# SYSTEM-LEVEL ENERGY-AWARE DESIGN OF CYBER-PHYSICAL SYSTEMS

**Electrical and Computer Engineering**
Technical Report ECE-TR-16

AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

# DATA SHEET

**Title**: System-Level Energy-Aware Design of Cyber-Physical Systems
**Subtitle**: Electrical and Computer Engineering
**Series title and no.**: Technical report ECE-TR-16

**Author**: José Antonio Esparza Isasa
Department of Engineering – Electrical and Computer Engineering, Aarhus University

**Internet version**: The report is available in electronic format (pdf) at the Department of Engineering website http://www.eng.au.dk.

**Publisher**: Aarhus University©
**URL**: http://www.eng.au.dk

**Year of publication**: 2013 Pages: 40
**Editing completed**: October 2013

**Abstract**: In this technical report we present the work conducted during the first part of the PhD thesis "System-Level Energy-Aware Design of Cyber-Physical Systems". We present the application of modelling techniques and methodologies to study energy consumption during the design and implementation of cyber-physical systems. This study is made from the electro-mechanical and computation angle. Additionally we present a setup that allows the combination of abstract models with hardware and software preliminary realizations. This allows a stepwise model to implementation transformation and improved model accuracy. Some of these techniques have been applied to the case study e-Stocking and others have been studied with more simple experimental setups.
In addition to the scientific content, we also present a description of the envisioned future work and the plans that will lead to completion of this PhD thesis by April 2015.

**Keywords**: energy-aware design, embedded systems, modelling.

**Supervisor**: Peter Gorm Larsen and Finn Overgaard Hansen
**Financial support**: This work is financed by the Ambient Assisted Living project e-Stockings.

**Please cite as**: Esparza Isasa, José Antonio, 2013. System-Level Energy-Aware Design of Cyber-Physical Systems. Department of Engineering, Aarhus University. Denmark. 40 pp. - Technical report ECE-TR-16

**Cover photo**: Polymer Lithium Ion Battery. Author: Sparkfun sparkfun.com Image license: CC BY-NC-SA 3.0

**ISSN**: 2245-2087

# SYSTEM-LEVEL ENERGY-AWARE DESIGN OF CYBER-PHYSICAL SYSTEMS

José Antonio Esparza Isasa,
Department of Engineering, Aarhus University

**Abstract**

In this technical report we present the work conducted during the first part of the PhD thesis "System-Level Energy-Aware Design of Cyber-Physical Systems". We present the application of modelling techniques and methodologies to study energy consumption during the design and implementation of cyber-physical systems. This study is made from the electro-mechanical and computation angle. Additionally we present a setup that allows the combination of abstract models with hardware and software preliminary realizations. This allows a stepwise model to implementation transformation and improved model accuracy. Some of these techniques have been applied to the case study e-Stocking and others have been studied with more simple experimental setups.

In addition to the scientific content, we also present a description of the envisioned future work and the plans that will lead to completion of this PhD thesis by April 2015.

# Table of Contents

# List of Figures

# 1

# Introduction

*This chapter introduces the PhD project "System-Level Energy-Aware Design of Cyber-Physical Systems". The chapter provides an introduction to the research field and frames the purpose of this project. Finally it provides a description of the remainder of this document.*

This document is presented as a "mid-term" report for the PhD qualification exam at the Department of Engineering at Aarhus University. This report presents a review of the work conducted during the first and a half year of the PhD studies, that have a nominal duration of three years. It also outlines the future work that will be conducted during the reminder of the PhD.

## 1.1 Short introduction to the research field

Today's embedded solutions are a combination of computing, communication, electronics and mechanical subsystems. These solutions operate autonomously or as part of a network together with external systems in order to provide a certain service and they are also known as Cyber-Physical Systems (CPS) [1]. The heterogeneity in these systems makes them especially complex to design [2]. Many CPS are battery powered and therefore have a limited amount of energy available to ensure their operation. Banerjee et al. presents the sustainability from the energy perspective as one of the key issues[1] to address when designing CPS [3]. Due to the heterogeneous composition of these systems a sustainable solution requires an holistic approach that is able to take into consideration computing and non-computing aspects [4] in what is called system-level design.

A design process that takes power and or energy related aspects into consideration is called power or energy-aware design. Power-aware design was originally applied at the device level[2] and it saw its expansion to other fields in the mid 90s, when it jumped to higher levels of abstraction. Going up in abstraction layers implied jumping from the electronic engineering to the computer engineering field [5, 6]. This approach was considered "system-level power-aware design".

Due to the heterogeneity of today's solutions and to the broader implications of the term "system", power-aware design has been extended now to the non-computing aspects. Multiple authors agree that this is the approach required in order to create sustainable systems [4, 3]. This implies that all the subsystems and components that comprises the solution under design, including

[1] Besides sustainability they also consider safety and security. This is also referred as the S3.

[2] In the electrical engineering field the device level is considered the lowest level of abstraction and takes the transistor as building blocks.

mechanical, electrical and software elements must be addressed in a single design effort. As a result of this, engineers need to deal with multiple factors spread across different disciplines. A possible way to cope with this high complexity is to apply abstract modelling [2, 7].

System-level power-awareness for CPS can benefit from the extensive work carried out in several fields separately until now in power and energy consumption analysis and management. Energy consumption in computing has been extensively researched and characterized at different levels of abstraction, ranging from the microarchitectural level and the instruction level [8, 9] all the way up to software design for low power [10, 11]. Additionally work have been done to characterize the energy cost of communicating computing units with other embedded system peripherals [12].

Energy consumption in communication has been especially researched since the introduction of Wireless Sensor Networks (WSN). Extensive work has been carried out in order to analyze hardware and protocol optimizations at different layers including improvements in the RF technology, network protocols [13, 14, 15] and specific battery discharge characteristics [16]. Previous work on the network side is not limited to wireless networks, Chabarek et al. propose the introduction of power awareness in wired network design and routing in [17].

While the energy consumption of mechanical and electrical devices has been widely studied separately, a combined approach to study their energy consumption is still not widely researched. However, there is an increasing interest and tool support for mechatronic system design through tools like DESTECS or MODELLICA [18, 19]. Some authors highlight the importance of characterizing the energy supply in mechatronic systems. Following this approach Zhang et al [20] and Rao et al. [21] propose specific battery performance models.

As explained above, there are existing research results in the communications, computation and electromecanical field separatedly and, in some cases, as a combination of two of those. However, it is our view that a unifying approach that considers the system as a whole is still missing.

## 1.2 Purpose of this PhD project

It is the purpose of this PhD work to address some of the challenges present in today's approaches to the design of energy constrained CPS.

> It is our hypothesis that: **A model-driven engineering approach, that applies heterogeneous modelling techniques and combined with partial prototyping, will enable the holistic approach that is needed to take energy consumption into account during the system level design of cyber-physical systems.**

During this PhD project we will try to validate this hypothesis by:

- Determining a coarse grained representation of systems energy consumption in computation in order to support system-level design decisions.
- Determining a coarse grained representation of systems and small-scale networks energy consumption in communication in order to support system-level design decisions.
- Proposing a methodology to apply existing co-simulation tools to represent energy consumption in electromechanical devices, in order to evaluate the impact of different control and supervision algorithms to the overall system energy consumption.

- Creating the necessary hardware and software infrastructure to apply Hardware In the Loop (HIL) in some of the selected modelling technologies and to enable combined co-execution of models and partial implementations.

- Applying accurate energy and power measurement techniques to improve the fidelity of the models to the level required for the analysis conducted.

It is the intention of this work to provide the systems engineer developing energy-constrained CPS with an approach that supports him in the process of exploring the design space when looking for solutions that comply with the energy limitations of the system under design. Furthermore this approach should also assist him in gradually transforming the models developed to conduct this analysis into final implementations.

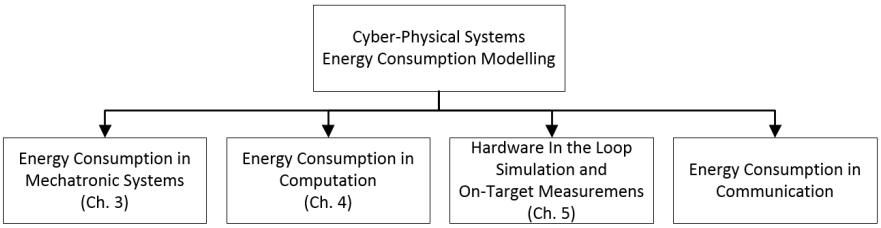


Figure 1.1: Overview of the approach followed in this PhD project

An overview of the modelling approach proposed in this PhD is presented in figure 1.1. This has guided the work conducted until now, that could be summarized in the following three main contributions so far:

**Contribution 1:** Representation of the energy consumption caused by electromechanical components in CPS. This analysis is conducted by models composed of discrete event and continuous time models. This is represented in figure 1.2.

**Contribution 2:** Enabling the energy consumption analysis across multiple CPU states with the VDM-Real Time modelling language. This analysis is conducted by applying a model structure and examining the logs produced after model execution. This is represented in figure 1.3.

**Contribution 3:** Enabling Hardware In the Loop (HIL) combined with VDM-Real Time model executions. This is represented in figure 1.4.

The application of modelling to study the energy consumption in communication has not been addressed yet and it will be a topic for discussion later on in this document.

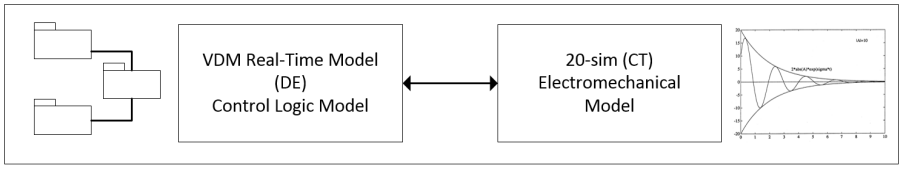


Figure 1.2: Mechatronic energy consumption analysis.

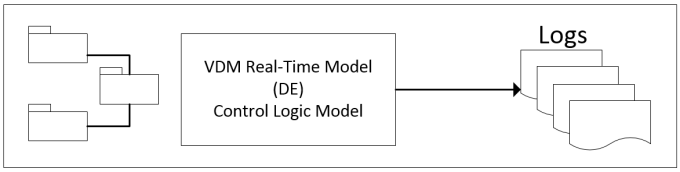


Figure 1.3: CPU energy consumption modelling in VDM-RT.

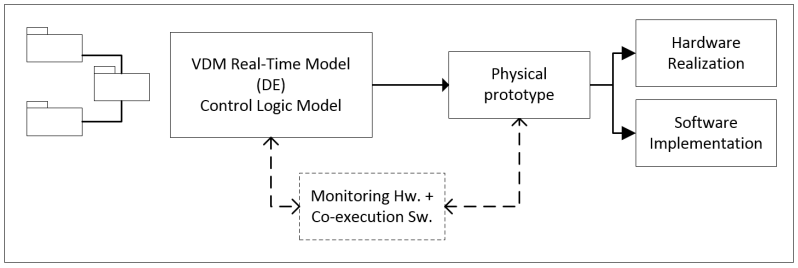


Figure 1.4: HIL for VDM-RT simulations.

