Details of the Conference

Name:

International Conference on Information Systems Engineering

Venue:

April 20-22, 2016, Los Angeles, USA

Website:

http://www.icise.org/

Current Status:

The article has been presented at the conference and published in the proceedings. It will be available in IEEE Explore Digital Library.

Combining SysML and Marte/CCSL to Model Complex Electronic Systems

Aamir M. Khan
Department of Electrical & Computer Engineering
College of Engineering, University of Buraimi
Buraimi, Oman.
e-mail: aamir.m@uob.edu.om

Frédéric Mallet
Université Nice Sophia Antipolis
INRIA Sophia Antipolis Méditerranée,
Sophia Antipolis, France.
e-mail: Frederic.Mallet@inria.fr

Muhammad Rashid
Department of Computer Engineering,
College of Computer & Information Systems,
Umm Al-Qura University, Saudi Arabia.
e-mail: mfelahi@uqu.edu.sa

***Abstract*—SystemVerilog is a popular hardware description and verification language aimed at designing and verifying present-day complex embedded systems. With the increasing number of design verification assertions, engineers always feel it difficult to manage the gap between the system specification and the design validation efforts and to cope with the time-to-market factors. An approach is presented for the modeling of system design as well as validation features using the UML standards like SysML, MARTE and CCSL. Finally the approach is demonstrated using an example of traffic light controller.**

Keywords—SystemVerilog; assertions; SysML; MARTE; CCSL; modeling; embedded systems

I. INTRODUCTION

With the increasing complexity of electronic systems and the embedded applications, there is a continual need for the abstract representations of such systems [1], [2]. Modeling such systems is a challenging task as these systems are not only huge in magnitude but are also significantly diverse [3]. Model Based System Engineering (MBSE) is the latest trend in system development methodology which focuses on creating and exploiting the abstract representations of systems. The MBSE approach is meant to increase the design productivity by simplifying the design process, reuse of standard models, and providing the basis for implementing systems. Various research works have mentioned the use MBSE techniques to improve the productivity [4].

Traditionally, Unified Modeling Language (UML) has been used to model software systems [5]. However, it supports MBSE approach for embedded systems through the use of profile extensions. UML profile for Modeling and Analysis of Real-time Embedded Systems (MARTE) and Systems Modeling Language (SysML) are leading examples of this trend [6], [7]. These UML profiles have features to support the specification of diverse structural, behavioral and temporal aspects of complex embedded systems. SysML profile is commonly used to model embedded systems requirements [8] while the UML profile for MARTE specializes in modeling real-time embedded systems [9], [10].

Although UML and its associated SysML/MARTE profiles support variety of modeling features, it is still sometimes insufficient to cater complex and large embedded systems requirements in the model using a single profile [11]. Therefore the combined use of distinct SysML and MARTE profiles has frequently been advocated by various authors and several efforts followed to exhibit the combined use of SysML/MARTE to model structural, behavioral and temporal aspects of complex embedded systems [12], [13].

Once modeled, these complex systems still face the chances of failures due to software or hardware issues or mere human error. Design verification of a system is a process intended to ensure that a system is designed and works as per the specifications. According to the analysts, about 50% of the design effort for an electronic chip consists of the verification process [14]. Assertion-based verification (ABV) plays an important role to handle such complex systems [15]. Assertions are the executable hardware description language (HDL) code that capture specifications and the design intent, usually written in hardware verification languages like SystemVerilog [16], PSL (Property Specification Language) [17] or Sugar [18]. In industrial setups, based on the design specification, dedicated team of verification engineers propose hundreds of assertions to verify the design integrity. But as the system specification is usually given in a natural language, it is not easy to translate those directly into assertions unless they are described in a formal language [19], [20], [21].

This paper proposes a unified approach of system design and its verification based on a formal language. It uses the SysML and the UML profile for MARTE to model the system structure and then embed the design verification properties to these structural models. It proposes to model the assertions (used for design verification) using the SysML parametric diagrams along with the Clock Constraint Specification Language (CCSL) [19]. CCSL is a constraint language, associated with the UML profile for MARTE. It is used here as a formal language to bridge the gap between the realms of models or system specifications (represented by SysML parametrics) and the application/code level at which SystemVerilog assertions can be applied. It gives a precise semantics of SysML parametrics so that SystemVerilog assertions can be derived. Hence the major contribution through this effort is to provide a unified modeling and verification framework based on sound formal semantics.

