- DRAFT -

Co-Design at Chess-iT

by

G. Bosman

Supervisors:

Dr. Ir. A. M. Bos (Chess-iT)

Ing. P. G. C. Eussen (Chess-iT)

Dr. Ing. R. Lämmel (Vrije Universiteit)

vrije Universiteit *amsterdam*

Contents

Chapter 1

Introduction

A very important choice to be made when designing an embedded computer system is how to divide the functionality of the system in hardware and software parts. Traditionally the choice what to implement in hardware and what to implement in software is made early in the design process. Both parts are then made in separate tracks by separate designers. Goal of co-design is to explore the whole design-space to be able to make well-informed decisions and to be able to make this decision in a later phase in the design process.

There is a wide range of architectures and design methods so the hardware/software co-design problem is treated in many different ways.

Chapter 2

Assignment

2.1 Development methods

In my assignment I will select 3 development methods. This selection will be based on:

- Complete coverage of the process from design to implementation,
- Having a representative candidate of each class of developments methods,
- Expectations on which language is the most likely candidate to become an industry standard in the field.

These methods will be chosen in close coordination with my internship supervisors. A refinement will have to be made for some languages. For example UML is such a broad language that a description of the parts of UML used has to be given.

I will investigate the relevance and relationships between these languages in the track from specification to implementation. I'll research to which degree existing development and implementation methods fit into a single paradigm.

2.2 Scenarios

I'll carry out two small experiments using these methods, to get a good understanding of the problem of distribution. These experiments will be based on existing problems Chess faces and should have different 'natural' distributions of hardware and software. The following two projects have been given by Chess:

Table 2.1: 3 methods, 2 scenarios.

	Method 1	Method 2	Method 3
DSP chip	A1	A2	A3
Fuel processing system controller	B1	B3	B3

- Design and implement a DSP chip that compresses or encrypts information (streaming). This is an example of a **transformation system**.
- Design and implement a controller that is used in a fuel processing system to send information back to the gas-company. This is a **reactive system**.

2.3 Methods vs. scenarios

The methods and scenarios can be seen in Table 2.1.

For each combination I will try to do the whole process of specification to an (abstract) implementation in hardware and software. The hardware will be made using a FPGA.

Chapter 3

System Level Modelling

Mooney et al.[9]: Approaches to hardware/software co-design of embedded systems can be differentiated in several ways. One way is to consider the system-level specification, which is either homogeneous (i.e., in a single specification language) or heterogeneous (i.,e. involving multiple modelling paradigms). Another way is to distinguish how the CAD tool partitions the system specification: fine-grained or coarse-grained.

3.1 Homogenous modelling

Homogeneous modelling implies the use of single specification language for the modelling of the overall system. Co-design starts with a global specification given in a single language. This specification may be independent of the future implementation and the partitioning of the system into hardware and software parts. In this case co-design includes a partitioning step aimed to split this initial model into hardware and software. The outcome is an architecture made of hardware processors and software processors. The is generally called a virtual prototype and may be given in a single language or different languages. [8]

3.2 Heterogeneous modelling

Heterogeneous modelling allows the use of specific language s for the hardware and software parts. The co-design starts with a virtual prototype where the hardware/software partitioning is already made. Here the emphasis is on the integral designing of the parts to make sure the overall system has the required properties. The key issues are validation and interfacing [8]. A lot of research is done on the integration of different system parts tat enables system optimization across language boundaries. [this sentence from the SPI Workbench].

3.3 Computational Models

Very often real-time systems are specified by means of concurrent processes, which communicate asynchronously. [7]This model serves as a good implementation model.

The synchrony hypothesis forms the base for the family of synchronous languages. It assumes, that the outputs of a systems are synchronized with the system inputs, while the reaction of the system takes no observable time. The synchrony hypothesis abstracts from physical time and serves as a base of a mathematical formalism. All synchronous languages are defined formally and system models are deterministic.

Chapter 4

Co-Design methods

In this thesis we will deal with homogeneous modelling. Several models are available. There are a number of authors who made an overview of various development methods, i.e. [11], [8].

Traditionally hardware is designed in Verilog or VHDL. Many methods for Co-Design focus on generating C for the software part and VHDL for the hardware part.

A flat representation of a realistic embedded system would be too complicated to understand in many situations. A layered approach is therefore mandatory.

An overview of the more commonly used methods (or design-languages):

- System C
- Cosyma
- VULCAN (Hardware C)
- Polis (Esterel)
- Spec-syn (Spec-Charts)
- SDL
- Haskell
- HandelC

4.1 System C

The SystemC language is becoming a new standard in the field and many designers are starting to use it to model complex systems. SystemC has been mainly adopted to define abstract models of hardware/software components, since they can be easily integrated for rapid prototyping. However it can also be used to describe modules at a higher level of detail, E.g., RT-level hardware descriptions and assembly software modules. Thus it would be possible to imagine a SystemC-based design flow where the system description is translated from on e abstraction level to the following one by always using SystemC representations.[1].

