International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 29857-29865

https://dx.doi.org/10.37622/IJAER/10.11.2015.29857-29865

Model Based Design Approach in Automotive Software and Systems

P.Sivakumar [*1], B.Vinod [2], R.S.Sandhya Devi [3], S.Poorani Nithilavallee [4]

*[*1] Assistant Professor, PSG College of Technology, Department of Electrical and Electronics Engineering, Peelamedu, Coimbatore, Tamil Nadu, India.*

[2]Head, Department of Robotics & Automation Engineering
PSG College of Technology, Peelamedu, Coimbatore, Tamil Nadu, India,

[3]Assistant Professor, Kumaraguru College of Technology, Department of Electrical and Electronics Engineering, Chinnavedampatti, Coimbatore, Tamil Nadu, India.

[4] PG scholar, PSG College of Technology,Department of Electrical and Electronics Engineering, Peelamedu, Coimbatore, Tamil Nadu, India,

Abstract

Electronics has become the brain behind a vehicle's functioning in today's world. Increased complexity due to increase in need for customized functionality coupled with limited time to market have caused the automotive industry to switch to Model Based Design (MBD). In MBD the model remains to be the central artifact which is refined throughout the entire development process. Process and technology advancements in the semiconductor industry have helped revolutionize the automotive electronics. Certainly, the advancements in embedded software tools such as static code checkers, debuggers and hardware emulators have helped to solve some problems associated with system design and verification. The MBD approach, which has been followed by the aerospace industry for decades, has now moved on to the automotive industry because from a technical standpoint, this improved process makes use of tools and methodologies that facilitate a systematic and methodological control design, alleviates the need for experimental system calibration and allows for system integration and requirements testing in early development stage.

Keywords: MBD, Tools, Design Language, Auto code, Embedded coder.

Model Based Design – An Overview

In the last few decades, car producers and OEMs have worked on drastic improvements in the area of mechanics, improvement in quality and logistic area. Considering the past decade, it is in the area of electronics where tremendous change has taken place with the cost of electronics development occupying nearly 40% of the total production costs [1] as many engine parameters are actively observed and controlled in real-time. Inevitable demands of today's automotive industry is to meet the short development time for the cars in total versus the longer software development time at the same time catering to the high safety requirements, growing complex functions and increased interaction between them. To meet the above challenges hand-coding based software development methods is substituted by model based development. These changes are mainly due to two forces [1] say evolutionary automotive system development dealing with iterated integration of new functions into existing functionality; and the platform independent development reducing the amount of reengineering/maintenance caused by changing hardware generations. Model based development process is vital in improving software quality and enforcement of consistent structures and architectures. However model driven development requires the team members to be systematically trained in modeling (UML).

Model-Based Design (MBD) [9] is a math-based visual method for designing complex control systems and is being used successfully in many industrial, automotive and aerospace applications. The design methodology includes the four key elements of the development process: modeling a system (from first principles or system identification), identifying a controller for the system, simulating the system, and programming the controller. Various development phases involved in system design is supported by a number of approaches from Model-in-the-Loop (MIL), Software-in-the-Loop (SIL), Hardware-in-the-Loop (HIL), Rapid-Control-Prototyping (RCP), or Component-in-the-Loop (CIL).

Traditional Design Process Vs Model Based Design Process

The traditional development process [2] involves using paper specification generated with requirements gathered from different sources, constructing physical prototypes for design, writing code to implement algorithm and control logic and finally using test vehicle for verification and validation. The major difference between the traditional development process and the model based development process is the former relies on verification, validation and testing at the end of the process whereas the later helps in doing the same at an earlier stage and hence the higher cost of fixing errors is reduced. In model based development the system is central throughout development from requirements to final validation and simulation at each step verifies the design requirements and hence this method helps to focus on design instead of coding. The major setbacks for adopting this design methodology is lack of features that supports model synchronization and dependence on the tool vendor's capability for assuring accuracy of the model. Model based development process of a design provides greater communication between the colleagues and earlier verification but has the disadvantage of high process redesign cost that includes tool cost, employees training, re-generation of hand coded projects.

