Software Reliability Growth Model for FPGA-based Safety Critical Software System

Satrio Pradana, Jaecheon Jung*
Department of Nuclear Power Plant Engineering
KEPCO International Nuclear Graduate School, Ulsan, South Korea 45014
**Corresponding author: jcjung@kings.ac.kr*

1. Introduction

In the NPP design, FPGA technology is mainly applied for safety critical I&C systems such as the Reactor Protection System (RPS). Recently The FPGA technology is more and more extensively applied both for the new NPP I&C system design and for updating the obsolete systems of operating plants especially the safety system. The role of FPGA based systems has become very significant; therefore, the reliability evaluation of the FPGA based systems has drawn the attention of researchers [1].

As digital system are continuously being introduce into nuclear power plants, the needs of reliability analysis for digital system is increasing. Kang and Sung [2] identified (1) a piece of software's reliability, (2) common-cause failures (CCFs), and (3) fault coverage as the three most critical factors during the reliability analysis of digital systems. For a reliability estimation of the safety-critical software (the software that is used in safety-critical digital systems), the FPGA based need an approach to estimate the reliability and predicting the failure of software.

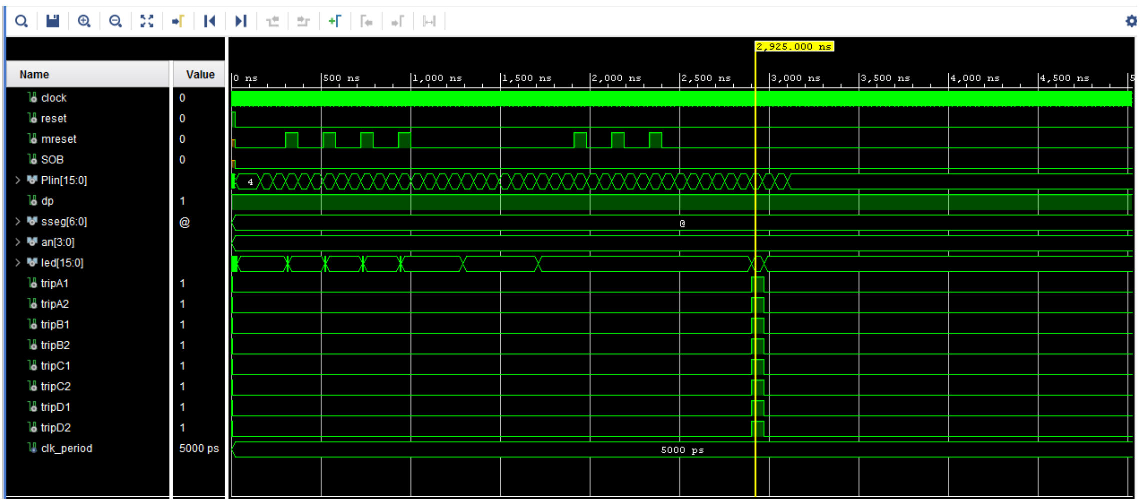
In this work, an attempt is made to analyze the reliability of FPGA based system considering the software reliability growth model (SRGM) methodology – as the NRC Technical reference NUREG/CR-6101 reports and IEEE Std. 1633 that the SRGM is one of the possible methodologies to model the instrumentation and control system [3][4].

2. Methods

Several steps have to be taken in order to obtain failure data and analyze the reliability. Plant Protection System (PPS) are one of safety critical safety system in Nuclear Power Plant. It has a function to trip the reactor through bistable logic controller and generate trip signals based on the measurement channel value exceeding a setpoint then transmit the signals to the Coincidence Processor (CP) located in four redundant channels. PPS receives sixteen (16) signals indicating safety-related plant conditions; fourteen (14) analog signals and two (2) digital signals. Besides the sixteen signals, manual trip signals by operator are provided. In this work, we only focus on Low Pressurizer Pressure Trip (LPPT).

