ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 2, Issue 7, July 2014

Search-Based Simulink Model Testing

Videet M Visavadiya[1], Bhavna K Pancholi[2]

PG Scholar, Electrical Engg.Dept,Faculty Of Technology & Engg.,The M.S.University Of Baroda,Vadodara, India[1]

Assistant Professor, Electrical Engg.Dept , Faculty Of Technology & Engg., The M.S.University Of Baroda, Vadodara,India[2]

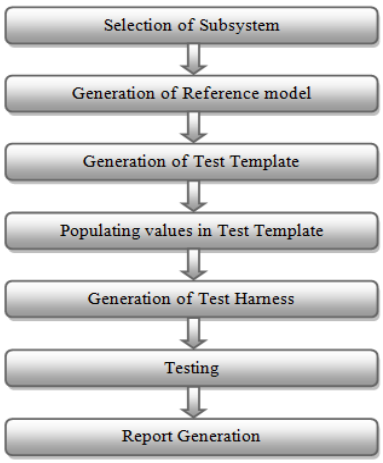
Abstract: Search-based test-data generation techniques have been widely used in code level testing. Testing of various functional and non-functional properties has been targeted, as has fulfilling certain structural coverage criteria. This work takes a step further to attempt test-data generation at the higher levels of abstraction offered by Simulink models.
Code testing has been proven successfully at industrial level and industries are working on it since many years. Hence, a flexible tool for testing of models can be a great boon to the industry. Presently, less number of tools is available for this purpose and that was the motivation for me to make this framework. In my work I have investigated such models and have made a wide-ranging, adaptable, automatable framework for testing these models. This flexible testing framework allows analysis of even smallest unit of blocks as well as high level design blocks.

Keywords: Model Testing, Search-based Testing Techniques, Testing Framework, High level model testing techniques.

I. INTRODUCTION

Software is taking an increasingly important role in high integrity applications, such as automotive and aircraft control systems, chemical processing plants, etc. Software development typically includes activities for ensuring the correct and safe operation of the software. Such activities are known collectively as Verification and Validation (V&V). During verification, the software is checked against some model of the system that represents the intentions of the designers and the wishes of the customer. The validation process ensures that any assumptions made during its development were correct and the delivered system satisfies both the customer and safety certification authorities. Simulink is a block diagram environment for multi domain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modelling and simulating dynamic systems. It is integrated with MATLAB, enabling us to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis. In many application domains, such as automotive application, embedded systems become increasingly complex and interdependent. This development is intensified by the demand for even more efficient and safe transportation systems. In the late development phases, when system integration testing is performed, this development poses a challenge to validation and verification in industry. The desired system reaction of one input stimulus sequence is determined by many factors. Moreover, the evaluation of the system reaction is complex as it consists of discrete and continuous signals as e.g. actuators and network messages.
All of them are to be checked w.r.t. the functional and non-functional requirements. It is hardly possible to integrate all this information in one test model. Especially when it comes to the testing of hybrid systems, advanced approaches are needed. One should not limit the test evaluation to some outputs. For a sound estimation of the reliability all outputs should be verified.
For the verification and validation we present a model-driven approach in which the complexity is handled by a combination of reference model, test template and test harness with models in MATLAB/Simulink that describe aspects of the requirements. Test-data generation problems can be categorized into three classes according to the source artefact the test generation is based on: specification-based, architecture-based and program-based. Given the source, one needs to define how rigorous a set of tests will constitute sufficient testing. This leads to the notion of coverage criteria. Coverage criteria can be structure-oriented, fault-oriented or error-oriented. These concepts were initially put forward for program-based test-data generation. However, they should also apply to specification or architecture based testing.

II. FLOW DIAGRAM



ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 2, Issue 7, July 2014

III. TESTING PROCESS

In order to test any system one should consider the possible blocks which might be used in model. Every time it is not necessary that it requirements are of testing only higher level model or whole system. It may require testing some part of system in order to testing small modules of system. Hence vast coverage ranging from small system to complicated large systems is required.

