Chapter 10

UML-EXECUTABLE FUNCTIONAL MODELS OF ELECTRONIC SYSTEMS IN THE VIPERS VIRTUAL PROTOTYPING METHODOLOGY

P.F. Lister, V. Trignano, M.C. Bassett and P.L. Watten
Centre of VLSI and Computer Graphics,
University of Sussex,
Brighton, BN1 9QH, UK
Tel: +44 1273 678050
Fax: +44 1273 678030
P.F.Lister@sussex.ac.uk

Abstract

This paper presents the use of UML-Executable Functional Models (UML-EFM) in the context of the ViPERS virtual prototyping methodology [Lister et al., 2004a, Lister et al., 2004b] for System-on-Chip design. The concepts, the implementation and the experiments presented in this paper were developed at the University of Sussex (UoS) in the Centre of VLSI and Computer Graphics as part of an EU project [VIPERS]. The ViPERS methodology and its employment of the executable functional models have been developed to face the contemporary challenges of System-On-Chips by integrating key design methodologies with the graphical and interactive features of virtual prototyping. The fast evolution in silicon technology and its consequences on the market of hand held electronic products, is making the adoption of new design methodologies mandatory, with modern techniques for the design, development and manufacturing of consumer electronics. Executable functional models provide a means to simulate the target device in different phases of the design flow and analyse its requirements (behaviours, interfaces, etc), architecture (HW/SW partitioning) and finally its digital implementation. A key contribution includes the combination of an interactive 2D photorealistic model with its functional executable model implemented as a UML state machine; the experiment is applied to an RF home-based remote control used to control a cooking stack.

P. Boulet (ed.) Advances in Design and Specification Languages for SoCs, 161–178.

Keywords: Virtual Prototyping; UML; SystemC; executable specification; handheld devices; SoC modelling; ViPERS methodology.

1. Introduction

A close look at the market of consumer electronics reveals that nowadays a significant slice of it is occupied by hand-held devices. The rapid advance in silicon technology is enabling a substantial increase in the number of transistors per chip [ITRS, 2003]. This growth in complexity is parallel to other phenomena, for example the shortened time to market, and the high competition among manufacturing companies. Designers and engineers are therefore facing the dilemma of having to produce highly technological and complex systems in a limited time. To reduce the gap between complexity and time to market new design methodologies are being proposed. The ViPERS methodology links key trends in SoC design with modern interactive and graphical features of virtual prototyping. At the heart of this methodology is the desire to test virtual prototypes of electronic products at different stages of the design, development and manufacturing processes.

The first step in the ViPERS methodology is the analysis phase and the consequent derivation of an UML-executable functional specification. It is clear from the research in the field of requirements and specification development [RUP] that the specification work is unlikely to be confined to the period before implementation begins. Determining accurate product requirements and specifications is a vital stage in the development of a commercially viable device and executable functional models can help extend the value and meaning of the requirements and specification phase to further ensure the validity of this work prior to implementation [Kimura and Verlag, 2002]. Hence there is a need to rapidly feed changes in the requirements into the implementation tool chain in an evolutionary way. It is common for a design house to be given a written specification for a prototype device. Often the specification is not complete enough for the first resultant prototype to be satisfactory to the client, resulting in some design iterations. If the design house were to build a virtual prototype or even several alternative schemes, the client can clarify the functional specification before any hardware or software is built. The virtual prototype is a form of communication and reference in addition to the functional specification and any other requirements of the design [Preece et al., 2002]. A key aspect being highlighted is to ensure that effort spent in the early product definition phase should be reused as much as possible in the later implementation and test of the device. Several methods of requirements gathering have been explored including traditional written reports and UML based tooling. UML [OMG, 2003] provides the means to document detailed requirements which can lead, with the aid of software tools such as Rational Rose RealTime and

the use of state machines, to the production of the first behavioural model of the electronic device. Rational Rose RealTime is built on the UML-RT profile [Selic and Rumbaugh, 1998], which, due to its limited architecture and performance modelling capabilities, should be considered complimentary to the UML Profile for Schedulability, Performance and Time [OMG, 2002] (also called the Real-Time UML Profile) standardised by the Object Management Group (OMG). Rational Rose RealTime was chosen upon other UML real-time software tools because of the intention by Rational Rose to implement a SystemC profile [Sardini, 2002], which would simplify the route to hardware for the ViPERS methodology.