2.1 Test Case

Safety critical software applications often require proof that they have been thoroughly tested. Hence, programmers and testers are expected to write good test cases [5] which can verify the behavior of the entire system. However, in real life applications, exhaustive testing is impractical as the input domain could be extremely large or infinite. Thus, the main challenge is to demonstrate the adequacy of testing effectively.



Name	Value
clock	0
reset	0
mreset	0
SOB	0
Plin[15:0]	
dp	1
sseg[6:0]	@
an[3:0]	
led[15:0]	
tripA1	1
tripA2	1
tripB1	1
tripB2	1
tripC1	1
tripC2	1
tripD1	1
tripD2	1
clk_period	5000 ps

Figure 1. Vivado simulation for PPS LPPT

According to IEEE 1633-2016, Recommended Practice on Software Reliability, test cases from black box testing can be used as operational profiles to support test selection both for collect failure data and verification & validation activities.

Test cases were then combined into test benches for the entire system. The design simulation was performed using Vivado. Figure 1 illustrates the simulation environment. Based on the generated waveforms, the verification of the operation of the different system modules was performed. All the faults from this testing phase are gathered. It helps collected and making a dataset to be applied in software reliability growth model.

2.2 Test Bench

A well-established test benches are required to get high quality of testing.

2.3 White Box Testing

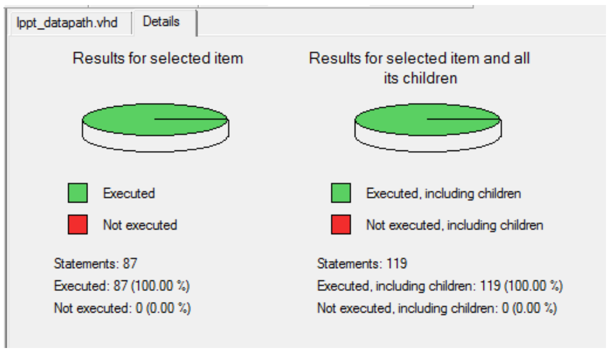
White-box testing is a method of testing the application at the level of the source code. It focuses on internal coding, flow of inputs and outputs through the application. White-box testing is developed with the test cases by executing methods and often used for verification phase by the programmer or independent test. Since this development of FPGA-based PPS has the VHDL code, the white-box testing can be used for verification process. In the other way, black box testing is the functional and behavioral testing, focuses on determining whether or not a program does what it is supposed to do based on its functional requirements [6].

Instead of black box testing, white box testing are used to collect failure data [7] as well as for V&V activities. In this research, white box testing was performed for PPS LPP bistable function using VIVIDO simulator and Aldec HDL Program. Figure 2 show a simulation of white box testing using Aldec HDL program.

This white box testing is only focus on internal behavior from input and output. Simulation on Vivado and Aldec are behavioral simulation. Syntesis, place & route for actual implementation of FPGA is not covered in this thesis.

Code coverage is a technique that allows engineers to collect the statistics on the execution of each line of HDL code, and evaluate the quality of their test. Code coverage can be roughly divided into statement coverage and branch coverage. Statement coverage provides information on which statements inside the VHDL or Verilog code were executed during simulation and how many times. Branch coverage examines the execution of conditional statements. It provides the data on which branches were executed during the simulation, how many times each branch was executed, and how many times the branch condition evaluated to true or false [8].

Code coverage and functional coverage are extensively used during validation to evaluate the effectiveness of testing. Ideally, designs must reach 100% statement, branch and functional coverage, however exceptions are made if it is known that a given coverage point is unreachable or not important.


lppt_datapath.vhd | Details

Results for selected item	Results for selected item and all its children
🟩 Executed	🟩 Executed, including children
🟥 Not executed	🟥 Not executed, including children
Statements: 87	Statements: 119
Executed: 87 (100.00 %)	Executed, including children: 119 (100.00 %)
Not executed: 0 (0.00 %)	Not executed, including children: 0 (0.00 %)

