International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:3, 2016

Modeling and Simulation Methods Using MATLAB/Simulink

Jamuna Konda, Umamaheswara Reddy Karumuri, Sriramya Muthugi, Varun Pishati, Ravi Shakya,

***Abstract*—**This paper investigates the challenges involved in mathematical modeling of plant simulation models ensuring the performance of the plant models much closer to the real time physical model. The paper includes the analysis performed and investigation on different methods of modeling, design and development for plant model. Issues which impact the design time, model accuracy as real time model, tool dependence are analyzed. The real time hardware plant would be a combination of multiple physical models. It is more challenging to test the complete system with all possible test scenarios. There are possibilities of failure or damage of the system due to any unwanted test execution on real time.

***Keywords*—**Model Based Design, MATLAB, Simulink, Stateflow, plant model, real time model, real-time workshop, target language compiler.

I. Introduction

DESIGN and development of real time physical model (plant model), in simulation environment, have challenges to meet the requirement of real time hardware plant system. MATLAB/Simulink environment models would be much helpful in such scenarios where multiple plant models integrated and tested for all possible scenarios of real time operation. Here the challenge is accuracy depiction of MATLAB/Simulink model with that of real time model would cause an issue for correct validation of the Simulink control algorithm through MBD.

Challenges identified in the process of developing plant models at simulation level:

- Real Time Model Development: Analysis through Mathematical model, Component Physical model (Plant model) would be used to develop simulation model much closer to real time plant model are investigated [4], [5].
- Scheduler Design and Development Multiple function call subsystem: Developing function calls is tedious task. This would impact the total design time of the complete system. Analysis performed to reduce the effort for developing function calls for multiple blocks using State Flow Charts which would be simpler and less time consuming [8].
- Multiple Targets Configuration: Sometimes, plant model with controller would demand execution of the controller code generation in multiple targets (Hardware). A generic configuration has to be set, but this would restrict the code generation possibility for multiple targets at single block execution [9], [10]. Analysis and possible methods to address this problem are investigated.

The fidelity of simulated physical model with hardware plant system has challenging aspects in terms of performance and behavior. Pre-design verification of control algorithm at simulation environment is very useful. Design of plant model requires overcoming the challenges of building simulation environment physical model as much as equal to real time hardware plant system. In such scenarios, few approaches, of designing and developing simulation environment physical model, are investigated in this paper.

Scheduler design in simulation environment is essential for sequential execution of multiple controllers or multiple blocks, where data dependency exists and timely execution is mandatory. Function call subsystems are widely used in model based design. Earlier S-Function methodology was extensively used for developing function calls [7]. Function-call subsystems using S-Function are not executed directly by the Simulink engine; rather, the S-function determines when to execute the subsystem. When the subsystem completes execution, control returns to the S-Function. S-Function is more time consuming as it requires developing the S-Function code, creating MEX-Files for every update in the function call without errors. The alternate way of developing function calls for scheduler design is using Stateflow Charts. In Part III, a simple example following how to replace function-call subsystem blocks in a Simulink model with Simulink functions is mentioned. This procedure reduces the number of objects in the model while retaining the same simulation results.

Simulink Stateflow Charts are used for scheduling the order of execution of Simulink subsystems explicitly in a model [8]. Stateflow schedulers extend control of subsystem execution in a Simulink model, which determines order of execution implicitly based on block connectivity via sample time propagation.

The Target Language Compiler is an advanced feature within Real-Time Workshop that allows you to customize the code generated by Real-Time Workshop [9].

The Target Language Compiler is an integral part of the Real-Time Workshop. The Target Language Compiler transforms a specially compiled Simulink block diagram into ANSI C files that can be compiled and linked manually or automatically by Real-Time Workshop. TLC enables you to customize the C code generated from any Simulink model and generate optimal, inline code for your own Simulink blocks.

Jamuna Konda is with Cyient Ltd, Hyderabad, India. (e-mail: Jamuna.konda@cyient.com).

