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 implementation 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 [8], 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) [6] 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 [4] or Petri nets [9], or a textual one such as SystemC [3] or VHDL [2]. 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.

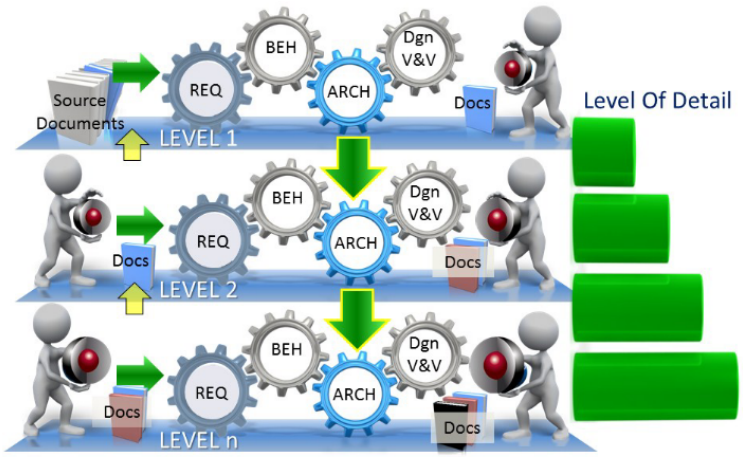


FIGURE 1.1: A Model-Based Systems Engineering process; REQ stands for requirements, BEH for behavior, ARCH for architecture, Dgn V&V for design verification and validation. This figure is an excerpt from [6].

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 INRIA CAMIN team (former DEMAR team) has developed a new technology of neuroprotheses [5]. Neuroprotheses are medical devices which purpose is to electro-stimulate the nerves of patients suffering from moving disabilities. The nerves are responding to the stimulation, i.e an electric influx, in order to activate the muscles and so that the patient can recover some movements. Thus, controlling the intensity and the form of the electric signal sent to the patient's nerve is critical point of the device overall functioning. These two parameters are controlled by a digital hardware circuit (i.e. a microcontroller) that is a part of the neuroprosthesis. Therefore, the design of such digital systems becomes utterly critical as a faulty circuit could result in the injury of patients. To assist the engineers in the design and the implementation of these critical digital systems, the CAMIN team came up with a process called the "HILECOP methodology" [1]. This methodology follows the principles of a MBSE process and relies on several transformations going from abstract models to concrete FPGA implementations through the production of VHDL code. Figure 1.2 details the global workflow of HILECOP.

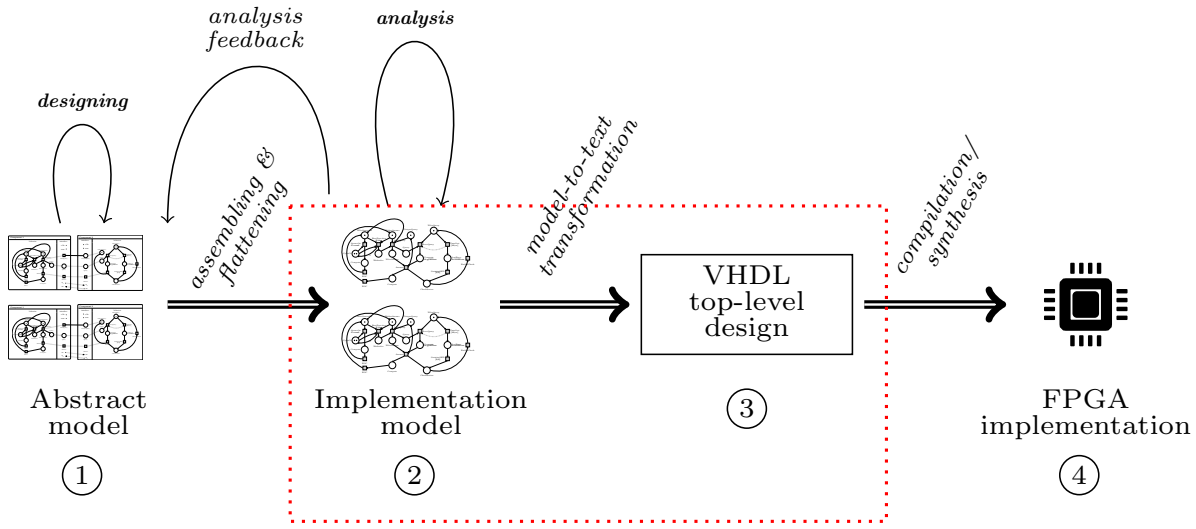


FIGURE 1.2: Workflow of the HILECOP methodology; horizontal double arrows indicate the transformation phases, i.e. the refinement phases in MBSE terms; simple arrows indicate different kinds of operations performed at a given step.