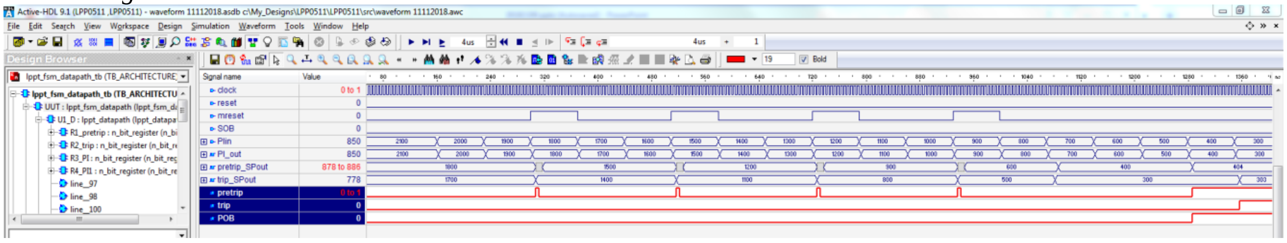
Figure 3 Coverage testing result

PPS LPPT VHDL test cases are generated to meet the 100% code coverage. ALDEC software tool is used to perform the code coverage test. Figure 3 shows coverage testing result.

3. Software Reliability Tools

The Several software reliability tools are available to apply one or more of the software reliability model to a development effort and to determine the applicability of a particular model to a set of failure data. A major issue in modeling software reliability lies in the ease-of-use of currently available tools.

2.4 Coverage Test


Active-HDL 9.1 (LPP0511 ,LPP0511) - waveform 11112018.asdb c:\My_Designs\LPP0511\LPP0511\src\waveform 11112018.awc

Signal name	Value
clock	0 to 1
reset	0
mreset	0
SOB	0
PIin	850
PI_out	850
pretrip_SPout	878 to 886
trip_SPout	778
pretrip	0 to 1
trip	0
POB	0

Figure 2 White box testing of LPPT using Aldec HDL Program

Following tasks are handled by the SRE tools:

a) Collecting failure and test time information
b) Calculating estimates of model parameters using information available.
c) Testing to fit a model against the collected information.
d) Selecting a model to make predictions of remaining faults, time to test, etc.
e) Applying the model

As a recommendation from literature review as shown in the comparison [9] Table 1 and according to the availability of using free version of the tool, SMERFS tool has been selected.

Table 1. Comparison table of tools

Factors			CASRE		SMERFS		SOFTREL		MEADEP		SRMP		SOREL		SREPT		SRTPRO	
Language			FORTRAN		FORTRAN		C		VC++		FORTRAN		PASCAL		JAVA		C#	
Performance & usability	Reliability of failure rate		Y		Y		Y		Y		Y		Y		Y		Y	
	Total Failure		Y		Y		Y		N		Y		Y		Y		Y	
	Remaining failure		Y		Y		Y		N		Y		Y		Y		Y	
Number of models supported			16		12		2		1		9		4		1		14	
Available models for	Estimation	Prediction	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Graphics			N		N		N		Y		N		N		Y		Y	
User assistance			Y		Y		Y		Y		Y		Y		Y		Y	

The software reliability prediction tool is SMERFS (Statistical Modeling and Estimation of Reliability Functions for Software), a well-known and widely accepted software application for evaluation of test data for failure rate and defect discovery rate prediction. The version of SMERFS used in this study included a total of 15 different reliability growth models. The input to SMERFS is a set of values consisting either of the time between discoveries of defects or the number of defects discovered per time period.

SMERFS then uses maximum likelihood methods or least squares methods to estimate the parameters used for one or more of these models (depending on the type of input and user selected options). Its output includes the parameter estimates, predicted values and a measure of the goodness-of-fit using the chi-squared distribution.

SMERFS is used to evaluate the different models and validate the software reliability according to the following steps: Perform Assessment/ Prediction Analysis and Forecast Additional Test duration. There are two types of models in SMERFS:

a) Interval data counts models
Typical interval data count models, including Generalized Poisson model (GPO), Brooks and Motley Poisson model (BMP), and binomial model (BMB), GO (also called NHPP) model, and Yamada delayed S-shaped model (YAM).
b) Failure-count models.
Typical FC models, including LV, geometric model (GEO), Musa Basic (MB), Jelinski/Moranda (JM), and Musa Okumoto (MO) [10].

