IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 21, Issue 1, Ver. I (Jan - Feb 2019), PP 26-31
www.iosrjournals.org

Automatic test data generation for time-continuous embedded system using built-in signals

Do ThiBich Ngoc[1], Nguyen ThiAnh Phuong[2]
[1](Posts and Telecommunications Institute of Technology, Hanoi, Vietnam)
[2](Institute of Information Technology, Vietnam Academy of Science and Technology, HaNoi, VietNam)
Corresponding Author: Do ThiBich Ngoc

Abstract: *Nowadays, embedded systems are becoming critical applications and play important roles in modern society. Therefore, quality assurance for these types of systems has attracted many attentions in software engineering research and development community. In these systems, testing procedure needs high coverage criteria such that test data cover all possible run. However, it is difficult to generate test data to coverage all possible runs due to the complex embedded system. In this paper, we propose a method to automatically generate test data based on built-in source signal in order to obtain test suite with high coverage. The experimental show that the proposed method generates a better test data than that of random test and Matlab/MBT test tool.*

Keywords: *embedded system, Matlab/simulink, MCDC coverage, testing, continuous signal*

Date of Submission: 28-12-2018 Date of acceptance: 12-01-2019

I. Introduction

1.1. Embeded model and testing problems

In this IoT century, embedded devices are playing more important role in several technological areas. In the embedded system development, it is common to use simulation for hardware design. The reason is that the hardware may not be available for testing, or may even not exist. Besides, the simulation can also avoid dangerous situations, such as equipment damage or human lack of safety. In most cases, testing is a feasible and reasonable method to guarantee the quality of practical models. This is because industrial models are very large and complex, and therefore formal verification takes very high cost, or is undecidable at the worst case. There are two main problems as follows.

Problem 1: Inputs of embedded systems are often time-continuous signals. The existing approaches to testing and verifying embedded models almost entirely focus on models with time discrete behavior, i.e., code generation models. These approaches generate discrete test inputs for embedded models with the goal of reaching runtime errors to reveal faults [3, 10], violating assertions inserted into the models based on some formal specification [8], and achieving high structural coverage [7]. Discrete test inputs, however, are seldom sufficient for testing embedded models, in particular, for those models with time-continuous behaviors.

Problem 2: Embedded system are complex. Embedded systems may have a high number of combinations of inputs and events, which can result in many different outputs. That means covering all posible runs in a manual testing activity is imposible. Thus, it is difficult to apply formal method or static analysis to generate the test data with high coverage. Moreover, the number of test data can be huge.

1.2. Proposed method

We propose a test data generation for time-continuous embedded system. To solve the first problem, each input signal is allowed to select any built-in signals in source blocks of Matlab/Simulink tool (e.g., sine, step signal…). Besides, we propose a method to modify Simulink model for testing purpose by modify Simulink model in order to allow each input can "select" one signal from a set of potential signals.

Next, to solve the second problem, we propose a system to automatically modify Simulink model and run simulation and check coveragage.

1.3. Related works

There are several works about test data generation for embeded systems [1,2,3,4,5,6,7,8,9,12,13].

Matinnejad. et.al. (in [4, 5]) generated test data focus on time-continuous simulink model. The authors aim to design each input port a special signal based on search algorithm in order to maximize diversity among output signals. The method focuses on evaluating the relation between the change of output and input, does not consider to check all posible runs of the models. Thus, it cannot be applied to test data generation with MC/DC coverage like our methods.

In [1], Godboley et.al. generated test data for Simulink models with branch coverage. The method firstly generated C code for Simulink model, then applied a consolic testing tool for generated C program to obtain test suite with MC/DC coverage. The only one problem of this method is it cannot work well with big model because it applied a static analysis tool for generated C program. Thus, it is difficult to generate test data for pratical models, which often have thousand of blocks. Our method uses testing technique, thus it can apply for hug models.

Tomita et.al [13] proposed a random test method in that each input signal is assigned for a signal template. Then, the a test data is intance of the signal template. We also use signal template for input signal. However, in this paper, we extend the idea by allow each input signal can use several built-in signal which allow input signal are more flexible.

