ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 515-521
doi: 10.18178/wcse.2019.06.076

A Method for the measurement of FPGA software safety in its whole life cycle

Xiaohui Jiang [1], Chuyuan Peng[+ 2], Yong Hu [1] and Wei Meng [2]

[1] Beijing Research Institute of Mechanical & Electrical Technology
[2] Beijing Jinghang Institute of Computing and Communication

Abstract. In this article a method for the measurement of FPGA software is proposed using the analogy of the methods from traditional software safety. The full life cycle of FPGA software development is decomposed into each single process according to its different stages and functions. The features of each process is analyzed separately. Some other features are also considered to ensure the model fits real requirements of FPGA software safety measurement. Generally the measurement method is acquired through making two different calculations. An example is extracted from a large FPGA project used in aerospace to better illustrate the measurement process. The measurement method is demonstrated for each process of its whole life cycle. By comparing the value generated by the method with specialists' ranking value, this method shows its strong reliability when treating complicated FPGA software safety problems.

Keywords: Software metrics, FPGA software safety, Safety measurement

1. Introduction

The development process and technical requirements of FPGA software are quite different from traditional software as the former includes HDL code, electronics, circuit timing, interface protocol, verification, EDA tool and other aspects. Although the FPGA software and traditional software differ a lot, there are also similarities between them, such as: system requirements, requirements analysis, design implementation, verification and validation, product delivery acceptance, maintenance and engineering management. In the process of implementation, similar to traditional software, FPGA software can also be coded using a variety of programming language to achieve the design requirements. In view of the above similarities, in this paper, theory and technology of traditional software engineering are combined as a reference in this research of the FPGA software safety metrics.

The industry has conducted in-depth research and extensive use of the measurement and evaluation of software quality, but the software quality model can only be used to measure the external quality or internal quality of the released software products. Thus cannot well reflect how to measure the quality during the process of software design and development. By referencing to the software quality model and combining the related requirements in the process of the FPGA development, this paper proposed a measurement model for the whole life cycle of FPGA software development. The measurement model can be used during the development process to measure the software safety as well as to recheck or modify the FPGA software safety problems after the product delivery.

[+] Corresponding author. Chuyuan Peng
E-mail address: chuyuan.peng@qq.com

2. The full life cycle of FPGA development

The full life cycle of FPGA includes the following processes: system requirements process, requirements analysis process, design process, implementation and integration process, validation process, test and verification process, delivery and acceptance process, operation and maintenance process.

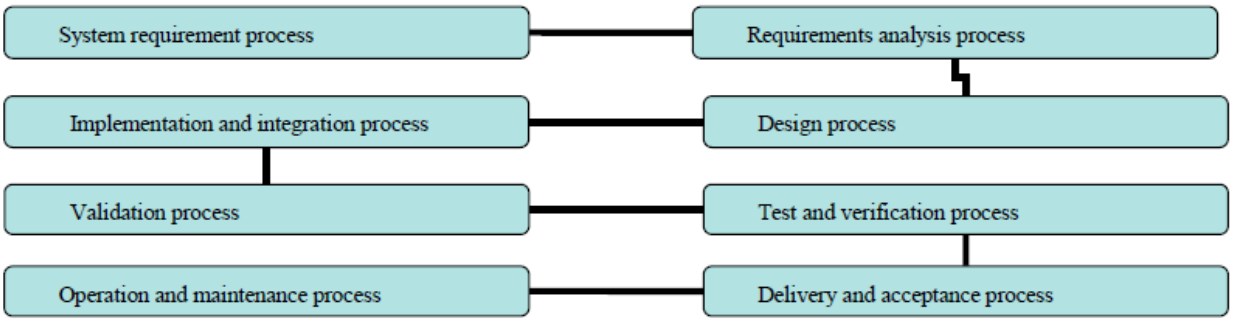

Fig. 1 : Full life cycle process.

Definition and brief requirements for each process.

(1) System requirements process: analyse system, equipment requirements for FPGA, FPGA definition, clarify key levels, determine FPGA operating environment and proposal of FPGA requirements including function, performance, reliability and safety,.

(2) Requirements analysis process: this process is used to identify and record the requirements of FPGA. It contains the description about FPGA architecture, technology selection, functionality, performance, interface, environment, reliability and safety requirements. For example: communication protocol, timing requirements, resource margin and time sequence margin, etc. These requirements should be recognizable, well defined and verifiable.