4. Predicting Software Reliability Growth Model

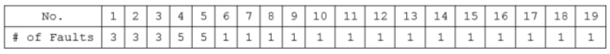
For this task, SMERFS program are used. SMERFS pull-down menus are concise and leave little room for misunderstanding. Firstly, specify whether the input is time-between-failure or interval data. For interval models, failure counts for every interval, even those without failures, must be provided in a text file with care not to have a blank line at the end of the file. The models can be selected and executed but better to let SMERFS do the selections with accuracy analyses for the models. If the data are grossly inappropriate for a model, the tool will informs.

The following models are abbreviated in SMERFS Program

BMB	: Brook and Motley's Binomial
BMP	: Brook and Motley's Poisson
GP-(1-3)	: Generalized Poisson (Treatment 1-3)
NHP	: Non-homogeneous Poisson
S-T(1-3)	: Schneidewind (Treatment 1-3)
YAM	: Yamada S-shaped

The data collected during 19 times execution of testing. When considering the whole dataset, the Table 2 shows the data gathered in format count of failure by interval. Failure data obtained during development phase. From several testing, data was chosen for 19 execution times with 33 failures.

Table 2 PPS LPPT Failure Dataset



No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
# of Faults	3	3	3	5	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1

From dataset given above, SMERFS program performs prediction for software reliability growth with accuracy analyses. After execution, Figure 4 shows the graph from all the model executed. The green nodes indicates the observed faults, and the graphs with different color indicates prediction of the models.

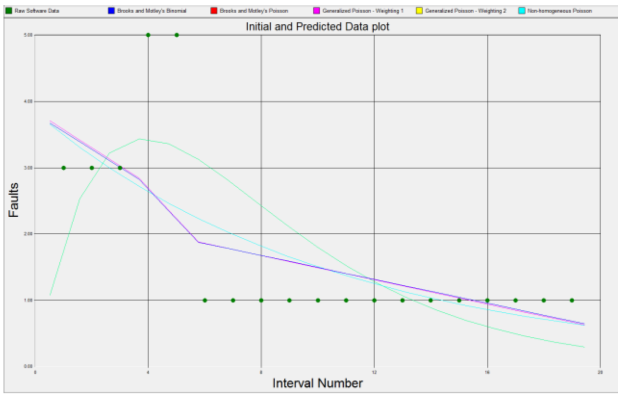
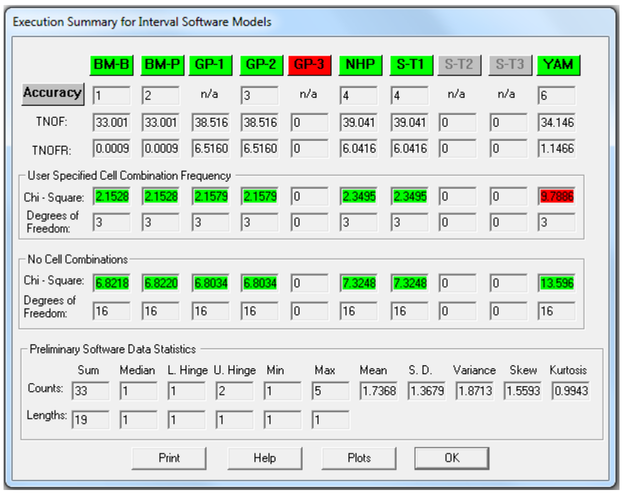


