Automated Test Data Generation for Embedded System Models Using Combinatorial Testing

Do Thi Bich Ngoc

***Abstract*— Embedded systems have been playing very important roles in modern society. They have appeared in every aspect of life from automotive to avionics industries to home appliances, etc. These embedded systems therefore must satisfy high quality requirements. As a consequence, quality assurance for these kinds of systems has attracted much attention and investment from both academic research and industry communities. In developing embedded systems, testing often requires high coverage with different measures with respect to international standards like Decision Coverage (DC), Condition Coverage (CC), Modified Condition/Decision Coverage (MC/DC) respected to ISO 26262. However, it is difficult to generate test cases with high coverage due to the complexity of the embedded system, the larger number of inputs, and the complex continuous signals of inputs.**

In order to have good test cases with high DC, CC, MC/DC coverage measures, this paper proposes a method to automatically generate test-cases by applying Combinatorial testing technique. The requirements of the inputs of the embedded model will be encoded to the input of Combinatorial testing technique in order to generate test cases that cover all pair values of any two inputs. Experiments on case studies showed that proposed method has coverage result better than that of the random testing method.

***Tóm tắt*— Hệ thống nhúng đã và đang đóng vai trò quan trọng trong xã hội hiện đại. Các hệ thống này xuất hiện ở mọi mặt của đời sống từ xe ô tô tự hành, hàng không tới các thiết bị gia đình. Vì vậy, các hệ thống nhúng phải đảm bảo các yêu cầu cao về chất lượng, an toàn. Việc đảm bảo chất lượng cho các hệ thống này đã thu hút cả các nhà nghiên cứu và những người làm công nghiệp. Trong phát triển các hệ thống nhúng, việc kiểm thử thường yêu cầu độ phủ cao với các độ đo khác nhau theo các chuẩn quốc tế như Phủ điểm quyết định (Decision Coverage), Phủ điều kiện (Condition Coverage), Phủ Điều kiện/Điểm quyết định sửa đổi (Modified Condition/Decision Coverage) theo chuẩn ISO 26262. Tuy nhiên, rất khó để sinh dữ liệu kiểm thử với độ phủ cao do các hệ thống nhúng thường phức tạp, có nhiều đầu vào, và các tín hiệu đầu vào có thể là liên tục, phức tạp.**

Để có các ca kiểm thử tốt với độ phủ cao, bài báo này đề xuất một phương pháp sinh tự động các dữ liệu kiểm thử bằng cách áp dụng phương pháp kiểm thử dựa trên tổ hợp. Các yêu cầu đầu vào của mô hình hệ thống nhúng sẽ được biến đổi để thành đầu vào của phương pháp kiểm thử dựa trên tổ hợp, từ đó dữ liệu kiểm thử có khả năng phủ hết các cặp giá trị xảy ra của 2 đầu vào sẽ được tạo ra. Kết quả thực nghiệm cho thấy phương pháp đề xuất cho độ phủ tốt hơn so với phương pháp dựa trên kiểm thử ngẫu nhiên.

Keywords*— *Embedded systems; model-based development; test cases generation; MC/DC coverage; MATLAB/Simulink; combinatorial testing.

Từ khoá*— *Hệ thống nhúng; phát triển dựa trên mô hình; sinh ca kiểm thử; độ phủ MC/DC; MATLAB/Simulink; kiểm thử dựa trên tổ hợp.

I. INTRODUCTION

Embedded systems are used widely in society, such as in transportation, healthcare, and the broader infrastructure. However, the system failures can cause or contribute to serious accidents that result in death, injury, significant environmental damage, or major financial loss. Therefore, quality assurance for these types of embedded systems has attracted much attention and investment from both academic research and industry communities [2, 5, 7, 8, 9, 10, 12, 13, 14]. There are many

This manuscript is received on December 01, 2022. It is commented on December 03, 2022 and is accepted on December 05, 2022 by the first reviewer. It is commented on December 17, 2022 and is accepted on December 24, 2022 by the second reviewer.

international standards for quality and security of embedded systems such as ISO 26262, IEC 61508, EN-50128, etc. All these standards have more strict testing requirements, for example the test suites must cover all possible runs of the system.

