embedded world 2014
Exhibition&Conference
... it´s a smarter world

Combining the Power of DAVE™ and SIMULINK®

From a High Level Model to Embedded Implementation

Pedro Costa
Infineon
Munich, Germany
pedro.costa@infineon.com

***Abstract*—In a modern real-time control system, the main control algorithm is executed from a microcontroller with a rich subset of peripherals. The microcontroller is in fact part of a much larger system, which due to the high performance of the technology, demands high complexity deliveries, from all the engineers within the development process. Developing and maintaining an optimized real-time control system, while tackling the optimization of the resources of a complex microcontroller, is a difficult task. This paper focuses on using a high level model development flow that combines DAVE™ version 3 and Simulink® to make this task easier. This flow will bring a competitive advantage by reducing time to market, reduce budget overruns through early identification of missing requirements, and still ensure microcontroller resource optimization and portability. The demonstration of the flow is done via a lighting example using the XMC1300 microcontroller.**

Keywords—microcontroller; DAVE™; XMC; Simulink®; high-level; modeling;V-Model; embedded;

I. INTRODUCTION

The engineering world is dominated by time to market, percentage optimization and high algorithm complexity. These facts impose an additional pressure on the development process for already, highly complex implementations, which can lead to project delays, budget overruns and missing requirements. With the increasing system complexity, errors tend to be identified towards the later stage of the design process. This can be due to missing checkpoints or simply because the development process – design, verification and testing – relies heavily on documentation. A development flow that uses model-based design can be used to minimize the impact of these issues within product development. With a model-based design flow, it is possible to evaluate the conceptual work in an early stage without waiting for prototypes, evaluate the system within an iterative environment and to have continuous verification and test of the requirements. A model-based design approach will also permit evaluation and simulation of the multi-domain complexity of the given system, while giving a good ratio between effort and maintenance [1].

A state of the art, real-time control system, can be comprised of several computing units; e.g. microcontrollers and large numbers of discrete components that can be coupled together with mechanical devices. Attempting to optimize a complete system as a whole would be a cumbersome task, which would not only hinder the efficiency of the development process, but would also heavily impact the maintenance and future expandability of the system. Current model-based design flows therefore implement system partitioning that helps to tackle these compounded complexity problems [2][3]. There is no single rule that can be applied to any given system when it comes to partitioning (although a generic guideline is to always sub-divide the system into computation clusters). An important step within the model-based design flow is the possibility to generate the control algorithm for each of the control clusters; e.g. generating the code for a specific microcontroller. This step can be done very comprehensibly in tools like Simulink® from Mathworks®. Using Simulink® the model-based design flow can implement a V-Model process for the control software in a seamless way, and be coupled with specific requirements. In Figure 1 the "Coding" step is then automated code generation within the model-based design flow. Several embedded targets for the control algorithm can be selected within Simulink®.

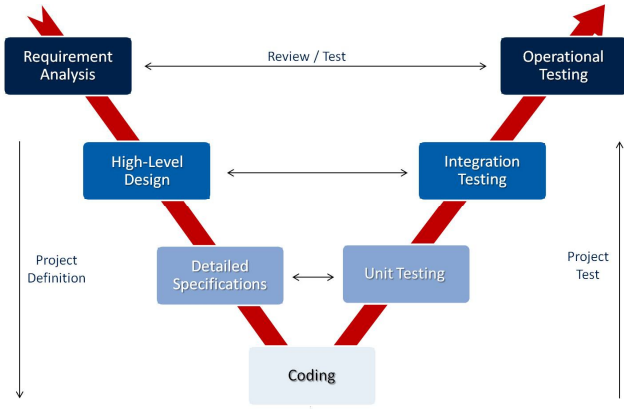

Fig. 1. Generic V-Model process for software development

Software generation for multi-embedded targets (that can coexist in complex control systems), can then also be addressed by the model-based design flow and the subsequent V-Model process [4].