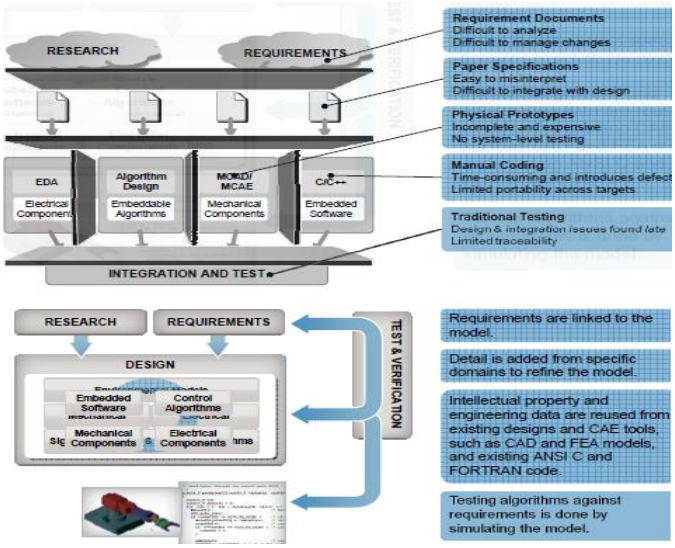

Figure 1: Traditonal vs Model Based Design Process

Various Aspects of Model Based Design

There are numerous references explaining a different approach and exploring different tool strength for achieving MBD implementation. Model based process supported by the tool DOORS [3] is used for requirement management whereas the CASE tool ARTiSAN Real Time Studio is based on UML and CASE tool Ascet SD is for specification and design purpose. These tools plays a vital role in certain phase of the design. For e.g. The tool DOORS is one of the leading tools in automotive with widespread use supporting extensive feature analysis and requirements tracing. Artisan Rts supports UML diagrams and helps in modeling the system architecture. Ascet SD offers a developing environment for ECUs and supports code generation. The transition from UML tool Artisan Real Time Studio to Ascet SD is a major problem [3] because of the semantics being different and a rule checker is proposed that supports model transformation. Embedded Systems Modeling language [4] helps in streamlining control design with software modeling employed on an experimental platform with time triggered paradigm. Integration of components from control design phase with components from software generation, deployment on an actual framework is a major challenge which is met by modeling approach. As the role of software in-vehicle has grown dramatically there are many tools for embedded software design boasting features like object-oriented concepts for structuring and state based behavioral models. However these tools differ in the modeling elements and semantic details and thus exchange of models between different tools is difficult because of the semantic transformations to be done. Thus integration of tools that had in-built supporting features for model transformation marked one aspect of model based design.

Improvement in process technology and electronic design automation has transformed the semiconductor integrated circuits design. Development tools such as

assemblers, linkers, compilers, debuggers are used in making the software platform independent. But the increased complexity with more and more integrated systems has made the software verification activities more time consuming than actual software creation. Hence software verification activities have been pushed to system level with the help of virtual systems and model based methodology [5] helps in a complete virtual implementation with the use of architecture exploration tool that facilitates the design of CPUs, memories, and peripherals.

Code generation is an important benefit of MBD methodology and standardization is necessary to handle software complexity and enhance modularity, flexibility, portability, reusability, etc. Standardization can be achieved via a set of procedures and guidelines, well defined architectural and structural requirements. Standardization can be achieved with the help of guidelines provided by MathWorks Automotive Advisory Board for Matlab based Models used for code generation and modeling. The significant initiatives in automotive industry is AUTOSAR (Automotive Open System Architecture) a standard architecture for designing ECUs and has been the unified result of several companies, automobile manufacturers, suppliers and tool vendors. AUTOSAR helps in the development of application software that is abstracted from the underlying hardware, communication technology and operating system. Hence the models developed should be such that the component configuration, interface definition and abstraction must allow easier implementation of AUTOSAR compliant software.