3.1 Generation of reference model

In order to test any such systems first step is to generate reference model. For generating reference model, first the person who is testing the system should select proper subsystem to test.

After selection of proper subsystem another model is being generated, which reflects the original System Under Test (SUT). In the procedure of reference model generation some parameters are being considered. The reason of taking care of such parameter is that it is not require that every time inputs are given by inports only. It is possible that some parameters are defined differently and values are given other way. One of the example is suppose constant block is defined inside the model with variable name define instead of constant value and that variable is defined in workspace, value of such variable will not change for particular case but it may be different for different test cases. Another example is use of memory blocks, combination of data set memory blocks, memory write blocks and memory read blocks may be used to change the value at run time. So such configurable parameter can be change at the run time of particular case. Rather than this Enable and/or Trigger subsystems are also should be consider.

Reference model is being generated from original model. It is another model which just takes selected model/subsystem with all the implemented blocks inside it and creates new model.

3.2 Generation of Test Template

For testing any model or subsystem testing vector and test cases plays major role. In order to provide such test cases to reference model Test Template (TT) is generated. TT consists of inputs, outputs and possible variables from reference model with its data type.

This blank TT should be populated with numerical values and number of Test cases can be generated for single SUT.

Multiple test cases can be tested at a time by single TT. TT is generated in the excel file format for ease of person who is testing. By adding multiple sheets, multiple test cases may be added.

3.3 Generation of Test Harness

Test harness is building a blocks surrounding SUT. It is being generated from populated test template and automatically generated reference model using test framework. Test harness is generated by taking care of values populated in test template w.r.t time. In other word it takes care of giving required value to particular input at particular time specified in test cases. It also takes care of data conversion at required stages. Test harness gives provision of selecting particular test case.

3.4 Appending Test Harness

In order to change test vector values, there is provision of changing value in TT and appending test harness according to changes made in TT. It simply changes input and output values as per new values w.r.t time.

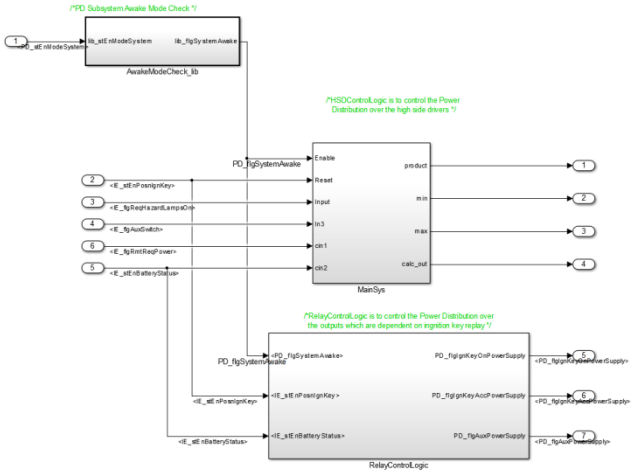
3.5 Testing

Testing runs SUT for the time period as per test case and gets results from that compilation. The results are being recorded as actual outputs. Comparison of actual and expected outputs gives final result that either particular test vector passes or fails. This is displayed accordingly. There is provision of testing single or multiple test cases at a time.

3.6 Automatic report generation

After testing is done the detailed report of testing results is required. Report generation facilitates this requirement. It generates report according to selection. It also facilitate of getting some results to TT for comparison.