MATLAB/Simulink provides a numeric computation, modeling, testing, and simulation environment and is used as a de-facto standard Model-based development (MBD) tool in many fields. The embedded system can be modeled using a Simulink model. When testing Simulink models, we first build a test suite that consists of test cases, which are groups of input signals, and then simulate and check the behaviors of the model for the test suite. Several test criteria are introduced to guarantee an acceptable quality, e.g., decision coverage (DC - every decision in the system has taken all possible outcomes at least once), condition coverage (every condition in a decision in a system has taken all possible outcomes at least once), and modified condition/decision coverage (MC/DC - every condition in a decision has been shown to dependently affect that decision’s outcome). It is typically required to test a model by using a high coverage test suite (e.g., “Road vehicles – Functional safety” in ISO 26262 requires conducting an MC/DC coverage test).

There are two main obstacles of testing complex embedded systems. First, the inputs of embedded systems are often real-time continuous signals. The current verification and validation techniques mostly work well with discrete time signals [5, 10, 12]. However, discrete test data rarely occurs in embedded systems. Second, embedded system models are complex with many input signals, output signals and events. Therefore, manual testing to cover all possible cases is impossible. In addition, it is very hard to apply formal methods or static analysis to generate test cases with high coverage measures.

II. Related works

In order to deal with these twoobstacles, there have been several research approaches recently. In [7, 8], Matinnejad et al. introduced a method to generate test data for Simulink models with continuous time signals. Their approach is that, for each inport, they applied a searching algorithm to design input signals in order to get the most varied output signals from the corresponding outport. However, this method cannot be applied to generate test cases with coverage criteria like DC, CC, MC/DC because there are constraints and relationships between inports and outports. For CC, DC, MC/DC criteria, we have to take care of all conditions inside the models. In another work, Godboley et al. [4] generated test data for Simulink models with branch coverage criteria. First, the C code for Simulink models is generated, then a C code static analysis tool is applied to generate test cases with respect to DC, CC, MC/DC coverage criteria. The weakness of this method is using a static analysis tool for C Code. Therefore, it cannot work well with large and complex Simulink models. Recently, Tomita et. al [13] proposed a testing method where each signal will be modeled as a given pattern (template-based) like sine, step signal, etc. Next, each test data will be an instance of the pattern. This method allows generating test data easily through some parameters of pattern (e.g., frequency, border ...) instead of generating data for all signals. However, when the models become large and complex, the number of inports increases dramatically. The random signals generated by this method may not create good test cases, namely the test cases cannot have high CC, DC, MC/DC coverage measures as expected.

Combinatorial testing [6, 15] is a combinatorial technique for software testing. Practice has shown that most faults are caused by interactions among six or fewer parameters. A number of tools for n-wise testing have been developed so far, including PICT [3], and ACTS[1]. These tools automatically construct a test-suite (a collection of test cases) of the n-wise testing for a given test model.

This paper proposes a method to strengthen the Template-based method with Combinatorial testing method to generate good test cases with higher coverage measurements.

III. PRELIMINARIES

A. Embedded Systems: Model-Based Development

Model-based development (MBD) is widely employed in the development of safe and reliable software for ECUs. In MBD, the model can be validated and verified because the models are mathematical and simulatable. The quality of the system is guaranteed by traceable model refinement and automatic code generation. MATLAB/Simulink is widely used as a de facto standard MBD tool, which provides an environment for numerical operations, graphical modeling, simulation, and code generation.