In Figure 1.2, Step 1 corresponds to the design phase of a critical digital system. At this step, the engineers produce a model of the wanted system; the leveraged model formalism is a graphical formalism specially designed for the methodology and based on component diagrams. Figure 1.3 provides an example of such a model.

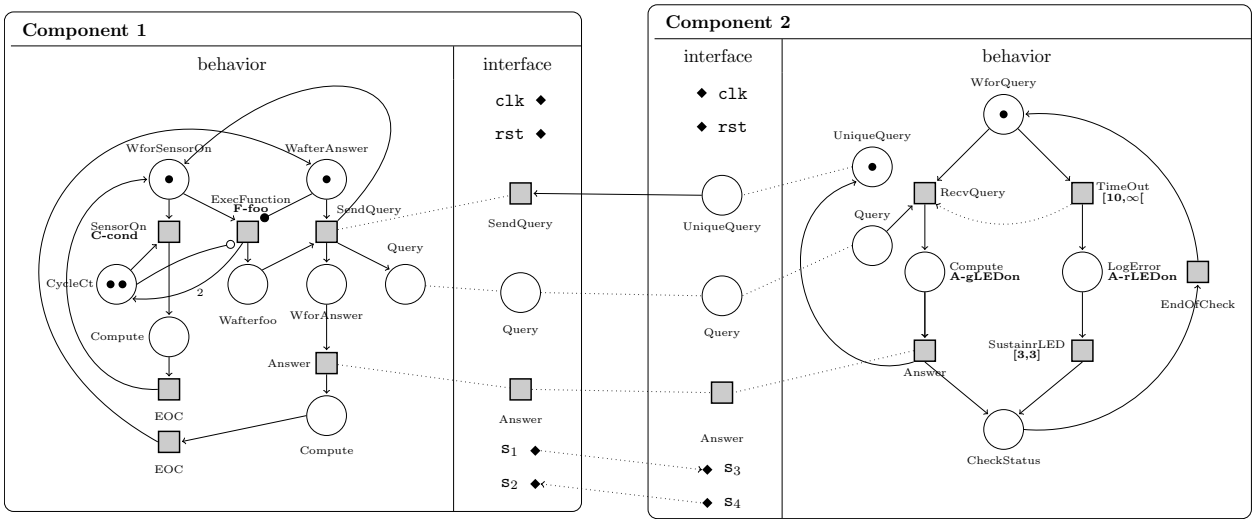


FIGURE 1.3: An Example of HILECOP high-level model. Black diamonds represent VHDL signals.

As shown in Figure 1.3, a component of the HILECOP high-level model formalism is represented by a box having an internal behavior and an interface that permits the connection to other components. The internal behavior of a component is defined with a specific kind of Petri Net (PN) model. These PNs and their distiguishing features will be thoroughly presented in Chapter **??**. The component interface exposes places, transitions, and signals, which are references to

the elements of its internal behavior, to the outside so that multiple components can be assembled. Each component has a clock and a reset input port (`clk` and `rst`) in its interface. This is the mark that the HILECOP has been built for the design of synchronous digital systems. To a certain extent, VHDL signals can be integrated to the high-level components to represent a direct wiring between components. A component behavior can also be defined through the composition of other components. In that case, we talk about a composite structure.

Next, in Figure 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. Figure 1.4 gives the result of the flattening phase for the model of Figure 1.3.