## 1.3 Document structure

This document is structured in 6 chapters. This chapter and the background chapter provide the necessary information to frame the work conducted in this project. Chapters 3 to 5 present the main contributions of this PhD project to the research field. Chapter 6 concludes this report and presents further work to be conducted.

**Chapter 1 Introduction:** presents the PhD project, its relation to existing research and its goals.

**Chapter 2 Background:** provides the information necessary to understand the work conducted in this project.

**Chapter 3 Modelling energy consumption in mechatronic systems:** presents the work conducted on energy consumption analysis in electromechanical systems through co-simulation. It also presents its application in the case study eStockings and therefore it provides a greater level of detail.

**Chapter 4 Modelling energy consumption of different CPU states:** presents the work conducted to represent and analyze transitions to low-power consuming states using the abstractions provided in the VDM-RT modelling language.

**Chapter 5 Model Validation and Verification through HIL in VDM-RT:** presents the work conducted to enable the co-execution of a model together with partial system implementations and/or realizations in software and/or hardware respectively.

**Chapter 6 Current status and future plans:** reflects on the progress of this PhD project and its current status. Additionally it presents the future plans for the reminder of this PhD.

Finally, the core of the document is complemented by three appendices:

**Appendix A Publications:** presents the publications produced so far during this PhD project.

**Appendix B Courses:** presents the courses followed as part of the PhD programme.

**Appendix C Gantt diagram:** gives an overview of the current plan to complete this PhD project.

# 2

# Background

*This chapter presents the background information necessary to understand the problem that this PhD project is addressing and its relevance. Additionally this chapter introduces low-power*[1] *embedded platforms and the main tools used in this work. Finally, this chapter presents the case study in which the modelling techniques developed in this PhD are being applied.*

## 2.1 Application areas

The system level energy-aware design approach presented in this work is targeting Cyber-Physical Systems (CPS) that a) operate under a tight energy budget (battery powered) or b) that aim at being energy-efficient even though the energy supply is ensured. We have identified three main application areas in which this work is relevant, involving the development of:

**Cyber-physical systems:** These are composed of electromechanical devices to interface the physical world (sensors and actuators) and one or more embedded control units that uses them. The way these interfaces are controlled have a significant impact on the total system energy consumption and that is why they are typically the first elements to optimize. CPS can operate as standalone devices or as part of a network. An example of this kind of systems could be an Unmanned Aerial Vehicle or the system under consideration in the case study presented later in this chapter.

**Personal or industrial embedded devices:** These are resource constrained computational units. The most energy demanding feature in these systems are typically on the computational side. An example of this kind of systems could be a portable multimedia player in the first case or a production process logging device.

**Small-scale networked embedded systems:** These are composed of multiple embedded systems equipped with a network interface. These networks are typically responsible for monitoring and/or control and therefore they can be composed by both sensors and actuators. A concrete example of networked embedded systems is a wireless sensor network. These are composed of multiple computing nodes equipped with a radio transceiver. They typically

[1] This is a common term used by electronic manufacturers to refer to both "low-power" or "low-energy consuming" technologies.

sense physical parameters and report them via radio. The most energy demanding component in this kind of systems is typically the radio transceiver and therefore even minor improvements in communication protocols or hardware technologies can have a relevant impact in the device autonomy in the long run.

The techniques presented in here aim at exploring these different domains by using a system level design approach.

## 2.2 Power and energy consumption definition

In order to calculate the power consumption of the systems under study in this work we have used the general formulation of power consumption in electrical DC systems:

$$P \text{ [Watts]} = V \times I \tag{2.1}$$

Based on the power consumption the energy consumption is calculated as its integral over a certain period of time:

$$E \text{ [Joules]} = \int_{t_0}^{t_1} P \mathrm{d}t \tag{2.2}$$

The total power consumption in Complementary Metal-Oxide-Semiconductor (CMOS) devices is defined by the static power consumption and the dynamic power consumption. The static power consumption is caused by the internal device current leakage while the dynamic power consumption is caused by the bit switching within the digital circuitry.

$$P_{total} \text{ [Watts]} = P_{static} + P_{dynamic} \tag{2.3}$$

The dynamic power consumption is defined as presented in equation 2.4 [22]. The dynamic power consumption is directly proportional to the capacitance load presented by the CMOS circuitry, the square of the voltage and the frequency. Lower frequencies will imply directly lower power consumption. In addition to this, today's processors operate at a lower voltage when operating at lower frequencies, making it possible to achieve even lower power consumption.

$$P_{dynamic} \text{ [Watts]} = C \times V^2 \times f \tag{2.4}$$

The equations related to the power consumption in CMOS devices (eq. 2.3 and 2.4) have been used in this work as theoretical basis to discuss power consumption reduction in computation. However they have not been used to calculate concrete power consumption figures of different CMOS circuits.

## 2.3 Platforms and implementation techniques

### 2.3.1 System-On-a-Chip

In this work we have used a System-On-a-Chip (SoC) hardware platform to conduct experiments involving partial system implementations and measurements. A SoC is a hardware platform that contains multiple hardware blocks within the same substrate. These hardware blocks can be digital, analog or both (hence conforming a mixed signal Integrated Circuit).

The concrete platform we have used in this work is a PSoC5 [23] from the manufacturer Cypress. It integrates a 32-bit ARM Cortex M3 CPU and two reprogrammable analog and digital silicon areas. This platform offers a flexible piece of silicon that can be reconfigured from an Integrated Development Environment (IDE). This IDE contains the necessary software facilities to develop hardware and software for the PSoC5. Most of the PSoC5 hardware blocks give the possibility of selective power-off controlled by embedded software running on the CPU, making it possible to save energy. The flexibility offered by this platform is advantageous when it comes to the fast evaluation of different configurations and prototype development.

### 2.3.2 Power saving techniques

Commercial processors achieve a reduction in energy consumption by applying a combination of hardware and software techniques. The most relevant for this work are:

**Dynamic Frequency Scaling (DFS):** This consist of adjusting the systems operating frequency at run-time in order to achieve a power reduction. A reduction in a CPU's operating frequency will imply slower software execution, which could lead to real-time problems. Applied in the PSoC5 platform, this enables a current draw reduction from 18 mA to 3.8 mA if scaling from 62 to 3 MHz (70 mWatts power consumption reduction if operating at 5 Volts).

**Dynamic Voltage Scaling (DVS):** This consist of adjusting the systems operational voltage at run-time. This technique is widely used in high-end processors integrated in laptops. Voltage reduction results on lower operating temperature and therefore lower power consumption. Due to the CMOS technology, a reduction in voltage difference between high and low states also implies a reduction in the maximum operating frequency. This technique is not supported by the PSoC5.

**CPU operational modes switching:** This consist of changing CPU operational states depending on the computational demands. Typical microcontroller CPUs incorporate two operational modes: active and sleep. In active mode the CPU is able to perform computations and operate all the peripherals. This mode is the most power demanding. In sleep mode the CPU is not able to perform computations and some peripherals have been shut down. An event external to the CPU should wake it up in order to transit back to active. In the PSoC5 platform the ARM CPU draws up to 10.5 mA while active and 4$\mu$A while sleeping. Implementing the transitions between these modes can lead to a reduction from 52.5mW to 20$\mu$W when operating at 5 Volts.

This work will be focused on the benefits of applying DFS and switching between CPU operational modes and their trade-offs with performance. Further details on this are provided in chapter 4. DVS is not covered in this work, however it is worthwhile mentioning due to its relevance among power management techniques.

## 2.4 Tools

### 2.4.1 Modelling languages tools

In this work we are using different modelling tools to analyze the systems energy consumption. The work conducted until now has been conducted with VDM-RT and 20-sim:

**VDM-RT:** is an extension to the software modelling language VDM++ [24] and enables the modelling of real-time embedded control software. This modelling language is ideal to represent Discrete Event (DE) systems. VDM-RT models can be created and executed in the Overture tool [25].

**20-sim**[2]**:** is a physical modelling tool capable of representing electrical and mechanical systems among others. Modelling can be done by using bond graphs, iconic diagrams or differential equations. This modelling language and tool is best suited to represent Continuous Time (CT) systems. Standalone 20-sim models can be created and executed in the 20-sim tool [26, 27].

**Crescendo**[3]**:** is a co-simulation tool that integrates Overture/VDM-RT and 20-sim developed on the DESTECS project [18]. It communicates the VDM-RT interpreter with 20-sim and provides a common notion of time to synchronize the parallel execution of DE and CT models, now considered a single co-model. This also provides methodological guidelines to design mechatronic systems [28].

We have applied previously VDM-RT in the development of embedded hardware and software systems in the MSc. thesis work [29], that was finally presented in [30]. This previous application was focused on the hardware/software co-design of systems.

In addition to the languages presented above we have used the Unified Modelling Language (UML) [31] and its extended subset System Modelling Language (SysML) [32] to represent software and systems. We have not used these languages to conduct any kind of analysis over the system energy consumption but to represent software and system structure and behaviour. The diagrams in this document are fairly self explanatory, but descriptions of their notations can be found in the literature.

### 2.4.2 Hardware tools

In the PhD work two different Digital AcQuisition (DAQ) boards to conduct measurements in hardware prototypes have been used:

**Logic analyzer**[4]**:** This hardware makes it possible to capture the evolution of digital signals over a period of time. This enables the analysis of real-time deadlines, interfacing with external components and verification of signals generated.

**Digital storage scope**[5]**:** This hardware enables the analysis of the evolution of analog signals over a period of time. This makes it possible to analyze the evolution of the power and energy consumption among multiple parameters.

Both DAQ boards provide a .NET Application Programming Interface (API), that allows to control the DAQ process from custom software. This possibility will be exploited later in order to monitor Hardware In the Loop (HIL) simulations in the work presented in chapter 5.

[3]20-Sim official website: http://www.20sim.com/

[3]Formerly known as the DESTECS platform. Official website: http://www.destecs.org/

[4]The concrete logic analyzer used in this work is the Saleae Logic http://www.saleae.com/logic

[5]The concrete analog storage socope used in this work is the Digilent Analog Discovery Kit http://www.digilentinc.com/Products/Detail.cfm?Prod=ANALOG-DISCOVERY

## 2.5 Case study

This PhD project uses as case study the design and development of an electronic compression stocking to treat leg venous insufficiency. This system is being developed under the e-Stocking Ambient Assisted Living EU project[6]. The goal of this project is to deliver quality treatment and to enable a higher degree of independent living to the elderly population suffering from this condition. This stocking is required to deliver a continuous compression gradient that ranges from 40 mmHg at the ankle to 20 mmHg below the knee. The stocking must reach the pressurized state and keep compression using as little energy as possible as it is operated by batteries. The stocking prototype is composed of a textile garment that contains the air-bladders responsible for the compression and a number of electronic and electromechanical components controlled from software. These hardware elements are represented in the SysML Block Definition Diagram (BDD) shown in figure 2.1.