The rest of the paper is organized as follows. Section 2 provides a brief introduction to the formalisms used in this paper: SystemVerilog, MARTE, and CCSL. Section 3 discusses various related research work and their limitations. Section 4 presents the approach for modeling assertions in SysML using CCSL with the help of a case study. Lastly section 5 concludes with the summary and future work.

II. FORMALISMS UNDER CONSIDERATION

A. SystemVerilog and SystemVerilog Assertions (SVAs)

SystemVerilog is a joint hardware description and verification language (HDVL) based on the Verilog language and standardized by the IEEE standard 1800-2005. For Register Transfer Level (RTL) design, it provides an extended feature-set of Verilog-2005 while for verification it introduces object-oriented programming features more closely related to languages like Java and C++. Assertions are the integral part of the verification feature of this language. An assertion is a statement about a specific functional characteristic or property that is expected to hold for a design. For the electronic components, assertions are the executable HDL code that captures the specification and design intent.

SystemVerilog concurrent assertions are based on a synchronous clock-based semantics. Thus they provide the ability to specify sequential behavior concisely and to evaluate them at discrete points in time. Two of the most commonly used conditional assertion operators are same-cycle implication operator (|->) and the next-cycle implication operator (|=>).

B. Time Structure in MARTE and CCSL

MARTE time model provides a sufficiently expressive time structure to represent time requirements [20] of embedded systems. MARTE time model is a set of logical clocks and each clock can be represented by $\langle I, \prec \rangle$, where I represents the set of instants and $\prec$ is the binary relation on I.

The CCSL [19]is a declarative language annexed to the specification of the MARTE UML Profile. It is used to specify constraints imposed on the clocks consisting of at least one clock relation. A clock relation relates two clock specifications. A clock specification can be either a simple reference to a clock or a clock expression. A clock expression refers to one or more clock specifications and possibly to additional operands. There are three basic clock relations in CCSL: precedence ($\boxed{\preccurlyeq}$), coincidence ($\boxed{\equiv}$), and exclusion ($\boxed{\#}$). A forth relation, the strict precedence ($\boxed{\prec}$) is derived from the precedence relation while the fifth/last relation subclocking ($\boxed{\subset}$) is an injective form of coincidence.

The clock relations can be classified as synchronous, asynchronous, or a combination of both. Subclocking constraint ($\boxed{\subset}$) is an example of synchronous clock constraint based-on coincidence. Each instant of the subclock must coincide with one instant of the superclock in an order-preserving fashion. The exclusion constraint ($\boxed{\#}$) states that the instants of the two clocks never occur at the same time. Non-strict precede constraint ($\boxed{\preccurlyeq}$) is an example of asynchronous clock constraint based-on precedence. Given the relation a $\boxed{\preccurlyeq}$ b, for all natural numbers k, the kth instant of 'a' precedes or is coincident with the kth instant of 'b' ($\forall k \in \mathbb{N}, a[k] \preccurlyeq b[k]$). Mixed clock constraints combine coincidence and precedence relations. An example is defer constraint ($\rightsquigarrow$) which enforces delayed coincidences. The expression $c = a(ns) \rightsquigarrow b$ (read as a deferred b for ns) imposes c to tick synchronously with the nth tick of b following a tick of a.

Hence, CCSL provides synchronous and asynchronous relationships on clocks and supports non-deterministic specifications and provides the adequate semantics to deal with the SystemVerilog assertions. Such an approach will also bridge the gap between system specification and assertions that validate the design and formally describe the specification.

III. RELATED WORK

The presented design approach advocates the combined use of multiple UML profiles to model complex embedded systems. This is endorsed by various distinguished researchers who have raised several concerns and opportunities regarding the combined use of SysML and MARTE profiles. Many researchers have proposed to use the SysML and UML profile for MARTE to model different sorts of electronic and embedded systems [22] - [24]. SysML and MARTE have been used together in several other projects like SATURN Project [26] and MeMVaTEx [27]. SATURN is an EU project aimed at a UML/SysML-based approach for hardware/software co-design. It is based on Artisan Studio software [28] and uses an enhanced SysML profile used to generate SystemC and VHDL code. MeMVaTEx is a model-based methodology for expressing requirements and traceability mechanisms during the modeling process. It relies on standards like EAST-ADL2 [29] and UML2 profiles like MARTE and SysML.