Figure 4 Failure Predicted Data plot from SMERFS program


Execution Summary for Interval Software Models

	BM-B	BM-P	GP-1	GP-2	GP-3	NHP	S-T1	S-T2	S-T3	YAM
Accuracy	1	2	n/a	3	n/a	4	4	n/a	n/a	6
TNOF:	33.001	33.001	38.516	38.516	0	39.041	39.041	0	0	34.146
TNOFR:	0.0009	0.0009	6.5160	6.5160	0	6.0416	6.0416	0	0	1.1466
User Specified Cell Combination Frequency										
Chi - Square:	2.1528	2.1528	2.1579	2.1579	0	2.3495	2.3495	0	0	9.7886
Degrees of Freedom:	3	3	3	3	0	3	3	0	0	3
No Cell Combinations										
Chi - Square:	6.8218	6.8220	6.8034	6.8034	0	7.3248	7.3248	0	0	13.596
Degrees of Freedom:	16	16	16	16	0	16	16	0	0	16

Preliminary Software Data Statistics

	Sum	Median	L. Hinge	U. Hinge	Min	Max	Mean	S. D.	Variance	Skew	Kurtosis
Counts:	33	1	1	2	1	5	1.7368	1.3679	1.8713	1.5593	0.9943
Lengths:	19	1	1	1	1	1					

Figure 5 Summary for execution of dataset

Figure 5 shows the summary for execution of dataset. It gives statistics of the data, accuracy rank, total number of faults (TNOF), total number of faults remaining (TNOFR), and chi squares are also included. Based on execution result, almost all model are executed except Generalized Poisson-3 (GP-3). It marked by red color that shown in Figure 5, means the model are not executed by reason of model are not suit for the dataset. For Schneidewind Treatment 2 (S-T2) and 3 (S-T3) model are marked by grey color, means the dataset are not applicable for the model.

5. Conclusion

This paper presents an approach for assessing a reliability measurement of safety critical software FPGA-based for plant protection system. The quality of model of the software reliability model also presented based on several test in verification and validation activities of FPGA-based system. The approach require numerous testing and management engineering before beginning of testing. Evaluating the model of software reliability growth is important to select the best model are fit the observed fault trends. After model are selected, reliability estimation can be performed to reduce the number of iteration in fixing the failure during development and reduces the likelihood of design errors because it allows tests quality to be increased in respect to an objective measurement.

Acknowledgements

This work was supported by the 2018 Research fund of KEPCO International Nuclear Graduate School (KINGS), Republic of Korea.

REFERENCES

[1] Ma Z., Yoshikawa H., and Yang M., 2017, “The Reliability Model for the FPGA-based Instrument and Control System Using Colored Petri Net”, Proceedings of 25th International Conference on Nuclear Engineering ICONE25, Shanghai, China July, 2017
[2] H. G. Kang and T. Sung, “An Analysis of Safety-Critical Digital Systems for Risk-Informed Design,” Reliability Engineering and System Safety, Vol. 78, No. 3, 2002.
[3] U.S. Nuclear Regulatory Commission, "Software Reliability and Safety in Nuclear Reactor Protection Systems," U.S. Nuclear Regulatory Commission, Washington, DC, NUREG/CR-6101, June 1993. [Online]. https://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr6101/cr6101.pdf
[4] “IEEE Recommended Practice on Software Reliability,” IEEE Std 1633, 2008.
[5] C. Kaner, “What is a good test case,” Relation, vol. 10, no. 1.100, p. 5569, 2003.
[6] Williams, L., (2008). “A (partial) introduction to software engineering practices and methods”, vol. 2009
[7] K. K. Mohan, ‘’ Integration of Black-Box and White-Box Modeling Approaches For Software Reliability Estimation” International Journal of Reliability, Quality and Safety Engineering Vol. 17, No. 3 pp. 261–273, 2010
[8] ALDEC, Collecting Code Coverage in Active-HDL, August, 2018
[9] Manohar Singh “Software Reliability Testing Tools: An Overview and Comparison” International Journal of Engineering And Computer Science Vol. 5 Issue 11, Page No. 18886-18891, Nov. 2016.
[10] Dolores R Wallace “Practical software reliability modelling”, Proceedings 26th Annual NASA Goddard Software Engineering Workshop, USA, Nov 2001