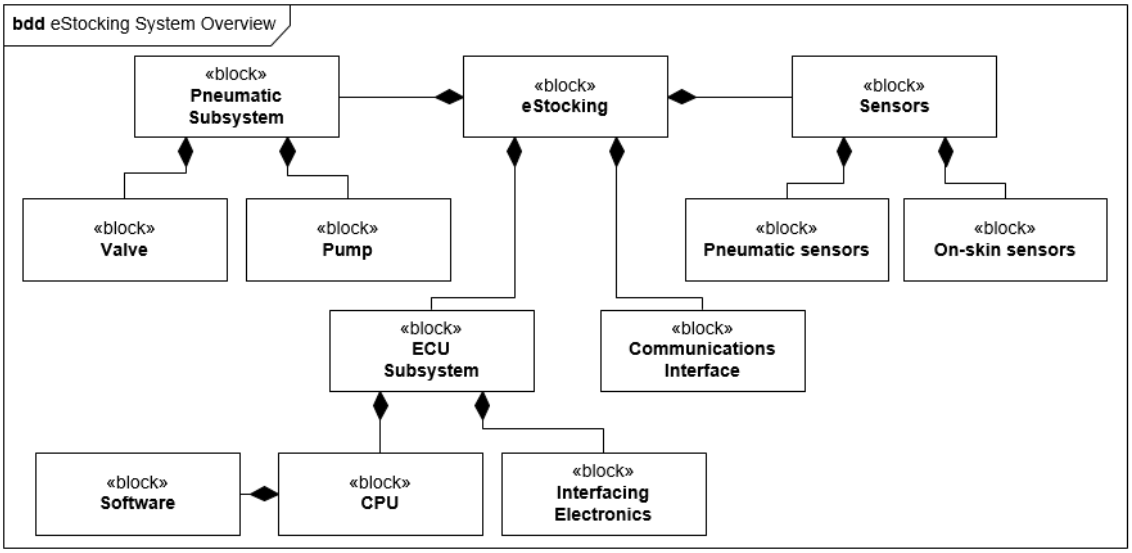


Figure 2.1: eStocking system overview.

The e-stocking hardware system is composed of two major subsystems:

**Pneumatic subsystem:** composed of electrically controlled valves and pump. It is responsible for conducting the air-flow and building the air pressure in the bladders required to deliver the compression.

**Electronic Control Unit (ECU) subsystem:** composed of two boards, one with interfacing electronics to sensors and actuators and a second one mounting a PSoC5 featuring the control software.

In addition to the subsystems presented above, the eStocking features a communication interface (Bluetooth link) and two different kind of sensors (pneumatic and On-skin). More specific details on some parts of the system that are of interest for this work will be provided in the subsequent sections. Additional information on this work can be found in [33, 34].

[6]E-stockings official website: http://www.e-stockings.eu/

# 3

# Modelling energy consumption in mechatronic systems

*Energy consumption in mechatronic systems is determined by the energy consumed in the electronics (including the CPU and other peripheral hardware) and in the electromechanical devices used to interface the physical environment. Typically the sensors and the actuators are commanded from a control logic running in a CPU, hence it could be concluded that the energy consumption in most mechatronic systems is software dependent. This chapter shows how to study the energy implications of different control algorithms in a mechatronic system. This chapter contains extracts from the work presented in the International Conference on Modelling and Simulation 2013 [35].*

## 3.1 Challenges addressed

In order to optimize overall system energy consumption, embedded software engineers developing control algorithms need to be able to understand the energy consumption in the electromechanical domain. This can be achieved by creating a system prototype or a model and test different control algorithms over it. The creation of a prototype can be complex, expensive and it can easily take a considerable amount of time. Additionally, measuring physical parameters is often complex and influenced by many factors such as measurement techniques and/or noise. These factors make such a study cumbersome and typically complex to repeat under exactly the same circumstances. A way to overcome these limitations is to use models of the physical system under control. Besides simplifying the study of energy consumption, the application of modelling brings the possibility of conducting design space exploration in a more efficient manner. This process allows the engineers to find electromechanical architectures that make better use of the energy available.

Mechatronic systems are composed of elements that are best represented by different modelling paradigms. Control logic is best modelled using Discrete Event (DE) modelling languages such as VDM-RT. Physical phenomena is best modelled using Continuous Time (CT) representations, such as differential equations. The work presented in here makes use of these two modelling paradigms to co-simulate mechatronic systems, explore the design space and analyse energy consumption.

## 3.2 Methodology

We propose the application of a methodology composed of five phases. An overview of this modelling process is presented in Fig. 3.1. These phases progress in a sequential manner, illustrated by a solid arrow between the phases. Additionally it is possible to iterate over the different phases depending on the modelling progress (illustrated by the dashed lines to previous phases).

In this methodology we take as input the Physical Requirements and Component Characteristics in the Continuous Time modelling phase and Control Requirements in the Discrete Event modelling phase (shown by dot-dashed arrows in the diagram). Additionally, we highlight the result of the Co-model Execution phase; Consumption Estimates, that serve as input to the Trade-off Analysis phase. In case the results of the Trade-off Analysis phase shows that it will not be possible to meet the energy and/or power consumption requirements, it is possible to revisit the previous phases and consider alternative physical realizations or different components.

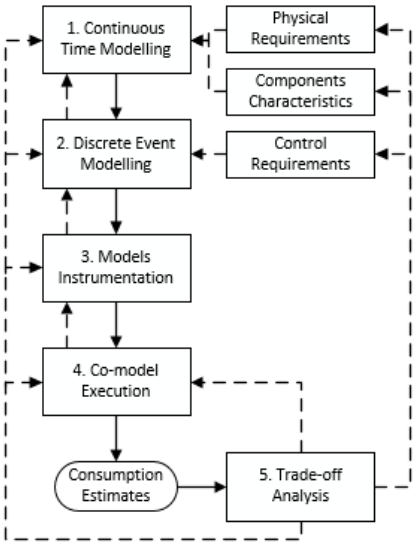


Figure 3.1: Methodology overview.

These five phase can be explained as:

**1.- Continuous Time (CT) Modelling:** We model the system from a mechanical perspective, taking into account the physical requirements and components characteristics. At this point the physical model can be exercised with control signals to check that the physical model is appropriate.

**2.- Discrete Event (DE) Modelling:** We take the control requirements as input and focus on capturing the control logic. We start with a simple control strategy that exercises the model created in the previous phase. The purpose of this is to validate the physical models further before evaluating more complex control strategies. In case we find errors in the physical model, these can be corrected and iterate over the DE Modelling phase again. Once this additional validation is completed, we can model further more complex control strategies and iterate again over the CT Model to evaluate its behaviour.

**3.- Models Instrumentation:** We incorporate the interfaces and monitored variables to evaluate energy consumption. The instrumentation takes place mostly at the CT model and is performed by adding target variables as monitored in the physical simulator. In order to measure the power

consumption we propose the introduction of a power meter block per component consuming energy in the CT model. Each power meter block will be responsible for computing the power consumption of that single component. Each power meter will take the component operating voltage and current as input. The calculation of the consumed power and energy will depend on the control signal provided by the DE model. Optionally, the DE model can take power and energy measurements into account at runtime with the purpose of modelling power or energy-aware operations at the component level. The calculated power consumption is represented as an output flow from the power meter for later processing in the CT model. The application of this block is shown in Fig. 3.2. This diagram uses ports with arrows inside to represent continuous data flow (stream ports) and empty ports to represent discrete data communication (standard ports).

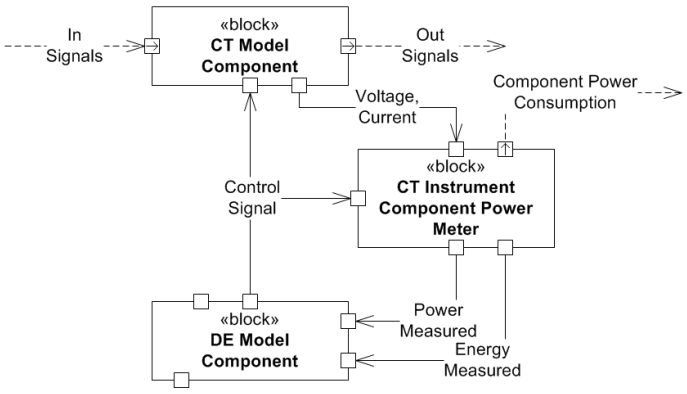


Figure 3.2: Power Meter measuring component power and energy consumption.

In order to compute the system total power consumption, we communicate all the component power meters with the system power meter. This block adds the power consumption logged by each component meter and integrates it to calculate the total system energy consumption.

**4.- Co-model Execution:** In this phase we execute the co-model and further study the results of the control algorithms for the mechanical system. Thanks to the model instrumentation it is possible to monitor power and energy consumption over time and study the performance of the different control algorithms. As a result of this phase we will produce a number of system consumption estimates.

**5.- Trade-off Analysis:** In this phase we evaluate the final energy and power consumption results and compare them against the system requirements. An example of the trade-offs we might encounter could be: energy consumption vs. accuracy, autonomy or size. Additionally we will be able to analyze the power and energy consumption of the individual components. In case the results we obtain through the trade-off analysis show that the requirements cannot be met with the solution considered, we can revisit the chosen components and reconsider their suitability. Optionally and depending on the kind of system under development the engineers may need to revisit the requirements and adjust them to what it is technologically feasible.

## 3.3 Abstractions and model limitations

The purpose of the models is to get an overview of how the energy is spent on the physical side of the system and how different control strategies affect the energy consumption. The level of detail present in the models should be enough to perform trade-off analysis at the system level and therefore to achieve a *coarse-grained estimation*.

As part of the process of setting the level of abstraction, some of the physical components will be left out in the physical models. We propose to follow these steps in order to decide which components to incorporate in the models:

1. Identify the most energy consuming sensors and actuators and incorporate them in the CT model.
2. Model the control logic of the sensors and actuators detected in the previous step in the DE model.
3. Study sensors and actuators that have lower energy consumption than the ones detected in step 1. In case the frequency of use makes their energy consumption comparable to those identified in step 1, incorporate them into the models.
4. Model the control logic of the sensors and actuators identified in step 3.

Regarding the level of abstraction in the DE models, we propose to limit the DE modelling only to the control logic. In case the systems engineer is interested on performing additional modelling on other software functionality, we advice to create a separate VDM-RT model targeting only software related issues and creating a VDM-RT mock-up model representing the interaction with the physical world in a signal based fashion.