II. Backgrounds

2.1. Simulink model

A Simulink model is constructed with various types of blocks, e.g., input/output, mathematical operator, logical/relational operator, (multiport) switch, and delay; blocks are connected with lines, transfer Boolean, integer or floating/fixed point data among them. Simulink supports hierarchical structure by subsystems with which a port may be equipped for selective activation/inactivation by external signals. Temporal behavior of a block is either based on continuous-time or discrete-time. Each model receipts the input from input blocks and returns the result in output blocks.

For example, Figure 1 is an embedded model for Modeling Clutch Lock-Up Using If Blocks. The clutch system in this example consists of two plates that transmit torque between the engine and transmission. There are two distinct modes of operation: (1) slipping - the two plates have differing angular velocities; (2) lockup - the two plates rotate together. In this model, the inputs are two From Work Space blocks (Engine Torque and Clutche Pedal); output are 6 outport blocks (we, wv, Locked Flag, Lockup Flag, Break-Apart Flag, FrictionTorque Required for Lockup); other blocks are if block, subsystem blocks (Friction model, Friction Mode Logic,…)

When testing Simulink models, we first build a test suite that consists of test data, which are groups of input signals, and then simulate and check behaviors of the model for the test criteria. Several test criteria are introduced to guarantee an acceptable quality, e.g., decision coverage, condition coverage, and modified condition/decision coverage (MC/DC coverage).

2.2. MC/DC coverage

It is typically required to test a model by using a high-coverage test suite. MC/DC originally defined in the standard DO-178B [1], intended to be an efficient coverage metric for the evaluation of the testing process of software incorporating decisions with complex Boolean expressions. The upcoming standard ISO 26262 [2] for safety-relevant automotive systems prescribes MC/DC for ASIL D as a highly recommended coverage metric. Actually, ISO 26262 ("Road vehicles – Functional safety") requires to conduct an MC/DC coverage test. MATLAB/Simulink has toolboxes for this purpose: Simulink Design Verifier6 (SLDV) and Simulink Verification and Validation7 (V&V).

MC/DC (modified condition/decision coverage) is a coverage criterion that requires all of the below during testing [1]:

- Each entry and exit point is invoked
- Each decision takes every possible outcome (DC)
- Each condition in a decision takes every possible outcome (MC)
- Each condition in a decision is shown to independently affect the outcome of the decision (MC/DC).

Independence of a condition is shown by proving that only one condition changes at a time.

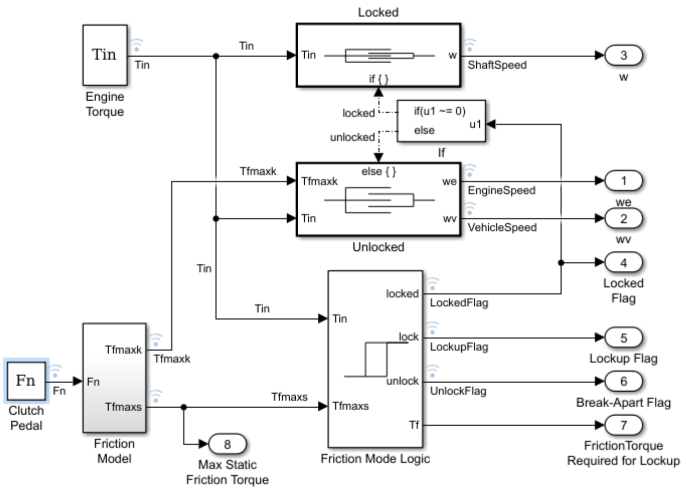

Figure 1: Modeling Clutch Lock-Up Using If Blocks [11]

2.3. Test data of Simulink model

Definition 1 (Test data/Suite). A test data is a vector of input signals which assigned for each input port of Simulink model a signals. A test suite is a set of test data.

Given a test input, the model coverage is a measure of how exhaustively the model objects are exercised. The model coverage for a test suite is an accumulation of that of every test input in the test suite. In decision, condition and MC/DC coverage criteria, we need to consider block whose activity changes logical characteristics (i.e., data flow pattern) of the model. In the case of Simulink, they are, e.g., relational/logical operator blocks, (multi-port) switch blocks and subsystems with active-control ports.