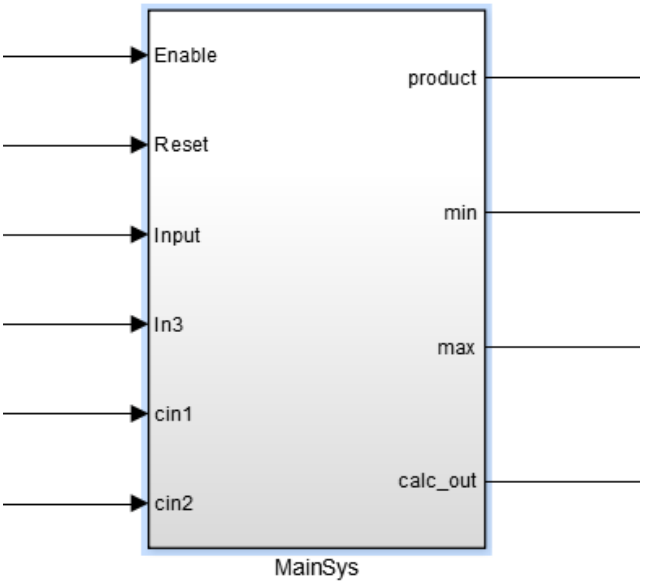
IV. RESULTS



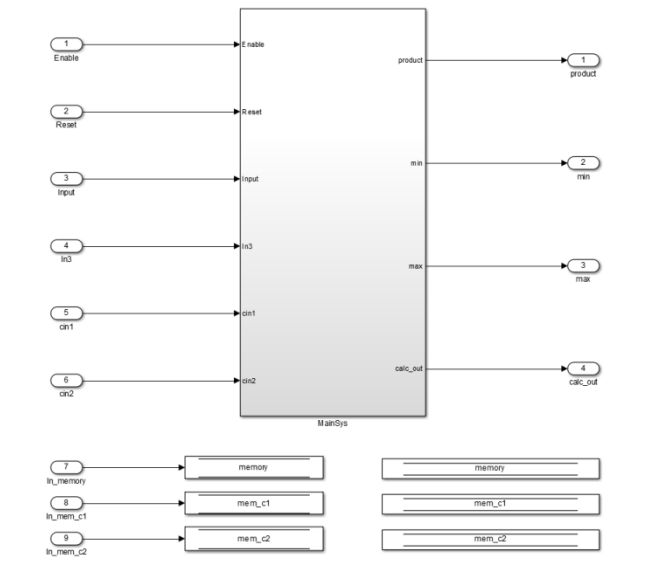
(Figure 1: Example of Model)

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

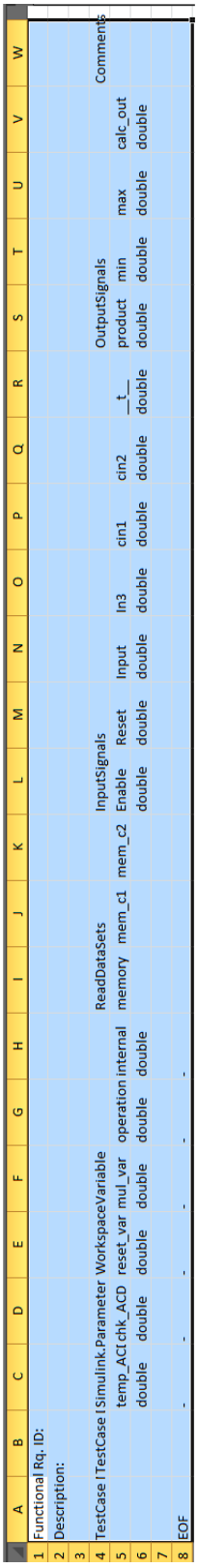
INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 2, Issue 7, July 2014

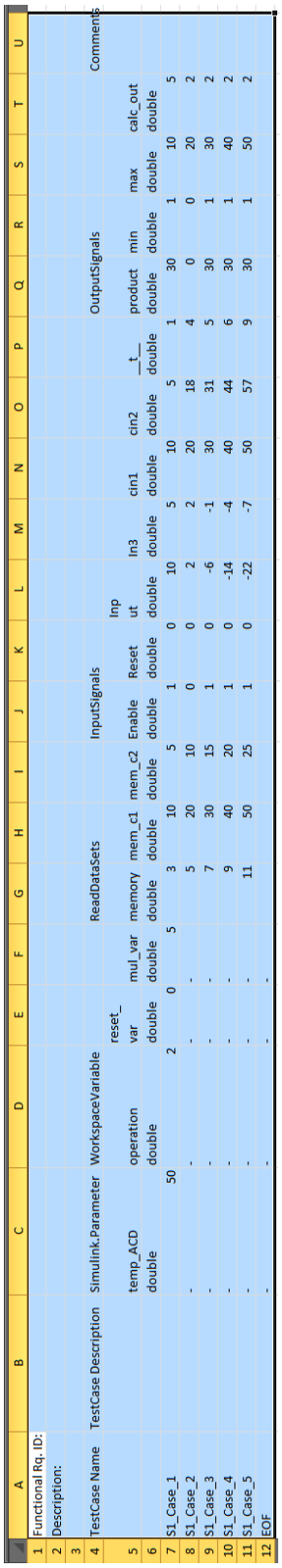


(Figure 2: Example of Subsystem)



