THÈSE POUR OBTENIR LE GRADE DE DOCTEUR
DE L'UNIVERSITÉ DE MONTPELLIER

En Informatique

École doctorale : Information, Structures, Systèmes

Unité de recherche LIRMM

Vérification d'une méthodologie pour la conception de systèmes numériques critiques

Présenté par Vincent IAMPIETRO

Le Date de la soutenance

Sous la direction de David Delahaye
et David Andreu

Devant le jury composé de

[Nom Prénom], [Titre], [Labo]	[Statut jury]
[Nom Prénom], [Titre], [Labo]	[Statut jury]
[Nom Prénom], [Titre], [Labo]	[Statut jury]

UNIVERSITÉ
DE MONTPELLIER

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

List of Abbreviations

SITPN	**S**ynchronously executed **I**nterpreted **T**ime **P**etri **N**et with priorities
VHDL	**V**ery high speed integrated circuit **H**ardware **D**escription **L**anguage
PCI	**P**lace **C**omponent **I**nstance
TCI	**T**ransition **C**omponent **I**nstance
GPL	**G**eneric **P**rogramming **L**anguage
HDL	**H**ardware **D**escription **L**anguage

For/Dedicated to/To my…

Chapter 1

The HILECOP methodology

In this chapter, we present the context of our work, and more specifically, the subject of our verification task, i.e. HILECOP, a methodology for the design and production of critical digital systems. In Section 1.1, we motivate the use of Model-Based Systems Engineering (MBSE) and formal methods in the design and production of safety-critical digital systems; in Section 1.2, we give an overall presentation of the HILECOP methodology which applies both the principles of MBSE and formal methods; in Section 1.3, we point out the specific transformation phase, intervening in the HILECOP methodology, that we propose to verify and discuss on how we propose to verify it and which research questions does it give rise to.

1.1 Designing critical digital systems

According to Moore's law [5], the complexity of digital integrated circuits is always increasing. To give an example, the cut-of-the-edge *AMD Epyc Rome* microprocessor (2019) is made out of 50 billions of transistors. Composing billions of transistors on a wired circuit is no more a task for humans but is very suited to computers. However, engineers need to think about the design of digital circuits in a way that is understandable for humans. Therefore, they need high-level views of the circuits they are designing in order to work together and to communicate about the designs. The domain of Model-Based Systems Engineering (MBSE) [4] proposes a framework to help engineers to design and produce digital circuits, in a well-documented, safe and reliable way. Comparable to what Model Driven Engineering (MDE) does in the world of software engineering, models are first order concepts in MBSE. A model represents a simplified view of real object. As illustrated in Figure 1.1, a MBSE process describes a way to design a digital circuit starting from a high-level view of the system. This high-level view can follow a graphical formalism such as SysML [3] or Petri nets [6], or a textual one such as SystemC [2] or VHDL [1]. Then, the MBSE process describe many refinements phases (the green arrows in Figure 1.1) during which the input model will be transformed; at each refinement phase, the model goes down in abstraction towards its final implementation as a hardware circuit. A refinement phase, which is also a transformation phase, can be performed automatically or manually.

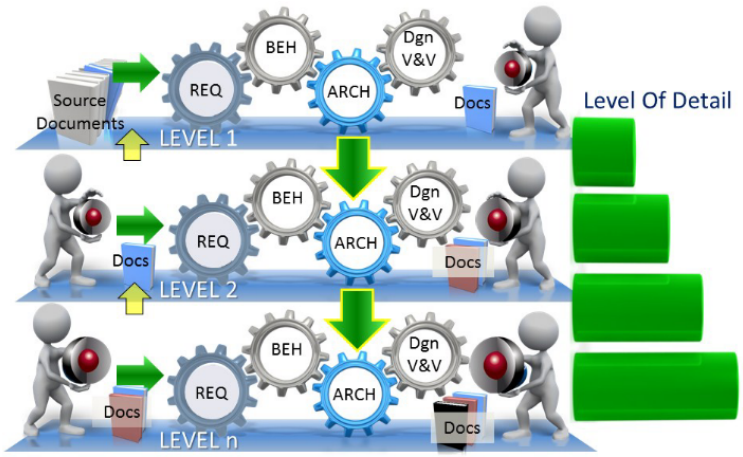

FIGURE 1.1: A Model-Based Systems Engineering process; REQ stands for requirements, BEH for behavior, ARCH for architecture, Dgn V&V for design verification and validation. This figure is an excerpt from [4].

In the case where the digital circuit being designed is a safety-critical system, i.e. on its behavior depends the life of people, an MBSE process will often employ formal models, i.e. models with a formal mathematical definition, as the design formalism. Thus, these models enable a certain extent of mathematical reasoning to prove that safety properties are met during the design V&V phase (cf. Figure 1.1).

1.2 Introducing the HILECOP methodology

The HILECOP methodology consists of a process for the design and implementation of critical digital systems. This methodology relies on several transformations going from abstract models to concrete FPGA implementations through the production of VHDL code. Fig. 1.2 details the global workflow of HILECOP. In this figure, Step 1 corresponds to the model of a digital system. This model is built with component diagrams and the behavior of each component is described by means of PNs. Fig. 1.3 provides an example of such model. As shown in this figure, an internal behavior and an interface outline the structure of components. The component interface exposes places, transitions, and signals, which are references to nodes of the internal behavior, from which the components can be assembled to get the global behavior of the digital system.