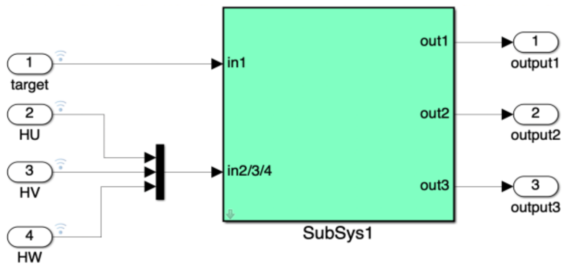

Figure 1. A Simulink model

Simulink model is constructed with various types of blocks, e.g., input/output, mathematical operator, logical/relational operator, (multiport) switch, and delay. These blocks are connected by lines that pass Boolean, integer, floating-point, or fixed-point data among them. Simulink supports hierarchical structure by subsystems. Simulink supports both continuous-time, discrete-time signals/values. Figure 1 is an example of a Simulink model. The input signals/values of a Simulink model will be obtained by the input blocks target, HU, HV and HW. The model will then be processed using the SubSys1 block to produce output signals/values, thus output blocks output1, output2 and output3.

B. Test Objectives

The objective of a model test is to obtain high model coverage, which measures how exhaustively the objects in a model are executed, given a test case. For a test suite, the coverage of every test case is accumulated. Various coverage criteria are proposed. In the DC, CC and MC/DC criteria, a target object is a block whose activity affects the logical characteristics (i.e., data flow pattern) of the model. Therefore, the test objective is broken down into the objectives for those blocks. The definition of coverage criteria is given as follows:

Decision Coverage (DC): checks that all the possible decisional outcomes of every block occur at some time step in a simulation. The coverage presents the percentage of the number of observed outcomes.

Condition Coverage (CC): checks that all the (satisfiable) atomic conditions of every block become true at some time step and become false at another time step. The coverage represents the percentage of observed conditional outcomes.

Modified Condition/Decision Coverage (MC/DC): checks that, for every block, each atomic condition affects its decisional outcomes independently. The coverage is defined as the percentage of observed witness pairs of atomic conditions.

A test case (generated signal) is graphically visualized by Simulink SLDV Tool in Figure 2, includes 4 signals: constant signal for input block target, sine signal for input block HU, linear signal for input block HV and triangle-wave signal for input block HW.

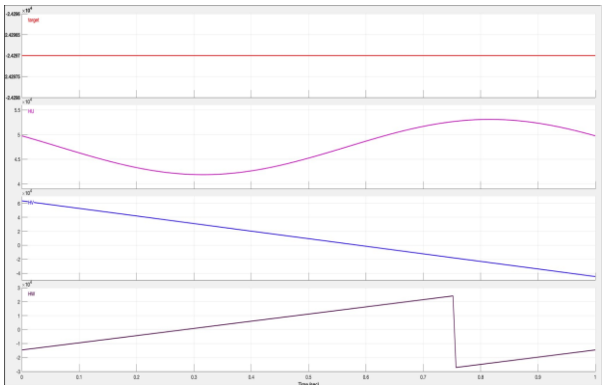

Figure 2. The generated signal (test case) visualized by Simulink SLDV Tool

C. Combinatorial testing

Combinatorial testing [6, 15] is a combinatorial technique for software testing. Combinatorial testing is a cost-effective way to test every combination of a set of parameters because: (1) the number of test cases is reduced; and (2) the test coverage is adequate.

It is a testing strategy, or coverage criteria, for constructing a test-suite using combinatorial techniques, i.e., given an input domain, which consists of a list of parameters with a set of their values and often with constraints (a.k.a., forbidden rules; expressed usually as a propositional formula), it stipulates that all combinations of values of n parameters should be covered at least once by test cases.

Combinatorial testing algorithms only accept the input domain in the format of parameters, and each parameter must be corresponded with discrete values like numbers or strings. In order to apply combinatorial testing, we need to prepare the input domain in the format of a list of parameters, and each parameter consists of a set of values.