## 3.4 Application in the case study

We have applied the approach proposed in this chapter to the e-stockings case study presented in section 2.5.

### 3.4.1 Modelling

#### CT modelling

The preliminary mechanical architecture is composed off three electrically controlled pneumatic actuators. These components are mounted as presented in figure 3.3. The functionality provided by each component is:

**Pump:** able to maintain a constant airflow that is delivered to the air chambers around the leg.

**Air distribution valve:** able to direct the air flow to the first or the second air chamber. It also allows to vent the bladders individually.

**Pass valve:** able to lock the air bladder or open it when activated. Each air bladder has one pass valve.

Applying mechanics and pneumatics theory we have been able to establish a physical relation between the shape of the air bladder, the contact surface with the leg, the flow of air into it and the final pressure exerted over the leg. This is presented in equation 3.1.

$$Pressure_{Leg} = \frac{BulkModulus_{air} \times \int_{t_0}^{t_{fin}} \frac{\varphi}{Volume} \mathrm{d}t \times Area_{AirBladder}}{Area_{Leg}} \tag{3.1}$$

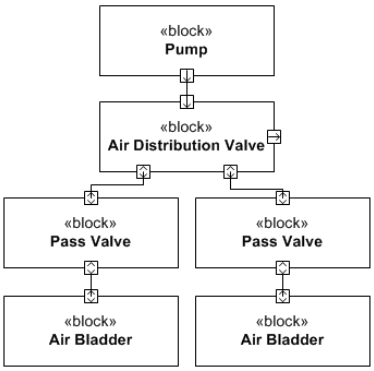


Figure 3.3: Overview of the CT model.

We have used two pressure equations to model the behaviour of the two different air bladders that compose the stocking.

**DE modelling**

The control software of the system is composed of a dummy controller and two regulation strategies. These strategies are wrapped in different classes:

**Dummy control:** contains a sequence of test signals for the different actuators deployed in the model and a number of commands for pressure reading. It allows the execution of a simple scenario to test the DE-CT model communication and validate the CT model behaviour.

**Dummy regulation:** provides a basic logical regulation based on conditions. It mainly checks if the desired pressure has been reached and if that is the case it stops the pumping. In case there is overpressure it vents the air bladders.

**PID regulation:** applies a proportional regulator [36] to reach the target pressure. It modifies the pump duty cycle at runtime depending on the difference between existing and target pressure.

These classes are instantiated within the controller class, that is able to interface to sensors and actuators. The controller class is a periodic thread that can be run at different frequencies. The logic to access concrete sensors and actuators is wrapped in additional classes to make a cleaner design. An overview of the control software is provided in the class diagram in figure 3.4.

### 3.4.2 Results analysis

After running the models developed for this case study we analysed the power and energy-wise implications of different control strategies. Additionally we considered the application of different mechanical architectures and the improvements they could bring.

**Power consumption**

After running the different control strategies modelled on the DE side with the same CT model, we have obtained four different power consumption estimates (one per DE strategy considered). System power consumption over time is shown in Fig. 3.5. Graphs a, b and c show the power consumption of the proportional controller with gains 2, 1 and 0.5 respectively. Finally, graph d shows the evolution of the power consumption under the dummy control algorithm. The dashed lines in each graph show the average power consumption in each case. The maximum average

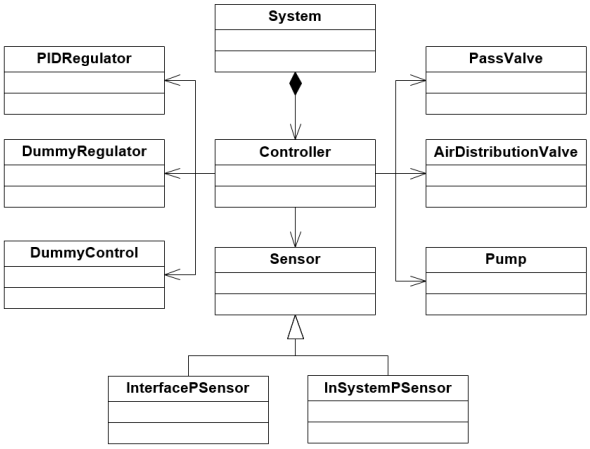


Figure 3.4: UML class diagram giving an overview of the DE model.

power consumption is 1.3 Watts, present when the dummy control is applied. The least power consuming control strategy is the proportional controller with gain 1, with an average consumption of 1.1 Watts. In all four cases the peak power consumption is 1.5 Watts.

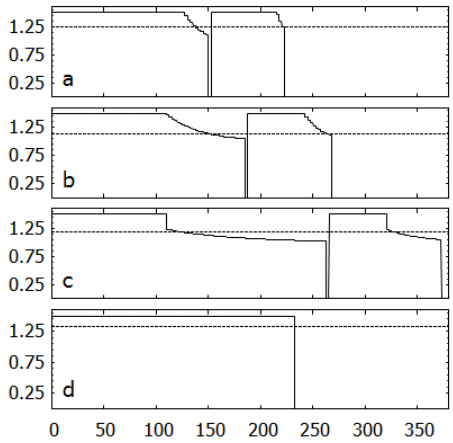


Figure 3.5: System power consumption over time under different control algorithms for stocking compression. y axis: power in watts, x axis: time in seconds.

### Energy consumption

We have calculated the system energy consumption by integrating the power consumption curves presented in Fig. 3.5. Additionally we have paid special attention to the energy consumed by the pump under the different algorithms. These estimates together with the total time the system has been active in each case are shown in Table 3.1.

The pump energy consumption under the different proportional control strategies is practically the same. The dummy control makes the most inefficient use of the pump. At the system level it is evident that, even though the pump consumption was almost the same in the first three cases, there is a considerable difference in consumed energy between the proportional controllers (up to 149 Joules). This happens because the controller with a higher proportional gain remains active for less time, and therefore requires the valves that enable inflation to be active for a shorter period of time.

Table 3.1: Energy consumed under different control algorithms

| Control Algorithm | System Energy Consumption [Joules] | Time Active [Seconds] | Pump Energy Consumption [Joules] |
|---|---|---|---|
| p gain 2 | 323 | 222 | 103 |
| p gain 1 | 367 | 267 | 102 |
| p gain 0.5 | 472 | 372 | 102 |
| dummy control | 348 | 232 | 116 |

**Suggestions for mechanical modifications**

With the current system mechanical architecture we need to keep running the pump, the air distribution valve and one pass valve, corresponding to the air bladder under inflation at the same time. This makes the power consumption peak at 1.5 Watts. We found out that it is possible to avoid the air distribution valve and cover its distribution functionality with a fixed air splitter and its selective air-bladder venting functionality with an additional pass valve per air bladder configured as sealed by default and connecting the air bladder to the exterior. This would make it possible to reduce peak power consumption to 1 Watt, a reduction of 33%. The total energy savings with this new architecture developed after model analysis are shown in Table 3.2.

Table 3.2: Potential Energy Savings

| Control Algorithm | System Energy Savings [Joules] | Reduction | Final System Energy Consumption [Joules] |
|---|---|---|---|
| p gain 2 | 111 | 34% | 211 |
| p gain 1 | 133 | 36% | 234 |
| p gain 0.5 | 186 | 39% | 286 |
| dummy control | 116 | 33% | 232 |

## 3.5 Final remarks

While these models were under development, one of the partners in the e-stocking project was creating prototypes of the compression stocking implementing different compression strategies and pneumatic configurations. Through the models created in this study we were able to foresee the unfeasibility of one of the compression strategies without investing man-hours and resources in prototyping. This was later confirmed by the prototyping work conducted by the other partner. Additionally, through the modelling work we were able to foresee the importance of a transfer function that establish the relation between air-pressure in the air-bladders and delivered compression over the skin. This was detected already during the second month of the project (within an 36 months project) and this has turned out to be the best strategy for pressure regulation and compression delivery. Finally, the idea of applying a proportional regulation to the compression process was explored through modelling and finally incorporated to the implementation due to its benefits in energy consumption and noise reduction. The application of modelling has benefited the project allowing us to find system-level problems in a cost-effective manner and to determine critical factors to optimize in the design.

# 4

# Modelling energy consumption of different CPU states

*Microcontroller-based embedded systems can achieve considerable energy savings by using low-power CPU states. These low-power states stop the on-going computations in the CPU and wait for external or self-triggered events to wake up. One of the challenges when designing embedded software is to decide when to use low-power states and for how long while still meeting the system requirements. One way of studying this is by applying the real-time software modelling language VDM-RT. This chapter presents how multi-state CPUs could be modelled using the VDM-RT CPU abstraction and what improvements could be beneficial to ease this analysis. This chapter contains extracts from the work presented in the 11th Overture workshop, 2013 [37]*

## 4.1 Challenges addressed

The VDM-RT modelling language introduces the CPU abstraction, that represents an execution node where parts of a model can be deployed. Each execution environment represents the hardware support for the computation (the CPU itself) and a Real Time Operating System (RTOS). These CPUs are configured by the parameters clock frequency and scheduling policy[1]. An example of a VDM-RT CPU is presented in the listing 4.1. In this case a CPU is created and configured with an Fixed Priority scheduling policy and working at a clock frequency of 24 MHz.

```
mcu : CPU := new CPU(<FP>, 24e6);
```

Listing 4.1: Example of a CPU in VDM-RT.

VDM-RT CPUs are constantly able to compute and communicate and thus represent a CPU that is active continuously. This evaluation of abstraction is not appropriate if we are interested in studying the power consumption of a CPU that uses low-power consumption states. The challenge here is to incorporate a way to represent CPU sleeping states without requiring language, scheduler or platform modifications at first.

[1] When configured as a fixed priority RTOS and only with a single thread Real-Time or procedural, the RTOS can be "bypassed" simulating an OS-less execution environment.

## 4.2 A design pattern structure to model CPU power modes

We propose the application of a design pattern structure (following the terminology proposed in [38] that makes it possible to represent different CPU power states. This structure makes use of a `VirtualCPUstate` class to represent different CPU operational modes that controls whether code can be executed or not. An overview of this is shown in the UML class diagram in Figure 4.1. The general idea in this structure is that whenever a thread running application logic is scheduled-in by the VDM-RT scheduler, it will check if the state registered in the `VirtualCPUstate` class is active or sleeping before running. There is a second case in which the thread will always run but forces a change in the Virtual CPU state before doing so. Additional details will be provided in the subsections below. This structure makes use of the following classes:

**`VirtualCPUstate`:** represents the CPU state. This class is protected against race conditions and thread interference by mutexes and history counters.

**`PowerLogger`:** keeps track of the CPU state changes and logs them so they can be analyzed further and represented in a graph.

**`ProcTRunner`:** represents functionality that should be implemented as a procedural thread. This class is a concrete instantiation of the `ProceduralThread` wrapper.