Next, in Fig. 1.2, the transformation from Step 1 to Step 2 flattens the model. The internal behaviors are connected according to the interface compositions, and embedding component structures are removed. The analysis phase, going from Step 2 to Step 1, aims to produce a model that is conflict-free (see Sec. **??** for more details about the definition of a conflict), bounded, and deadlock-free, using model-checking techniques. After several iterations, the model should reach soundness and is then said to be implementation-ready.

From Step 2 to Step 3, VHDL source code is then generated by means of a model-to-text transformation. This generated code describes the hardware system that will be implemented in a FPGA circuit. From Step 3 to Step 4, the VHDL compilation/synthesis and the FPGA programming are finally performed using industrial tools.

In this work, we focus on the part of the workflow in Fig 1.2 that is framed with dotted lines, i.e. the model-to-text transformation between Step 2 and Step 3. In particular, we aim to prove

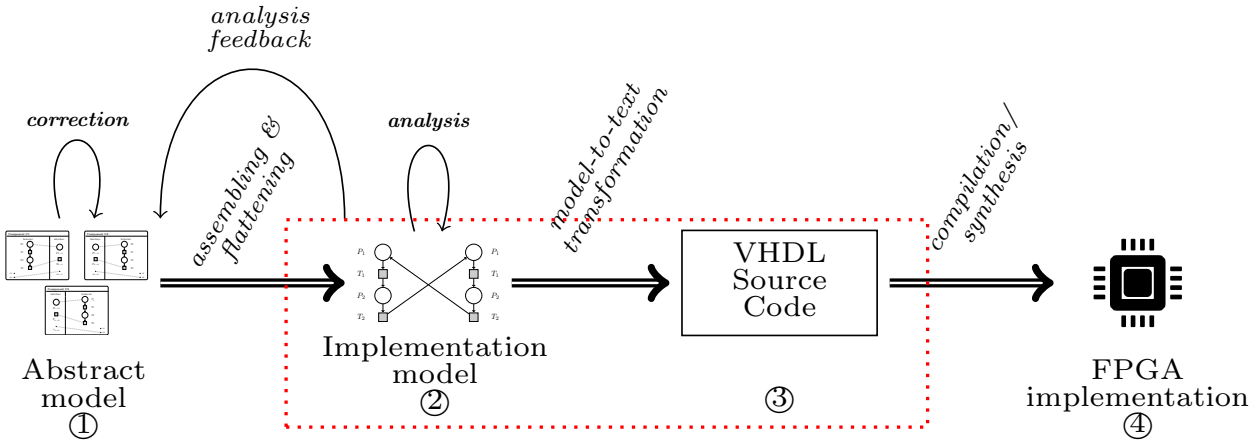

FIGURE 1.2: Workflow of the HILECOP Methodology

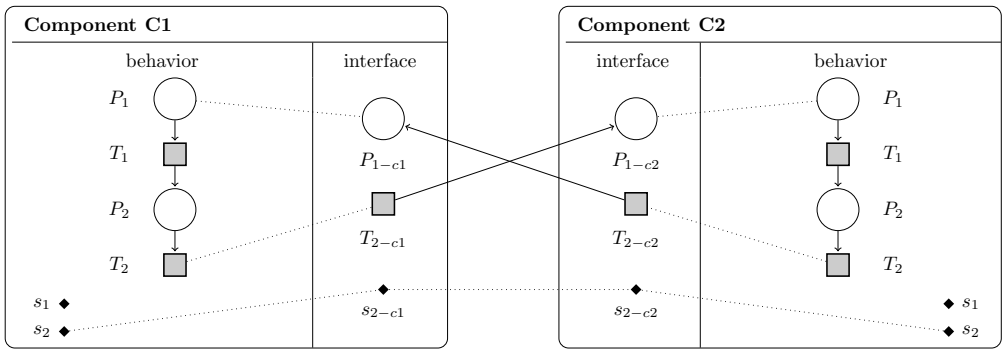

FIGURE 1.3: An Example of HILECOP Model

that through this model-to-text transformation, the behavior described by the initial model is preserved in the generated VHDL code. To do so, we have to implement the structure of PNs used in the HILECOP models, together with the semantics providing the evolution rules of these PNs.

1.3 Verifying the HILECOP methodology

Bibliography

[1] Peter J. Ashenden. *The Designer's Guide to VHDL*. Morgan Kaufmann, Oct. 7, 2010. 933 pp. ISBN: 978-0-08-056885-0. Google Books: XbZr8DurZYEC.

[2] David C. Black et al. *SystemC: From the Ground Up, Second Edition*. Springer Science & Business Media, Dec. 18, 2009. 291 pp. ISBN: 978-0-387-69958-5. Google Books: Op2a6yiO9jwC.

[3] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann, Oct. 23, 2014. 631 pp. ISBN: 978-0-12-800800-3. Google Books: Ze60AwAAQBAJ.

[4] David Long and Zane Scott. *A Primer for Model-Based Systems Engineering*. Lulu.com, 2011. 126 pp. ISBN: 978-1-105-58810-5. Google Books: pCaoAwAAQBAJ.

[5] Gordon E. Moore. "Cramming More Components onto Integrated Circuits, Reprinted from Electronics, Volume 38, Number 8, April 19, 1965, Pp.114 Ff." In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (Sept. 2006), pp. 33–35. ISSN: 1098-4232. DOI: 10.1109/N-SSC.2006.4785860.

[6] Carl Adam Petri. "Kommunikation mit Automaten". In: *http://edoc.sub.uni-hamburg.de/informatik/volltexte/petri.pdf* (1962). URL: https://edoc.sub.uni-hamburg.de//informatik/volltexte/2011/160/ (visited on 06/10/2021).