Umamaheswara Reddy Karumuri is with Cyient Ltd, Hyderabad, India (e-mail: Umamaheswara.Karumuri@cyient.com).

Sriramya, Mr. Varun Pishati, Mr. Ravi Shakya are associated with CYIENT Ltd, Hyderabad, India – 500032.

The TLC files are ASCII files that explicitly control the way code is generated. The Target Language Compiler provides a complete set of ready-to-use TLC files for generating ANSI C or C++ code. The Target Language Compiler (TLC) is designed to convert the model description file model.rtw into target-specific code or text. The word target in Target Language Compiler refers not only to the high-level language to be output, but also to the nature of the real-time system on which the code will be executed.

In this paper, Section II describes about real time physical model development in simulation environment and analyzed the solution to the challenge of matching it with the real time hardware plant system. Section III describes the scheduler design and development, where issue of designing multiple function calls has been addressed. Section IV describes about handling the multiple controller blocks with different target configuration set in parent model for code generation. Finally, conclusions are summarized in Section V.

II. Real Time Model Development

Many times MATLAB/Simulink models are designed and developed for control algorithm. The control algorithm would later be used for code generation for embedded target platform where the rapid prototyping on a real hardware physical model (plant model) is possible. Here, the challenge would be the control software performance depicted on the simulation environment might defer from the real time rapid prototyping with a huge variation. One of the possible reasons could be the plant model development in simulation environment is not as accurate as the real time hardware physical model. Simulation environment would not guarantee the model to be the same as the real time physical model (plant model). The problem would be critical when it involves multiple physical models (plant model) integrated under one simulation environment. Reference [1] shows the design and development of the Real Time Plant Model using MATLAB/Simulink.

The paper investigates approaches to develop plant model in simulation environment closer to real time physical model using an example plant model. A simple power electronic buck converter model [2] is designed and developed through mathematical and component level modeling at simulation level [3]. Performed analysis on the parameters of the physical or plant model that would directly impact the performance differentiating from the simulation build model and real time model. The results of the mathematical model and component level model are analyzed comparatively. Results of these two models would be later extrapolated over real time plant model output results, which is not included in this paper. Further study would be comparative analysis of the mathematical model, component model and real time hardware plant model results. Simulation environment MATLAB/Simulink is used for the analysis.

A. Mathematical Modeling Approach

Mathematical model is designed and developed through functional equations used to define the physical model (Plant model). Simulink library commonly used block sets are used for this level of modeling [4]. Buck converter mathematical model is built through common Simulink block sets and results are analyzed.

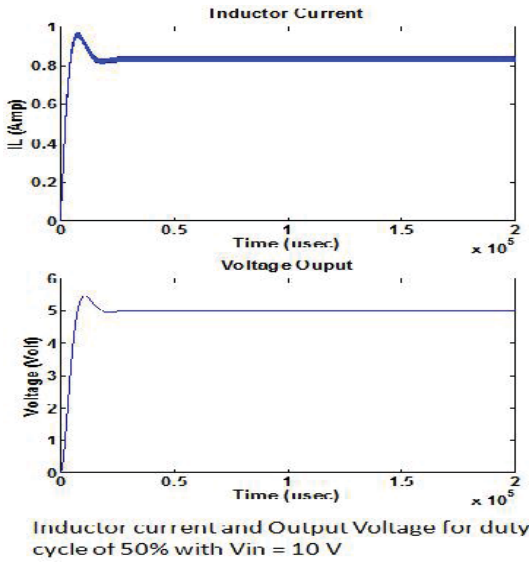
See Appendix A for Fig. 7 which shows a voltage step-down converter model, which converts a dc high input voltage Vin (t), to a dc low output voltage Vout (t). Output results of Inductor current and Output Voltage are shown in Fig. 1 for variation in the duty cycle producing the corresponding PWM. Mathematical buck converter model is developed using (1)-(3) considering the on time condition and off time condition of the PWM.