If the graphical model has been implemented at this stage then the requirements can be explored through the connection of the graphical model to its behavioural correspondent in UML as shown in Figure 1 and feedback from the stakeholders can be gathered.

Figure 1 shows the refinement steps related to the ViPERS methodology and the consequent creation of virtual prototypes that result from the four main phases (analysis, design, implementation and test). Virtual prototypes are distinguished based on which phase of the design flow they are generated from, and therefore which features of the target device they incorporate; each virtual prototype implements an EFM. The virtual prototypes are:

1 Functional Prototype; this is a product of the analysis phase, where the requirements and specification of the target product are analysed and defined.

2 Architectural Prototype; this is a product of the design phase, where architectural design takes place and hardware/software partitioning is defined.

3 Digital Prototype; this is a product of the implementation and test phases, where all the hardware and software blocks that constitute the target electronic device are implemented and then tested.

To explore the use of UML-EFMs in the context of the ViPERS methodology this paper presents the combination of an interactive 2D photorealistic model with its functional executable model implemented as a state machine in UML with the support of Rational Rose RealTime. The SxUMLSocket Package is employed to link the state machine of the EFM to its correspondent graphical model. Communication between the functional and the graphical model conforms to a XML-like communication protocol which defines the criteria that models need to be consistent with, in order to establish a connection with each other and communicate.

Traditional UML techniques are used to explore the requirements of the target electronic product. Once the system has been specified purely by means of

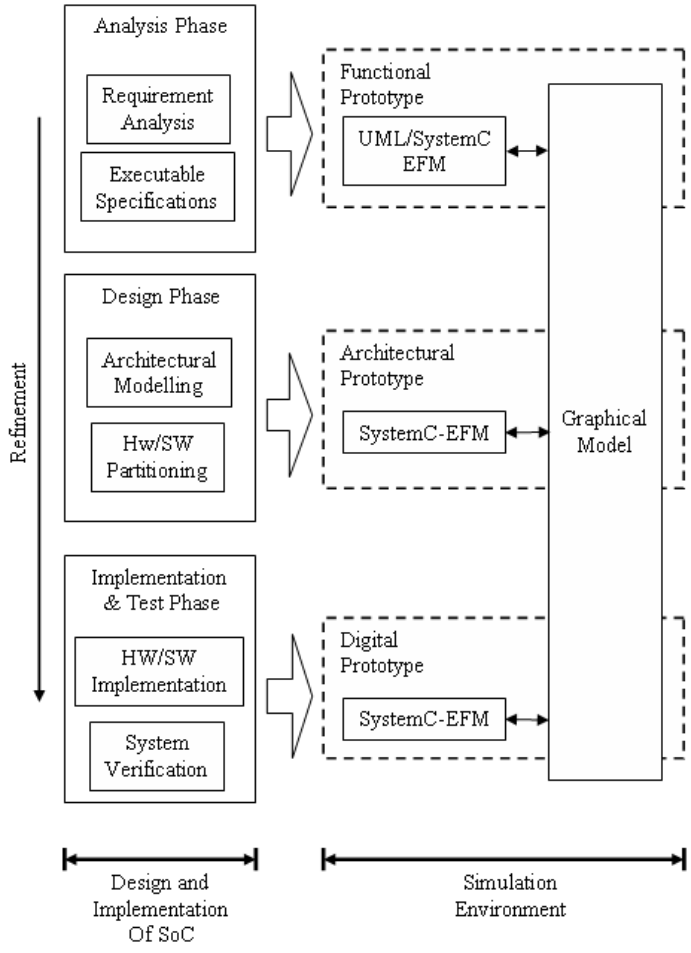

Figure 10.1. EFMs allocation in the ViPERS methodology.

