1 – the area of multiplexers

Sh – the area overheads, Sh1 – the area overheads when the first way is used, Sh2 – the area overheads when the second

$$Sh1 = Sr1 - Sb = K * Sb + Sm \quad (3)$$

$$Sh2 = Sr2 - Sb = (K+1) * Sc + Sm2 \quad (4)$$

way is used

IV. The coordinates of the point obtained values on the axis of fault probability

To calculate ALU fail probability in time Markov chain were constructed. In [14] was described method of calculating Markov chain with discrete time. This method was used to calculate probabilities for constructed schemes. It need to be noticed that all calculations are made with discrete time.

The following symbols are used in the schemes:

p_w – transferring probability to stay in W-state (work without failures)

p_{wn} – transferring probability to move from state W to N-state, where N – number of the failed ALU

p_{wr} – transferring probability to move from state W to state R, where R – state of reserve element or it's multiplexor failure

p_f – transferring probability to stay in F-state (system failed). As F is the finish state, and there are no ways from it this probability is equal 1

p_{nf} - transferring probability to move from N-state to F-state

p_{rf} - transferring probability to move from R-state to F-state

p_n – transferring probability to stay in N-state

Markov chain for the device with one fully redundant ALU is presented on fig. 5. Марковская цепь для устройства с единичным резервированием представлена на рисунке 5.

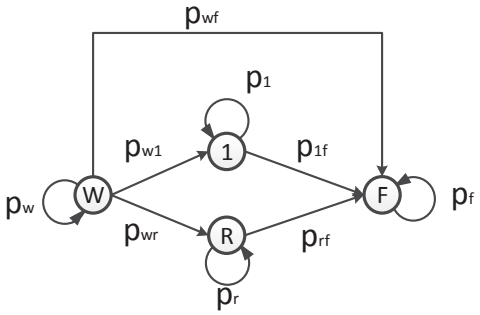


Fig. 5. Markov chain for fully redundant ALU

Calculations of fail probability for fully redundant ALU were made using equations that were described in [14].

Fig. 6 describes ALU with sliding redundancy, organized as combination of 2 ALU (2·8/2·16) that forms 1 ALU of 16/32 input bit vector with 1 reserve ALU.

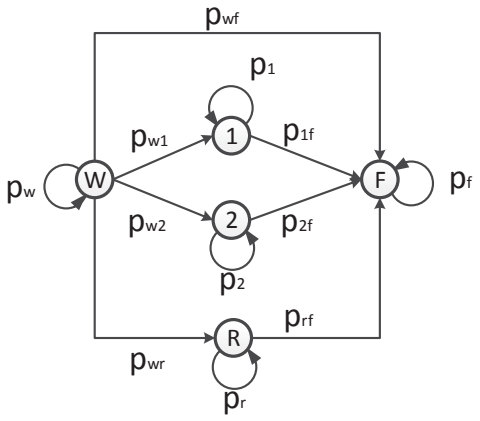


Fig. 6. Markov chain for ALU with sliding redundancy (2·8/2·16/2·32 with 16/32/64 input bit vector)

Fig. 7 shows ALU with sliding redundancy, organized as combination of 8 ALU (8·8) that forms 1 ALU of 64 input bit vector with 1 reserve ALU

