DEPENDABILITY MODELS BASED ON PETRI NETS AND MARKOV CHAINS

Martin Kohlík
Information Science and Computer Engineering, 1st class, full-time study
Supervisor: Hana Kubátová

Faculty of Electrical Engineering, Czech Technical University in Prague
Karlovo náměstí 13, 121 35 Praha 2

kohlim1@fel.cvut.cz

Abstract. This paper shows a way to use stochastic Petri nets as formal availability models instead of Markov chains. Advantages of stochastic Petri nets over Markov models are illustrated on example models. The ability of stochastic Petri nets to represent the structure of modelled design and its use in further research is introduced.

Keywords. FPGA, dynamic reconfiguration, Markov chain, stochastic Petri net, formal availability model.

1 Introduction

FPGA-based designs are sensitive to many effects that can change their programmed function. [1] These changes are most unwelcome when designs are used in safety-critical applications, where the material loss or mortality can be caused because of their failure. The improvement of the reliability of the design is required to minimize the impact of such effects. The availability of the system is the probability that the system is operating at a specified time t.

The definition of the availability: [2]

$$A_{ss} = \frac{t_{uptime}}{t_{uptime} + t_{downtime}}$$

Time t_{uptime} corresponds to the expected value of the uptime of the system and $t_{downtime}$ corresponds to the expected value of the downtime of the system.

Markov chains [3] are the common way to calculate availability parameters. Markov chains are easy to create and they allow calculations of the availability of the design by solving a system of linear equations.

Basic Place-Transition Petri nets with immediate transitions are not designed to calculate availability parameters. Stochastic Petri net [4] is a subset of timed Petri nets that add nondeterministic time.

Stochastic Petri nets (SPNs) can be created as easy as Markov chains and allow the same calculations. Mathematical properties, that are available for Petri Nets, can be used to analyze SPNs. Moreover, SPNs are not only dependability models, they are able to represent the structure of the design. This feature will be used in further work in our research group.

The paper is structured as follows: A short introduction to Markov chains, stochastic Petri nets, the method how to transform Markov chains to Petri nets and vice versa and basic types of FPGA faults and errors are in Section 2. Section 3 contains example models. The conclusion and topics for further research are in section 4.

2 Background

2.1 Markov chains

Markov chains were originally proposed by the Russian mathematician Markov in 1907. Over the many decades since, they have been extensively applied to problems in social science, economics and finance, Computer science, computer-generated music, and other fields.

The steady-state distribution of probabilities of states is computed in reliability calculations. The distribution is the result of the system of linear equations.

Fig. 1 shows an example of Markov chain. It contains two states (*S1* and *S2*) and two edges (λ_1 and λ_2).

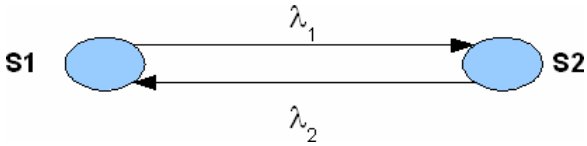

Figure 1: A simple Markov chain example.

The following system of linear equations corresponds to a previous example. The variable *p(Sx)* is the probability of the state *Sx*. The sum of probabilities of non-hazard states is the availability of the system.

$$p(S1).\lambda_1 - p(S2).\lambda_2 = 0$$
$$p(S1) + p(S2) = 1$$

2.2 Stochastic Petri nets and their extensions

Stochastic Petri nets are formed from Place-Transition nets by adding the transition rate of the transition t_i. It means that the firing time is exponentially distributed and the distribution of the random variable i of the firing time of transition t_i is given by

$$F(x) = 1 - e^{-\lambda_i x}$$

The average time for t_i to fire is $\frac{1}{\lambda_i}$.

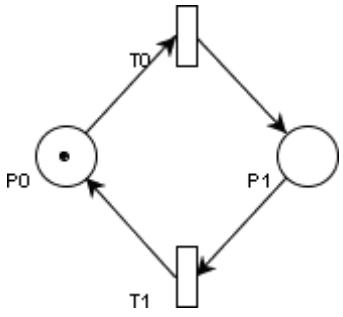

Figure 2: A simple SPN example.

A simple example of SPN is illustrated in Fig. 2. This SPN contains two places (*P0* and *P1*) and two transitions with an exponentially distributed firing time (*T0* and *T1*). When transition *T0* is fired, using the firing rule of Place-Transition nets, the token is moved from place *P0* to *P1*.