4.2 VULCAN/HardwareC

The input to the co-synthesis system is described in hardware description language called HardwareC, a subset of C defined for hardware description. The architecture is limited to a one-CPU-one-bus architecture. Type of the CPU must be a general purpose register machine with one-level memory. [11]

4.3 Polis/Esterel

The Polis system developed at the university of California, Berekely, implements a HW/SW Co-Desgin system using the CSFM model as its basis. The textual language used in the Polis project is called Esterel. There is work in progress on a system that uses Statem-ateTM, a graphical specification tool, to create specifications for the Polis system[6].

CSFM stands for 'Codesign Finite State Machine'. Is focussed on reactive systems or control oriented applications[7]. Esterel is a synchronous language.

4.4 SDL

SDL (specification and description language) is intended for the modelling and simulation of real-time, distributed and telecommunication systems and is standardized by the ITU. A system described in SDL is regarded as a set of concurrent processes that communicate width each other using signals. SDL supports different concepts for describing systems such as structure, behavior and communication. SDL is intended for describing large designs at the system level. There are two SDL formats, a textual and a graphical one.[3] In [3] a system is defined that allows the generation of VHDL from system level specifications in SDL.

In [2] a case-study is presented that uses SDL modelling to generate a VHDL description. It is then implemented in a FPGA.

4.5 Haskell

A Haskell program is a function, which cosists of a composition of other functions. Functions produce only one result, althought this results can be a tuple consisting of values multiple data-types.[7]. I.e. there are no side-effects.

Chapter 5

Scenarios

5.1 DSP

I'll select an typical example of a Digital Signal Processing system here. Currently we're thinking of an encrypting or a compressing device.

5.2 Embedded controller

In Chess there has just been released a demo-version of a product that is a controller for a heating system with remote control features.

5.3 Selection of method/scenario combinations

Chapter 6

Method A

6.1 Implementing a DSP

6.2 Implementing an Embedded Controller

Chapter 7

Method B

7.1 Implementing an Embedded Controller

Chapter 8

Method C

8.1 Implementing a DSP

8.2 Redefining a DSP, different HW/SW configuration

Chapter 9

Conclusions

Bibliography

[1] A. Fin, F. Fummi, M. Martignano, and M. Signoretto. SystemC: A homogenous environment to test embedded systems. In Proceedings of the Ninth International Symposium on Hardware/Software Codesign (CODES-01), pages 17–22. ACMPress, Apr. 2001.

[2] C. A. Marcon, F. P. Hessel, A. M. Amory, L. H. L. Ries, F. G. Moraes, and N. L. V. Calazans. Prototyping of embedded digital systems from sdl language: a case study. In Proc. Seventh Annual IEEE International Workshop on High Level Design Validation and Test, 2002. To appear.

[3] J.-M. Daveau, G. F. Marchioro, and A. A. Jerraya. VHDL generation from SDL specification. In C. Delgado Kloos and E. Cerny, editors, Hardware Description Languages and their Applications (CHDL '97), Toledo, Spain, Apr. 1997. IFIP WG 10.5, Chapman and Hall.

[4] F. Slomka, M. Dörfel, and R. Münzenberger. Generating mixed hardware/software systems from SDL specifications. In Proceedings of the Ninth International Symposium on Hardware/Software Codesign (CODES-01), pages 116–121. ACMPress, Apr. 2001.

[5] D. Gajski and F. Vahid. Specification and design of embedded software-hardware systems. IEEE Design & Test of Computers, 12(1), 1995.

[6] I. D. Bates, E. G. Chester, and D. J. Kinniment. A statechard based HW/SW codesign system. In Proceedings of the Seventh International Workshop on Hardware/Software Codesign (CODES-99), pages 162–166. ACMPress, May 1999.

[7] I. Sander and A. Jantsch. System synthesis utilizing a layered functional model. In Proceedings of the Seventh International Workshop on Hardware/Software Codesign (CODES-99), pages 136–140. ACMPress, May 1999.

[8] A. A. Jerraya, M. Romdhani, P. L. Marrec, F. Hessel, P. Coste, C. Valderrama, G. F. Marchioro, J. M. Daveau, and N.-E. Zergainoh. Multilanguage Specification for System Design and Codesign. Kluwer academic Publishers, 1999.

[9] V. J. Mooney III and G. De Micheli. Real time analysis and priority scheduler generation for hardware-software systems with a synthesized run-time system. In Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, pages 605–612. IEEE Computer Society, 1997.

[10] D. E. Thomas, J. K. Adams, and H. Schmit. A model and methodology for hardware-software codesign. IEEE Design & Test of Computers, Sept. 1993.

[11] T.-Y. Yen and W. Wolf. Hardware-Software Co-Synthesis of Distributed Embedded Systems. Kluwer Academic Publishers, 1996.