(3) Design process: The design process includes the outline design process and detailed design process. According to the specification requirements of FPGA, the outline design process establishes the architecture of FPGA, and define the functions and performance indicators of each module in the architecture. The detailed design process is to carry out detailed design for each module generated in the outline design, divide each sub-module, determine the design method of each sub-module and carry out design activities. For example, determine the appropriate clock and reset scheme, describe the referenced IP kernel and its attributes.

(4) Implementation and integration process: Design input, logic synthesis, layout and wiring are conducted in the process of implementation and integration according to the detailed FPGA design specifications, and eventually generate downloadable configuration files, then solidify them into on-board configuration memory.

(5) Confirmation process: Carry out system test of FPGA. System tests include loading the configuration file into the target system, running in the target system environment, and testing whether the functions meet the requirements of the task through external observation equipment.

(6) Testing and validation process: Check the results of the FPGA activities in different stages to determine conformance with the specified requirements of the activity.

(7) Delivery and acceptance process: Delivery and installation, acceptance for FPGA.

(8) Operation and maintenance process: After FPGA delivery and acceptance, due to the defect of the FPGA or new requirements assigned by assignment agent, the task holder implements FPGA software corrective maintenance, integrity maintenance and adaptive maintenance, to eliminate defects or meet the demands.

3. FPGA Software Safety Measurement Model

FPGA software safety work are conducted throughout the whole life cycle of the FPGA development. Among them, the system requirements process, requirements analysis process, design process, implementation and integration process, testing and validation process is the key point of this article. Further

research content mainly focus on these five process to establish the FPGA software safety metrics. The process of system requirements and demand analysis are similar as they both propose requests on the function of FPGA, performance, interface, reliability, and safety, so the two processes are discussed together.

(1) System requirements process and requirements analysis process.

The main task of system requirements and demand analysis is to analyse and determine the requirements of the FPGA, including the requirements of hardware assigned by the system and the requirements FPGA derived demand generated in the whole life cycle. Derived demand should be able to feedback to the system requirements in the process, in order to assess the effect of derived demand on the system requirements. From the perspective of FPGA safety, the safety requirements of FPGA are divided into performance requirements, functional requirements, data requirements, interface requirements, etc. The safety requirements of FPGA should be correct and complete. Full compliance of FPGA safety requirements means that FPGA is reliable and safe enough. Therefore, the premise of FPGA's reliability and safety is to ensure the correctness and completeness of safety requirements.

(2) Design process.

The design process of FPGA includes the outline design process and detailed design process. Outline design finish the system top-level design, such as module function block diagram design, structure description and design sketch, etc. Detailed design process is based on the conceptual design process. It designs in detail each module generated in outline design, divide them into sub modules, determine the design method of each module, and carry out design activities

The process of safety analysis of FPGA design process is to analyse whether its design meets the safety requirements. The analysis objects include the interface design, communication design, module design, clock use, IP core use, redundant fault-tolerant design and so on. In the FPGA design process, we should strictly abide by the safety design criteria, so as to effectively ensure the safety of FPGA. In addition, the traceability relationship between design process and requirement analysis process should be established to ensure that all safety requirements are realized in the design process.

(3) Implementation and integration process

FPGA implementation and integration process contains four sub process: design input, logic synthesis, layout design and configuration loading. In the FPGA implementation and integration phase, in addition to the use of hardware description language (HDL) to complete the RTL coding, a variety of professional tools are needed. Such as RTL code synthesis tools, layout, wiring, configuration file generation tools, etc. Coding and the use of tools and asked could introduce different levels of safety.

(4) Testing and verification process

The testing and verification process is an important stage to improve the safety of FPGA. Static test and dynamic test are conducted and the quality of test work will affect whether the FPGA can work safely after it is put into use. Therefore, the two measures of "safety test adequacy" and "safety test coverage" are very important. Whether the test is sufficient and comprehensive is related to whether the safety of FPGA software is guaranteed.

4. FPGA Software Safety Metrics

4.1. Metric acquisition.

The content of the FPGA software safety measurement is called the FPGA software safety metrics. Since the calculation parameters for the metrics involves lots of data, it is necessary to obtain the measurement data through a variety of means. Mainly including: review, testing and expert rating. The following three methods are introduced respectively:

(1) Reviewing.