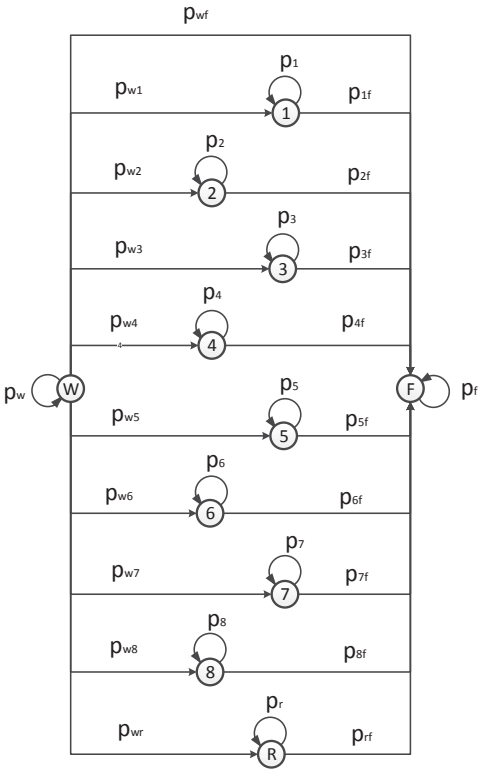


Fig. 7. Markov chain for ALU with sliding redundancy (8·8 with 64 input bit vector)

Fig. 8 shows ALU with sliding redundancy, organized as

$$p_{rf} = \sum_{k=1}^{n-1} (C_n^k p^k) \qquad (7)$$

combination of 4 ALU (4·4/4·8/4·16) that forms 1 ALU of 16/32/64 input bit vector with 1 reserve ALU

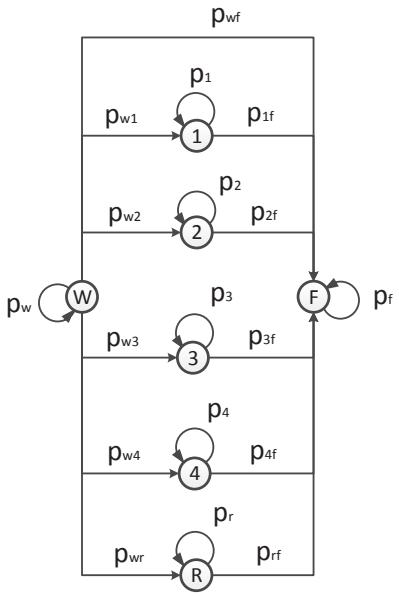


Fig. 8. Markov chain for ALU with sliding redundancy (4·4/4·8/4·16 with 16/32/64 input bit vector)

Calculating transferring probabilities for sliding redundancy schemes new equations are needed.

Probability p_{wn} is equal ALU's or it's multiplexor's probability to fail, or both of them.

Probability p_{nf} calculates as a number of combinations for other ALU/multiplexors to fail (5). To make equation easier multiplexors fails are added as logical 'or'. From the area of

$$p_{nf} = p^{n-2} p_r + \sum_{k=1}^{n-2} (C_{n-2}^{k-1} p^{k-1} p_r + C_{n-2}^k p^k) \qquad (5)$$

components and sub-components point of view it's enough to conclude the results of calculations for these schemes.

where:

p_{nf} - transferring probability to move from N-state to F-state

p – not reserve ALU fail probability

p_r – reserve ALU fail probability

n – number of states in the scheme (excluding W и F states)

k – minimal number of failed elements at which state

$$p_{wf} = p^{n-1} p_r + \sum_{k=2}^{n-1} (C_{n-1}^{k-1} p^{k-1} p_r + C_{n-1}^k p^k) \qquad (6)$$

transfers

Probability p_{wf} may be calculated using equation (6)

Since including multiplexors fail probability in reserved ALU it becomes different from the ALU components' fail probability in case of sliding redundancy. Therefore, p_{wf} may be calculated as (7)

As a base probability to fail was used value 2·10-9 multiplied on the total area of the ALU (or element of complex ALU) or multiplexor.

V. Example of method using on ALU component

The proposed construction of the redundancy options have been implemented and synthesized using Cadence RTL Compiler 16.1. As the result of compiler work areas of ALUs and multiplexors were obtained. Initially, the basic circuit elements were synthesized – ALU with 4/8/16/32/64 bit width of input/output vectors. ALU component was considered as "simple" (only sum operation) and "complex" (sum and multiplying operations) variant. Results of modeling and RTL Compiler work are presented in Table I.

TABLE I SYNTHESIS OF BASE ALU COMPONENTS

Input vector Bit Width	Total Area for SUM	Total Area for SUM & MULT
4	2510	9100
8	5600	51382
16	11289	187211
32	25034	633754
64	61733	2407552

On the next step ALU were implemented as complex elements. Areas of complex ALU component constructed of 2/4/8 subcomponents are presented in Tables II и III.