SVAs represent the system constraints. As the constraints are closely related to time, it causes special interest of time representation in UML. Frederic [30] has discussed the use of SysML parametric diagrams along with CCSL (Clock Constraint Specification Language) to represent the laws related to time. The contribution in the paper extends this work and proposes assertions modeling based on this approach. Similarly, Huascar [31] also advocates the use of SysML parametric diagrams and MARTE VSL (Value Specification Language) to specify non-causal models with algebraic and time constraint expressions. Another research-work in the form of TEPE Language [32] defines a SysML based framework for specifying time-constrained properties. This approach has also utilized the SysML parametric diagrams to specify their models. The work in [33] compares the time semantics of CCSL and PSL (which is quite similar to SVAs). Similarly the work in [34] considers the representation/implementation of CCSL in VHDL using the concept of observers. None of these mentioned approaches focus directly on SystemVerilog assertions which contributes heavily to design verification efforts.

The only research work directly targeting the SystemVerilog assertions (SVAs) [35] considers the system behavior implemented in the form of state machines. Then they coupled the assertions, in the form of properties, with these states. But for representing SVAs, they relied only on the comments attached to the classes (effectively not being part of the model). These comments are retrieved from the model and pasted directly in the SystemVerilog code.

IV. MODELING SYSTEMVERILOG ASSERTIONS

The presented framework proposes the modeling of SystemVerilog assertions in SysML parametric diagrams and CCSL. This section firstly presents the case study used in modeling approach followed by the CCSL

representation for the SystemVerilog assertions. Lastly assertions modeled in SysML are presented.

A. *Case Study*

The case study considered is of the traffic light controller taken from the SystemVerilog Handbook [36], as shown in Figure 1. It consists of a cross-road with a North-South highway traffic signal (*ns_light* in the HDL module interface) and the East-West farm road traffic signal (*ew_light)*. There are sensors installed for the emergency vehicles (*emgcy_sensor*) and for the farm road traffic (*ew_sensor*). Highway traffic is only interrupted if there is a vehicle detected by the farm road sensor.

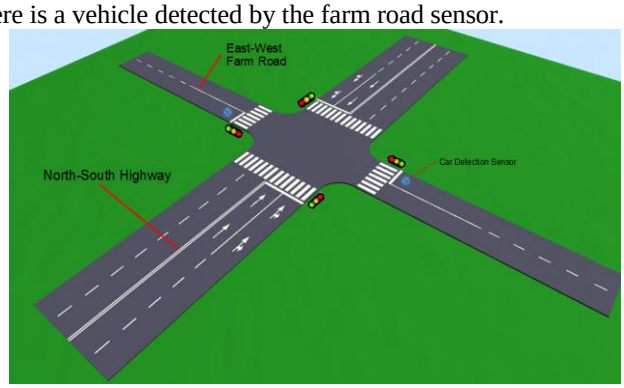

Figure 1. Highway and Farm Road Traffic Intersection

The architecture for the traffic light controller consists of two finite state machines (FSMs) and the timers. The timers ensure the proper delay for all the signal transition as shown in Figure 2. Module interface contains all the ports described earlier used by the SVAs for design verification.

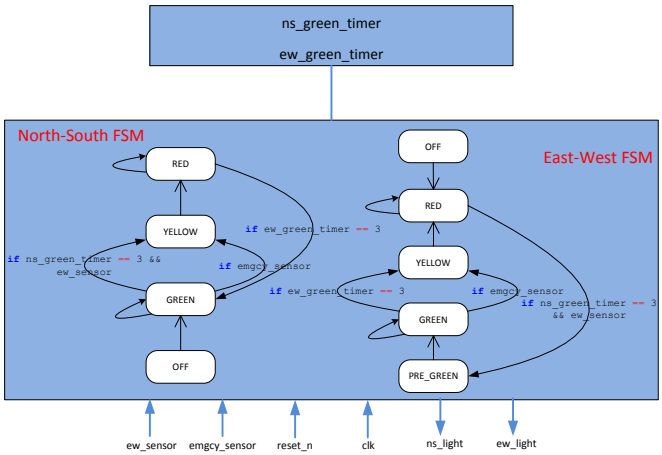

Figure 2. Block Diagram of Traffic Light Module

B. *Mapping SVAs to CCSL*

CCSL enables to assign logical clocks to selected modeling elements, and hence proposes a way to express system safety properties using logical time patterns. Instants of the CCSL clocks represent the occurrences of the events, as discussed in detail in [37]. In the case study, the 'highway light is green' (*ns_light == GREEN*) is considered as an event to trigger conditional assertions in SystemVerilog. Such events representing state machines directly mapped to CCSL logical clocks with notions like *ns_light.GREEN* which will represent a clock that will tick throughout when the corresponding variable has the value GREEN. Next the case study properties are discussed one-by-one to further understand how SVAs are represented in CCSL.