**`RealTimeRunner`:** represents functionality that is executed periodically in a periodic thread. This class is a concrete instantiation of the `RTthread` wrapper.

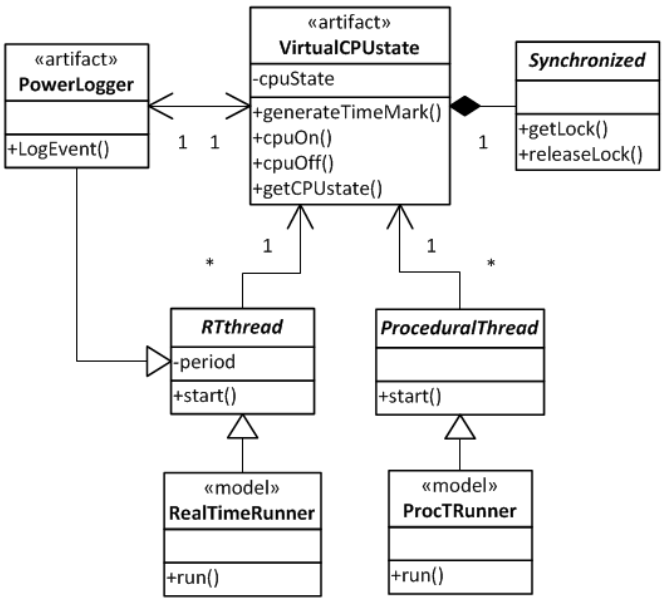


Figure 4.1: Design pattern structure to model multiple CPU states.

The `VirtualCPUstate` class might be accessed by several threads in a concurrent manner. Therefore it is necessary to ensure thread safety in the operations that read and modify the `cpuState` atrribute that represents the CPU operational mode. We have used mutexes to prevent data corruption. Additionally and besides preventing the corruption of `cpuState`, we must ensure that the `getCPU` state function is executed every time there is a change in the power model.

We have reinforced this policy by using VDM-RT permission predicates and history counters. This is shown in listing 4.2.

```
per turnOn =>  #req(turnOn) - #act(turnOn) = 1;
per turnOff => #req(turnOff) - #act(turnOff) = 1;
per getCPUstate => #active(getCPUstate) = 0  and
    cpuOn => #fin(turnOff) = #fin(getCPUstate);
```

Listing 4.2: Protection in VirtualState with history counters.

Whenever there is a change in the state of the Virtual CPU a method pushes this change to the `PowerLogger` class. The operation responsible for this is preceded by a **`duration(0)`** statement to avoid interfering the application logic timing.

This sets a basic infrastructure to model upon different behaviour. We have modelled two different scenarios that are explained in the subsections below.

## 4.3 Pattern application

This section presents the application of the pattern introduced above in two different scenarios common in embedded system design.

### 4.3.1 Modelling functionality that makes a CPU become active

In this case we model a scenario in which whenever certain functionality has to be executed the CPU is awakened. This is incorporated to the `run` method modelled in the thread. This is modelled as shown in listing 4.3: the thread marks the state in the `VirtualCPUstate` class as `on`, executes the application logic and marks the CPU state as `off` again. The state changes are preceded by a 0 nanosecond duration statement, hence we are not accounting for the transition time between states.

```
duration (0) state.turnOn();
executeLogic();
duration (0) state.turnOff();
```

Listing 4.3: Conditional controlling the execution.

### 4.3.2 Modelling functionality that runs if CPU is active

In this second scenario we model the situation in which the application logic runs only when the CPU is active. We have used the structure presented in listing 4.4. This structure is modelled as part of the thread operation `run` that is executed when it is scheduled-in. In this case, when the thread is scheduled-in it will check the state of the Virtual CPU and determine if it is able to execute its application logic or not.

```
if not state.getCPUstate()
then duration (0) IO`print("CPU is off");
else executeLogic();
```

Listing 4.4: Conditional controlling the execution.

### 4.3.3 Data analysis

The execution of a model making use of this pattern produces an output log file that registers the states in which the CPU have been and when the transitions between them have taken place. By processing this file it is possible to create a graph showing the duty cycle of the CPU. An example is presented in figure 4.2. In this particular case a CPU has been active in the intervals 20 to 25 ms and 50 to 70 ms, and remained sleeping during the rest of the time. If we take a concrete CPU model it is possible to calculate concrete power and energy consumption figures. Taking the ARM Cortex M3 platform; the manufacturer specifies a CPU consumption of 6.5 mA when operating at 5 Volts. This results on a power consumption of 32.5m Watts.

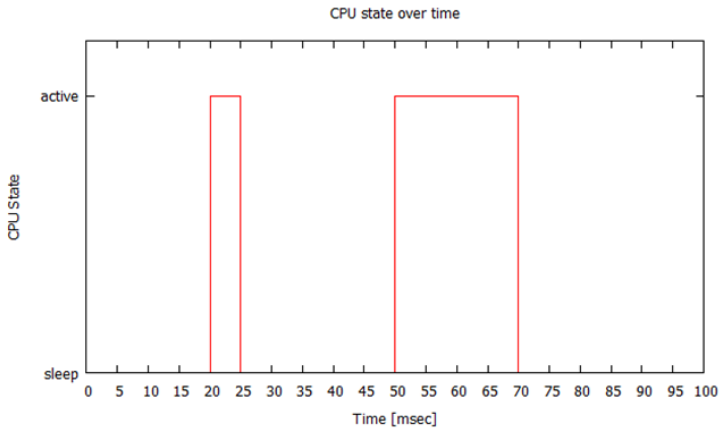


Figure 4.2: Evolution of the CPU power consumption over time.

It is possible to calculate the system energy consumption by integrating the power consumption over the period of time in which the system has been active. This is illustrated in figure 4.3 and yields a total power consumption of 650 mJoules. In this case, by making use of the sleep states, it has been possible to save approximately 2600 mJ.

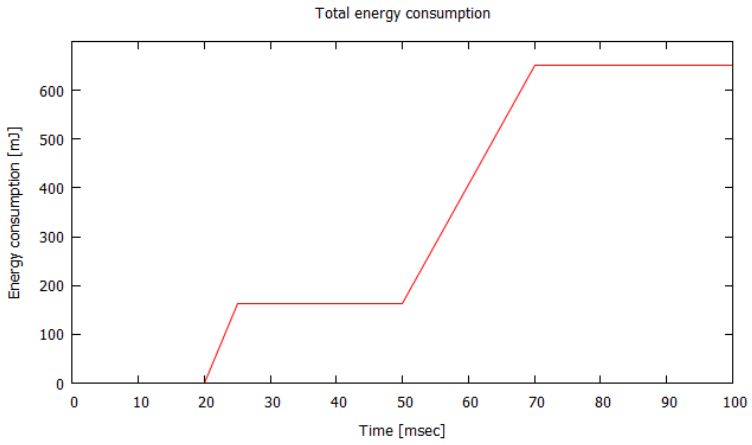


Figure 4.3: Evolution of the CPU energy consumption over time.

## 4.4 Proposed expansions and improvements to the VDM-RT language

The design pattern presented in this chapter is a way to overcome at the modelling level current limitations of the Overture platform and VDM-RT CPU abstractions. We would like to take this idea further and possibly incorporate changes in the Overture tool and the VDM-RT language.

### 4.4.1 Unaddressed issues

This design pattern does not take into account the communication aspect when the CPU is sleeping. Here we should consider two scenarios: a) Information is received at the communication interface and this generates an interrupt that wakes up the CPU so the information can be processed and b) Information is received and discarded since the CPU is sleeping. These aspects have not been studied but are worthwhile exploring further in later work.

The co-simulation approach presented in chapter 3 should be revisited to review its applicability to study computation and communication power consumption. Our initial thesis is that this could be especially beneficial for systems with multiple components (such a SoC with analog and digital blocks). However additional work is needed to confirm this.

Finally, it could be interesting to explore how dynamic frequency scaling could be incorporated to VDM-RT CPUs. As presented in section 2.2 the power consumption in CMOS devices is directly proportional to the frequency at which the device is running. Slower CPU clocks would lead to power and energy savings but it could imply missing real-time deadlines as well. Exploring different CPU frequencies at the modelling level in VDM-RT could be beneficial since it would make it possible to check whether or not real-time requirements are fulfilled or not when the CPU runs at different frequencies.

### 4.4.2 Tool and language modifications

We propose to implement the sleep functionality in the VDM-RT java engine rather than at the modelling level. This would simplify the process of creating the real-time models since the engineer could then focus on modelling the application logic without having to take into account the pattern presented above. This modification would make it possible to sleep the CPU with just one operation call in the model (`cpu.sleep()`). In addition the language should incorporate the necessary facilities to wake CPUs up again (`cpu.active()`).

Besides these language modifications, additional tool work will be needed to generate consumption graphs automatically and to produce real-time logs in a more efficient way.

## 4.5 Final remarks

The work presented in this chapter is on-going at the moment and some of the unaddressed issues presented above are under consideration. At this point we have incorporated the operations sleep and active to the VDM-RT CPUs and we have created a platform to conduct accurate power consumption measurements in physical CPUs. Additional efforts are needed in order to test these language modifications and relate predicted energy consumption figures with actual ones in an scenario related to the case study.

# 5

# Model Validation & Verification through Hardware In the Loop in VDM-RT

*The previous chapters have provided an insight into cyber-physical system (CPS) modelling with focus on energy consumption. The techniques presented so far aim at exploring the design space. This chapter introduces the application of Hardware In the Loop (HIL) for VDM-RT models, enabling the combination of a complete or partial system implementation with a System Level Model. HIL is typically used as a way to provide a system with external stimuli with the purpose of mocking-up an external piece of equipment. In this case we propose its application to enable the stepwise transformation of models into candidate realisations and to incorporate physical measurements on target into the models. This chapter contains extracts of the paper submitted to the 2nd International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2014 [39].*

## 5.1 Challenges addressed

The HIL system proposed in this work is intended to allow the execution flow handover from a VDM-RT model to an external target embedded system (hereafter called the Device Under Test (DUT)). Additionally it supports the execution flow return from the DUT to the VDM-RT model execution environment. The main objective of this is to allow a stepwise transformation of models of functionality into components that realise that functionality. In order to benefit from the models one can combine the implemented components with the models of those that still have not been realised. In this way it is possible to simulate the components in a single model-implementation co-execution. This is illustrated in Figure 5.1 with an example based on the design of an embedded system that enters three stages: Acquire Data, Process Data and Provide Output. The design of such a system starts with the modelling of the complete system functionality. This is illustrated in the upper row of Figure 5.1 (Full Modelling). As it can be seen all the components are modelled (represented on the VDM side) and no components have been implemented on the target (the HW side). After the system has been modelled the component providing the Process Data functionality is implemented. It is the intention to substitute the modelled Process Data component with its correspondent implementation and still use the rest of the model for system simulation. This

approach is shown in the lower row of Figure 5.1. At this point it is possible to start the system simulation on the VDM-RT side in the Acquire Data stage and then the HIL system will hand-over the execution of the Process Data stage to the implementation deployed on target. Once Process Data has completed, execution will return from the target embedded system to the VDM-RT model of this functionality and continue with the Provide Output stage.