State of the art embedded targets, such as the XMC family from Infineon, can contain hundreds of functional pins, tenths of Timer, Analog-to-Digital or Communication resources. The complexity and the subsequent optimization of the embedded target is not sufficiently addressed in the current design flows and subsequent V-Model processes. The development teams still need to be concerned about the growing complexity of these embedded targets – in terms of optimization, configuration and maintenance. We will address how the DAVE™ version 3 tool from Infineon together with Simulink® from Mathworks®, can be used to implement a V-Model process for a specific XMC target. This flow will not only take the advantages of the model-based design flow but will also allow a complete optimization of the embedded target. The development teams are then offloaded from writing efficient peripherals drivers, maintaining a conflict free resource allocation or optimizing pin mapping for a specific XMC family target. All these actions are performed by the DAVE™ version 3 tool from Infineon. DAVE™ offers support for the complete XMC family, while the link between the model-based design flow in Simulink® is maintained by APIs (Application Programming Interface), which are only visible during code generation. We are going also to demonstrate the flow with a simple lighting system that uses the XMC1300 microcontroller.

II. Model-based Design Flow and the V-Model

Both the model-based design flow and the V-Model process in this paper are tailored for an embedded target implementation. This notion is important to understand due to the fact that the system partitioning and the control algorithm development are conditioned by it. It is also worth noting, that the model-based design flow structure can vary from system to system, depending on the type of components or the need for simulating multi-domain sub parts [5]. It is of the upmost importance for a good concept realization, to be aware of all the model-based design stages.

A. Model-based Design Flow

A generic model-based design flow can be divided in to sub-stages/tasks. The number of stages can of course be influenced by several facts, such as the maturity of the system concept, the compounded or intrinsic complexity levels, and so on. We assume that a Simulink® model-based design flow [6], for an embedded target will be constituted by at least the following stages (with or without breaking the main stages into multiple sub-stages):

- System definition – here the system needs to be defined and the sub-components need to be identified; e.g. a motor control system using a BLDC (Brushless DC) motor. If several sub-systems are identified then each one of them should be addressed separately.
- System modeling – at this stage the system is modeled according to the requirements. The system modeling can have several steps and/or components; e.g. modeling the system with mathematical equations; modeling the system with Simulink® block; etc. At the end of this stage, one should have a block diagram of the specific system.
- Simulation – here the system or sub-system is simulated and the results are analyzed against the requirements.
- Validation – the model results are validated taking into consideration the embedded target (using microcontroller prototyping kits, development boards, etc). The control algorithm is automatically generated from Simulink®.

B. System Partition

Different strategies may exist to properly partition a complete system. Detailed analysis of the system requirements will lead to an optimal sub-partitioning of the complete system and should allow the minimum dependency between the tasks/development stages.

For this discussion we consider that the partitioning is made per computation cluster; e.g. per microcontroller. Sub-partitioning can of course be implemented. A straightforward example can be a motor control system that runs with an XMC4500 microcontroller (therefore considered as a computation cluster), but that could be sub partitioned into: real-time control (main motor control algorithm) and non-real-time control (per example communication routines). As depicted in Figure 2, if a system contains two computation clusters, then each one of them should be addressed independently. If one computation cluster contains more than one control algorithm, then this cluster can be subsequently partitioned or not – this will be decided taking into consideration the dependencies between the control algorithms and future maintenance and/or portability requirements.