For example, in Figure 1, a test data is a vector of two signals corresponding with pair of inputs (Engine Torque and Clutche Pedal). To obtain the coverage of if block, we have to create test data such that both conditions (u1 == 0), (u1 ~=0) must happen sometimes.

2.4. Built-in signal as test data

A test data for Simulink model is characterized as a vector of input signals. Each signal may have an arbitrary shape of a waveform. However, input signals are provided by other electric modules which are likely well-controlled, or by physical objects which are dominated by physical laws. So it may be unnecessary to consider arbitrary signals. Additionally, considering well-controlled types of input signals (e.g., sine-wave, constant, step…), can be characterized with a few parameters, for achieving high decision, condition and MC/DC coverage. Besides, Simulink already provides a group of signals with such types. [13] also uses such built-in signal for randomly test data generation in Simulink models.

Therefore, we also use built-in input signals to design test data. To improve the coverage, the test data will use various of built-in signals for an input port instead of a single one.

III. Test Data Generation Method Using Built-In Signals

3.1 Proposed Model for testing

As mention in Subsection 2.4, built-in signals are used for input ports to easily control the parameters. Besides, we propose a method in that each test input will set input port a different built-in signal, not only a single one like [13]. By this, the test data are more varied. However, in Matlab/Simulink, a model M, each input can only be connteded with only one source block (stand for a built-in signal).

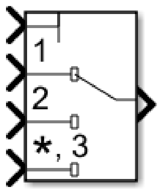

Figure 2: Multiport Switch block

We have an observation that in Matlab/Simulink, the Multiport Switch block (Figure 2) determines which of several inputs to the block passes to the output. The first input is the control input and the remaining inputs are the data inputs. The value of the control input determines which data input passes to the output. Thus, multiport switch can be used to automatically set any source block for a input port. For examle, in Figure 3, if the input port In1 = 1the output of Multiport switch is repeating sequence signal; if In1 = 2; the output of multiport switch is Sine wave signal; and if In1 = 3, the output of multiport switch is Step signal.

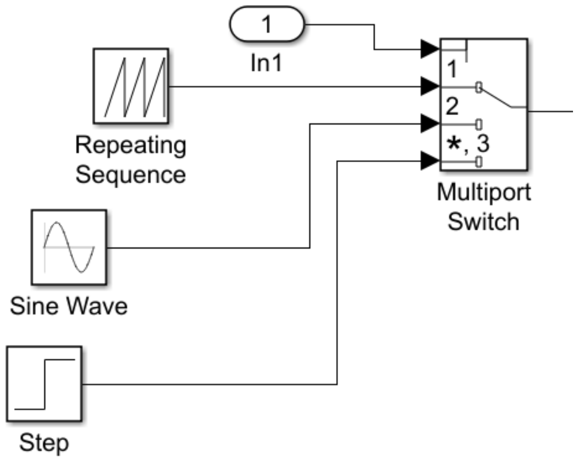

Figure 3: Multiport Switch block with 3 source blocks

The modified Simulink model is called *model for testing*. The proposed method to obtain model for testing is shown in Algorithm 1.

Algorithm 1: Generate model for testing

```
Input:
- Original model M with list of input ports P = (p1,…,pn)
- Set of possible signal blocks S = {s1,…, sk}
Output: model for testing MT
begin
for each input port pi
if (pi is continuous signal) then
begin
add Multiport switch mi with source blocks s1,…,sk as data input of multiport switch mi, control input is range from 1..k;
replace pi by mi;
endif;
end;
```

Thus, for model M and a test data (s_{i1},…s_{in}), with $s_{ij} \in$ S (i.e., set of posible signal blocks). We now need not to manually modify M by connecting each input port pi a source block s_{ij}. It can be automatically excute in Model for testing MT by only setting the value of control input of multiport switch mi to s_{ij}.