A review is an independent examination to assess whether an FPGA software product meets the requirements of the FPGA software requirement specifications and design specifications. After reviewing FPGA software code, FPGA software assignment book, FPGA software requirement specifications and FPGA software design specifications, the quantified review results yields some values of the metric.

(2) Testing.

The testing here includes both the FPGA function verification test and the data obtained by using EDA tools. Some metrics' value can be obtained from the testing data or results.

Functional verification test includes: verifying whether all functions of FPGA software can be implemented correctly, verifying whether performance and interface requirements are satisfied, verifying compliance with requirements specifications and design specifications, etc.

The data obtained by the EDA tool include: using simulation tool Questasim to get statement coverage and branch coverage, using static timing analysis tool Prime Time to get time series analysis report.

(3) Expert rating.

Experts are invited to score FPGA safety based on the FPGA software safety metric input, and the score obtained is the comprehensive evaluation result of FPGA software safety. Since this method has some subjective and artificial factors, many experts should be invited to grade it. The obviously too high or too low grade should be removed, and take the average value to obtain objective safety evaluation results.

4.2. Evaluation method for metrics.

The basement of quantitative measurement on FPGA software safety is to define the way of obtaining the FPGA software safety metrics. In this paper the measurement metrics and the FPGA software safety evaluation results are chosen in the closed interval [0, 1]. And the closer measurement value is to 1, the better measure result it indicates. There are two approaches: binary values and ratio values.

(1) The rule of binary value is: the value of binary value is either the maximum value 1 or the minimum value 0. When the answer to a question is "yes", the value of measurement element is 1; when the answer to a question is "no", the value of measurement element is 0.

(2) The rule of ratio value is as follows: for metric that is the larger the better, the ratio is obtained by dividing the number of verified statistics objects by the total number of statistical objects, shown as "A/B". For metric that is the smaller the better, the ratio is obtained by 1 minus the number of verified statistics objects divided by the total number of such statistics objects. The ratio is expressed in the form of "1-A /B".

For measure value that is the larger the better, and the upper and lower bounds are given, the rule to set the value is:

$$\text{measure value} = \frac{actual\ measure\ value - lower\ bound}{upper\ bound - lower\ bound} \tag{1}$$

If the actual measured value is less than the lower limit value, the metric element value is 0. If the actual measured value is greater than the upper limit value, the metric element value is 1.

For measure value that is the smaller the better and the upper and lower bounds are given, the rule to set the value is:

$$\text{measure value} = \frac{upper\ bound - actual\ measure\ value}{upper\ bound - lower\ bound} \tag{2}$$

If the actual measured value is less than the lower limit value, the metric element value is 1. If the actual measured value is greater than the upper limit value, the metric element value is 0.

4.3. Extraction of FPGA software metrics.

Based on the above discussion and the problems encountered in the process of FPGA evaluation and accumulated experience, FPGA software safety metrics are extracted as shown in table 1. Should be noted that the metrics do not necessarily apply to all the FPGA software. When measuring a certain FPGA software safety, the metrics should be reasonable selected according to specific circumstances.

Table. I: FPGA software safety metric

Development Progress	Safety metrics category		Name of safety measure element	Method of measuring element value
System requirement process and demand analysis process	Performance requirements safety		1.Margin requirement	Binary:1
			2.Timing relations between signals	Binary:1
			3.FPGA working environment	Binary:0
	Functional requirement safety		4.Safety critical function identification	Binary:1
			5.Safety critical function assurance measures	Binary:0
			6.Anti single event effect design	Binary:0
	Data requirement safety		7.Data definition	Binary:1
			8.Data acquisition requirements	Binary:1
			9.Important data validation	Binary:1
	Interface requirement safety		10.Interface characteristics analysis	Binary:0
			11.Interface transmission key signal description	Binary:1
Design process	Interface design safety		12.Interface consistency	Ratio:9/9=1.00
			13.Interface fault tolerance	Ratio:12/12=1.00
			14.Correctness of signal connection between modules	Binary:1
			15.Validity of input data	Ratio:17/19=0.89
	Communication design safety		16.Protocol conformance	Ratio:3/4=0.75
	Module design safety		17.Module level	Binary:1
			18.Separate storage logic into modules	Binary:1
	Clock usage safety		19.Avoid direct use of combinational logic to generate clocks	Ratio:1/1=1.00
	IP core usage safety		20.IP kernel verification	Binary:0
	Redundancy fault tolerance design safety		21.Redundant design of safety critical functions	Ratio:2/3=0.67
	Traceability of safety requirements		22.Safety requirements traceability	Ratio:7/7=1.00
Implementation and integration process	Design input safety	Cross clock domain processing safety	23.Signal synchronization in cross clock domain	Ratio:3/6=0.50
		Reset and initialization safety	24.Asynchronous reset synchronous release	Ratio:13/15=0.87
			25.Register initialization	Ratio:30/34=0.88
		State machine design safety	26.Invalid state handling effectiveness	Ratio:4/4=1.00
			27.Setting state machine initial state	Binary:1
		Code specification compliance	28.No unreachable branch or redundant code	Binary:0
			29.Correctness and integrity of sensitive list	Ratio:38/45=0.84
			30.Conditional judgment uses logical operators in expressions	Binary:1