UML diagrams (class, sequence, use-case, activity, etc), Rational Rose Real-Time is used to implement a state machine that describes the functionality of the system. The ViPERS methodology suggests the use of UML state machines for the creation of the functional model in this analysis phase. The model can optionally be described in SystemC [SystemC], using the freely downloadable toolkit or taking advantage of the graphical interface and automatic code generation that software tools as CoCentric System Studio from Synopsis offer.

The advantages offered by the use of UML-EFMs are clear when the simulation environment is running and the functional model is linked to the graphical model; the first analysis of the functionality of the target product can then commence. Designers as well as investors, hardware engineers as well as end-users with no technical background can test the usability of the product; interaction is achieved, for example, by pressing a button or moving a slider on the graphical model and viewing in real-time the changes on the display or other output means. The designer can also view the progress of the simulation through the graphical state machine at run time, and track the changes between states that result from the interaction with the graphical model implemented in VDM [Lister et al., 2004a] (virtual device model). VDM is an integrated development environment which enables designers to create photorealistic models of the target electronic product and define its interactivity through the scripting of the graphical objects that constitute the model.

The interaction of the executable functional model with the graphical model is a very effective approach to test features such as the graphical user interface and user interactivity issues of the device prior to any implementation. The two elements that make up the virtual prototype (graphical and functional model) can be packaged together in order to be distributed between stakeholders or end-users for feedback.

2. Executable Functional Model (EFM)

For the purpose of this paper we define the Executable Functional Models as descriptions of a number of properties associated with the functionality of an electronic system. They are independent executables which contain the means to communicate with an external application for simulation purposes. The nucleus of the ViPERS prototyping environment is VDM. Figure 1 shows that executable functional models connected to a VDM graphical model constitute the simulation environment of the ViPERS methodology.

The aim of the executable functional models is to simulate the behaviour of the virtual prototype (functional, architectural and digital) through the design and implementation stages. Based on a modified version of the ROPES process [Douglas, 1999], as shown in Figure 2, the ViPERS methodology with

its environment provides the specific tools, libraries, packages, and services needed to connect EFMs to the target graphical model.

EFMs are developed to test various features of the target electronic product; features include: Itemized lists:

- Functional properties,
- Graphical User Interfaces and User Interactivity properties,
- Architectural issues such as HW/SW partitioning,
- Digital properties, explored through the verification of the completed hardware/software implementation.

The ViPERS methodology currently provides support for two types of EFMs, these are:

1 UML-EFM, which use Rational Rose RealTime and Visual Studio version 6.0.

2 SystemC-EFM, which uses either:

   (a) The SystemC free-toolkit and Visual Studio version 6.0 (Windows-based),and

   (b) CoCentric System Studio from Synopsys (Linux-based)

The first type of SystemC-EFMs was demonstrated in [Lister et al., 2004b] and uses a SystemC-container application, developed by the ViPERS team, for the communication of the SystemC free toolkit with the VDM model. The second type uses a SystemC library for CoCentric, SxSockets Library [Trigano et al., 2003] and a Linux-based application server - the local communication control service (LCCS). Drawbacks and advantages of the different approaches in different stages of the methodology will be described in the following section of this paper.

The communication framework which allows real-time simulations provides a fast interaction mechanism between the functional and graphical model; the framework relies on the passing of small tagged messages. Further considerations on time-related issues for the simulation are explored later in this paper.

2.1 Executable Functional Models in UML (UML-EFM)

UML-EFMs are allocated in the analysis phase of the ViPERS methodology to take advantage of the many aspects that a description language such as UML. With its vast tooling support, it can bring to the definition of a system when details of the implementation have yet to be defined [Douglass, 1999]. The models are developed as state machines using Rational Rose RealTime,