Generalized Stochastic Petri nets (GSPNs) [4] have two different classes of transitions: immediate transitions and timed transitions. Once enabled, immediate transitions fire in zero time. Timed transitions fire after a random, exponentially distributed enabling time as in the case of SPNs.

Fig. 3 shows an example of GSPN. Transition *T4* is an immediate transition. The other transitions are exponentially distributed. The arc heading from transition *T5* to place *P0* has weight set to 2 to ensure liveness of GSPN.

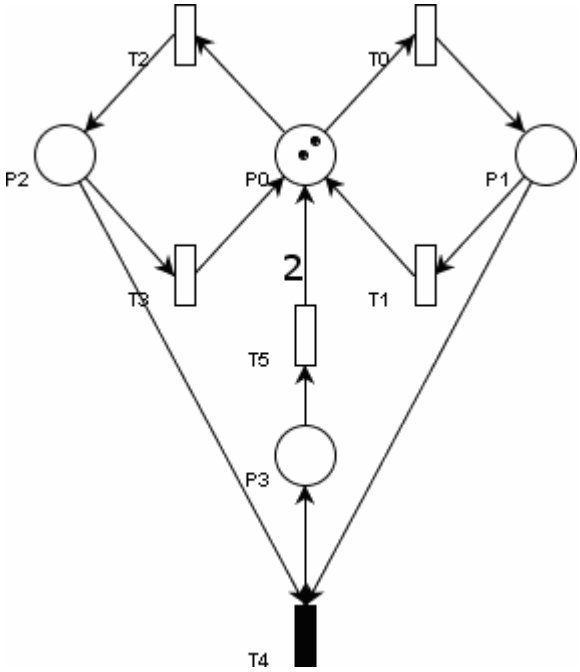

Figure 3: A GSPN example with one immediate transition.

Mathematical properties, that are available for Petri Nets, can be used to analyze both SPNs and GSPNs as well. All example models in this paper are live, bounded and reversible. [5]

- Boundeness: If the model is not bounded, then the set of reachable markings is infinite. The probability of any marking from this reachability set will approach 0.
- Reversibility: All proposed models suppose non-destructive actions that can be undone. Undoing the actions will lead the model to its original state.
- Liveness: Every transition in proposed models represents a certain action. The transaction, that cannot be fired, represents an impossible action. An impossible action does not need to be modelled. This fact along with the reversibility property implies the liveness property.

2.3 Transformation from Markov chain to stochastic Petri net and vice versa

Following sections briefly describe methods how to convert Markov chains into SPNs and vice versa. More detailed descriptions of the conversion to Markov chain are available in [4]. All examples in both forms (Markov chain and (G)SPN) lead to the same steady-state distribution results.

2.3.1 Markov chain to SPN

This transformation is very simple:

- Convert every Markov chain state into SPN place
- Convert every Markov chain edge into SPN transition (keep intensity rates)
- Add one token to Petri net place that corresponds to the Markov chain default state

2.3.2 SPN to Markov chain

This transformation is not as simple as the previous one. SPNs may have more than one token and they may contain transitions with multiple arcs. These two facts cannot be included in Markov chains.

The transformation is based on the reachability graph of SPN. The reachability graph condenses each marking of SPN to one state and eliminates transitions with multiple arcs.

This transformation is made as follows:

- Create reachability graph of SPN
- Convert every reachability graph state into Markov chain state
- Convert every reachability graph edge into Markov chain edge (keep intensity rates)

2.3.3 GSPN to Markov chain

This is the most complicated transformation. It is based on the reachability graph of GSPN, too. The reachability graph of GSPN contains vanishing states (states with at least one immediate transition enabled), that cannot be included in Markov chains. The reachability graph must be reduced and vanishing states removed. All edges leading to a removed vanishing state have to be connected to all edges starting from a removed vanishing state. The rates of edges have to be fixed as it is illustrated in Fig. 4.