$$\frac{diL}{dt} = \frac{Vin - Vout}{L} \tag{1}$$

$$Vin - Vout = L\frac{diL}{dt} \tag{2}$$

$$Vc = \frac{1}{C}\int idt \text{ where 'i' = } IL - Iout \tag{3}$$

where L: Inductor, R: Resistor, C: Capacitor



Inductor current and Output Voltage for duty cycle of 50% with Vin = 10 V

Fig. 1 Buck Converter Mathematical Model Outputs

B. Component Physical Modeling Approach

Component Physical (Plant model) model is designed and developed using the Simulink library using specific tool for example SIMPOWER System Library. Mostly Simulink SIMPOWER library block sets are used for this level of modeling [5].

Buck converter component level model is built through SIMPOWER library blocks and results are analyzed. See Appendix B for Fig. 8 which shows voltage step-down converter model designed and developed using component elements similar to real time hardware plant.

Fig. 2 shows the outputs inductor current and output voltage for 50% duty cycle ratio. The observed output voltage is less

than 5 Volts for an expected output voltage of 5 Volt. The difference is due to the parameters effects of the MOSFET, Diode used in the component model.

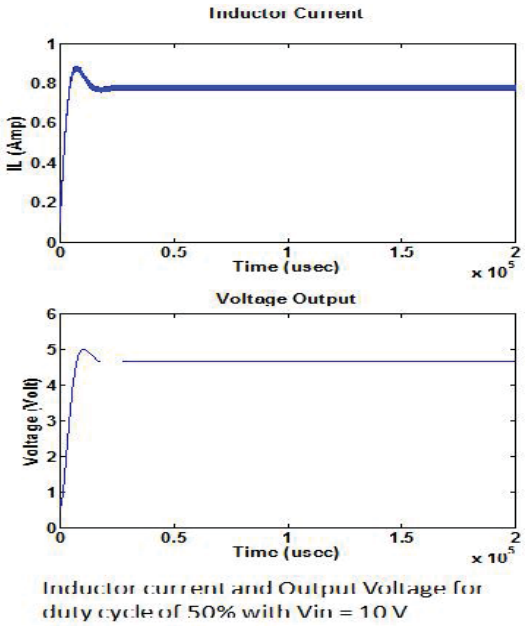


Fig. 2 Buck Converter Component Model Outputs

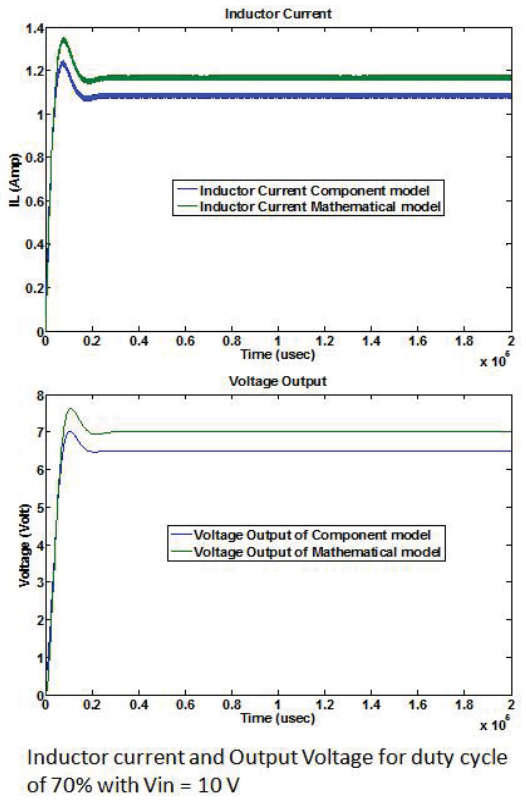


Fig. 3 Comparison of Component Model and Mathematical Model – Current & Voltage

As observed in the model, physical components from Simulink library are used, but still the performance in terms of transient and steady state output might not be similar to the real time hardware plant and mathematical model. Building the component model as much as equal to the real time hardware plant requires fine tuning the parameters of the elements which might deviate from the original data sheet parameters of the elements used in the hardware plant. The behavioral performance cannot be accurate as that of real time hardware plant as few aspects like thermal effect not addressed in this component level modeling.