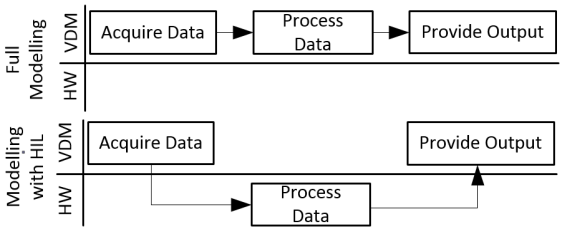


Figure 5.1: Device modelling and Combined device modelling with HIL.

By applying the approach proposed in this paper a systems engineer is able to benefit from: 1) the expressiveness of VDM-RT to represent real-time constraints and system properties 2) the simulation capabilities of Overture and 3) the insight gained through prototyping using a combined-modelling prototyping approach.

In addition of the execution hand-over the system should incorporate the necessary facilities to provide the external stimuli to the DUT, monitor its output lines and communicate with it.

## 5.2 Design of the HIL system

This section introduces the architectural design of the HIL system presenting hardware and software aspects.

### 5.2.1 Hardware

The system is composed of a number of external hardware components. The hardware architecture is presented in Figure 5.2 through a Block Definition Diagram (BDD) and shows the components that comprise the system in a hierarchical manner. The HIL setup block is the top-level representation of the system. The rest of the components are:

**Workstation:** This is a computer running Overture and the required software infrastructure to communicate with the rest of the hardware components connected to it.

**DUT:** This is the external device that contains a partial implementation of the system. This partial implementation contains a certain functionality that will be triggered from the model execution. The DUT is composed of Embedded Software and Embedded Hardware.

**Stimuli Provider:** This is an output board able to provide digital signals to the DUT. These external inputs can represent any device with a logical interface, e.g. a sensor measuring physical parameters or a different digital hardware block.

**Logic Analyzer:** This is a DAQ board enabling capturing of the evolution of a number of digital signals over time. The main purpose of this block is to monitor the digital outputs provided by the DUT.

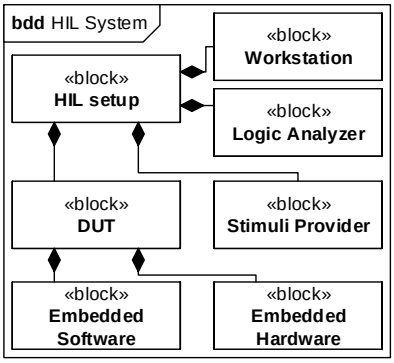


Figure 5.2: Main components of the HIL setup shown in a SysML BDD.

The connection between the hardware components is presented by the Internal Block Definition Diagram (IBD) shown in Figure 5.3. Standard ports represent discrete logical signals and flow ports continuous signals. The Workstation uses serial busses to communicate with the rest of the blocks. The communication between the Stimuli Provider and DUT is made through a logical bus that maps the Stimuli Provider outputs to the DUT inputs. The same kind of bus is used between the DUT and the Logical Analyzer to monitor the DUT output values. Additionally two dedicated inputs in the Logic Analyzer are used to analyze a DUT Pulse Width Modulated (PWM) output and the function execution time (Duration Pin).

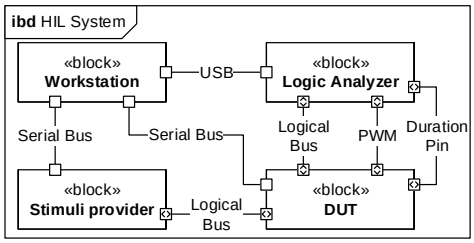


Figure 5.3: SysML Internal Block Diagram showing the hardware connections to the DUT.

The software components running on the Workstation are presented later in this section.

### 5.2.2 Software and modelling components

The workstation software and modelling components of the HIL setup are shown in relation to each other in the class diagram in Figure 5.4. This system makes use of the VDM-RT modelling language and interpreter, already introduced in section 2.4.1 The main components are treated individually in the description below.

**Hardware Proxy:** This specifies the interface of the hardware being controlled from the VDM-RT execution. All invocations are initiated in the VDM-RT model and communicated through the Java Bridge before they reach the physical hardware.

**Java Bridge:** This enables delegation of VDM-RT functionality execution to Java. In this way one can invoke a VDM-RT operation or function and have the body specified and executed

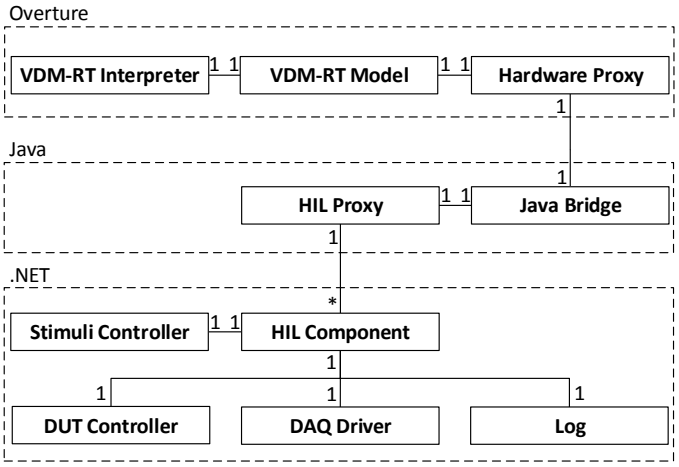


Figure 5.4: The software and modelling components of the HIL setup shown in a class diagram.

as a Java method. The Java Bridge does automatic conversion and transferring of input and output values between Java and VDM-RT.

**HIL Proxy:** This is a thin software component that accepts inputs from the Java Bridge inside Overture and relays the invocations originating from the VDM-RT execution to a HIL component responsible for communicating with the hardware used in the HIL setup. The HIL proxy is interoperability glue that enables communication between Overture and the hardware.

**HIL Component:** This is responsible for communicating directly with the hardware used in the HIL setup through serial connections. Although we only connect to a single instance of the HIL Component it would be possible to extend this to multiple instances.

**Stimuli Controller:** This component manages the logical signals that are provided to the DUT as external stimuli. Its behaviour is controlled by the HIL Component.

**DUT Controller:** This is instructing the external DUT to execute the Functionality Under Test. This can be realised in software or hardware.

**DAQ Driver:** This provides the software interface enabling the HIL component to launch the DAQ so the data needed can be acquired from the DUT execution.

In addition to the components presented above the system also features a "Log" making it possible to write data acquired to the file system.

### 5.2.3 Embedded Software

The embedded software executing on the DUT is composed of three main blocks: Hardware Drivers, Functionality Under Test (FUT) and HIL Support as shown in Figure 5.5. Hardware Drivers provide software support to interface the hardware blocks. HIL Support contains logic to start the execution of the FUT upon request from the DUT Controller, running on the workstation. It is structured in three phases:

1. **Wait for command:** The system waits for a possibly parametrized command over UART describing the functionality to execute.

2. **Serve HIL request:** The system initiates the execution of the FUT. This can be surrounded by pin toggling operations to measure the execution time of the function. Optionally additional pin toggling can be performed inside the function to measure the time it takes to reach different points during the execution.

3. **Return to DUT controller:** Once the system has completed the execution of the FUT it returns to the DUT controller so it can stop signal sampling and notify the model execution environment to resume model execution. Additionally it can return result values produced by the execution of the FUT.

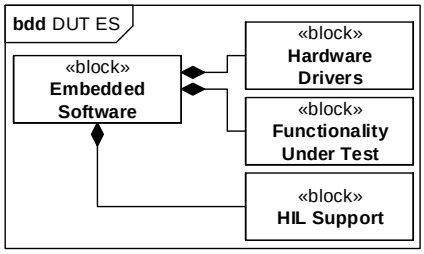


Figure 5.5: The embedded software of the DUT shown in a SysML BDD.

### 5.2.4 Model co-execution

The model co-execution starts on the VDM-RT side with the interpreter executing the model until it reaches an operation registered as "not specified" in the Hardware Proxy. This indicates that the operation modelling that functionality is implemented in the DUT. This invocation is relayed by the HIL Proxy to the HIL component.

Hardware control is initiated through the Hardware Proxy that specifies the functionality of the physical hardware accessible to the VDM-RT model (e.g. providing the DUT with input signals or performing hardware or software computations). Invocations to the Hardware Proxy are initiated in the VDM-RT model, and then relayed by the HIL Proxy to the HIL Component. From here the HIL Component either invokes the Stimuli Controller or the DUT Controller. This process is represented with a UML sequence diagram in Figure 5.6. In this diagram the HIL Component first signals the Stimuli Controller to supply the DUT with input signals. Next it launches the DAQ using the corresponding driver and finally the DUT is instructed to execute the FUT via the DUT controller. While the DUT is executing, the DAQ is continuously sampling the state of the DUT, which is logged to a file at the end of execution. Invocations from the VDM-RT interpreter, Hardware Proxy, HIL Proxy and HIL Component (including data logging) have been omitted from Figure 5.6 for clarity.

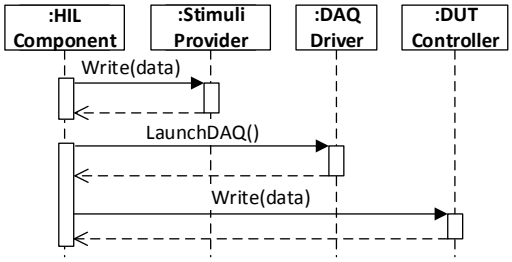


Figure 5.6: Sequence diagram showing the Workstation communicating with the DAQ Driver and the DUT Controller.

## 5.3 Execution results

We have conducted preliminary co-executions of model and implementations and we can conclude that with the current HIL setup it is possible to validate the real-time behaviour of the FUT component as well as the correct operation of its control logic. We have been able to validate that a certain time slot allocated to the FUT is not overrun and obtained a precise time measurement that can be incorporated to the models. Additionally it has been possible to exercise the implemented logic together with the modelled components on a number of scenarios. Additional work is needed in order to apply this technique to a complex case study.

## 5.4 Final remarks

This work is currently under further development and we are working on a number of improvements to the current HIL system. Additional details on this will be provided in the section 6.2.1.

# 6

# Current status and future plans

*This chapter concludes this report with a summary of the work conducted during the first part of this PhD and explains how this work will be extended.*

## 6.1 Summary of work conducted

During this first part of the PhD project we have successfully applied the VDM-RT modelling language and the DESTECS platform (Crescendo) to the system-level energy-aware design of Cyber-Physical Systems (CPS). The main contributions have been focused on the study of energy consumption in the electromechanical and computing domain.