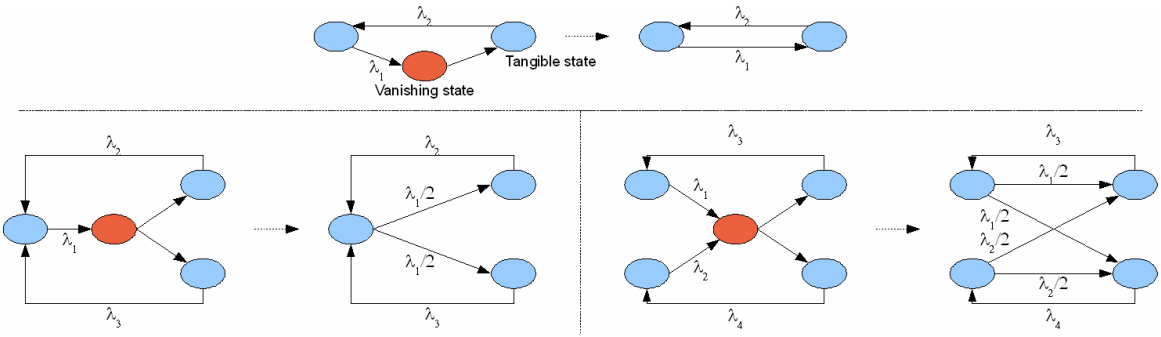

Figure 4: Examples showing the reduction of reachability graph.

2.4 FPGA in-operation faults and errors

The function of the FPGA may be changed during the operation due to many reasons. [1][6] Some of them are listed below:

- Aging
- Stress (voltage, heat …)
- High-energy particle impact

The effect of the impact of high-energy particle can be divided into the following groups:

- Destructive (Single Event Burnout …)
- Non-destructive (Single Event Upset …)

Only non-destructive Single Event Upsets (SEUs) will be taken into account in this paper. The SEU can change the configuration information – FPGA bitstream. This negative effect of the SEU can be removed by the reconfiguration of the FPGA. During the process of the reconfiguration, the original bitstream is restored. Some types of the FPGA are able to reconfigure its content dynamically. This feature allows the partial reconfiguration of the FPGA while the rest of the FPGA can be in the operational mode. The reconfiguration of the whole FPGA is required when the dynamic reconfiguration cannot be used. Some parts of the design are not possible or proper to reconfigure dynamically (e.g. module interconnections, module-pin connections …). A non-reconfigurable overhead may have a significant effect to the availability of the design.

The reconfiguration of the FPGA takes about 50 ms (it depends on the size of the reconfigured part of the FPGA and the period of the configuration unit's clock signal). If the design in the FPGA is not operational during the reconfiguration, the availability parameter of the design is decreased.

Both Markov chains and SPNs can be used to calculate the decrement. The availability of the system with a dynamically reconfigurable module is increased significantly in comparison to the system without a dynamic reconfiguration until the size of an overhead part does not get over critical size. [7] It is possible, that some design will have small overhead and will derive advantage from a dynamic reconfiguration. The other design will have a big overhead part and its advantage from a dynamic reconfiguration will be small or none.

3 Example models

3.1 Models conditions

Conditions for all example formal models mentioned in this paper are defined as follows: [7][8]

- Only a "single fault" may appear:
  - It will occur at a single time instant that is arbitrarily located at the time axis.
  - The fault can destroy a data item located within the configuration memory of the FPGA. The assumed "width" of the fault is one bit in a configuration memory. Every bit of the bitstream memory of the FPGA can be attacked with the same probability.
  - The time distance between any two successive faults is large enough to recover the system from the first one (otherwise it is a multiple fault).

- The design is defined as follows:
  - The design contents a dynamically reconfigurable module and an overhead part that cannot be reconfigured dynamically.
  - The module can be divided into n (e.g. 2 or 3) fractions with identical configuration data size or can be left undivided ($n = 1$).
  - Each fraction is able to detect the SEU that impacted it on. An overhead part has the same ability.
  - The reconfiguration unit loads a correct configuration data after the fault is detected. The time needed to reconfigure a faulty part depends on the configuration data size of the part. The reconfiguration unit is able to reconfigure each fraction of the module independently.
  - SEUs impacting an unused logic do not change the function of the used part. This type of impact is not considered in calculations.

Fig. 5 shows the schematic model of an example design. Possible fractions of a reconfigurable module are represented by green lines.