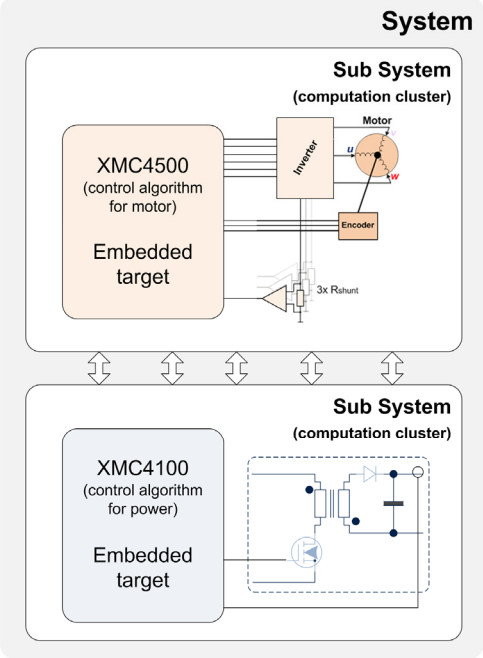

Fig. 2. System partitioning for embedded model-based design flow

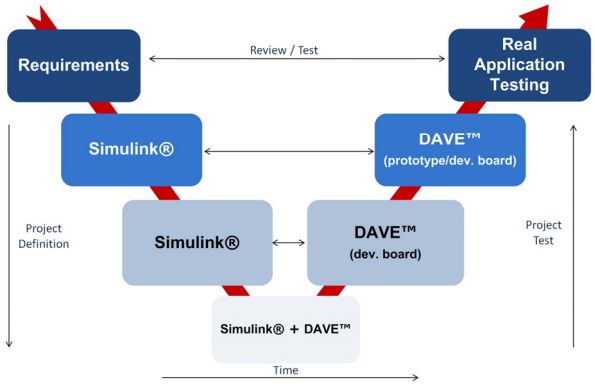

Fig. 3. V-Model stages break down by tool type

C. *V-Model for Embedded Target*

After defining the design stages and the system partitioning, the V-Model process that can combine the advantages of the model-based design and embedded target resource optimization needs to be defined. This V-Model will comprise the automatic generation of two software libraries – one containing the control algorithm generated by Simulink® and the other the optimized embedded target library for the given system, generated by DAVE™.

There are different possible implementations of a V-Model process, but for our purposes we consider the realization of a V-Model process tailored for an embedded control algorithm development. The process will follow the generic rules that dictate: the development stage on the left-side; the verification/testing/validation stages on the right-side; the x axis represents project completeness/time; the y axis indicates abstraction (lowest abstraction at the bottom); horizontal relation between the stages on left-side and right-side. These rules are depicted in Figure 1. For the enclosure of the V-Model process within our model-based design flow, we should have on the left-side the stages based in Simulink® and in the right-side the stages based in DAVE™. Of course the left-side stages are highly dependent on the validation strategy of the system, which could use development boards and/or prototypes to validate with. For our purposes here, we consider that the last stage of the validation is real application testing which DAVE™ is would not be used for – Figure 3.

The stage at the bottom of the V-Model process contains the code for the control algorithm and the complete optimization of the embedded target. This stage is achieved with the combination of both tools – control algorithm generation from Simulink® plus the generation of the optimized configuration of the embedded target for the system by DAVE™.

The integration of the two generated code parts is made in DAVE™. The seamless link between the generated code parts is achieved by using S-functions in Simulink® [7] that allow, during code generation the insertion of the proper APIs that are then recognized in DAVE™. The following section will show how the proposed model-based design and V-Model processes are implemented for a specific example.

III. Development Process Example

To illustrate the advantages of the proposed model-based design and V-Model flow, we have an example using the XMC1300 microcontroller as embedded target. The requirements for the target application/system are:

- A system that needs to control a LED matrix depending on the intensity of the ambient light.
- A light sensor is used to sense the ambient light intensity.
- An LED matrix responds inversely to the light intensity.
- The transition of the LED matrix should be soft/flicker free.

A. *Modeling and Simulation in Simulink®*

After the requirements are gathered, the system can be defined accordingly with the high-level model-based design flow (Figure 4). The first modeling stage can simply contain a mathematical representation of an algorithm that permits the fulfillment of the defined system requirements. It should also be used to identify all the variables of the control algorithm. In the "Detailed Specification" stage, all the sub components are identified and the specific Simulink® blocks are used to model the system. At this stage the components of the system are clustered in several groups – Figure 5: control algorithm; embedded target peripherals; external components. During this stage it is then possible to evaluate the behavior of the system with several other conditioning factors: Analog-to-Digital finite bit resolution; rounding factor for the dimming engine; etc.