Safety, simultaneous green lights are illegal: The North-South Highway traffic light and the East-West farm traffic lights cannot be GREEN at the same time. Using the same-cycle implication operator, the assertion is represented as,

```
ns_light == GREEN |-> ! ew_light == GREEN;
```

Here the same-cycle implication operator means ns_light and ew_light both cannot be green in the same cycle. Same intention can be represented in CCSL using the exclusion operator as,

```
ns_light.GREEN # ew_light.GREEN
```

State of lights at reset: The North-South Highway traffic light is in the OFF state when the system is reset. Using the next-cycle implication operator, the assertion is represented as,

```
reset_n == 1'b0 |=> ns_light == OFF;
```

Here the next-cycle implication operator means when the reset_n signal goes low, the ns_light turns OFF in the next clock cycle. Same intention can be represented in CCSL using the defer and sub-clock operators as,

```
reset_n (1) ⇝ clk ⊂ ns_light.OFF
```

The same property is repeated for the farm road with its CCSL representation as,

```
reset_n == 1'b0 |=> ew_light == OFF;
reset_n (1) ⇝ clk ⊂ ew_light.OFF
```

State of lights during emergency: Whenever the emergency vehicle sensor is activated, all the traffic signals turn to yellow and then red. Here the assertion combines the next-cycle implication operator with the delay operator. Use of `true [*2] is trivial.

```
emgcy_sensor |=> `true[*2] ##1
ns_light==RED;
```

When the emergency sensor is activated, the light turns red after a delay of two clock cycles (one explicit and the other implicit through |=> operator). In CCSL the same can be expressed as,

```
emgcy_sensor (2) ⇝ clk ⊂ ns_light.RED
```

Similar constraint will be repeated for the farm road as well.

Safety, green to red is illegal, need yellow: Whenever the traffic signal is GREEN, it must not become RED in the next clock cycle.

```
ns_light == GREEN |=> ! ns_light == RED;
```

This property can also be implemented using the CCSL exclusion operator.

```
ns_light.GREEN (1) ⇝ clk # ns_light.RED
```

Farm road light remains green for maximum 3 cycles: The farm road signal must turn back to RED when the timer reaches the count of 3.

```
(ew_green_timer==2'b11) |=> `true ##1
                ew_light==RED;
```

Its logical representation in CCSL can be as,

```
ew_green_timer.3 (2) ⇝ clk ⊂ ew_light.RED
```

Here the ew_green_timer.3 is considered to be a logical clock which ticks for all the instants when ew_green_timer attribute has the value 3. So for all such instants, the ew_light is checked to be red after two clock cycles.

Safety, green-yellow lights at the same time is illegal: To ensure the safety of all the commuters on the road, it is verified that green and yellow traffic lights are not ON at the same moment. Both traffic lights are tested using a single assertion.

```
(ew_light == GREEN |-> ! ns_light ==
                YELLOW;)
and
(ns_light == GREEN |-> ! ew_light ==
                YELLOW;)
```

In CCSL, two simpler constraints can be combined using the intersection (*) operator.

```
(ew_light.GREEN # ns_light.YELLOW)
  *
(ns_light.GREEN # ew_light.YELLOW)
```

In general, the mappings between the SystemVerilog assertions and the CCSL constraints are tabulated in Table I. Remember that these mappings are only for a small subset of the language. Just like SystemVerilog, multiple constraints can be combined together to make a complex sequence.

C. *Modeling Assertions in SysML*

The presented modeling efforts focus on the structural representation of traffic light controller using SysML block definition diagrams (BDDs) embedded with the verification FEATURES USING PARAMETRIC DIAGRAMS.