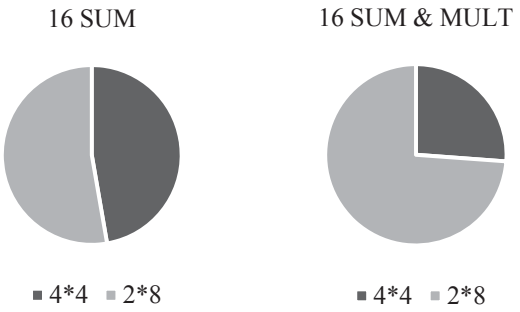
TABLE II SYNTHESIS OF BASE "SIMPLE " ALU CONSTRUCTED WITH SUB-COMPONENTS

SUM				
Input vector Bit Width	**Number of ALU**	**ALU Bit Width**	**Base Area**	**Total Area**
16	**4**	**4**	**2510**	**10040**
16	**2**	**8**	**5600**	**11200**
32	**4**	**8**	**5600**	**22400**
32	**2**	**16**	**11289**	**22578**
64	**8**	**8**	**5600**	**44800**
64	**4**	**16**	**11289**	**45156**

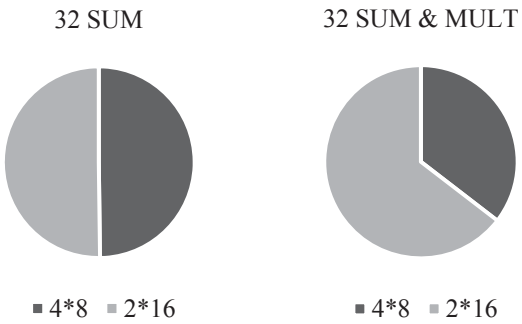
TABLE III SYNTHESIS OF BASE "COMPLEX " ALU CONSTRUCTED WITH SUB-COMPONENTS

SUM & MULT				
Input vector Bit Width	**Number of ALU**	**ALU Bit Width**	**Base Area**	**Total Area**
16	**4**	**4**	**9100**	**36400**
16	**2**	**8**	**51382**	**102764**
32	**4**	**8**	**51382**	**205528**
32	**2**	**16**	**187211**	**374422**
64	**8**	**8**	**51382**	**411056**
64	**4**	**16**	**187211**	**748844**

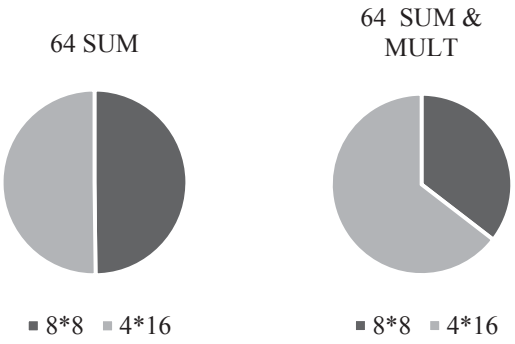
The charts on fig 9 shows the area ratio of ALU composed of a different number of ALU sub-components



a) 16 bit ALU constructed with 2 8-bit ALU sub-components and 4 4-bit sub-components



b) 32-bit ALU constructed with 2 16-bit ALU sub-components and 4 8-bit sub-components



c) 64 bit ALU constructed with 8 8-bit ALU sub-components and 4 16-bit sub-components

Fig. 9. Difference between ALU constructed of different ALU sub-components with the same input vector width

After synthesis results of base and complex ALU were received models of full redundancy and sliding redundancy were synthesized. Results of this synthesis for "simple" and "complex" ALU component with full redundancy are presented in Tables IV and V.