Example 1:

Orignal model in Figure 1 has two input ports, include: input p_1 is Engine torque, input p_2 is Clutch pedal. Assuming that, we set to each input port three types of signal: repeating sequence, sine wave, step. Thus, we can apply the multiport switch in Figure 3 with control input $\in$ {1,2,3}. The Model for testing is shown in Figure 4 with p_1 and p_2 are replace by m_1 (Multiport switch 1) and m_2 (Multiport switch 2). A *test data (1,2) mean in_1 = 1, in_2 = 2, or p_1 is Repeating sequence signal and p_2 is sine wave signal.*

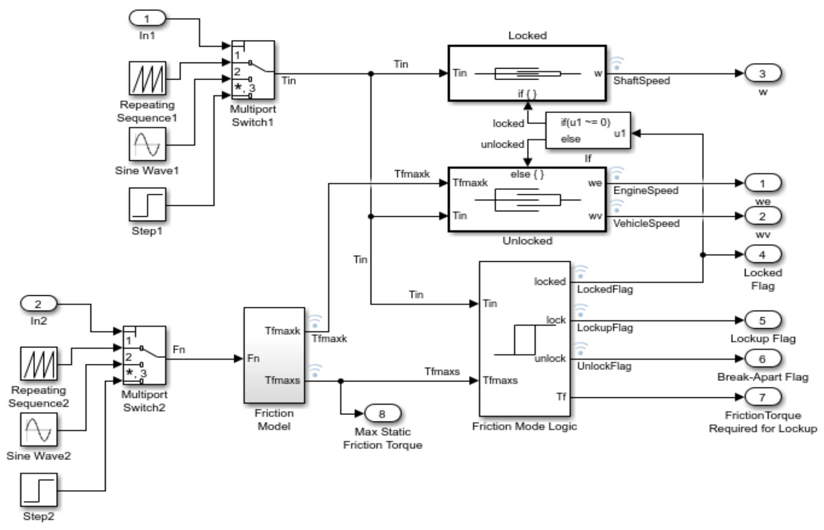

Figure 4: Model with multiport switches

3.2 Proposed test suite generation method

We next consider to select test input to obtain high coverage. The model for testing MT will be simulated with each input and evaluate the coverage.

Algorithm 2: GenerateTestsuite

```
Input: model for test MT
Output: test suite T
begin
        Let ST is set of test data over MT;

        Let T=∅
        Let Coverage = (0,0,0);
for each t in ST
begin
        simulation (MT,t);
        tCoverage = checkMCDC();
        if (Coverage <tCoverage)
        begin
                T = T∪ {temp};
                Coverage = tCoverage;
        endif;
endfor;
return T;
end;
```

In Algorithm 2, function simulation(MT,t) excutes simulation of model MT with input setting for control inputs of multiportswitchs is t. Function checkMCDC() returns the coverage values of DC, CC, and MCDC when adding simulation with t. The Algorithm only selects the test data t into test suite T if the simulation MT with t increases the coverage.

Example 2: For model MT in Figure 4, we have ST is {(1,1,), (1,2), (1,3), (2,1), (2,2,), (2,3), (3,1), (3,2), (3,3)}.

IV. Experimental Result

We encode the proposed method using matlab mscript. Table 1 shows the experimental result of MC/DC coverage with several Matlab/Simulink model using random test in [13] and the proposed method with set of source blocks are block*Sine, Step, Random Number, Signal Generator (sawtooth),Pulse*. The Model column shows the list of simulink models; Num of blocks column shows the number of blocks in the model; Num of Inputs shows the number of inputs in the model; %DC, %CC, %MCDC (Random) shows the coverage result of test data generation based on [13]; %DC, %CC, %MCDC (proposed method) shows the coverage result of proposed methods. The experimental result of model in Figure 4 is shown in row 6th, corresponing with Model 6.

The experimental shows that the proposed method generate test suite with better MC/DC coverage, expecially for the cases of big models with more than 400 blocks.