C. Comparative Analysis of Mathematical Model and Component Model Results

As explained in earlier sections, mathematical models and component level models are developed at simulation environment to depict the performance of the plant model with controller. The results of both the models would be bench marked with real time hardware plant results which are not listed for now. Pre-design verification of the control algorithm is much possible in MATLAB/Simulink environment [6] for which plant model should be developed as much as closer to the real time hardware plant model. Fine tuning of the mathematical model and component models might be required to ensure the performance similar to real time hardware plant.

D.Approach for Multiple Test Case Execution

Multiple test cases are executed for complete validation of the control algorithm on real hardware plant. When testing the complete hardware plant with all possible test scenarios, there are possibilities of failure or damage of the plant due to any unwanted test execution on real time hardware plant.

MATLAB/Simulink models are much helpful for such instances where few particular test cases cannot be directly acceptable on real time hardware plant. Multiple test cases are segregated as per behavioral, performance, fault check etc...

Analysis of the test cases and choosing the applicable model i.e. Mathematical model or Component model would be a challenging task. Some test cases would be applicable to Mathematical model, while other test cases would be applicable for Component level model. Correct listing of test cases to models would result complete testing of the system at simulation environment successfully. For illustration, consider few examples of control signals test case execution i.e. impulse signal test, power up test, power switching test cases and behavior test cases like voltage distortion, current distortion etc. The voltage distortion and current distortion are more applicable to component level model while the impulse test can be executed on mathematical model. There is no mandatory conclusion to execute a test case on particular model, but choosing appropriate test cases for each model and analyzing the results would give better approach for complete system testing at simulation environment.

Test cases applicable to mathematical model are run and results analyzed. Test cases applicable for component model are run and results analyzed. Combination of test cases execution on mathematical model and component model will

give better results for depicting the overall performance of the system to bench mark with the real time hardware plant. Fig. 3 shows the comparative results of mathematical and component model for Inductor current and Voltage outputs for a duty cycle of 70%. Fig. 4 shows the comparative analysis for the ripple content in the Inductor current for the mathematical model and component model in simulation environment.

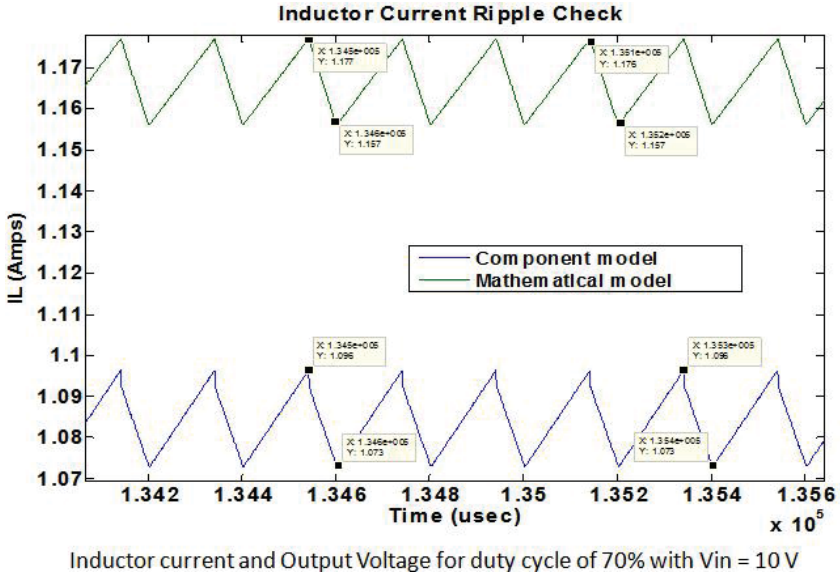


Fig. 4 Comparison of Component Model and Mathematical Model - Ripple check

III. Scheduler Design and Development

Some Simulink plant models with controller have multiple subsystems (function call subsystem), these subsystem blocks representing their functionality which needs to be trigger sequentially using function calls. Developing function calls for multiple blocks is tedious task. Simulink S-Functions are extensively used to generate function calls [7] but the implementation, updating, creating MEX-Files for every update, compilation for no errors shall be time consuming. This would impact the total design time of the complete system. Analysis performed to reduce the effort for developing function calls for multiple blocks.