B. *Creating the S-Functions for library linking in Simulink®*

After simulating and evaluating the "Detailed Specification" model against the requirements, it is then possible to create the S-Functions in Simulink® that are going to be used to create the link between the two libraries. These S-Functions are only going to be used during the code generation process of the control algorithm. For simulation, the S-Functions are seen as simple data pass through blocks. These S-Functions can be seen as the interface between the control algorithm and embedded target peripherals – in this case, we need two S-Functions: one to interface with the ADC and another to interface with the Dimming Engine Peripheral (the

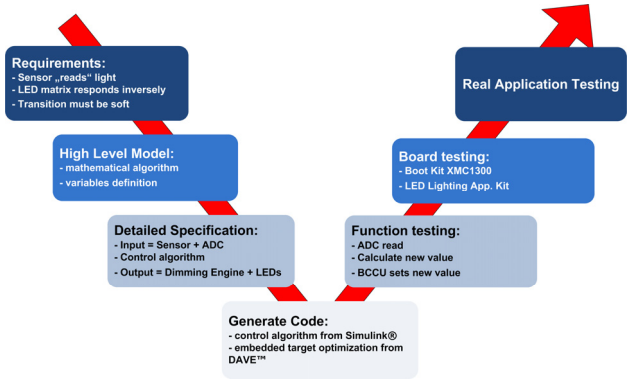

Fig. 4. V-Model stages for Light Sensor example

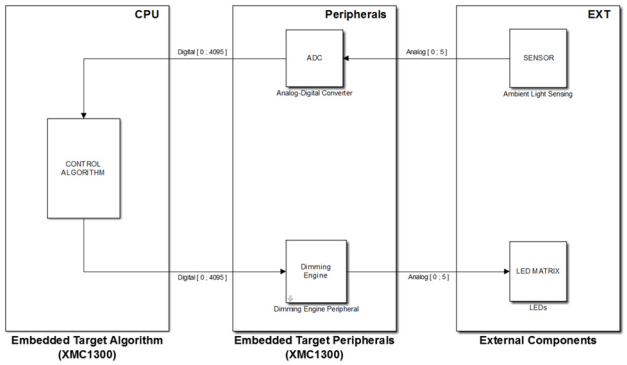

Fig. 5. Detailed Specification stage clustering

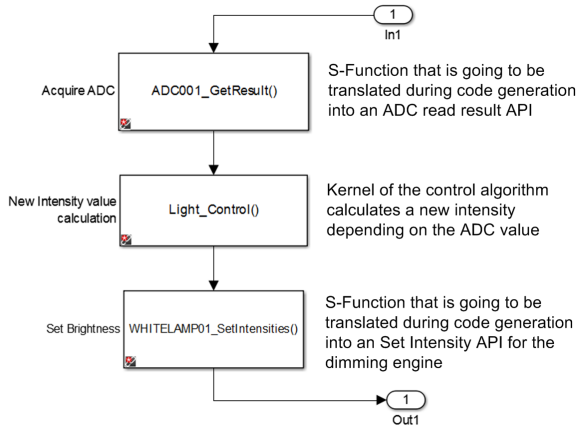

Fig. 6. S-Function interface for the control algorithm

Dimming Engine Peripheral in XMC1300 microcontroller is called the Brightness Color Control Unit, BCCU).

These S-Functions are placed at the input and output stage of the Control Algorithm component. These newly created sub components are just data pass through blocks during simulation. The replacement code for the specific API needs to be written inside the Target Language Compiler (*.tlc*) [8] of these S-Functions. The list of the available APIs for each peripheral needs to evaluated on the DAVE™ tool; e.g. for reading the converted data from the Analog-to-Digital peripheral the *ADC001_GetResult* API can be used. This replacement code is then going to be used during the code generation of the control algorithm in Simulink®

C. *Generating the control algorithm code in Simulink®*

The control algorithm library can at this step be generated automatically from Simulink®. After the S-Function interface has been created, we can generate the code of the control algorithm directly from the Simulink® environment. After the code generation is finished we will have the source code files for the control algorithm and the API interfaces created by the specific *.tlc* files.