Table 1: MCDC COVERAGES OF EMBEDDED MODELS

STT	Model	# blocks	#Inputs	%DC,CC,MCDC (Random)	%DC,CC,MCDC (proposed method)
1	Model 1	66	4	dc = 11/12, cc = 0/0, mcdc = 0/0	dc = 11/12, cc = 0/0, mcdc = 0/0
2	Model 2	65	4	Fail	**dc = 28/32, cc = 0/0, mcdc = 0/0**
3	Model 3	63	4	dc = 0/0, cc = 2/2, mcdc = 0/0	dc = 0/0, cc = 2/2, mcdc = 0/0
4	Model 4	77	4	dc = 0/0, cc = 8/8, mcdc = 0/0	dc = 0/0, cc = 8/8, mcdc = 0/0
5	Model 5	69	4	dc = 0/0, cc = 8/8, mcdc = 0/0	dc = 0/0, cc = 8/8, mcdc = 0/0
6	Model 6	117	2	dc = 8/8, cc = 8/8, mcdc = 2/2	dc = 8/8, cc = 8/8, mcdc = 2/2
7	Model 7	501	4	dc = 93/129, cc = 46/64, mcdc = 4/13	dc = **126/129**, cc = **64/64**, mcdc = **11/13**
8	Model 8	485	2	dc = 122/129, cc = 64/64, mcdc = 13/13	dc = **125/129**, cc = 64/64, mcdc = 13/13
9	Model 9	507	4	dc = 93/129, cc = 46/64, mcdc = 4/13	dc = **123/129**, cc = **64/64**, mcdc = **13/13**

V. Conclusion

The paper proposed a method to automatically generate test data for embedded model with time-continuous input. Instead of generating test data as arbitrary signals, the built-in signals (e.g., sine-wave, step-wave,…) are used. Besides, we improve the original embedded model by applying Algorithm 1. Then, the test data for embedded model can be generated and excuted automatically. Algorithm 2 supports selection and evaluatation of the MCDC coverage. The experimental result shown that the proposed method generates test data for time-continuous embedded system with higher coverage than random testing.

For future works, we plan to apply testing techniques and static analysis to generate testcases for embedded model with huge number of inputs.

Acknowledgements

The paper is partly supported by Project PTNTĐ17.08 of IOIT "Testing the security and safety of IOT system".

References

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

[2]. Godefroid, P., Klarlund, N. and Sen, K., 2005, June. DART: directed automated random testing. *In ACM Sigplan Notices (Vol. 40, No. 6, pp. 213-223). ACM.*

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

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

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

[6]. Pacheco, C., Lahiri, S.K., Ernst, M.D. and Ball, T., 2007, May. Feedback-directed random test generation. In Proceedings of the 29th international conference on Software Engineering (pp. 75-84). IEEE Computer Society.

[7]. Peranandam, P., Raviram, S., Satpathy, M., Yeolekar, A., Gadkari, A. and Ramesh, S., 2012, March. 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.

[8]. Richardson, D.J., Aha, S.L. and O'malley, T.O., 1992, June. Specification-based test oracles for reactive systems. In Proceedings of the 14th international conference on Software engineering (pp. 105-118). ACM.

[9]. Satpathy, M., Yeolekar, A. and Ramesh, S., 2008, October. Randomized directed testing (REDIRECT) for Simulink/Stateflow models. In Proceedings of the 8th ACM international conference on Embedded software (pp. 217-226). ACM.

[10]. Simulink Design Verifier: https://www.mathworks.com/products/sldesignverifier.html [Online; accessed 28-Oct-2018].

[11]. Simulink example: Modeling Clutch Lock-Up Using If Blocks: https://www.mathworks.com/help/simulink/examples/modeling-clutch-lock-up-using-if-blocks.html[Online; accessed 28-Oct-2018].

[12]. Sims, S. and DuVarney, D.C., 2007, October. Experience report: the reactis validation tool. In ACM SIGPLAN Notices (Vol. 42, No. 9, pp. 137-140). ACM.

[13]. T. Tomita, Daisuke.I, Toru. M, Shigeki.T, T. Aoki, Template-Based Monte-Carlo Test Generation for Simulink Models, Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy'17), LNCS 11267.

Do ThiBich Ngoc. " Automatic test data generation for time-continuous embedded system using built-in signals" IOSR Journal of Computer Engineering (IOSR-JCE) 21.1 (2019): 26-31.