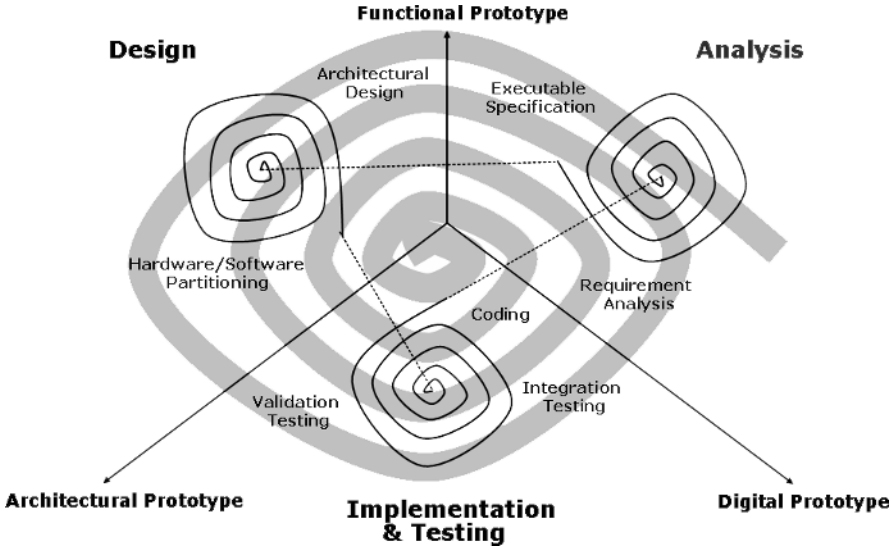

Figure 10.2. ViPERS modification of the ROPES process.

which provides a familiar framework for windows programmers. State machines are defined graphically and C++ code is added to the various states and transitions. The model created represents a very high level description of the system, the state machine in fact is supported by various diagrams (class, structure, sequence, etc), as well as communication protocols to communicate with other threads. UML-RT profile also introduces the use of capsules, which are differentiated from classes by their dynamic behaviour; capsules are active objects that represent system components, their internal behaviour is defined by state machines and they communicate with each other through stereotyped objects called ports which implement interfaces. The suggested approach for the development of a functional description is to create the various classes and diagrams which represent the initial high level description of the system. Once the first design of the system is achieved, it can be refined by adding attributes and operations that will be eventually used for the behavioural description of the model. A capsule is then developed to incorporate the classes and give dynamic support to the system; the behaviour is described with the use of state machines. The communication protocol of the capsule needs to be defined so that the designer can commence testing the capsule by injecting signals at simulation time. As soon as the model functions as expected the SxUMLSocket package is integrated into the system to provide the means to communicate with outside applications. The integration of the two capsules and the structure of a basic UML-EFM are shown in Figure 3.

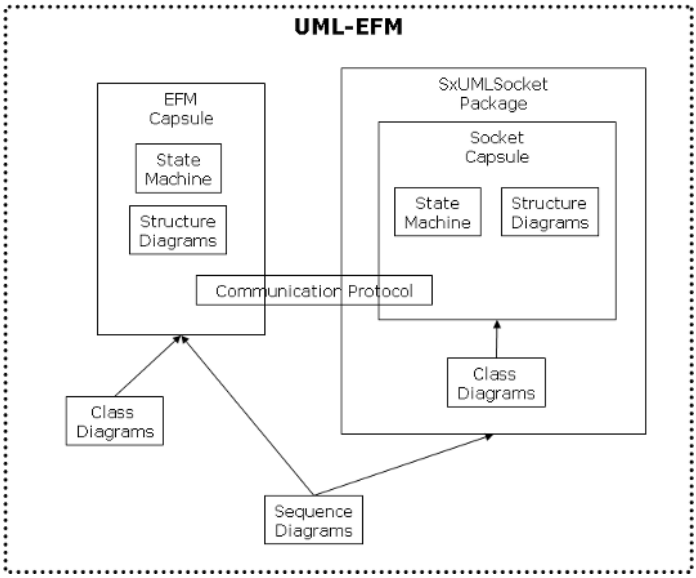

Figure 10.3. UML-EFM structure.

At this stage the functional model is an independent executable which is ready to communicate with the VDM graphical model. The state machine can be tested through enhanced visual means allowing users to interact with the graphical model of the electronic model and view how the state machine processes the input and output. The designer no longer needs to follow the simulation through a visual representation of the state machine, which requires a technical knowledge, but can actually interact with the virtual product as the end user would.