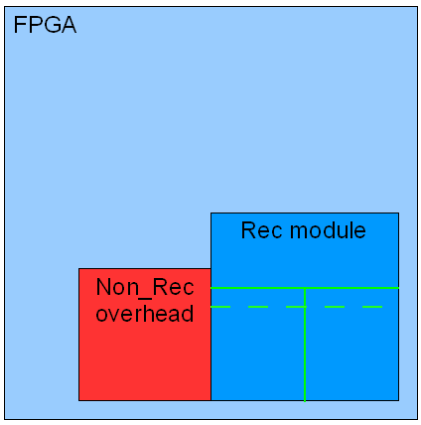

Figure 5: The schematic model of an example design.

3.2 Models comparison

An example GSPN model with a dynamically reconfigurable module undivided is illustrated in the left part of Fig. 6. In the right part of the same picture is corresponding Markov chain.

The model for an overhead part, that cannot be reconfigured dynamically, is located in the upper part of the left figure. SEU impact into the overhead part of the design is represented by *Non_Rec_SEU* transition. Firing this transition moves the token from *Non_Rec_OK* place to *Non_Rec_Error* place. *All_Repair* transition is enabled in a new marking. Firing this transition matches the end of the reconfiguration of the FPGA and moves the token from *Non_Rec_Error* place back to *Non_Rec_OK* place.

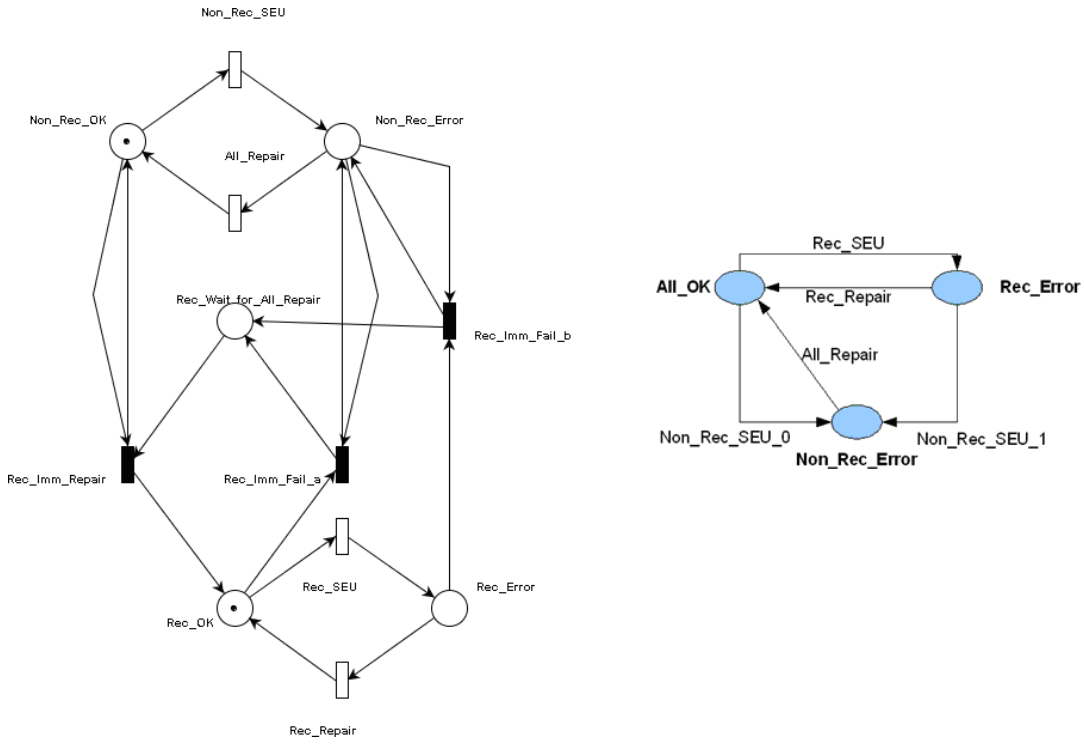

Fig. 6: An example GSPN and Markov chain with a dynamically reconfigurable module undivided

The model of a dynamically reconfigurable module is located in the lower part of the left figure. The meaning of places and transitions is very similar to the model for an overhead part. *Rec_Repair* transition corresponds to the dynamic reconfiguration of the module in this case.