TABLE IV FULL REDUNDANCY ALU SYNTHESIS OF "SIMPLE" ALU

SUM				
Input vector Bit Width	**Total Area for SUM**	**Number of MUX**	**MUX Area**	**Total Area with reserv**
4	**2510**	**1**	**1300**	**6320**
8	**5600**		**2600**	**13800**
16	**11289**		**5200**	**27778**
32	**25034**		**10400**	**60468**
64	**61733**		**20800**	**144266**

TABLE V FULL REDUNDANCY ALU SYNTHESIS OF "COMPLEX" ALU

SUM & MULT				
Input vector Bit Width	**Total Area for SUM & MULT**	**Number of MUX**	**MUX Area**	**Total Area with reserv**
4	**9100**	**1**	**1300**	**19500**
8	**51382**		**2600**	**105364**
16	**187211**		**5200**	**379622**
32	**633754**		**10400**	**1277908**
64	**2407552**		**20800**	**4835904**

Once redundant variant of full redundancy ALU was implemented. For this variant, it need one multiplexor in the scheme that is written in column Number of MUX.

Results of sliding redundancy ALU synthesis for simple and complex ALU are presented in Tables VI and VII. Sliding redundancy needs one multiplexor for each ALU sub-component. Therefore, column Number of ALU shows also number of used multiplexors.

TABLE VI SLIDING REDUNDANCY ALU SYNTHESIS OF "SIMPLE" ALU

SUM					
Input vector Bit Width	**Number of ALU**	**ALU Bit Width**	**Base Area**	**MUX Area**	**Total Area**
16	**5**	**4**	**2510**	**1820**	**21650**
16	**3**	**8**	**5600**	**2490**	**24270**
32	**5**	**8**	**5600**	**3400**	**45000**
32	**3**	**16**	**11289**	**4950**	**48717**
64	**9**	**8**	**5600**	**4100**	**87300**
64	**5**	**16**	**11289**	**6200**	**87445**

TABLE VII SLIDING REDUNDANCY ALU SYNTHESIS OF "COMPLEX" ALU

SUM & MULT					
Input vector Bit Width	**Number of ALU**	**ALU Bit Width**	**Base Area**	**MUX Area**	**Total Area**
16	**5**	**4**	**9100**	**1817**	**54585**
16	**3**	**8**	**51382**	**2255**	**160911**
32	**5**	**8**	**51382**	**2880**	**271310**
32	**3**	**16**	**187211**	**5148**	**577077**
64	**9**	**8**	**51382**	**4393**	**501975**
64	**5**	**16**	**187211**	**4529**	**958700**

Based on the results of the synthesis charts were made that shows difference between area of full and slide redundancy for different width of input bits.

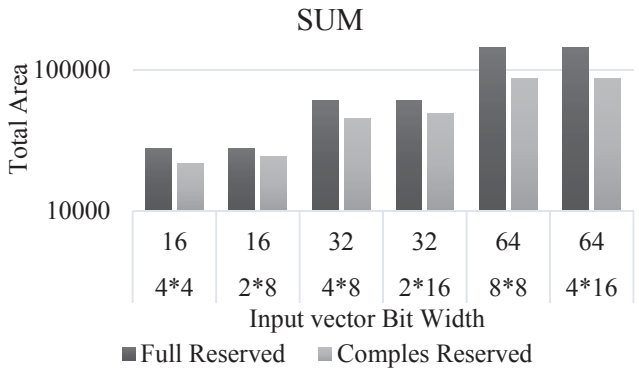


Fig. 10. Dependence of used area by ALU with sum operation constructed with different number of ALU sub-components

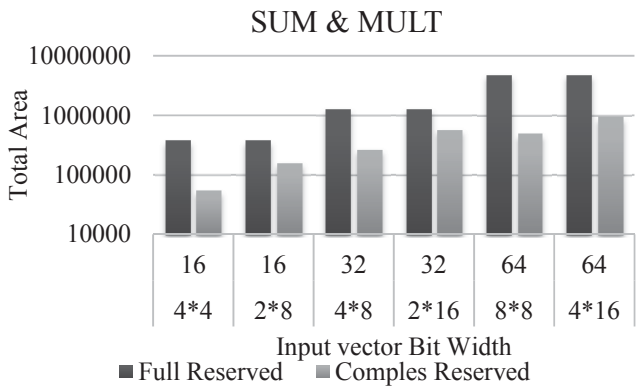


Fig. 11. Dependence of used area by ALU with sum and multiply operation constructed with different number of ALU sub-components

Dependence between overhead and ALU component constructed with different number of sub-components is presented on a chart in fig. 8 and 9 for the slide redundancy ALU.