Changes on the display of the graphical model or other outputs that the model might possess are now the means by which the designer tests the functionality of the target product.

Some of the benefits of using a UML state machine are:

- Easy to implement
- Fast to simulate
- Can be used as an independent executable
- Parts of the state machine can be reused in later stages of the implementation

The power of this approach during the analysis phase of the methodology is substantial. UML-EFMs provide a very high level description of the target device, plus Rational Rose RealTime provides various tools to help the designer

in the development and debugging stages. Designers are provided with all the means to test the usability of the target system, graphical user interfaces and interactivity issues; the model can easily be modified or new ones can be created to test different possibilities or to fix unexpected behaviours. Ideally designers would develop a set of possible candidate solutions for the target product, from which one would be chosen as the final design. The life of UML-EFMs in the methodology is not over at this stage and there is a good possibility that the designer will need to come back to it when technical constraints or unexpected bugs make mandatory the redesign of certain features. The nature of the model provides fast and ease means to achieve this goal. UML-EFMs have another two possible major uses in the ViPERS methodology, these are:

1 Automatic translation to SystemC-EFM; issues related with the translation mechanism are outlined in the final section of this paper.

2 Reuse of state-machines parts in later stages of the design flow as embedded code; this is highly dependent on the implementation of the state machine of the EFM, however Rational Rose RealTime provides the means to create code for both platform-specific models (PSMs) and platform-independent models (PIMs).

SxUMLSocket Package. SxUMLSocket Package is a UML Rational Rose RealTime package that provides the classes, the communication protocol, and the capsule needed to link the state machine describing the behaviour of the system to an external application. The package was developed using Rational Rose RealTime and Visual Studio version 6.0, and the generated code is C++. The package includes the class diagram, the structure diagram, the sequence diagram, and the state diagram that visually describe static and dynamic features of the socket capsule.

Figure 4 shows three type of UML diagrams:

1 Class diagrams; which show the classes and their relationship, with relative attributes and operations, that are used in the socket capsule to implement the server and its functionality. The class diagram contains all the classes needed to support the socket capsule plus other classes to support new developments. The class diagram includes:

- Socket Capsule Stereotype. It is a stereotype capsule which represents the active object of the TCP/IP socket and therefore implements its behaviour.
- Sock Class. It is the base class for both the server and the client class.
- Server and Client Classes. They define the functions and attributes that are specific for the server and the client implementation.

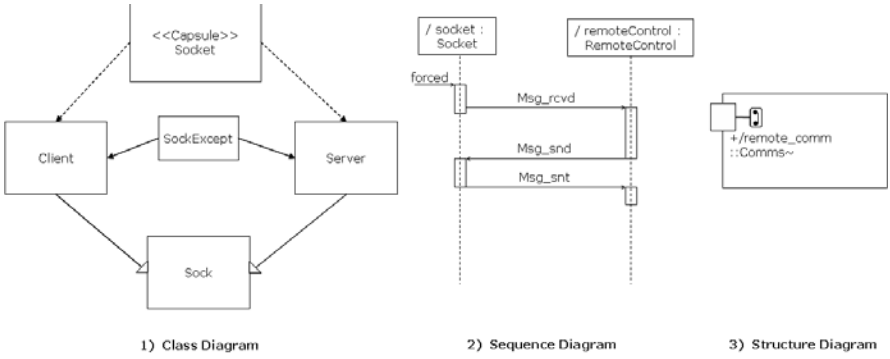

Figure 10.4. SxUMLSocket Package, sequence and structure diagrams.

- SockExcept Class. This class defines the exception cases for the socket.

2 Sequence diagrams; these show the sequence of events, by the passing of signals, between the socket capsule and the functional capsule (i.e. the capsule that describes the behaviour of the target device)

3 Structure diagrams; which show the structure of the socket capsule. The diagram shows the presence of a conjugate wired end port (*remotecomm*) which represents the means for this capsule to communicate with other capsules.