(Figure 3: Example of Reference model)

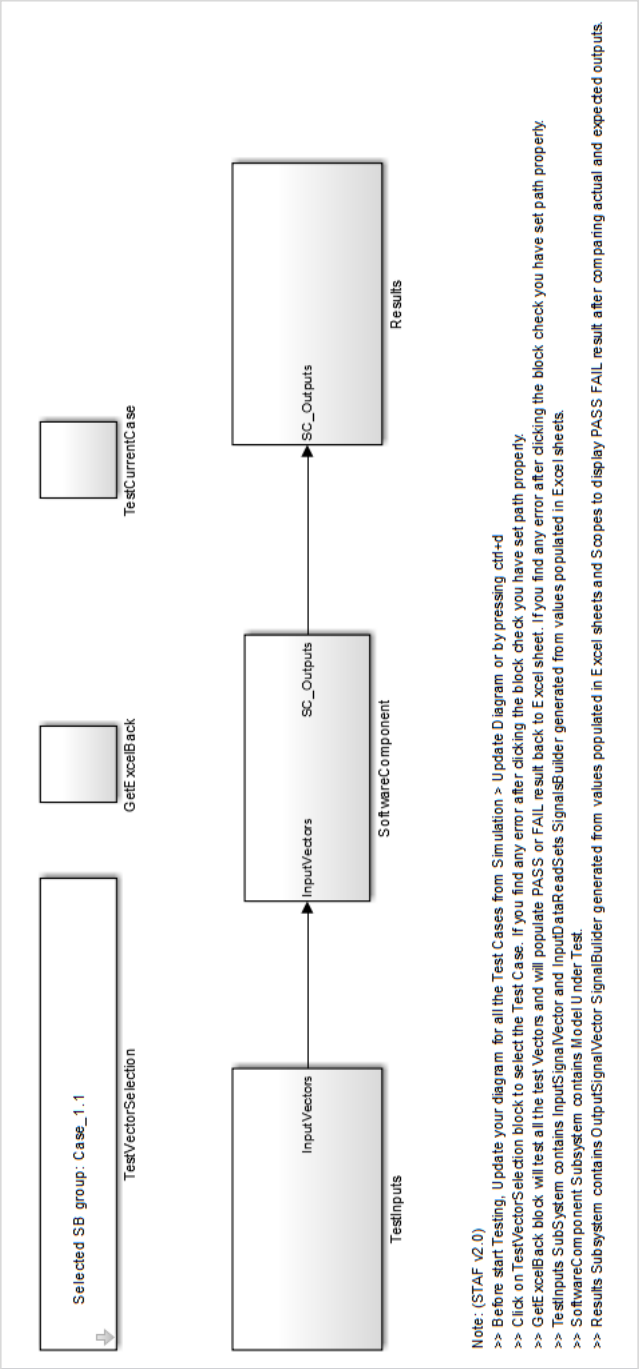




(Figure 4: Example of Test Template)

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 2, Issue 7, July 2014



(Figure 5: Example of Test Harness)

V. CONCLUSION

The results in this paper show that it is possible to use search techniques to provide an automatic test-set generation framework for testing Simulink models. The test-sets generated with this framework are both efficient (small size) and effective (high coverage). The dynamic test-data generation approaches have demonstrated its effectiveness (high coverage/success-rate) and efficiency (low computational cost) in the generation of certain test vectors. The test-set extraction facilities have been shown to be useful to improve the cost efficiency of testing.

This is effective tool for testing Simulink models and generating reports. This framework analyses parameters at all level of model and hence gives vast coverage.