Autocode Generation

Today Matlab is being widely used to design and control models. Two major autocode generation tools that supports Matlab/ Simulink/ Stateflow models are Real-Time Workshop Embedded Coder from The MathWorks. Targetlink makes code generation and the same tool can be used for scaling. Real Time Embedded Coder is an integrated tool with Simulink. For code generation Fixed Point Toolbox has to used. Both of these tools supports single subsystem code generation, provides control over the generated code interface and supports MISRA (Motor Industry Software Reliability Association) standard C code rules. Embedded Coder has few working limitations too as compared to TargetLink since it uses lesser RAM memory. However TargetLink has smallest execution time.

Matlab/Simulink toolboxes plays a dominant role in MBD. But the smaller firms have financial problems in buying and maintaining products of MathWorks and hence the possibility of open source alternatives such as GNU Octave, Scilab should be considered. GNU Octave is a Matlab clone with internalization of languages and command line support. However, there is no interface for building block diagrams like Simulimk. Scilab is an open source tool Scilab and comes with a convenient user interface with command window, command history, file browser and variable browser [8]. Besides internalization into several languages, it has a graphical interface for block diagrams called Xcos, and a library of components. M-scripts and VB macros developed in-house[6] are used as an alternative to the commercially available AUTOSAR block set from dSPACE [7].

Modelling Language and Tools – Additional Examples

Besides Matlab and dSpace, Autonomie software [10] provides an integrated environment and set of processes for managing, interconnecting, and integrating dynamic models of vehicle components and subsystems, to build and execute complete system simulations. Each module of a system designed using Autonomie software (e.g., plant or control) can be either represented by a set of equations or by its hardware.

UML is the widely used modeling language. Besides it, there are also modeling languages such as Clafer (class, feature, reference) which is a textual language [11] with minimal syntax. Clafer tools translates textual model into various formats and supports checking for various constraints.

Maestro [12] is a Model-Based Systems Engineering (MBSE) environment for design and simulation of complex electronic systems using Orchestra—a simulation tool developed at Sandia National Laboratories. It is deployed as a plugin for MagicDraw and uses Orchestra Domain-Specific Language (DSL) which is based on SysML. In this, the graphical design information is converted into executable java code however the need for close familiarity with a Java design environment is a barrier to wider adoption of the simulation tool. A number of other tools such as AIDA, Upaal, Ptolemy etc supports MBD.

Quality and Compliance To Safety Standards

Quality assurance and functional safety of safety critical automotive applications is an important segment of MBD. Safety critical functions and functions with high speed requirements are generally hand coded due to lack of trust in the code generator in comparison with the hand coded functions. However various tools such as Embedded Coder supports code generation in accordance with various safety standards like IEC 61508. For example, Model Advisor, a part of Simulink, checks the model or subsystem for conditions and configuration settings that can result in inaccurate or slow simulation or generation of inefficient code.

Numerous tools are available each one supporting AUTOSAR compliant code generation, MISRA standard, MAAB standard etc, but the tool efficiency determines the success of using model based approach in the development process. A tool should support the development lifecycle and its usage should decrease the risk of the systematic faults in the system under development.

MBD of a controller starts with function definitions, which represents real-time objects and interfaces. Data flow is established and functional networks are simulated. These functions are mapped on to its respective elements in the technical architecture, buses are configured and simulation is done for failure risk analysis. The code generated is implemented in target system and followed by testing, debugging, fault back tracking.