The socket capsule is implemented as a non-blocking server thread and the dynamic features of it are described in its state machine, visually shown in the state diagram of the capsule. Figure 5 shows that the state diagram consists of one state and two transitions; the "waiting for messages" state, the "initial" transition, and the "sending message" transition.

The operation of the socket capsule was purposely kept simple to allow designers to change features when needed. After the initial transition, determined by the initialisation of the socket capsule, the state machine enters the state and waits through a non-blocking receive function call for messages from the client model (VDM).

The non-blocking feature allows the capsule to perform other operations while waiting for the client to send a message. Once the socket capsule receives a message, it sends it to the EFM capsule to be processed by its state machine. The sate machine will output a signal carrying the message to be sent back to the VDM model, this signal will trigger the socket capsule to go through the "sending message" transition, which will force the socket capsule to send

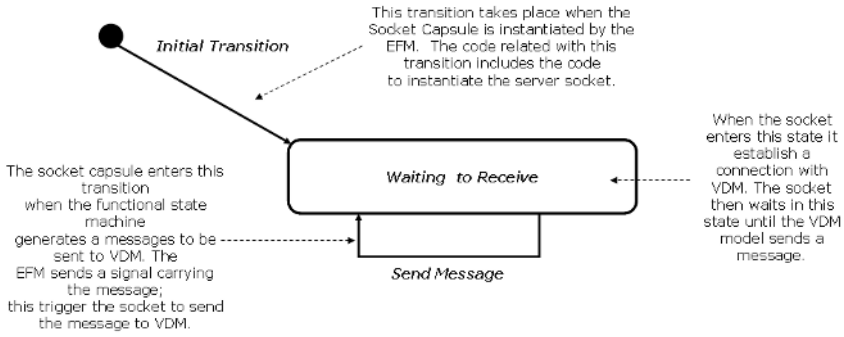

Figure 10.5. SxUMLSocket Package, state diagram.

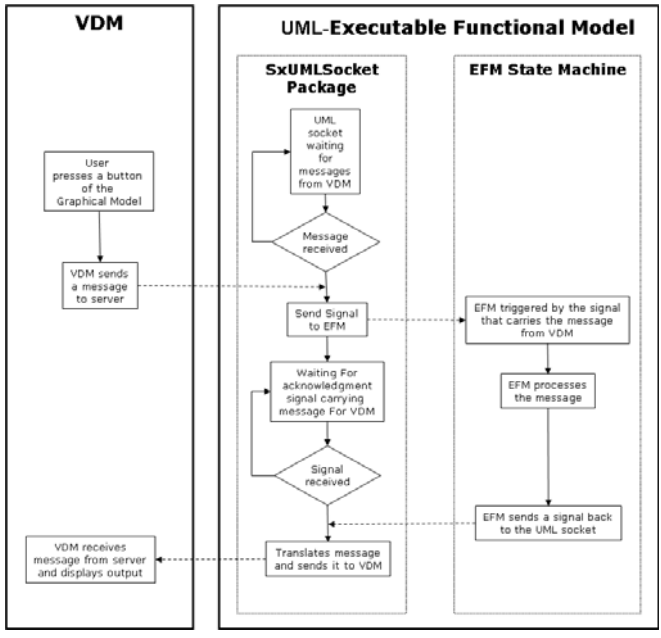

Figure 10.6. Interaction between UML-EFM and VDM.

the message to the graphical model. Details of the sequence of events and interaction that take place during a simulation between the two capsules (socket and EFM), and the VDM model are shown in Figure 6.

3. Comparing EFMs

This section compares the use of the UML-EFMs previously described, with the SystemC-EFMs in the context of the ViPERS virtual prototyping methodology. The aim of this comparison is to justify the specific allocation of each model in the methodology and to reason a suggested approach over another one in different phases. At present, the only SystemC-EFM that have been built and tested by the ViPERS team and therefore supported in the ViPERS prototyping environment are implemented in behavioural SystemC.