A. Approach to Design Scheduler

One such simpler approach can be using Stateflow Chart and using chart commands to develop function calls [8]. This was found very effective as implementation is simpler. It does reduce the design time. One such requirement is analyzed to perform the function call development showing the effectiveness in terms of time consuming, simplicity and accuracy. For example, consider the system requirement which says: "monitor an analog signal for every 500 ms and shall calculate the average value for the last 10 measurements". Here the requirement is to develop Simulink model using common block sets. Such requirement requires sequential order of block execution. First the analog signal represented using signal builder in Simulink has to be checked for every 500 ms, and later sum at iterative process has to be performed for the purpose of finding the average restricted to only 10 measurements. First block would calculate the iterative sum for every 500 ms, and later reset to previous input after 10 iterative steps. Second block shall calculate the average of the sum generated by the first block. So here, function calls are required to trigger the blocks as per requirement. Using Stateflow Charts the function call generation is simpler in design, easy to update and less time consuming. See Appendix C for Fig. 9 which shows the design of the system requirement with Stateflow Charts.

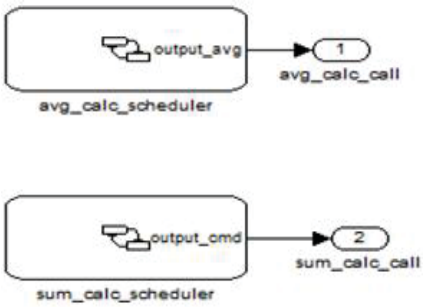


Fig. 5 State Chart used for function call generation

As shown in Fig. 5, State Charts are used to generate function calls at every specified interval time as per requirement. Here two function calls are generated by the Stateflow Chart "avg_calc_scheduler" & "sum_calc_scheduler". Function calls can also be generated using Simulink S-Function approach, but this approach would require more time for updating, creating MEX for every update and resolving compilation errors. State Charts function

call generation practice is much simpler as user can debug the State chart block operation.

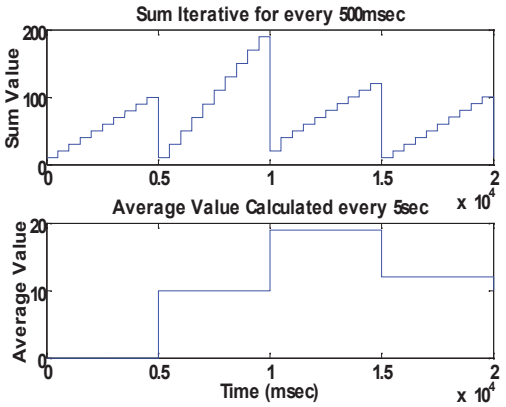


Fig. 6 Sum and Average value calculation – outputs

IV. Multiple Targets Configuration

Sometimes, plant model with controller in simulation environment would demand execution of the controller for code generation in multiple targets (Hardware). Simulink model designed for Plant model with multiple controllers will not simulate if the multiple controllers represent different target configurations. Code generation for plant model comprising controller with multiple blocks and configuration of blocks with different target reference is not possible. Analysis is performed to list the possible solutions for code generation with plant and controller model having multiple target configurations.

The Target Language Compiler (TLC) [9] is feature that is included within Real-Time Workshop (RTW) [10] and enables to customize the C code generated from any Simulink model [11]. Through customization, user can produce platform-specific code.