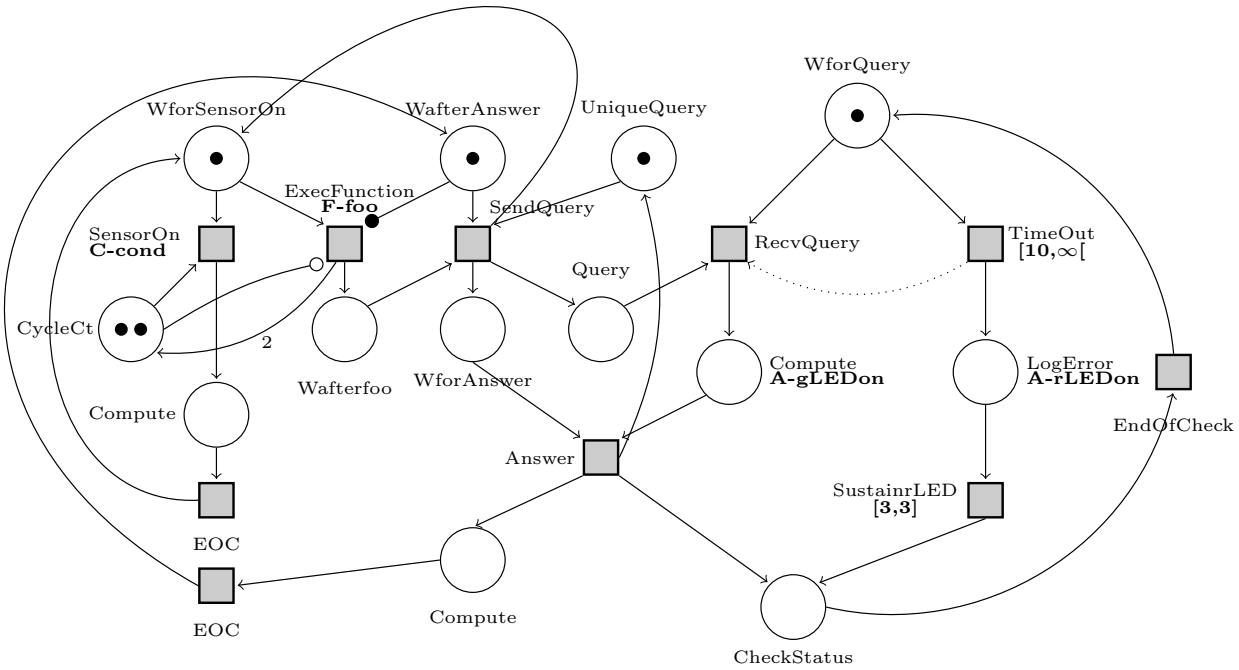


FIGURE 1.4: A global Petri net model obtained after the flattening of a HILECOP high level model.

The PN formalism is a formal model and therefore permits to apply mathematical reasoning on its instances. Particularly, a PN model can be analyzed, and a proof that a given model meet some properties can be automatically produced through the direct analysis of the structure or through the use of model checking techniques. This feature of PNs has been one of the reason of the adoption of this formalism as the basis of the design of critical digital systems. A whole thesis has been dedicated to developed new methods to analyze the HILECOP PN models [7]. In fact, the transformation of the abstract model is a bit different in preparation of the model analysis. The transformation adds new information to the flattened model for the purpose of the analysis. Figure 1.4 only gives the flattened version of the model that is not produced for the analysis but in preparation of the next transformation into VHDL design. The analysis phase is here to convince the engineers that they are designing a safe system. The analysis process is a round trip between Step 1 and Step 2. It aims at producing a model that is conflict-free (see Section **??** 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. The generated code describes a VHDL design, i.e. a textual description of a hardware system, which has an interface defining input and output ports and an internal behavior called an architecture. Details about the syntax and the semantics of the VHDL language will be given in Chapter **??**. Figure succinctly illustrates the transformation happening between Step 2 and Step 3.