Previous work included the implementation of executable models in SystemC [Vanderperren et al., 2002] with the aid of UML diagrams. However, UML was not used to construct executable models, but only for architectural modelling and therefore to demonstrate that the design could meet the requirements. One of the drawbacks of that approach was that engineers needed a UML background in order to understand the design. In our approach the UML model serves two purposes; first it can be used to graphically describe the system, in which case a UML knowledge is needed, and secondly it can be used in conjunction with a graphical model to execute its functionality in a real time simulation. In the second case no knowledge is needed; the user simply interacts with a photorealistic representation of the target product and studies its behaviour.

UML-EFMs present substantial differences with SystemC-EFMs. The development and use of UML-EFMs is almost entirely devoted to the first phase of the methodology, the analysis, while the SystemC model can be refined to RTL level within the same environment. A SystemC model offers the advantage that it can be implemented at very different levels of abstraction, and each level can be allocated in the methodology as part of the refinement process. Another advantage (a consequence of the previous one) is the reusability of the SystemC-EFMs; these models can ideally be reused from their first high level implementation down to the timed (cycle accurate) models, by the refinement process shown in the design flow for SoCs in Figure 1. But UML-EFMs have two major advantages over SystemC-EFMs, which drove the ViPERS team to explore their use in the analysis phase. These advantages are:

1 Ease of implementation. UML-EFMs are very easy to implement and to modify which makes it a much better candidate when, in the analysis phase, designers need to quickly put together and test the device and possibly implement modified version of it.

2 Simulation speed. UML-EFMs provide real-time simulation speed, which can ideally be achieved only by a high level SystemC description; that would strongly depend on the framework used (free-toolkit, CoCentric, etc) and on the implementation style.

3 Industrial Standard. UML, as well as SystemC, descriptions conform to an industrial standard and therefore are not tied to any particular proprietary tool.

The ViPERS team encountered considerable speed limitation when trying to simulate a behavioural SystemC-EFM implemented in CoCentric; considering the issues related with the different platform (Linux), the model was similarly re-implemented using the SystemC free-toolkit and the container application [Lister et al., 2004b]. The new model performed better giving a near to real-time simulation performance, but still not comparable with the simulation speed of a corresponding UML-EFM.

From these comparison considerations we concluded that the best possible solution for the ViPERS prototyping methodology was to employ both models in different phases in order to maximise their qualities and minimise their weakness. However different routes from this are possible.

4. UML-EFMs and ViPERS Virtual Prototyping Methodology

The final aim of EFMs is to be used in the context of a graphical simulation, where the user does not need to know any detail about the underlying processing of the EFM. At this stage of the development the benefits that virtual prototyping can bring to standard SoC design methodology can be appreciated. In the experiment presented here we linked together a UML-EFM of an RF remote control to its graphical counterpart; the simulation with both VDM and Rational Rose RealTime are shown in Figure 7. The experiment started with a design team writing a requirement document for the remote control. The document was simply a written document that attempted to describe the functionality of the device and all the possible action the user could undertake. It was clear from the start that an imagination gap was created between the understanding of the reader and the conceiver; another phenomenon was the fact that some scenarios were accidentally missed by the author of the document when trying to imagine all the possible scenarios.

Taking into consideration the popularity of UML in the field of requirements and specifications we produced a UML-type description document of the system; the description was more detailed and some ambiguity that natural languages can easily introduce was eliminated but the imagination gap was still not bridged. As described earlier in this paper, the adoption of UML to describe a system has an advantage in the fact that it is an industry standard