Two immediate transitions (*Rec_Imm_Fail_a* and *_b*) located in the middle of the figure are purposed for removing the token from the lower part when the reconfiguration of the whole FPGA is running (the token is located in *Non_Rec_Error* place). There is no need to calculate a SEU probability or the reconfiguration process of a dynamically reconfigurable module during the reconfiguration of the whole FPGA. A third immediate *Rec_Imm_Repair* transition is purposed to return the token to the lower part after the reconfiguration of the whole FPGA has been finished (the token has been returned to *Non_Rec_OK* place).

Markov chain contains three states that correspond to states of the system. Edges correspond to the same-named transitions of GSPN. Edges *Non_Rec_SEU_0* and *_1* correspond to the same event. It looks simpler than corresponding GSPN, but does not contain any sign of the structure of the design.

Fig. 7 shows an example GSPN model with a dynamically reconfigurable module divided into 2 fractions.

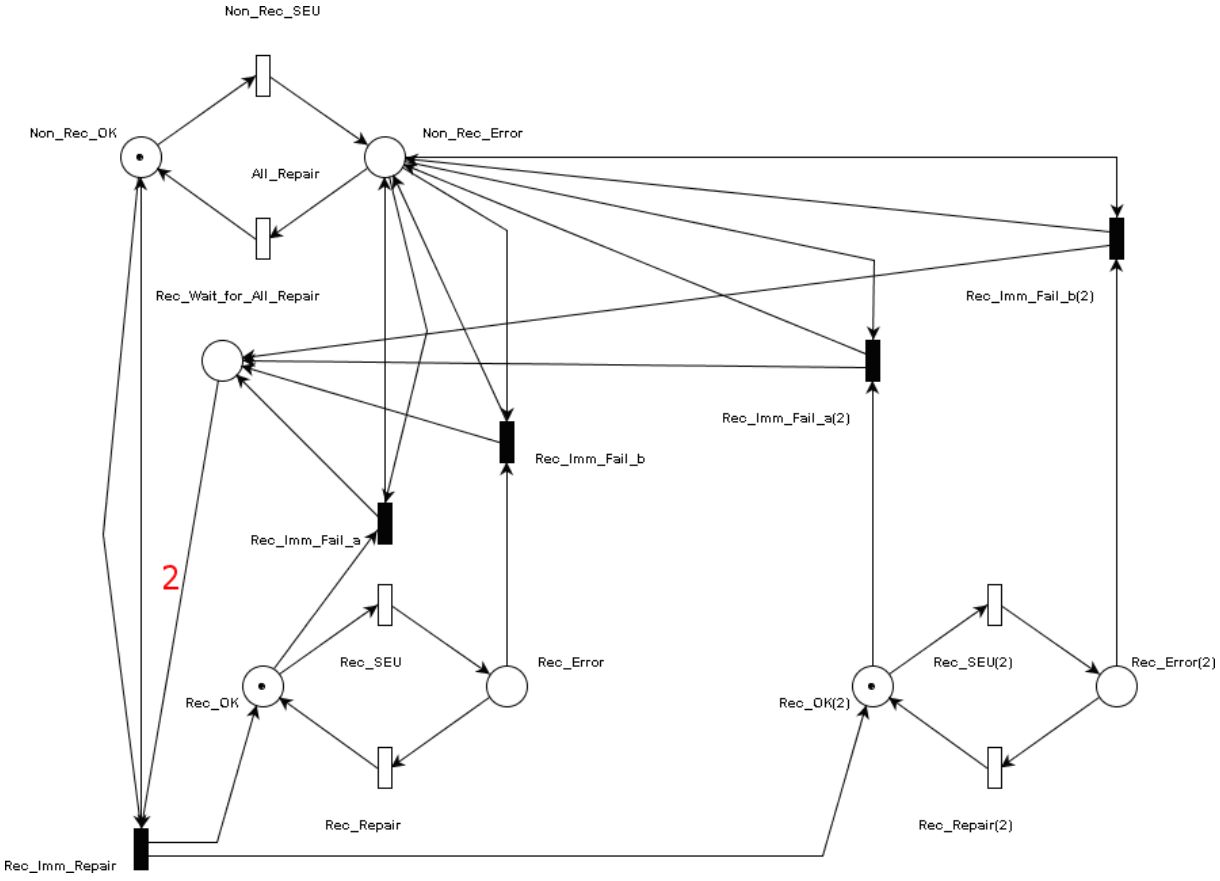

Fig. 7: An example GSPN chain with a dynamically reconfigurable module divided into 2 fractions.