D. Template-Based Method

This subsection recalls the template-based method proposed by Tomita et. al. in the recent papers [13]. In general, a test case is characterized as a group of input signals with arbitrary wave-forms. However, in the model-based development (MBD) method, the input signals to a controller are provided by other electronic modules that are likely to be well-controlled, or by physical objects that are dominated by physical laws. In practice, it is frequently sufficient to consider input signals with several specific wave forms to achieve high coverage. Therefore, Tomita et. al. [13] focused on the practical collection of such specific signals. For a test case consisting of them, the behavior of a model is relatively easy to understand for a design engineer. It is helpful for refining the model. In particular, they introduced the templates for Constant, Linear, NLinear, Step, NStep, Sine, Square, and Triangle to represent constant, linear, piecewise-linear, single-step, multiple-step, sine-wave, square-wave, or triangle-wave signals, respectively. Each signal template has several attributes and each attribute will be assigned a range of values. E.g., attributes of sine template are amplitude with range [-100,100], frequency with range [-10, 10],... We have a sine signal for the value of amplitude = 2 and frequency = 5.

IV. Proposed Method: Combinatorial Testing For Simulink Models

In order to generate test data from the signal template, we can randomly select the signal template and values of attributes for each input of the Simulink model [13]. However, this causes two problems: First, a signal template may be chosen several times while the others are chosen fewer. Second, for a chosen template, the values of the template's attributes are not well selected.

In order to deal with the limitations of the template-based method, this section proposes another approach using Combinatorial testing to improve the way to generate good test data.

A. Creating input domain for Combinatorial testing of Simulink models

As mentioned in Subsection II.C, the Combinatorial testing technique requires an input domain that is presented as a list of parameters with a set of values for each parameter. The following is a proposed procedure to prepare the input domain of the Simulink model for Combinatorial testing.

Step 1: Create a list of parameters by using a list of input blocks from Simulink models. E.g. the input domain for the model in Figure 1 has 4 parameters: target, HU, HV and HW.

Step 2: Create a set of values for each parameter as signal templates. E.g., for the model in Figure 1, target will have a set of values that are signal templates {Constant, Linear, N Linear, S tep, N S tep, S ine, Square}.

Step 3: Enhance the set of values from Step 2 by dividing each attribute range into multiple ranges. Besides, the boundary values are also considered. For example, the range of the Constant [−1024,1024] will be divided into two ranges: Constant_1 [−1024, 0]; Constant_2 [0, 1024], as well as two boundary values: Constant_3 -1024; Constant_4 1024.

Algorithm 1: Coverage measurement incremental test suite generation algorithm.

Input: A Simulink model M, an input domain Idomain.

Output: A selected test suite TS

```
1.  TS←∅
2.  TSCoverage←∅
3.  TestSuite←Combinatory(IDomain)
4.  for each testcase ∈ TestSuite do
5.      sd ← simulate(M, testcase)
6.      tCoverage ← measureCoverage(M, sd)
7.  if tCoverage ∩ T SCoverage ≠ ∅ then
8.      TS ← TS ∪ {tc}
9.      TSCoverage ← TSCoverage ∪ tCoverage
10.     end if
11. end for
12. return TS
```

B. Proposed Algorithm

This subsection presents the method for applying Combinatorial testing to generate good test cases for Simulink models. After creating the input domain for Combinatorial testing as a method in Subsection IV.A, a set of Combinatorial test suites is created by using a Combinatorial test tool. It is not necessary to obtain all Combinatorial test cases because we also need to consider the coverage assessment requirements. Since we only include good and helpful test cases, we only need to add new test cases when they significantly increase the testing's coverage metric. From this, Algorithm 1 is proposed. Function *combinatory (IDomain)* returns a set of Combinatorial test suites.

Function *simulate (M,testcase)* runs simulation tools for model M with inputs of M are form test case tc to get simulated data for all blocks of M. Function measure Coverage (M, sd) computes the coverage results CC, DC, MC/DC of model M.