Additionally we have developed the infrastructure necessary to enable the (Hardware In the Loop) HIL for VDM-RT models, facilitating the co-execution of models with prototype hardware/software realizations in target. These techniques need to be better supported with additional physical measurements. Additionally we have to explore the analysis of energy consumption in the communication area. Reflections about future work are presented in the sections below.

## 6.2 Future work

In this section we present a number of issues that are currently under development or scheduled as future action items.

### 6.2.1 Improvements to the existing work

Regarding energy consumption in mechatronic systems, the predictions of consumed energy obtained through modelling need to be validated further against real measurements taken on current eStockings prototypes. These measurements have not been conducted earlier since a usable prototype has not been available until few weeks ago. This would also imply minor modifications to the existing co-models represented in figure 1.2.

Regarding the work on HIL, introduced in figure 1.4, we are considering more elaborated profiling (time, power and energy-wise), the specification and validation of real-time deadlines through validation conjectures [41] and a better support for waveform analysis and logging by

adopting the VCD format [42]. Additionally, further efforts are needed in order to apply this HIL technique to the eStockings case study.

The work on computation modelling is currently under further development and the primitives necessary to sleep and activate the VDM-RT CPUs have been already incorporated to the VDM-RT language. These additions still need further validation through a more complex application of the abstractions to the case study. The analysis of the energy consumed would still be based on the logs presented in figure 1.3, but additional facilities will be built into the Overture tool in order to facilitate their analysis.

### 6.2.2 Modelling of communication energy consumption

The research conducted so far has not covered the energy consumed while communicating with other systems and/or subsystems. The modelling language VDM-RT incorporates a communication abstraction that enables to establish a communication channel between several CPUs present in a model. At this point this abstraction does not take the energy consumed while transmitting or receiving data into account. As part of this work we will try to explore how the notion of energy can be modelled using this abstraction and how this can be used to study performance at the node and network level. Our initial position is that it will be suitable to study small scale CPS networks such as a Body Area Network. We also consider the applicability of this technology to greater networks to be reduced due to the scalability problems of the modelling platform used.

### 6.2.3 Other possible work

The core part of this PhD is the application of modelling to system-level energy aware design of CPS. However we have identified some interesting contributions that could be carried out in case the main hypothesis of this PhD has already been covered. During the course of this PhD project we are reviewing extensively existing techniques, methodologies and modelling approaches to low-power design and energy consumption among many other related topics. This material could be summarized and compiled into a survey about the state of the art in power and energy-aware design. In case we detect common problems that affect the energy or power consumption in specific way, we could try to propose general solutions to those. These solutions could be formulated as design patterns and compiled in a small "pattern catalogue". Depending on the extension and the originality of the solutions detected here, this work could be incorporated in the survey mentioned above or as a standalone publication.

Finally, during this PhD we are collaborating extensively in the project e-Stockings developing software, hardware and integrating different actuating and sensing technologies. Publications about these developments are planned in the future.

## 6.3 Progress and future work overview

We present an overview of the state of the work conducted until now in table 6.1. Here we show the four main areas of work presented in figure 1.1 and their progress if compared with three different criteria: establishment of the problem foundation and formulation of the approach proposed (column Foundation), application of this approach to the case study e-Stockings (column Application) and generalization of the approach so it can be applied in other problems (column Generalization). Modelling of energy consumption in mechatronic systems is considered as a covered area by this PhD work including laying the foundation and applying the methodology proposed in the case study. Generalization of this is not considered since this work is applying

already existing techniques that are already generic (shown as Not Applicable). The foundation of modelling of energy consumption in computation is defined and at the moment its application to the case study and generalization of this technique is in progress. We have laid the foundation for modelling and HIL and at the moment we are applying this to the case study. The study of energy consumption in communication from the modelling perspective remains at the moment as future work. In table 6.1 we also show the planned publications to disseminate the research results during the remainder of this PhD project. Additional details on these planned publications (shown as [p#]) are provided in Appendix A.

Table 6.1: Progress overview showing complete (✓), in progress (P) and future (F) work.

| Area | Foundation | Application | Generalization |
|---|---|---|---|
| Energy consumption in mechatronic systems | ✓ | ✓ | N/A |
| Energy consumption in computation | ✓ | P [p1] | P [p4] |
| HIL and measurements | ✓ | P [p3] | F [p4] |
| Energy consumption in communication | F [p2] | F [p2] | F [p4] |

## 6.4 Concluding remarks

This report has presented the progress, current state and future plans that we believe will lead to completion of this PhD by April 2015. This work has already partially expanded the applicability of the VDM-RT and DESTECS technologies to the system-level energy-aware design of cyber-physical systems. We hope to contribute in this direction further during the remainder of this PhD and end up with a solid case that argues and demonstrates with practical results how this approach can benefit the development of future embedded solutions today.

# Appendices

# Publications

This appendix presents the published material during the course of this PhD. Additionally it presents articles that have been submitted and that are currently under preparation.

## A.1 Related to this PhD project

**Published**

- José Antonio Esparza Isasa and Finn Overgaard Hansen and Peter Gorm Larsen. "Embedded Systems Energy Consumption Analysis through Co-modelling and Simulation" in *Proceedings of the International Conference on Modelling and Simulation ICMS 2013*, 2013, June.
- José Antonio Esparza Isasa and Peter Gorm Larsen. "Modelling Different CPU Power States in VDM-RT" in *Proceedings of the 11th Overture Workshop* School of Computing Science, Newcastle University, 2013.

**Submitted**

- José Antonio Esparza Isasa and Peter Würtz Jørgensen and Peter Gorm Larsen. "Hardware In the Loop for VDM-Real Time Modelling of Embedded Systems" Submitted for publication in *Second International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2014* 2014, January.

**Under preparation**

- "ICT-Enabled Compression Stocking for Treatment and Supervision of Leg Venous Insufficiency". To be submitted to *7th International Conference on Biomedical Electronics and Devices, BIODEVICES 2014*, part of the BIOSTEC joint conference.
- "ICT Architecture for Calibration and Supervision of Treatment Quality in Personal Healthcare Systems" . To be submitted to *5th International Conference on Bioinformatics Models, Methods and Algorithms, BIOINFORMATICS 2014*, part of the BIOSTEC joint conference.

**Planned publications**

- p1. "Modelling Energy Consumption of Multiple CPU States in VDM-RT", conference paper.

- p2. "Modelling Energy Consumption of Small-Scale Embedded Systems Communication Networks with VDM-RT", workshop or conference paper.
- p3. "Hardware In the Loop for VDM-RT Models - System Implementation Co-execution", conference paper.
- p4. "System-Level Energy-Aware Design of Cyber-Pysical Systems", target journal submission.

## A.2 Not related to this PhD project

- José A. Esparza and Peter Gorm Larsen and Kim Bjerge, "Supporting the Partitioning Process in Hardware/Software Co-Design in VDM-RT" in *Proceedings of the 10th Overture Workshop 2012* School of Computing Science, Newcastle University, 2012.

B

# Courses completed

This appendix presents the courses that have been completed during the first half of the PhD programme. The courses fulfil the required 30 ECTS by Graduate School of Science and Technology (GSST), Aarhus University. The courses have been completed across three different countries and four different universities. This selection presents a variety in course types and complements the PhD studies from different angles.

**Telecommunications DTU Summer University - 5 ECTS:** International summer school targeting students from electronics, telecommunications and computer engineering disciplines. During this summer school we were introduced to communication modelling tools and fiber optics. Summer 2013 *Department of Photonics, Technical University of Denmark (DTU).*

**R&D project activity - 5 ECTS:** Research activity in which we developed an interface to an experimental capacitive-based pressure sensor and characterized it. The main focus areas of the project were analog interfacing and sensor technology. Spring 2013 *Department of Engineering, Aarhus University.*

**The world of research - 2 ECTS:** Transferable skills course in which we learn about general aspects of research, such as science and politics, scientific practice and funding. Spring 2013 *GSST, Aarhus University.*

**Project management - 5 ECTS:** Transferable skills course in which we were introduced to project management from a managerial perspective. Spring 2013 *GSST - Aarhus School of Business, Aarhus University.*

**Scientific writing and communication - 3 ECTS:** Transferable skills course in which we learnt how to communicate scientific results in different media. Spring 2013 *GSST, Aarhus University.*

**DESTECS summer school - 5 ECTS:** International multidisciplinary summer school in which we learnt the basics of the DESTECS co-modelling approach to cyber-physical systems design. Summer 2012 *Robotics and Mechatronics, University of Twente.*

**Innovation and Creativity for Complex Engineering activities - 5 ECTS:** International multidisciplinary summer school in solving complex industrial problems by applying a scientific approach and innovation skills. Summer 2013 *Department of Informatics, Universidade do Minho.*

C

# Gantt diagram

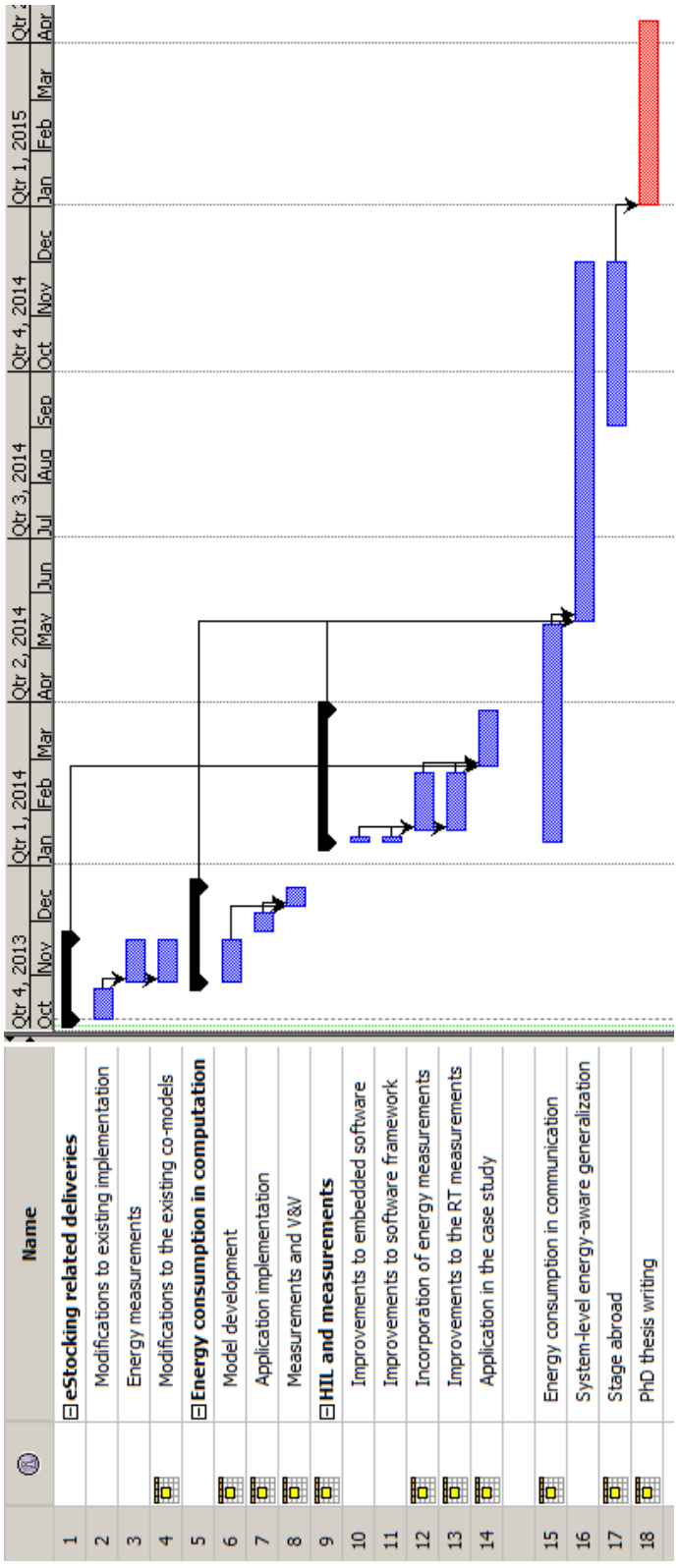


Figure C.1: Gantt diagram showing the plan for the remainder of this PhD.