This model is similar to that one shown in Fig. 6. The model of the second fraction of a dynamically reconfigurable module is located in the right part of the figure. It contents same places and transitions with same functions. The arc leading from *Rec_Wait_for_All_Repair* place to *Rec_Imm_Repair* transition has increased weight to ensure boundeness of the model. Both simple models of fractions of a dynamically reconfigurable module and the model of the overhead part of the

design can be found in GSPN. A similar method is used to extend GSPN to create the model for 3 and more fractions.

Fig. 8 shows an example Markov chain corresponding to GSPN shown in Fig. 7. This Markov chain cannot be created from that one shown in Fig. 6 by adding new part, it have to be revised completely.

Markov chain for more fractions looks like an *n*-dimensional cube with one place corresponding to the error of the overhead part of the model. Each vertex of the cube represents one combination of error and non-error states of all fractions of a dynamically reconfigurable module. A 2D cube (square) can be found in Fig. 8.

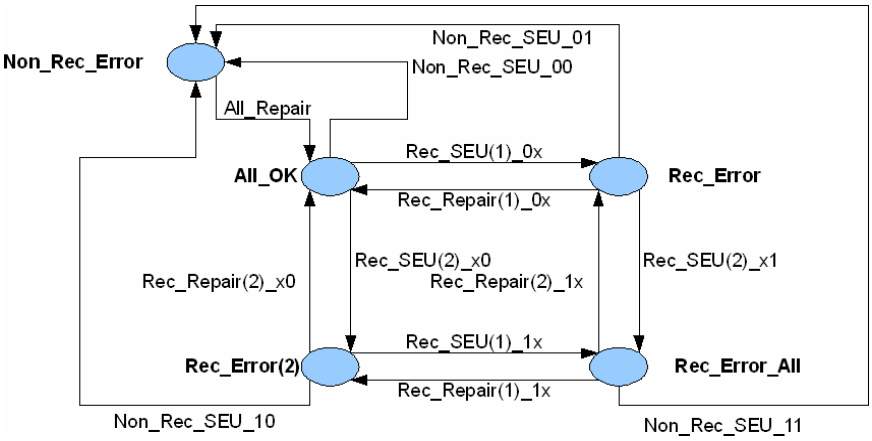

Fig. 8: An example Markov chain with a dynamically reconfigurable module divided into 2 fractions.

4 Conclusion and future work

Stochastic Petri nets and their use in reliability modelling are described in this paper. Examples presented in this paper have been used to calculate the availability of the system with a dynamically reconfigurable module and non-reconfigurable overhead. Detail results are available in [7]. One of advantages of GSPNs over Markov chains – the ability to represent the structure of a modelled design – will be used in further research.

The reconfiguration of the FPGA is a deterministic process, but it is represented as the transition with non-deterministic time. This approximation may cause errors in results. The calculation of the relevance of this error will be the first step in research.

DUBs – Dependable Universal Blocks [9] – will be used to build complex designs in our research group. Each of these DUBs will have its own security part and GSPN model depending on its structure. We will try to create the method to combine models of DUBs to one model of whole complex design. The next goal will be the automation of the process.

Acknowledgment

This research has been supported by MSMT under research program MSM6840770014.

References

[1] Kastensmidt, F. L., Carro, L., Reis, R.: Fault-Tolerance Techniques for SRAM-based FPGAs, 2006, 978-0-387-31068-8.

[2] Hlavička, J.: Spolehlivost a diagnostika (in Czech), 1989.

[3] Basic Probability Theory and Markov Chains, 2006, 978-3-540-27642-5.

[4] Bause, F., Kritzinger, P.: Stochastic Petri Nets – An Introduction to the Theory (2nd edition), 2002, 3-528-15535-3.

[5] Murata, T.: Petri Nets: Properties, Analysis and Applications, 1989.

[6] Novák, O.: Úvod do diagnostiky, základní pojmy (the lecture of 36DSP course; CTU in Prague; in Czech), 2008.

[7] Kohlík, M., Kubátová, H.: Reconfiguration Strategy for FPGA Dependability Characteristics Improvement based on Stochastic Petri Net, 2009.

[8] Kubalík, P., Kubátová, H.: Dependable Design Technique for System-on-Chip, 2008.

[9] Borecký, J: Dependable universal blocks (DUBs) (in Czech), 2009.