D. *Optimized embedded target library creation in DAVE™*

The DAVE™ version 3 tool from Infineon, is a high productivity development platform for the XMC microcontroller families. The code generation is based on predefined and tested application oriented software components, called DAVE™ Apps. The tool is capable of automatically optimizing the resource allocation (via the DAVE™ Solver) i.e. determining which timer should be used to control the Analog-to-Digital converter; which internal connection should be used between two sub resources; etc.

With DAVE™ version 3, we are able to generate an optimized embedded target library for any of the XMC microcontrollers. With the proposed development model, it is possible to combine the advantages of a model-based design approach for the system, with the advantages of having an optimized embedded target library. For any given system, a library for the control algorithm is generated from Simulink® – taking advantage of a model-based design approach; an optimized library for the required embedded target resources is generated in DAVE™.

For this specific example, the embedded target (XMC1300) resources have already been identified in previous stages:

- Analog-to-Digital Converter channel
- Dimming Engine peripheral (BCCU)

At this point a migration is required, from continuous algorithm calculation into to a periodic/interrupt driven calculation. This is achieved by using two additional resources:

- A Timer that controls the periodicity of the algorithm calculation
- An Interrupt that triggers the ARM® Cortex™-M0 CPU to perform the computation. The handle of this interrupt in DAVE™ should exactly match the name of the control algorithm function developed in Simulink

DAVE™ offers a graphical interface where the specific resource blocks can be dropped and connected together – a similar approach as with the Simulink® blocks. When all the required resources are dropped into the project, the DAVE™ solver can be executed to optimize the resource usage and embedded target connections. If this step is done successfully, then the optimized embedded target resource library is created. The DAVE™ graphical view of the resources needed for this development example can be seen in Figure 7.

E. *Integration of algorithm and embedded target library in DAVE™*

The main file for the DAVE™ project can be generated in two ways: using the Simulink® generation environment or using the main file created automatically by DAVE™. When using the Simulink® approach, under the "Custom Code" options, always use a main file template that includes and initializes the methods used by DAVE™.

The interface between the two libraries has been created in the two previous steps: Simulink® has generated the algorithm by using the S-Functions with the proper API replacement code; in DAVE™ the interrupt handler has been named with the control algorithm name. This means, that the next step is as simple as adding the generated files from Simulink® and the control algorithm into the DAVE™ project and compiling. The two libraries are then integrated.