		31.Annotation rate	Ratio:18%/20%=0.9
	Logic synthesis safety	32.All objects can be synthesized	Binary:1
	Temporal safety	33.Typical working time sequence satisfied	Binary:1
		34.Maximum time sequence satisfied	Binary:1
		35.Minimum time sequence satisfied	Binary:1
	Gate level net list safety	36.Logical equivalence	Ratio:730/730=1.00
		37.Compile chip model correctness	Binary:1
		38.No establishment/maintenance confliction	Ratio:2203/2215=0.99
		39.Bus lines avoid centralized wiring	Binary:1
	Configuration file loading safety	40.Resource utilization	Ratio:1
		41.Three module utilization ratio	Ratio:1
		42.Timing margin	Ratio:1
Test and verification process	Safety testing adequacy	43.Text case execution rate	Ratio:45/45=1.00
		44.Test case passing rate	Ratio:39/45=0.87
		45.Fault density	Ratio:1-21/2400=0.99
		46.Troubleshooting	Ratio:21/21=1.00
		47.Failure probability	Ratio:1-2/100=0.98
	Coverage of safety testing	48.Safety requirement coverage	Ratio:7/7=1.00
		49.Statement coverage	Ratio:0.95
		50.Branch coverage	Ratio:0.96

5. The Measurement Model Verification Based on the Full Life Cycle of FPGA Software Development.

The last column of table 1 lists the measurement value of each safety metric of one FPGA software. The safety key level for this software is level B, putting forward high demand to safety.

Multiple safety specialists were invited to evaluate and score the safety of the FPGA software based on the above safety measurements. The average score is 0.87 and the FPGA software safety evaluation result is "good". The result can objectively reflect the overall level of software safety

6. Summary

Starting from the full life cycle of FPGA development, this paper introduces the definition and requirements of each process of the life cycle. Taking system requirements process, demand analysis process, design process, implementation and integration process, test and verification process as the main analysis object, this paper proposes the FPGA software safety measurement model based on the whole life cycle of the development. By studying the safety issues that should be considered in each development process, we extract the FPGA software safety metrics from each development process, and give the way to select the metrics of each measure. Then we use the model to evaluate the FPGA software safety in an example. The conclusion is consistent with experts' evaluation conclusion, which verifies the model effectiveness.

7. References

[1] Morrison P , Moye D , Pandita R , et al. *Mapping the field of software life cycle security metrics*[J]. Information & Software Technology, 2018.

[2] Huda S, Alyahya S, Ali M M, et al. *A Framework for Software Defect Prediction and Metric Selection* [J]. IEEE Access, 2018:1-1.

[3] Wohlin C, Runeson P, Martin Höst, et al. *Experimentation in Software Engineering* [J]. IEEE Transactions on Software Engineering, 2012, SE-12(7):733-743.

[4] Voas J, Kuhn R. *What Happened to Software Metrics?* [J]. Computer, 2017, 50(5):88-98.

[5] Fenton N E, Pfleeger S L. *Software metrics: a rigorous and practical approach* [M]// Software Metrics: A Rigorous and Practical Approach. International Thomson Computer Press, 1997.

[6] Sasao T. *FPGA Design by Generalized Functional Decomposition* [M]// Logic Synthesis and Optimization. 1993.