Embedded Coder – Simulation

Embedded coder extends MATLAB Coder and Simulink Coder products with features that are important for embedded software development. Code is generated and optimized for a specific target environment. Autocode consists of C files that are derived from a higher-level design by software program. The sources files are compiled and linked using a normal compiler to produce an executable software application which runs on target hardware. Hand coding requires several iterations of design especially in prototyping work due to which the proportion of time spent increases. This scenario exemplifies the real gain of auto code generation in terms of time and cost savings but it has the disadvantage of the code being inefficient in terms of RAM, ROM or CPU consumption, difficulty in performing low level I/O and communication operations etc. At present software with additional functionality of generating ROM efficient code is also available.

The first step in applying Embedded Coder configuration options to the application development process is to consider the application objectives, with respect to efficiency, traceability, safety, and map to code generation options in a model configuration set.

Developing a model for code generation consists of various steps such as configuring the signal, input and output ports, initializing states, setting up configuration parameters, setting up model with state chart or Matlab function block

In mission-critical application development and certification goals, the models or subsystems and the code generated must comply with one or more of the standards and guidelines.

An example of window and door control using embedded coder was experimented and the simulation windows is as follows:

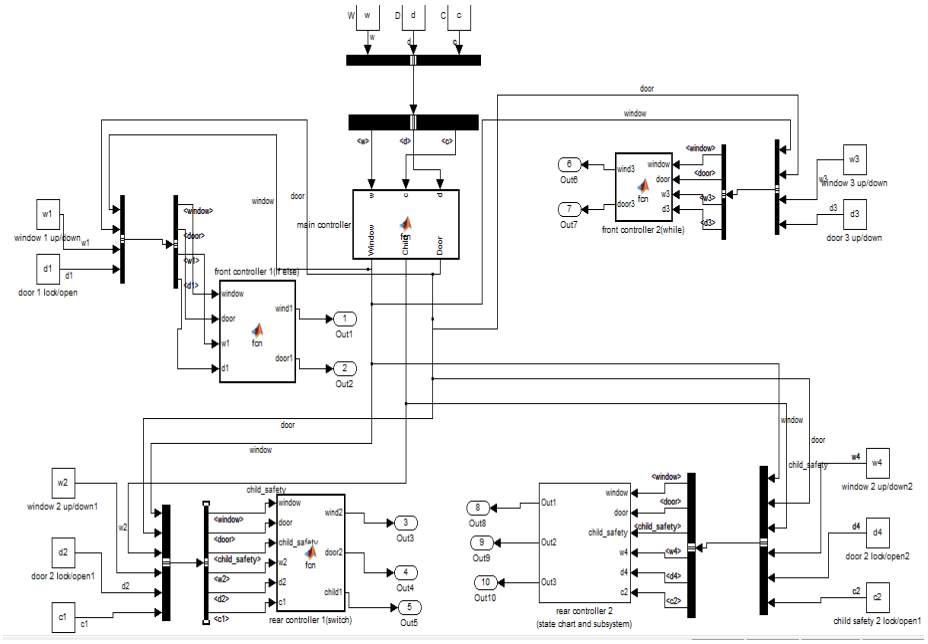

Figure 2: Window and Door Control Model in Matlab

The simulation is done with a main controller and four individual controllers (two for front and two for rear). The individual controllers are responsible for the particular system control while the main controller can exercise global control over all the windows and doors.

The following shows a subsystem modeled within the rear controller. The modeling can be defined using state charts or user defined functions.