V. IMPLEMENTATION AND EXPERIMENTS

A. Implementation

The proposed algorithm is implemented by mSCript in the MATLAB environment to generate test data for the MATLAB/Simulink model. Besides, the core Combinatorial testing algorithm is implemented in MATLAB. The input is a MATLAB/Simulink model, a .txt file describing the signal template setting and the output is a test suite and the corresponding CC, DC, MC/DC coverage information.

TABLE 1. RANDOM VS. COMBINATORIAL TESTING COVERAGE RESULTS

Model	**Num. of tc**	**Coverage type**	**Random – result**	**Random – time**	**Combinatory – result**	**Combinatory – time**
1	16	DC CC MC/DC	65/76 93/100 20/23	28(s)	65/76 93/100 20/23	29(s)
2	120	DC CC MC/DC	41/123 45/70 4/13	83(s)	79/123 61/70 8/13	107(s)
3	166	DC CC MC/DC	117/137 110/154 12/29	345(s)	118/137 120/154 17/29	359(s)
4	178	DC CC MC/DC	371/440 157/280 8/14	1540(s)	391/440 177/280 11/14	1663(s)

B. Experiments and Comparisons

Two experiments are executed: (1) random in [13]; (2) Combinatorial testing in our method. The experiment is executed with 4 practical models. For each model, 8 signal templates are used. In order to get a fair comparison, the random method will be executed with the number of test cases equal to a number of Combinatorial testing test cases.

All experiments are performed on a PC equipped with 1 Intel core i7-8570h @2.20GHz 2.21Ghz and 16 GB of RAM memory. Table 1 shows the experiment results for 4 models. Column Num. of tc shows the number of generated test cases by both random method and Combinatorial testing methods. Column Coverage type shows 3 types of coverage: DC, CC, MC/DC. Column Random - result (resp. Combinatory - result) shows the coverage results of random method (resp. Combinatorial method). The results are in the form a/b, in that a is the number of covered DC (or CC, or MC/DC) by test suites; b is the total number of DC (or CC, or MC/DC) in the model. Column Random - time (resp. Combinatory - time) shows the execution time in seconds.

The experimental results showed that, with the same number of test cases. Due to the proposed method of producing a combinatorial test case rather than a random test case as in [13], it will operate a little bit slower. However, while the simulation and calculating coverage results are time-consuming, the time difference is not much. For small models (like No.1) the random method and Combinatorial method have the same results. However, for large models, the Combinatorial method will return better CC, DC, MC/DC coverage. In addition, doing pre-processing before applying Combinatorial testing will increase the number of test cases, but the CC, DC, MC/DC coverage will be better. However, both methods cannot return full coverage (i.e., all CC, DC, MC/DC are covered). The reasons are: (1) the signal template setting is not good enough, the range setting is too large and the special constraint of the model (e.g., == c) is difficult to cover; (2) some uncovered CC, DC, MC/DC are dead logic in the model. Thus, full coverage will never be obtained.

V. CONCLUSION

This paper proposes a method for applying combinatorial testing to generate test cases for embedded models. In particular, a pre-processing step is applied to create the input domain for the Simulink model for Combinatorial testing. Then, Combinatorial testing technique is applied to generate the test suite. Finally, the simulation is performed to measure CC, DC, MC/DC coverage for each test case. Only the test cases that contribute incrementally to the CC, DC, MC/DC coverage are selected. Therefore, the main algorithm is proposed to generate only the necessary test cases. Experiments in Industrial case studies showed that the proposed method is orders of magnitude better than that of the random testing method. These selected test cases will support testing embedded models more efficiently.

The future work is a strategy to get a good range setting in order to generate test cases with higher coverage results. Moreover, it is an important and interesting research topic to find a way to check whether an uncovered CC, DC, MC/DC is dead logic.