TABLE I. SVA TO CCSL MAPPING OF SOME COMMON INSTRUCTIONS

SystemVerilog	CCSL
a |-> b	a ⊂ b
a |-> ! b	a # b
a |=> b	a (1) ⇝ clk ⊂ b
a |=> ! b	a (1) ⇝ clk # b
a ##n b	a (n) ⇝ clk ⊂ b
a && b	a * b
a || b	a + b

For the modeling approach, a library of CCSL constraints represented by SysML constraint blocks is created. The package also defines a logical clock signal typed as ClockType from the MARTE time model, as shown in Figure 3.

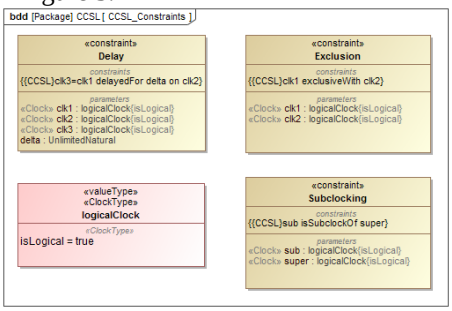

Figure 3. CCSL Constraints Package

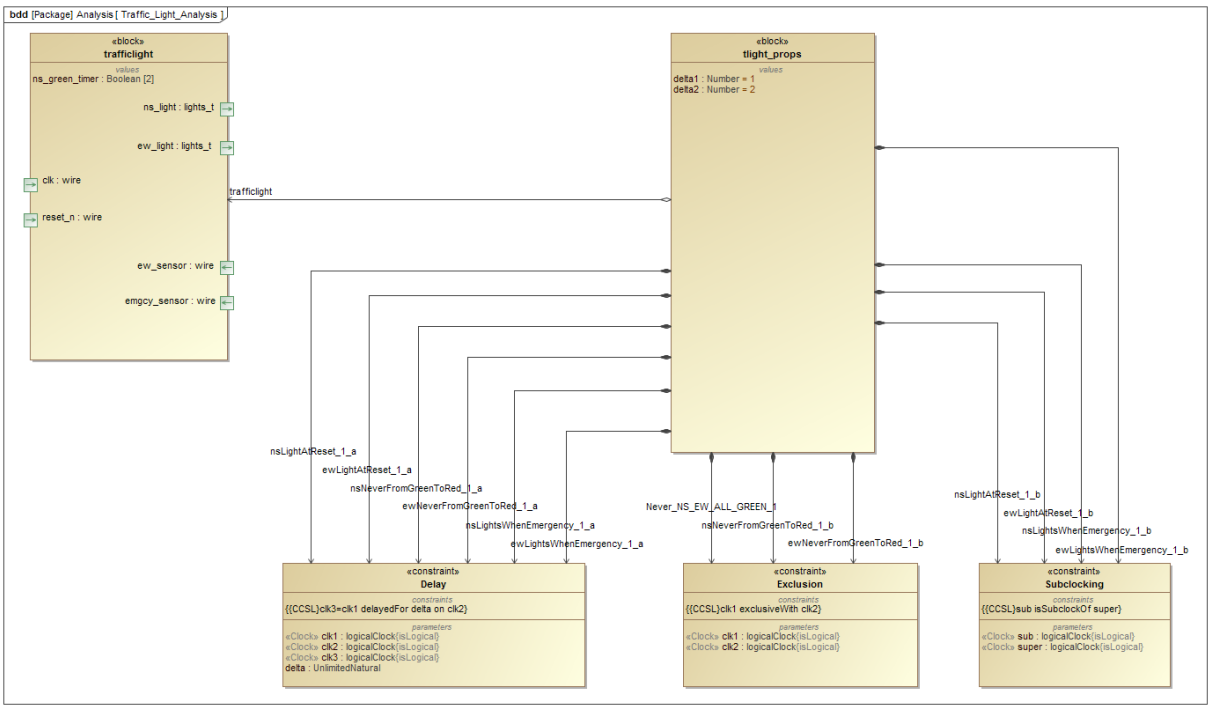

Figure 4. Using Constraint-based Analysis to Represent Assertions

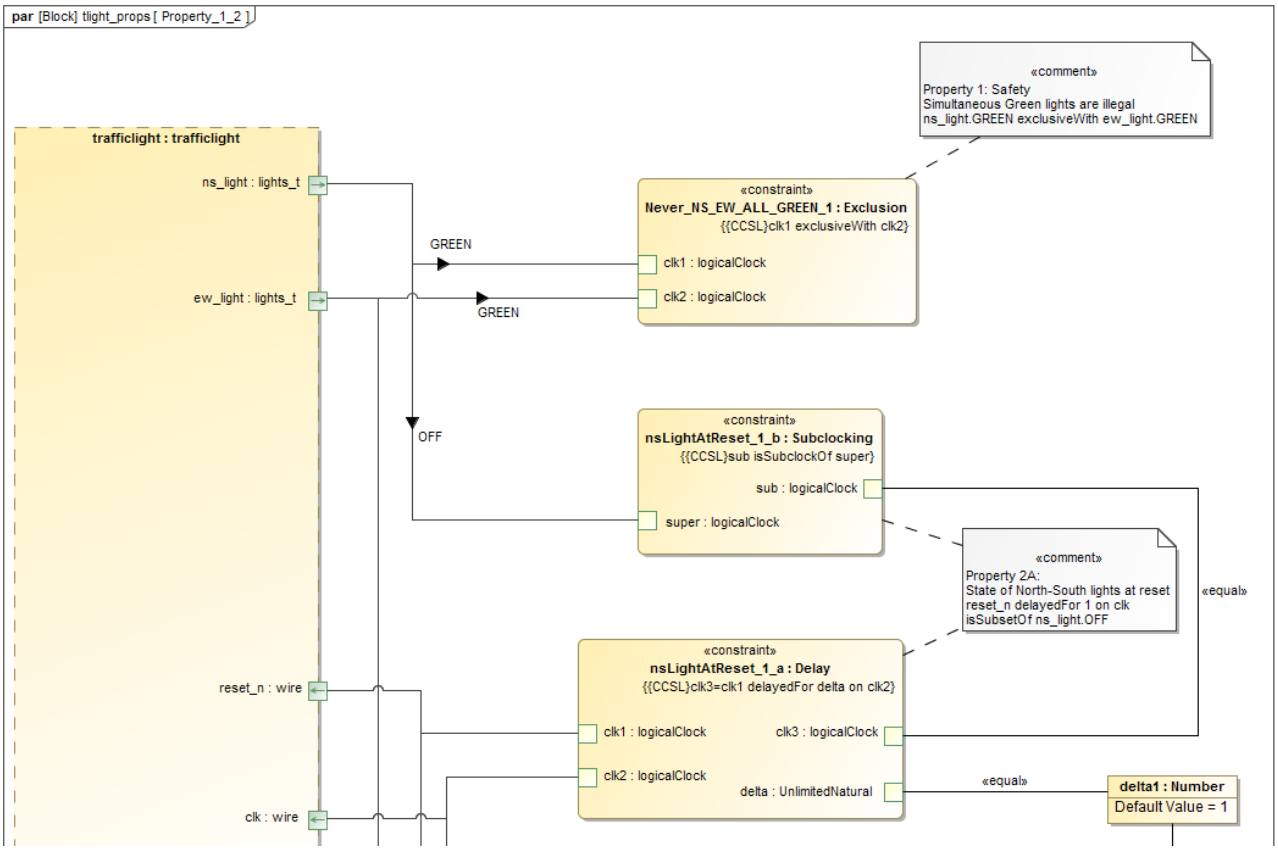

Figure 5. Modeling Assertions using Parametric Diagrams

Figure 4 shows the block definition diagram for modeling the case study verification features. This diagram gathers the design under verification (DUV), testbench and the CCSL constraints and links them together. The block *tlight_props* represents the design testbench having aggregation relation with the design under verification *trafficlight* and composition relation with the desired CCSL constraints. The CCSL constraints have no direct meaning in the design without the proper use with the testbench. Multiple composition relations between the testbench block and a CCSL constraint block refers to multiple use of same constraint to represent various SystemVerilog assertions.

TABLE II. VARIOUS CONCEPTS REPRESENTED IN UML/SYSML

System Verilog Concept	UML/SysML Components Used	SysML Diagram Used
Design under Verification	Block (trafficlight)	BDD
Testbench	Block (tlight_props)	BDD
-	Constraint Block (CCSL)	BDD
SV Property Instance	Combination of Roles of Constraint Blocks	Parametric
SV Assertion	Combination of CCSL constraints	Parametric

Parametric diagrams like in Figure 5 further elaborate the CCSL and testbench relations. The constraint properties shown here are the usages of their respective CCSL constraint blocks. Hence parametric diagrams are used to bind the design properties (ports, attributes) and testbench attributes (like *delta1*) to the constraint properties through their parameters (*sub*, *clk1*, etc.) to create a distinct relationship representing the SystemVerilog assertions. The CCSL mappings defined previously are used to model these constraint relations graphically. Based on the behavior of the target SystemVerilog assertion operator, it is represented by one or more constraint properties connected to design parameters. Table II summarizes the relationship between different types of UML/SysML components used to represent SystemVerilog and CCSL concepts.