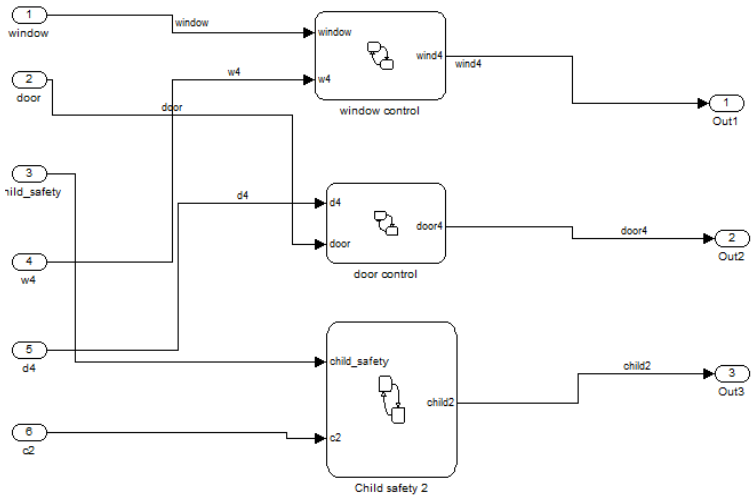

Figure 3: Subsytem in Rear Controller 2

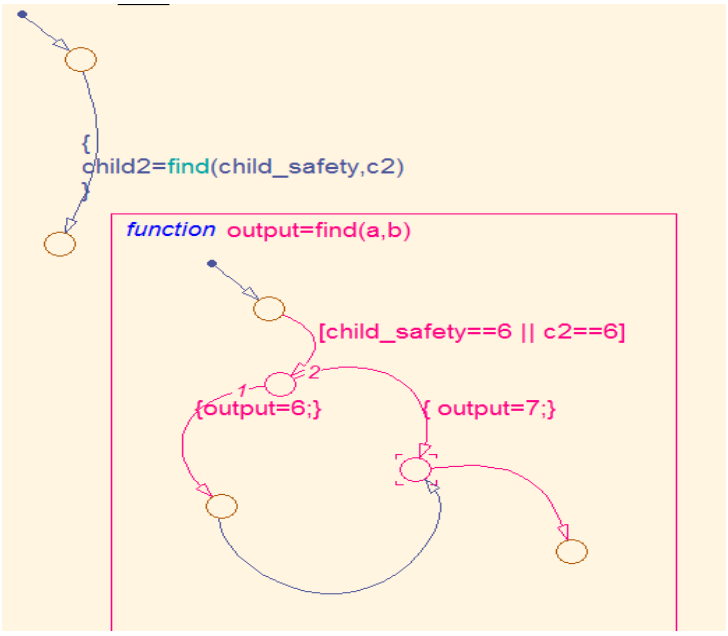

Figure 4: State Flow Chart for Child Safety in Rear Controller 2

Likewise the various controllers are defined and the code generation is implemented. The code generated can be tailored to specific needs such as ROM efficiency, RAM efficiency, compliance to safety standards, AUTOSAR compliance etc. Thus autocode is generated with more importance being given to the design rather than writing code or its implementation with importance to semantic details.