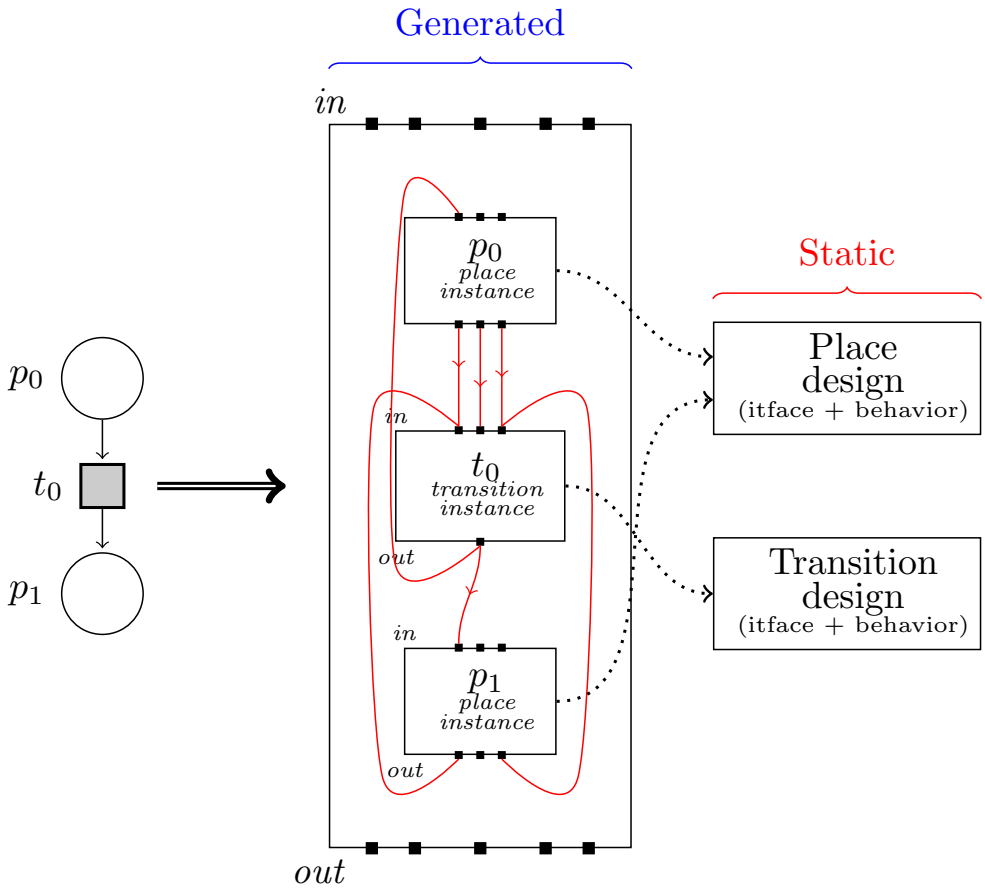


FIGURE 1.5: Generation of a top-level VHDL design from a Petri net.

For the purpose of the HILECOP methodology, two VHDL designs have been defined: the place design that is a hardware description of a PN place (circle nodes in a PN) and the transition design that is a hardware description of a PN transition (square nodes in a PN). Like all VHDL designs, the place and the transition design have an input and output port interface, and their own internal behavior. A VHDL design is a mould to describe a hardware component. Thus, a design can be instantiated in the behavior of other designs in order to obtain more complex behaviors. As illustrated in Figure, the transformation from Step 2 to Step 3 creates a place design (or component) instance (PCI) and a transition design (or component) instance (TCI) for each place and transition of the input Petri net. Then, the PCIs and TCIs are connected together through their input and output port interfaces. These connections mimic the arc connections between the places and the transitions of the input PN.

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 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

The development of these medical devices is at the base of the creation of the Neurrinov company. The Neurrinov company is now looking for the industrial development of such neuroprotheses.

# Bibliography


[1] David Andreu, David Guiraud, and Guillaume Souquet. "A Distributed Architecture for Activating the Peripheral Nervous System". In: *Journal of Neural Engineering* 6.2 (Apr. 1, 2009), p. 026001. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2560/6/2/026001. URL: https://iopscience.iop.org/article/10.1088/1741-2560/6/2/026001 (visited on 06/08/2020).

[2] 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.

[3] 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.

[4] 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.

[5] David Guiraud et al. "An Implantable Neuroprosthesis for Standing and Walking in Paraplegia: 5-Year Patient Follow-Up". In: *Journal of Neural Engineering* 3.4 (Sept. 2006), pp. 268–275. ISSN: 1741-2552. DOI: 10.1088/1741-2560/3/4/003. URL: https://doi.org/10.1088/1741-2560/3/4/003 (visited on 06/11/2021).

[6] 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.

[7] Ibrahim Merzoug. "Validation formelle des systèmes numériques critiques : génération de l'espace d'états de réseaux de Petri exécutés en synchrone". PhD thesis. Université Montpellier, Jan. 15, 2018. URL: https://tel.archives-ouvertes.fr/tel-01704776 (visited on 02/10/2020).

[8] 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.

[9] 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).