TLC is used to transform an intermediate form of a Simulink block diagram model.rtw into C code. The TLC works like a text processor, using the target files and the model.rtw file to generate ANSI C code. In order to create a target-specific application, RTW also requires a template makefile that specifies the appropriate C compiler and compiler options for the build process.

Simulink Coder is the tool used for code generation from Simulink models. Code could be individually generated for different target platforms including operating systems and hardware. Each target platform generated code is distinguished by data types fixed-point and floating-point float numbers, big-endian and little-endian addressing, 16/32/64 word size.

The issue is addressed through building a plant model in simulation environment which has two control blocks with different target specifications in the hardware implementation.

Two case studies are analyzed to represent the problem statement for this issue.

In Table I, under the Simulink plant model, there are two different controller blocks with different hardware implementation. The Simulink plant model is required to set to generic version for target hardware. In Simulink RTW configuration settings has System target file specification for the model and not for the specific block or subsystem. So the design of Simulink model with different System target file for each block is not possible.

TABLE I
Plant Model With Generic Hardware Target

	Hardware Implementation	System Target File	Language
Plant Model	Generic	grt.tlc	C
Controller Block-1	Hardware target -1	ert.tlc	C
Controller Block-2	Hardware target -2	ert.tlc	C

If the controller blocks are model referenced blocks, then Plant model can be specified with grt.tlc and controller models with ert.tlc with different hardware implementation. This plant model will not simulate as the controller models are referred in the plant model, which have different System target file specified.

Code generation for this model is restricted as the System Target File for plant model is different from the controller blocks.

One of the probable solutions is to set the plant model, the model referenced controller blocks to generic configuration. As the generated code is in the generic mode, this cannot be used directly onto the hardware setup, for real time behavior.

TABLE II
Plant Model With Specific Hardware Target

	Hardware Implementation	System Target File	Language
Plant Model	Hardware target -1	ert.tlc	C
Controller Block-1	Hardware target -1	ert.tlc	C
Controller Block-2	Hardware target -2	ert.tlc	C

In Table II, the Simulink plant model is set with the hardware configurations of any of the model referenced controller blocks. Code generation build process is restricted as the hardware implementation of one model referenced controller block does not match with another model referenced controller block.

A. Approach to Multiple Targets

One approach can be using hardware implementation as "generic" and System Target File as "grt.tlc" for the plant model and model referenced controller blocks. See Appendix D for Fig. 10. This configuration setting will allow the Simulink model to simulate. For code generation customized TLC file has to be specified to the configuration settings. The customized TLC file will have template to generate code for model referenced controller blocks with different target specification.

V. Conclusion

Approach to design and develop real time physical (plant) model at simulation level are investigated through mathematical and component models. Further analysis would

be implemented for comparative study of the two modeling results with real time hardware plant results.

Scheduler design in MATLAB/Simulink using State Chart are investigated through an example model and found to be very useful in terms of design time, update for every change etc.

Code generation for Simulink model comprising multiple controller blocks with different target configuration settings are investigated and analyzed. Probable solution of code generation with multiple hardware configurations is discussed in detail. Further investigations would be performed for enhancing the TLC options.