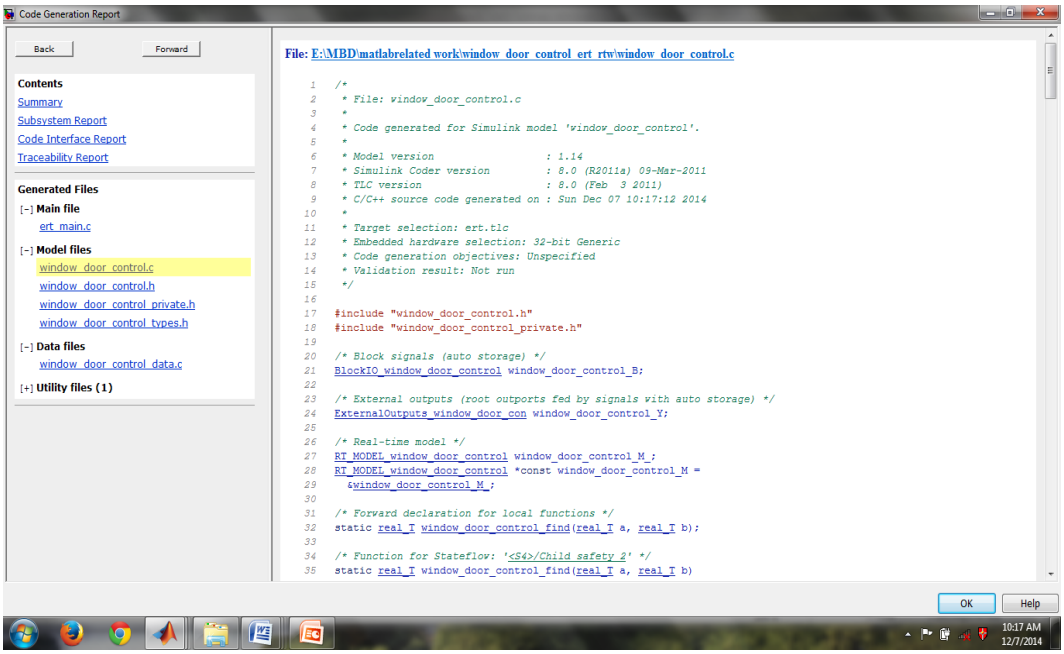

Figure 5: Code Generation Report

Conclusion

In this paper a concise explanation of Model Based Design, and its vital role in automotive industry, its benefit in the long run, the various pros and cons associated with it, the tool infrastructure available and the associated tool related constraints has been discussed. The growth of electronics has been exponential especially in the last decade. Customers require increased value for the expenditure they incur. This has added further pressure to the automotive industry struggling to meet the ever increasing requirements at a shorter development time taking advantage of the advances made in the semiconductor industry. Model Based Design approach ensures a definite reduction in development time after overcoming the initial setbacks.

References

[1] Broy, Manfred, Sascha Kirstan, Helmut Krcmar And Bernhard Schätz. "What Is The Benefit of A Model-Based Design of Embedded Software Systems In The Car Industry?". Emerging Technologies For The Evolution

And Maintenance Of Software Models. IGI Global, 2012. 343-369. Web. 25 Nov. 2014. Doi:10.4018/978-1-61350-438-3.Ch013

[2] Tom Egal, Michael Burke, Michael Carone, Wensi Jin. "Applying Model Based Deign to Commercial Electronics Systems". Copyright 2008 The Mathworks, Inc.

[3] Martin Mutz, Michaela Huhn, Ursula Goltz, Carsten Kromke. "Model Based System Development In Autotmotive". Copyright 2002 Society Of Automotive Engineers.

[4] Di Shang, Emeka Eyisi, Zhenkai Zhang, Xenofon Koutsoukos, Joseph Porter, Gabor Karsai, Janos Sztipanovits. "A Case Study on The Model Based Design and Integration Of Automotive Cypber Physical Systems". Control and Automation (MED), 2013 Mediterranen Conference.

[5] Everett Lumpkin, Michael Gabrick. "Hardware/Software Design and Development Process". Copyright 2006 SAE International.

[6] Devendra Rai, T.K. Jestin, Lev Vitkin. "Model-Based Development of AUTOSAR-Compliant Applications: Exterior Lights Module Case Study". Copyright 2008 SAE International.

[7] Guido Sandmann, Richard Thompson. "Development of AUTOSAR Software components with Model-Based Design". Copyright 2008 The Mathworks Inc.

[8] Manfred Lohofener, Hochshule Mersburg. "Model Based Design of Mechatronic System With Open Source Software".REM2013.

[9] Dr. Mirko Conrad. "Model Based Design For Safety Critical Automotive Applications". The Mathworks.

[10] Alexandr Murashkin, Automotive Electronic/Electric Architecture Modeling, Design Exploration and Optimization using Clafer. University of Waterloo 2014.

[11] Manas Bajaj, Andrew Scott, Douglas Deming, Gregory Wickstrom, Mark De Spain, Dirk Zwemer, Russell Peak. " Maestro –A model-based systems engineering Environment for complex electronic systems". INCOSE International Symposium, Jul 9-12 2012, Rome.

[12] Mirko Conrad, Ines Fey. "ISO 26262 – Exemplary Tool Classification Of Model Based Design Tools ". Copyright 2011 The Mathworks and samoconsult.