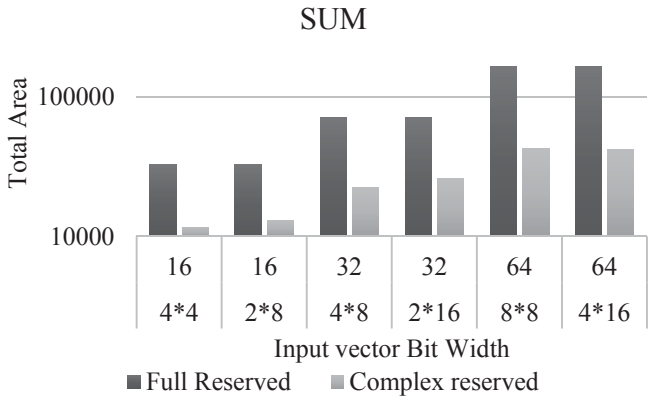


Fig. 12. Dependence of overhead used area constructing ALU with sum operation and different number of ALU sub-components

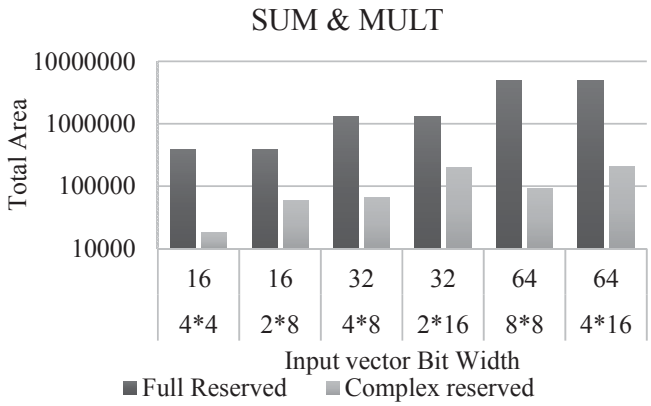


Fig. 13. Dependence of overhead used area constructing ALU with sum and multiply operation and different number of ALU sub-components

The results of Markov chain calculations are presented on charts in fig. 13 and fig. 14

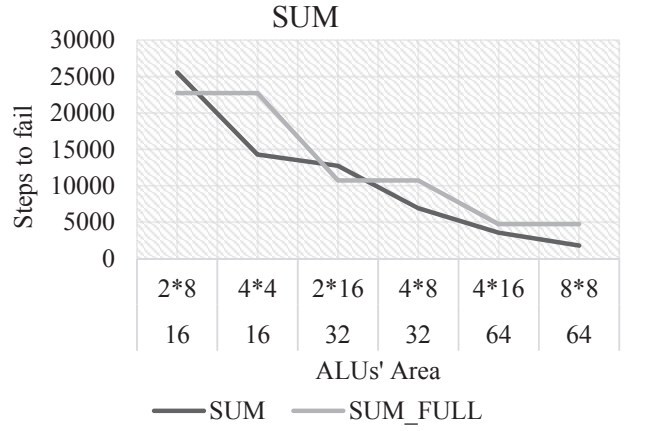


Fig. 14. Dependence of Steps to fail from combination of ALU component construction (for "simple" ALU)