APPENDIX

A. Buck Converter Mathematical Model

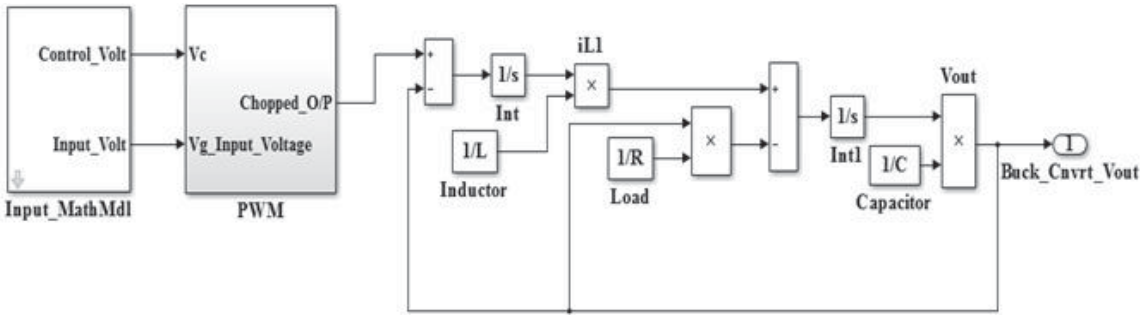


Fig. 7 Buck Converter Mathematical Model

B. Buck Converter Component Level Model

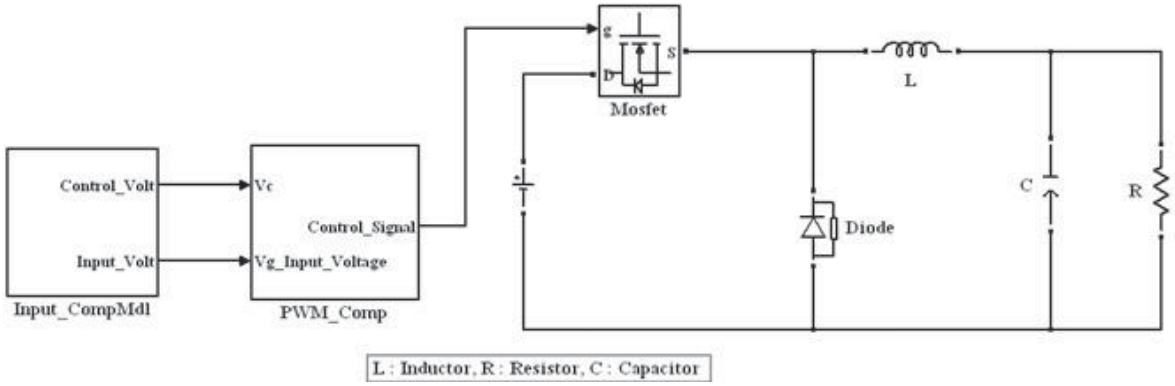


Fig. 8 Buck Converter: Component Level Model

C. Simulink Plant Model with Multiple Function Call Subsystems

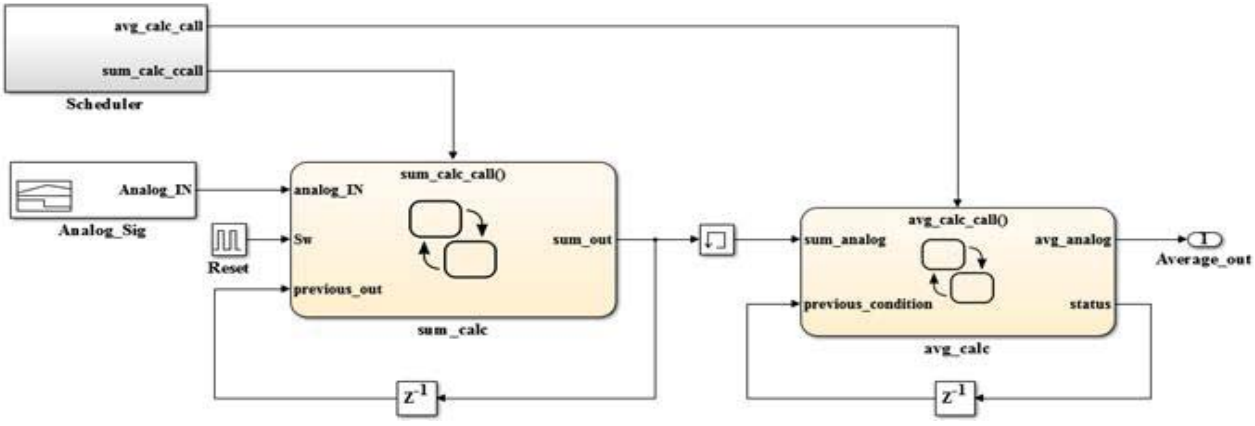


Fig. 9 Simulink Plant Model with multiple function call subsystems

D.Multiple Targets Configuration Approach

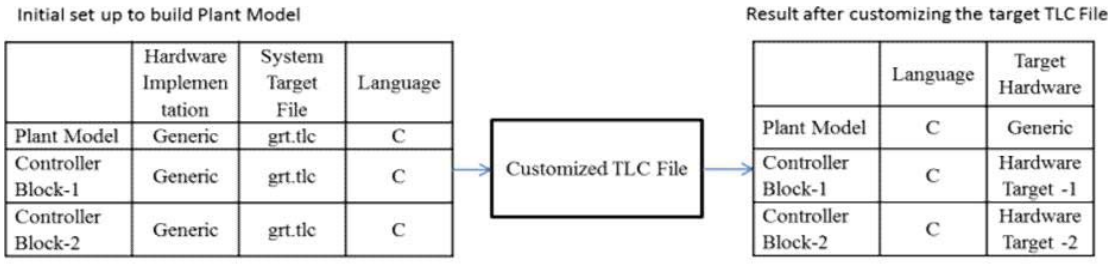


Fig. 10 Multiple Targets Configuration Approach

ACKNOWLEDGMENT

MATLAB Simulink product information is taken from the MathWorks website and we acknowledge the same. We acknowledge the information taken from the paper "Detailed SIMULINK Model of Real Time Three Tank System" on Real time model development in Simulation environment. The buck converter system example theory has been taken from the application note "Buck Converter Design" by Infineon Technologies and we acknowledge the same.

REFERENCES

[1] Petr Chalupa, Jakub Novak, Vladimir Bobal, "Detailed SIMULINK Model of Real Time Three Tank System", Recent Researches in Circuits, Systems, Communications and Computers, Available Online: http://www.wseas.us/e-library/conferences/2011/Tenerife/CSCC/CSCC-26.pdf.

[2] Jens Ejury "Buck Converter Design" Design Note DN 2013-01 V1.0, January 2013, Available Online: http://www.mouser.com/pdfdocs/buckconverterdesignnote.pdf.

[3] Supplementary notes and materials ECEN 5797 Fall 2015, "Converter System Modeling via MATLAB/Simulink", Available Online: http://ecee.colorado.edu/~ecen5797/course_material/SimulinkSlides.pdf.