# Bibliography


[1] J. Fitzgerald, K. Pierce, and P. G. Larsen, *Industry and Research Perspectives on Embedded System Design*, ch. Collaborative Development of Dependable Cyber-Physical Systems by Co-modelling and Co-simulation. 701 E. Chocolate Ave. Hershey, PA 17033, USA: IGI Global, 2014.

[2] Thomas A. Henzinger and Joseph Sifakis, "The Embedded Systems Design Challenge," in *FM 2006: Formal Methods, 14th International Symposium on Formal Methods, Hamilton, Canada, August 21-27, 2006, Proceedings*, pp. 1–15, 2006.

[3] Banerjee, A. and Venkatasubramanian, K.K. and Mukherjee, T. and Gupta, S. K S, "Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber-Physical Systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 283–299, 2012.

[4] S. K. Gupta, T. Mukherjee, G. Varsamopoulos, and A. Banerjee, "Research Directions in Energy-Sustainable Cyber–Physical Systems," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 57 – 74, 2011.

[5] Unsal, O.S. and Koren, I., "System-Level Power-aware Design Techniques in Real-Time Systems," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1055–1069, 2003.

[6] M. R. Stan and K. Skadron, "Guest Editors' Introduction: Power-Aware Computing," *Computer*, vol. 36, pp. 35–38, December 2003.

[7] J. Shi, "A Survey of Cyber-Physical Systems," in *Proceedings of the International Conference on Wireless Communications and Signal Processing*, November 2011.

[8] Mostafa E.A. Ibrahim and Markus Rupp and Hossam A. H. Fahmy, "A Precise High-Level Power Consumption Model for Embedded Systems Software," *EURASIP Journal on Embedded Systems*, vol. Volume 2011, January 2011.

[9] Sheayun Lee and Andreas Ermedahl and Sang Lyul Min, "An Accurate Instruction-Level Energy Consumption Model for Embedded RISC Processors," 2001.

[10] Bassem Ouni and Cecile Belleudy and Eric Senn, "Accurate Energy Characterization of OS Services in Embedded Systems," *EURASIP Journal on Embedded Systems*, vol. 2012, no. 1, p. 6, 2012.

[11] Young-Hwan Park and Sudeep Pasricha and Fadi J. Kurdahi and Nikil Dutt, "A Multi-Granularity Power Modeling Methodology for Embedded Processors," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 4, pp. 668–681, 2011.

[12] Ozgur Celebican and Tajana Simunic Rosing and Vincent J. III Mooney, "Energy Estimation of Peripheral Devices in Embedded Systems," in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, GLSVLSI '04, 2004.

[13] A. Wang and C. Sodini, "A Simple Energy Model for Wireless Microsensor Transceivers," in *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, vol. 5, pp. 3205–3209 Vol.5, 2004.

[14] Q. Wang, M. Hempstead, and W. Yang, "A Realistic Power Consumption Model for Wireless Sensor Network Devices," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society*, vol. 1, pp. 286 –295, sept. 2006.

[15] M. Nistor, D. Lucani, and J. Barros, "A Total Energy Approach to Protocol Design in Coded Wireless Sensor Networks," in *Network Coding (NetCod), 2012 International Symposium on*, pp. 31–36, 2012.

[16] C. Park, K. Lahiri, and A. Raghunathan, "Battery Discharge Characteristics of Wireless Sensor Nodes: an Experimental Analysis," in *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, pp. 430–440, 2005.

[17] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power Awareness in Network Design and Routing," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. –, 2008.

[18] J. F. Broenink, P. G. Larsen, M. Verhoef, C. Kleijn, D. Jovanovic, K. Pierce, and W. F., "Design Support and Tooling for Dependable Embedded Control Software," in *Proceedings of Serene 2010 International Workshop on Software Engineering for Resilient Systems*, pp. 77–82, ACM, April 2010.

[19] D. Henriksson and H. Elmqvist, "Cyber-Physical Systems Modeling and Simulation with Modelica," in *Proceedings of the 8th International Modelica Conference*, 2011.

[20] F. Zhang, Z. Shi, and W. Wolf, "A Dynamic Battery Model for Co-design in Cyber-Physical Systems," in *Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems Workshops*, ICDCSW '09, (Washington, DC, USA), pp. 51–56, IEEE Computer Society, 2009.

[21] R. Rao, S. Vrudhula, and D. N. Rakhmatov, "Battery Modeling for Energy-Aware System Design," *Computer*, vol. 36, pp. 77–87, Dec. 2003.

[22] Texas Instruments, "CMOS Power Consumption and Cpd Calculation," tech. rep., June 1997.

[23] Cypress, "Programmable System-on-Chip PSoC 5LP: CY8C58LP Family Datasheet," Tech. Rep. 001-84932, March 2013.

[24] J. S. Fitzgerald, P. G. Larsen, and M. Verhoef, "Vienna Development Method," *Wiley Encyclopedia of Computer Science and Engineering*, 2008. edited by Benjamin Wah, John Wiley & Sons, Inc.

[25] P. G. Larsen, N. Battle, M. Ferreira, J. Fitzgerald, K. Lausdahl, and M. Verhoef, "The Overture Initiative – Integrating Tools for VDM," *SIGSOFT Softw. Eng. Notes*, vol. 35, pp. 1–6, January 2010.

[26] Controllab products, "http://www.20sim.com/," January 2013. 20-Sim official website.

[27] C. Kleijn, *20-sim 4.1 Reference Manual*. Enschede: Controllab Products B.V., First ed., 2009. ISBN 978-90-79499-05-2.

[28] J. F. Broenink and Y. Ni, "Model-Driven Robot-Software Design using Integrated Models and Co-Simulation," in *Proceedings of SAMOS XII* (J. McAllister and S. Bhattacharyya, eds.), pp. 339 – 344, jul 2012.

[29] J. A. E. Isasa, "A VDM-RT Methodology for the Hardware/Software Co-design of Embedded Systems," Master's thesis, Aarhus University School of Engineering, Finlandsgade 22, 8200 Aarhus, Denmark, December 2011. Supervised by Prof. Peter Gorm Larsen. Available on-line at http://overture.sourceforge.net/docs/Esparza11.pdf.

[30] J. A. E. Isasa, P. G. Larsen, and K. Bjerge, "Supporting the Partitioning Process in Hardware/Software Co-design with VDM-RT," in *Proceedings of the 10th Overture Workshop 2012*, School of Computing Science, Newcastle University, 2012.

[31] I. J. Grady Booch and J. Rumbaugh, *The Unified Modelling Language User Guide*. Addison-Wesley, 1999.

[32] R. S. Sandford Friedenthal, Alan Moore, *A Practical Guide to SysML*. Friendenthal, Sanford: Morgan Kaufman OMG Press, First ed., 2008. ISBN 978-0-12-374379-4.

[33] F. O. Hansen, T. F. Jensen, and J. A. Esparza, "ICT Architecture for Calibration and Supervision of Treatment Quality in Home Healthcare Systems.," in *International Conference on Biomedical Electronics and Devices 2014. Part of the BIOSTEC conference. Manuscript under preparation*, March 2014.

[34] F. O. Hansen, T. F. Jensen, and J. A. Esparza, "ICT-Enabled Compression Stocking for Treatment and Supervision of Leg-Venous Insufficiency.," in *International Conference on Bioinformatic Models, Methods and Algorithms 2014. Part of the BIOSTEC conference. Manuscript under preparation*, March 2014.

[35] J. A. E. Isasa, F. O. Hansen, and P. G. Larsen, "Embedded Systems Energy Consumption Analysis Through Co-modelling and Simulation," in *Proceedings of the International Conference on Modeling and Simulation, ICMS 2013*, World Academy of Science, Engineering and Technology, June 2013.

[36] Tim Wescot, "PID Without a PhD," *Embedded Systems Programming*, October 2000.

[37] J. A. E. Isasa and P. G. Larsen, "Modelling Different CPU Power States in VDM-RT," in *Proceedings of the 11th Overture Workshop 2013*, School of Computing Science, Newcastle University, June 2013.

[38] R. E.Gamma, R.Helm and J.Vlissides, *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional Computing Series, Addison-Wesley Publishing Company, 1995.

[39] J. A. E. Isasa, P. W. V. Jørgensen, and P. G. Larsen, "Hardware In the Loop for VDM-Real Time Modelling of Embedded Systems," in *MODELSWARD 2014, Submitted for publication in Second International Conference on Model-Driven Engineering and Software Development*, January 2014.

[40] Cypress, "http://www.cypress.com/," July 2013. Cypress official website.

[41] J. S. Fitzgerald and P. G. Larsen, "Triumphs and Challenges for the Industrial Application of Model-Oriented Formal Methods," in *Proc. 2nd Intl. Symp. on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2007)* (T. Margaria, A. Philippou, and B. Steffen, eds.), 2007. Also Technical Report CS-TR-999, School of Computing Science, Newcastle University.

[42] IEEE, "IEEE Standard Hardware Description Language Based on the Verilog(R) Hardware Description Language," *IEEE Std 1364-1995*, 1996.

# José Antonio Esparza Isasa, System-Level Energy-Aware Design of Cyber-Physical Systems, 2013

**Department of Engineering**
Aarhus University
Edison, Finlandsgade 22
8200 Aarhus N
Denmark

Tel.: +45 4189 3000