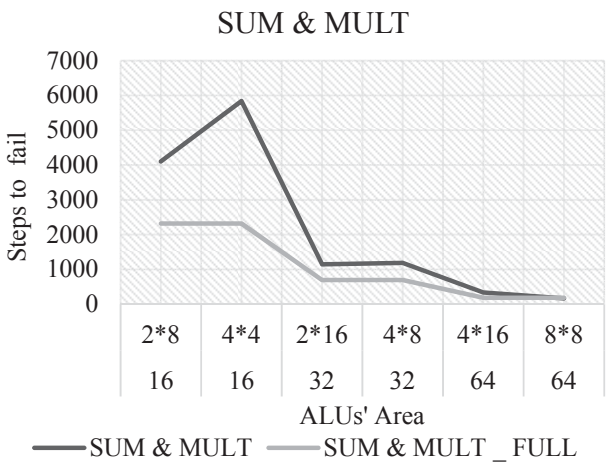


Fig. 15. Dependence of Steps to fail from combination of ALU component construction (for "complex" ALU)

Charts shows dependence of the failure from the ALU area. Calculations were made until the value to stay in F-state will rich 0.1 value. This value was chosen for the convenience of calculations, but it's enough to make conclusions of the method.

CONCLUSION

Selecting the NoC components redundancy scheme method based on design space exploration is proposed in this paper. Application of method in two types of redundancy is presented. We considered ALU component as the use case of our method. The proposed method may be used in other variants of components redundancy. It can be also applied for redundancy of different components' types (that are combinational schemes).

Our directions for further work are:

- Constructing mathematical models for evaluation of the system with 2 and more redundancy sub-components with different variants of redundancy;
- Research and evaluation of redundancy schemes at the level of topological design.

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), 2013.
[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] Pooria M. Yaghini, Ashkan Eghbal, Hossein Pedram and Hamid Reza Zarandi, "Investigation of transient fault effects in synchronous and asynchronous Network on Chip router", *Journal of Systems Architecture*, vol. 57, issue 1, Jan. 2011, pp. 61–68.
[5] Y.C. Chang, C.T. Chiu, S.Y. Lin and C.K. Liu, "On the design and analysis of fault tolerant NoC architecture using spare routers", in Proceedings of the Asia and South pacific design automation conference (ASPDAC), 2011, pp. 431–436.
[6] Yu Ren , Leibo Liu , Shouyi Yin, Jie, Qinghua Wu and Shaojun Wei, "A fault tolerant NoC architecture using quad-spare mesh topology and dynamic reconfiguration", *Journal of Systems Architecture*, vol. 59, 2013, pp. 482–491.
[7] C. Liu, L. Zhang, Y. Han and X. Li, "A resilient on-chip router design through data path salvaging", *in Proceedings of the Asia and South Pacific design automation conference (ASPDAC),* 2011, pp. 437–442
[8] D. Fick, A. De Orio, J. Hu, V. Bertacco, D. Blaauw and D. Sylvester, "Vicis: a reliable network for unreliable silicon", *in Proceedings of the ACM/IEEE design automation conference (DAC),* 2009, pp. 812–817.
[9] System Level Approach to NoC Design Space Exploration. R. K. Jena. *International Journal of Information and Electronics Engineering,* Vol. 2, No. 2, March 2012 / 5 p
[10] R. K. Jena and G. K. Sharma, "A Multi-Objective Evolutionary Algorithm Based Optimization Model for Network-on-Chip Synthesis," *in Proc. of 4th International conference on IT: New Generation,* April, 2-4, Las Vegas, Nevada, USA,2007 pp. 977-983.
[11] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast and Z. Wang, A NoC Tra_c Suite Based on Real Applications, *2011 IEEE Computer Society Annual Symposium on VLSI (ISVLSI),* Pages 66-71, ISSN 2159-3469, July 2011.
[12] Qualcomm, Snapdragon S4 Processors: System on Chip Solutions for a New Mobile Age, Qualcomm white paper, October 2011, https://developer.qualcomm.com/download/qusnapdragons4whitepaperfnlrev6.pdf, Retrieved 18 April 2014.
[13] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(4), 2005
[14] 14 Rozanov V. Suvorova E. Approaches to the SoC IP-blocks' Design with Errors' Mitigation.