ACKNOWLEDGMENT

The author would like to acknowledge the support, encouragement & valuable guidance provided by Mr. Shailesh Mohite ans Mr. Durvesh Kulkarni.

REFERENCES

[1] Shaukat Ali, Lionel C. Briand, Hadi Hemmati, Rajwinder K. Panesar-Walawege. "A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation" IEEE Transactions on SOFTWARE ENGINEERING November/ December 2010 (vol. 36 no. 6)

[2] Ramon Sagarna, Xin Yao, "Search-Based Testing of Complex Simulink Models containing Stateflow Diagrams" 2008 IEEE International Conference on Software Testing Verification and Validation Workshop

[3] Peranandam P, Raviram S, Satpathy M, Yeolekar A, Gadkari A, Ramesh S, "An integrated test generation tool for enhanced coverage of Simulink/Stateflow models" 2012 Design, Automation & Test in Europe Conference & Exhibition

[4] Yuan Zhan, "A search-based Framework for Automatic Test-Set Generation for MATLAB/SIMULINK Models" Submitted for degree of PhD, University of York, Department of Computer Science, December 2005

[5] Reactis. Available at: http://www.reactive-systems.com

[6] MathWorks, MATLAB/Simulink, http://www.mathworks.de/products/simulink / ,2013

[7] M. Beyer, W. Dulz, and K.-S. J. Hielscher, "Performance Issues in Statistical Testing," in MMB, R. German and A. Heindl, Eds. VDE Verlag VDE Verlag, 2006, pp. 191–208.

[8] R. Boot, J. Richert, H. Schutte, and A. Rukgauer, "Automated Test of ECUs in a Hardware-in-the-Loop Simulation Environment," in Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on, 1999, pp. 587–594.

[9] E. H. Spafford. Extending Mutation Testing to Find Environmental Bugs. Software – Practice and Experience, 20(2): 181-189. February 1990.

[10] G. Bernat, A. Colin and S. M. Petters. WCET Analysis of Probabilistic Hard Real-Time Systems. Proceedings of the 23rd Real-Time Systems Symposium. 2002.

[11] Reactis. Available at: http://www.reactive-systems.com.

[12] A. Baresel, M. Conrad, S. Sadeghipour and J. Wegener. The Interplay between Model Coverage and Code Coverage. Proceedings of the International European Conference on Software Testing, Analysis and Review (EuroSTAR03). 2003.

[13] Shaukat Ali, Lionel C. Briand, Hadi Hemmati, Rajwinder K. Panesar-Walawege. "A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation" IEEE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 2, Issue 7, July 2014

Transactions on SOFTWARE ENGINEERING November/ December 2010 (vol. 36 no. 6)

[14] Ramon Sagarna, Xin Yao, "Search-Based Testing of Complex Simulink Models containing Stateflow Diagrams" 2008 IEEE International Conference on Software Testing Verification and Validation Workshop

[15] Peranandam P, Raviram S, Satpathy M, Yeolekar A, Gadkari A, Ramesh S, "An integrated test generation tool for enhanced coverage of Simulink/Stateflow models" 2012 Design, Automation & Test in Europe Conference & Exhibition

[16] Yuan Zhan, "A search-based Framework for Automatic Test-Set Generation for MATLAB/SIMULINK Models" Submitted for degree of PhD, University of York, Department of Computer Science, December 2005

[17] Y. Zhan, and J. Clark. Search-Based Mutation Testing for Simulink Models. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005). ACM Press, pages 1061-1068. 2005.

[18] M.Z. Iqbal, A. Arcuri, and L. Briand, "Environment Modeling with UML/MARTE to Support Black-Box System Testing for Real-Time Embedded Systems: Methodology and Industrial Case Studies," Proc. ACM/IEEE Int'l Conf. Model Driven Eng. Languages and Systems, pp. 286-300, 2010.

[19] D. Knuth, The Art of Computer Programming, vol. 3. Addison Wesley 1973.

[20] M. Lyu, Handbook of Software Reliability Engineering. McGraw-Hill, Inc., 1996.