[4] The MathWorks, Inc. Solutions "Mathematical Modeling", Available Online: http://in.mathworks.com/solutions/mathematical-modeling/index.html.

[5] The MathWorks, Inc. Solutions "Physical Modeling", Available Online: http://in.mathworks.com/solutions/physical-modeling/index.html.

[6] The MathWorks, Inc. Solutions "Verification, Validation, and Test", Available Online: http://in.mathworks.com/solutions/verification-validation/index.html.

[7] The MathWorks, Inc. Documentation "Function-Call Subsystems and S-Functions", Available Online: http://in.mathworks.com/help/simulink/sfg/function-call-subsystems.html.

[8] The MathWorks, Inc. Documentation "Schedule Execution of Simulink Subsystems", Available Online: http://in.mathworks.com/help/stateflow/ug/scheduling-execution-of-simulink-subsystems.html.

[9] Petr Alexeev, "Customizing Simulink code generation by Target Language Compiler (TLC) files" Posted on 2011/11/03, Research activities, Available Online: http://blogs.abo.fi/alexeevpetr/2011/11/03/customizing-simulink-code-generation-by-target-language-compiler-tlc-files/.

[10] The MathWorks Inc. Reference Guide Version 5 "Target Language Compiler™ For Use with Real-Time Workshop®", Available Online: http://mntek3.ulb.ac.be/Matlab/pdf_doc/rtw/targetlanguagecompiler.pdf.

[11] The MathWorks, Inc. Documentation "Simulink® Coder™ Target Language Compiler" © Copyright 2011–2015 by The MathWorks, Inc., Available Online: http://www.mathworks.com/help/pdf_doc/rtw/rtw_tlc.pdf.