V. CONCLUSION AND FUTURE WORK

This paper explores a novel approach for modeling SystemVerilog assertions. A technically sound solution was proposed based on mature standards like SysML, MARTE and CCSL. The proposed solution can be used to model third-party designs and SVAs independently and can help the verification engineers to start working on the design validation efforts at the time of design specification. The presented work focuses on the verification/validation features of a system and future work will present a holistic approach including structural information augmented with design requirements which will then be satisfied through the design verification model elements.

Consequently, this will achieve a model-driven architecture approach by automatically generating SystemVerilog structural skeleton and assertions from the abstract model. Lastly future efforts will focus to exhaustively map various SystemVerilog assertion concepts to those of CCSL.

ACKNOWLEDGEMENT

This project is partially funded by NSTIP (National Science Technology and Innovative Plan), Saudi Arabia under the Track "Software Engineering and Innovated Systems" bearing the project code "13-INF761-10".

REFERENCES

[1] Muhammad Rashid, Damien Picard and Bernard Pottier: Application Analysis for Parallel Processing, In *11th Euro Micro Conference on Digital System Design, Architectures, Methods and Tools (DSD'08)*, Parma, Italy, 2008, pp. 633 - 640.

[2] Muhammad Rashid, Fabrizio Ferrandi and Koen Bertels, "HARTES Design Flow for Heterogeneous Platforms", In *10th International Symposium on Quality of Electronic Design (ISQED'09)*, CA, USA, 2009, pp.330—338.

[3] Muhammad Rashid, Bernard Pottier, "Visitor-based application analysis methodology for early design space exploration", *Design Automation for Embedded Systems*, vol. 16, no. 4, pp. 319-338, , Nov. 2012.

[4] Muhammad Rashid, Muhammad Waseem Anwar, Aamir M. Khan, "Towards the Tools Selection in Model Based System Engineering for Embedded Systems - A Systematic Literature Review, *Journal of Systems and Software*", vol.106, pp 150-163, Aug. 2015.

[5] Unified Modeling Language (UML). Object Management Group (OMG). [Online] http://www.omg.org.

[6] UML Marte. [Online] OMG. http://www.omgmarte.org/.

[7] Systems Modeling Language. Object Management Group (OMG). [Online] http://www.omgsysml.org.

[8] E. Andrade, P. Maciel, G. Callou and B. Nogueira. "A Methodology for Mapping SysML Activity Diagram to Time Petri Net for Requirement Validation of Embedded Real-Time Systems with Energy Constraints", In *3rd International Conference on Digital Society*, Cancun, 2009, pp. 266-271.

[9] Imran Rafiq Quadri et al. "Expressing embedded systems configurations at high abstraction levels with UML MARTE profile: Advantages, limitations and alternatives", *Journal of Systems Architecture*, vol. 58, no. 5, pp 178-194, Apr. 2012.

[10] Gianmaria DeTommasi, et al. "Modeling of MARTe-Based Real-Time Applications With SysML", *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp 2407-2415, Nov. 2013.

[11] Muhammad Rashid, Muhammad Waseem Anwar, Aamir M. Khan, "Identification of Trends for Model Based Development of Embedded Systems", In 12*th IEEE International Symposium on Programming and Systems (ISPS)*, Algiers, April 2015, pp. 1 – 8.

[12] Imran Rafiq Quadri, et al. "MADES FP7 EU Project: Effective High Level SysML/MARTE Methodology for Real-Time and Embedded Avionics Systems", *7th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2012, pp. 1-8,

[13] Milena Rota, et al. "Integrating UML, MARTE and SysML to improve requirements specification and traceability in the embedded domain", *12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 176-181.

[14] EETimes. [Online] http://eetimes.com/electronics-news/4153299/Speakers-call-for-innovation-in-verification.

[15] Muhammad Rashid, Muhammad Waseem Anwar, Farooque Azam and Muhammad Kashif, "Exploring the Platform for Expressing SystemVerilog Assertions in Model Based System Engineering", *7th International Conference on Information Science and Applications (ICISA)*, Vietnam. February 2016.