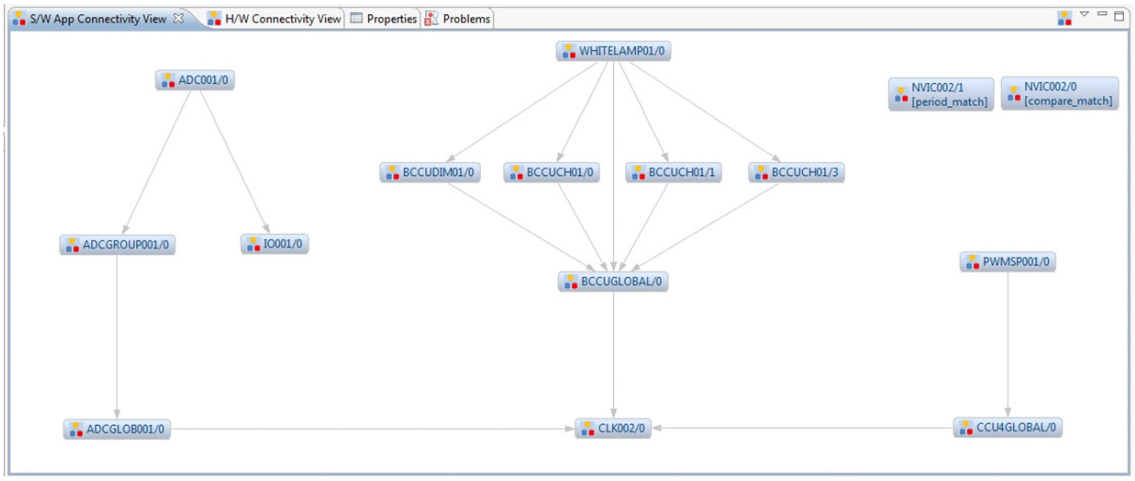

Fig. 7. DAVE version 3 graphical view

IV. BOARD TARGETING

The selected hardware for this system/application was the XMC1300 microcontroller. To perform the second level of validation the LED Lighting Application kit from Infineon was selected. This kit contains the inLight White LED board, which contains a matrix of white LEDs and a light sensor. This board together with the XMC1300 boot kit is then used to perform the intermediate validation of the system (after the unit testing).

After the two libraries are merged together with the DAVE™ tool, the process of deploying the code into the specific hardware is a seamless process – and the accuracy and behavior of the control algorithm can be evaluated against the initial requirements – Figure 8.

V. CONCLUSION

In this article we have described a V-Model process integrated in a model-based design flow, using Mathworks® Simulink® and the Infineon DAVE™ version 3 tool. This process is especially tailored for an embedded target control algorithm implementation. It comprises not only the advantages of a model-based design approach – minimizing time to market, and reducing budget overruns – but also the advantages of having an optimized embedded target resource library. The integration of the two libraries can be done in a seamless way for any of the XMC microcontroller devices by means of the DAVE™ version 3 tool – while the link between the libraries is created in Simulink® by means of S-Function blocks. The flow was demonstrated for a system/application using the XMC1300 microcontroller and the LED Lighting Application Kit.

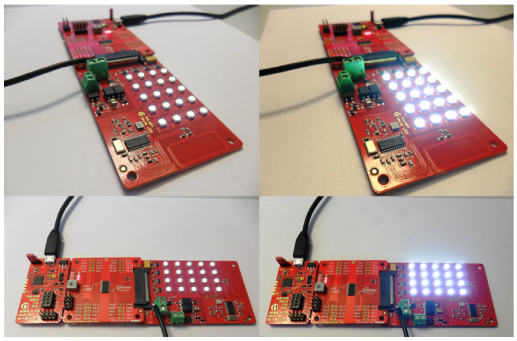

Fig. 8. XMC1300 lighting project validation with LED Lighting Kit

REFERENCES

[1] S. Mahapatra, T. Egel, R. Hassan, R. Shenoy and M. Carone, "Model-based design for hybrid electric vehicle system," The Mathworks Inc., January, 2008.

[2] P. J. Mosterman, J. Ghidella and J. Friedman, "Model-based design for system integration," in The Second CDEN International Condeference on Design Education, Innovation, and Practive, pp. TB-3-10, July, Kananaskis, Alberta, Canada, 2005.

[3] J. C. Jensen, D. H. Chang and E. A. Lee, "A model-based design methodology for cyber-physical systems," Proceedins of the 7th International Wireless Communications and Mobile Computing Conference, IWCMC 2011, Istanbul, Turkey, 4-8, July, 2011.

[4] G. Hodge, J. Ye and W. Stuart, "Multi-target modelling for embedded software development for automotive applications," SAE 2004-01-0269, 2004.

[5] D.J. Hatley, "Strategies for real-time system specification," Dorset House Publishing Co., New York, New York, 1988.

[6] The Mathworks Inc., "Simulink getting started guide", The Mathworks Inc., Natick, MA, September, 2013

[7] The Mathworks Inc., "Developing s-functions", The Mathworks Inc., Natick, MA, September, 2013.

[8] The Mathworks Inc., "Simulink coder target language compiler", The Mathworks Inc., Natick, MA, September, 2013.