REFERENCES

[1] Combinatorial Testing: https://csrc.nist.gov/projects/automated-combinatorial-testing-for-software [Online; accessed 22-Nov-2022].

[2] Dũng, N. Q., Hoàng, L. V., & Trung, N. H. (2020). Phát hiện mã độc IoT botnet dựa trên đồ thị PSI với mô hình Skip-gram. Journal of Science and Technology on Information Security, 7(1), 29-36. https://doi.org/10.54654/isj.v7i1.53.

[3] Jacek Czerwonka. Pairwise Testing in the Real World. Practical Extensions to Test Case Generators. Microsoft Corporation, Software Testing Technical Articles. February 2008.

[4] Godboley, S., Sridhar, A., Kharpuse, B., Mohapatra, D.P. and Majhi, B., Generation of branch coverage test data for simulink/stateflow models using crest tool. International Journal of Advanced Computer Research, 3(4), p.222, 2013.

[5] Holling, D., Pretschner, A. and Gemmar, M. 8cage: lightweight fault-based test generation for simulink. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (pp. 859-862), September 2014, ACM Press.

[6] Kuhn, D. R., Raghu N.K., and Lei, Y., Introduction to combinatorial testing, CRC press, 2013.

[7] Matinnejad, R., Nejati, S., Briand, L.C. and Bruckmann, T. Automated test suite generation for time-continuous simulink models. In Proceedings of the 38th international conference on software engineering (pp. 595-606), May 2016, ACM Press.

[8] Matinnejad, R., Nejati, S., Briand, L. C., Bruckmann, T. SimCoTest : A test suite generation tool for Simulink/Stateflow controllers. In Proceedings of the 38th International Conference on Software Engineering Companion (pp. 585-588), May 2016, ACM Press.

[9] Ngoc, D.T.B. Sinh Dữ Liệu Kiểm Thử Cho Mô Hình Hệ Thống Nhúng Sử Dụng Kỹ Thuật Kiểm Thử Theo Cặp, Journal of Science and Technology on Information and Communications. Vol 1 No 2 (2020).

[10] Peranandam, P., Raviram, S., Satpathy, M., Yeolekar, A., Gadkari, A. and Ramesh, S. An integrated test generation tool for enhanced coverage of Simulink/Stateflow models. In Proceedings of the Conference on Design, Automation and Test in Europe (pp. 308-311). EDA Consortium, March 2012.

[11] Mats G, and Jeff, O. Input parameter modeling for combination strategies. In Proceedings of the 25th conference on IASTED International MultiConference: Software Engineering, pp. 255–260, ACM Press 2017.

[12] Simulink Design Verifier: https://www.mathworks.com/products/sldesignverifier.html [Online; accessed 28-July-2022].

[13] Takashi, T., Daisuke, I., Toru, M., Shigeki, T., and Toshiaki, A. Template- Based Monte-Carlo Test Generation for Simulink Models. In IEICE TRANS. FUNDAMENTALS, VOL.E103–A, NO.2 FEBRUARY 2020. pp, 451-461.

[14] Tran, H. T., Hoang, P. V., Do, T. N., & Nguyen, D. H. (2020). Hardware Trojan Detection Technique Using Frequency Characteristic Analysis of Path Delay in Application Specific Integrated Circuits. Journal of Science and Technology on Information Security, 10(2), 36-43. https://doi.org/10.54654/isj.v10i2.64.

[15] Tzoref-Brill, R. Chapter Two - Advances in Combinatorial Testing. In series Advances in Computers, Volume 112, pp. 79-134, Elsevier 2019.

ABOUT THE AUTHORS

Do Thi Bich Ngoc

Work place: Posts and Telecommunications Institute of Technology.

Email: ngocdtb@ptit.edu.vn

Education: Received master degree in 2007 and PhD of Information Science in 2010.

Recent research direction: Test data generation; Numerical data analsyis; Sofware enginering; Software testing.