[16] SystemVerilog. [Online] Accellera. http://www.systemverilog.org/.

[17] Property Specification Language (PSL), Standard for. 1850-2010 - IEEE. IEEE Standards Association. [Online] http://standards.ieee.org/findstds/standard/1850-2010.html.

[18] PSL/Sugar. Formal Verification. [Online] IBM. https://www.research.ibm.com/haifa/projects/verification/sugar/.

[19] C. Andrć "Syntax and semantics of the clock constraint specification language (CCSL)". Research Report RR-6925, INRIA, 2009.

[20] C. Andrć F. Mallet and R. De Simone. "Modeling Time(s)". ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS), 2007, pp. 559-573.

[21] Muhammad Rashid, Muhammad Waseem Anwar and Farooque Azam, "Expressing Embedded Systems Verification Aspects at Higher Abstraction Level - SystemVerilog in Object Constraint Language (SVOCL), IEEE International Systems Conference (SysCon), Florida, USA, April 2016.

[22] M. Mura, L. G. Murillo, and M. Prevostini. "Model-based Design Space Exploration for RTES with SysML and MARTE" in Forum on Specification and Design Languages (FDL), 2008. pp. 203-208.

[23] M. Mura, A. Panda and M. Prevostini. "Executable Models and Verification from MARTE and SysML: A Comparative Study of Code Generation Capabilities" in Proceedings of MARTE Workshop (DATE'08), 2008.

[24] A. A. Kerzhner and J. J. Paredis, "Combining SysML and Model Transformations to Support Systems Engineering Analysis" in 4th International Workshop on Multi-Paradigm. 2011. Vol. 42. 1863-2122.

[25] Jorgiano Vidal et al., "A co-design approach for embedded system modeling and code generation with UML and MARTE, *Design Automation and Test in Europe (DATE)*, 2008, pp. 226-231.

[26] SATURN Project. [Online] http://www.saturnsysml.eu/.

[27] M-A Peraldi-Frati, et al., "The MeMVaTEx Methodology: From Requirements to Models in Automotive Application Design". *4th International Congress on Embedded Real Time Software ERTS2008*, Toulouse, Jan 2008.

[28] Artisan Studio Software. [Online] Atego. http://www.atego.com/products/artisan-studio/.

[29] East-ADL Language, Profile and Tools. East-ADL. [Online] ATESST. http://www.atesst.org/scripts/home/publigen/content/templates/show.asp?P=125&L=EN.

[30] F. Mallet. "Clock Constraint Specification Language: Specifying Clock Constraints with UML/MARTE", *ISSE, Springer*, vol 4, no 3, pp. 309-314, 2008.

[31] H. Espinoza, D. Cancila, B. Selic and S. Gérard. "Challenges in Combining SysML and MARTE for Model-Based Design of Embedded Systems", in *Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'09)*, Springer, 2009, pp 98-113.

[32] D. Knorreck, L. Apvrille, P. de Saqui-Sannes. "TEPE: A SysML Language for Time-Constrained Property Modeling and Formal Verification". *ACM SIGSOFT Software Engineering Notes*. vol. 1, no. 1, pp. 1-8, Jan. 2011.

[33] R. Gascon, F. Mallet, and J. DeAntoni. "Logical time and temporal logics: Comparing UML Marte/CCSL and PSL", *18th Int. Symposium on Temporal Representation and Reasoning (TIME)*, pp. 141,148 Sept. 2011

[34] F. Mallet. "Automatic generation of observers from MARTE/CCSL", in *Proceedings of 23rd IEEE International Symposium on Rapid System Prototyping (RSP)*, 2012, pp 86-92.

[35] L. Li, F. P. Coyle, and M. A. Thornton. "UML to SystemVerilog Synthesis for Embedded System Models with Support for Assertion Generation", in *Forum on Specification and Design Languages (FDL)*, Sept. 18-20. 2007.

[36] B. Cohen, S. Venkataramanan, A. Kumari, L. Piper. "SystemVerilog Assertions Handbook", 2nd ed., VhdlCohen Publishing, 2010.

[37] F. Mallet, C. Andre, R. de Simone, "Logical time: Specification vs. Implementation". *SIGSOFT Softw. Eng. Notes*, vol. 36, no.1, Jan. 2011.