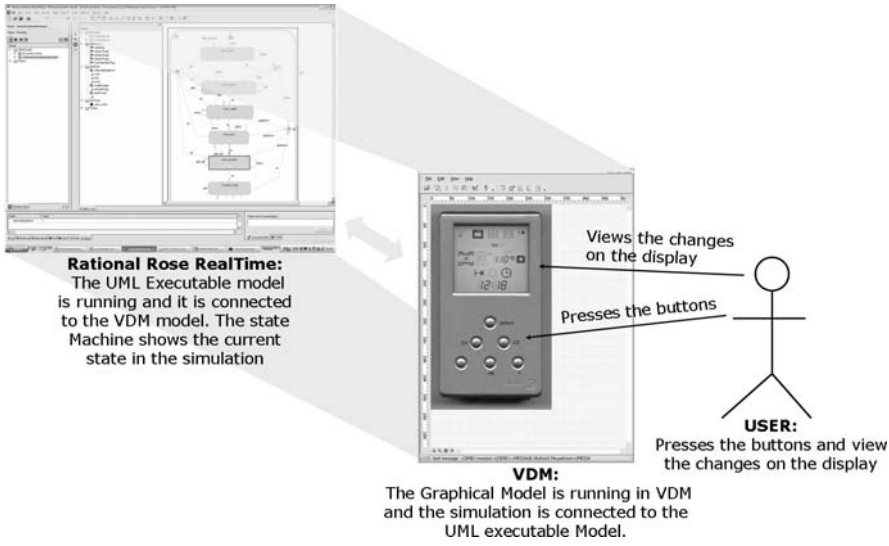

Figure 10.7. A Virtual Prototyping Experiment.

but a drawback since a UML knowledge is needed in order to understand the system. For this reason we decided to create a UML description that could be executed, so that the system could be seen running while interacting with a visual model, which represents the input/output means of the UML model. The UML description was then used as the basis for producing the UML-EFM of the device, while graphics designers were developing the photorealistic model of the remote control and determining its interactivity means.

The two models were then simulated, producing a fully functional virtual prototype, which is referred to as a 'functional prototype' in the context of the ViPERS methodology (to be distinguished from 'architectural and digital' prototypes shown in Figure 1). The team was able to test the behaviour of the device by interacting with the graphical prototype, pressing buttons and viewing the display updating. Considerations were made on the functionality and interaction means of the remote, and the imagination gap was eliminated. Figure 7 shows a use-case type diagram, where a user is interacting with the remote control. At this stage of the simulation anyone can test the functionality of the remote control since the mean is simply an interaction with the photorealistic model. Figure 7 also shows that the graphical representation of the remote control is linked to a state machine. The state diagram shown in the figure is running on Rational Rose RealTime and provides the functionality to the graphical model. Users with a knowledge of UML are also able to follow the simulation from the state changes and transitions in the state machine of the remote control.

The practical experiment illustrated in this paper was conducted at the University of Sussex. Both the functional and the graphical model were built and simulated on a windows-based PC with a 2GHz processor. The models communicate through TCP/IP sockets and therefore they can run on separate machines on a network to take advantage of the processing power; however the experiment showed that in this particular case this was not necessary due to the simple nature of the functional model.

5. Conclusions and Future Work

The experiment demonstrated the benefits that UML-EFMs can bring in the analysis phase of an electronic product. The advantages of using an industry standard such as UML were realised from the early stages of implementation. The standard diagrams provided the means for communication and understanding between the members of the team when trying to establish the requirements of the product. The major benefit though was introduced by the simulation of the EFM in conjunction with the VDM graphical model. The simulation environment provided all the means for the team to test the behaviour, the graphical interface and the input/output means of the target product. In the experiment only the final virtual prototype is showed. This was the result of many changes and iteration of both the graphical and the functional features of the initial prototype. Comparisons with the SystemC models enabled the team to establish the strength and the weakness of each approach and therefore locate their use in a specific stage of the methodology.

The requirement for virtual prototyping to become part of a wide ranging stakeholder evaluation process has driven the development of the ViPERS methodology. Early feedback from the application example suggests significant improvements can be made once stakeholder interaction is possible. The visual realism of the remote has allowed discussion beyond the engineering domain to more diverse stakeholders such as marketing and users. Although more rigour is required to evaluate the benefits, it seems the narrowing of the leap of imagination required to visualise the final product is a positive feature of this approach.

Inevitably the ViPERS environment contains similar features to existing virtual prototyping tools [Cybelius Software, Alita, RAPID], but our focus is on:

- The nature of the graphics (photorealism)
  - Alpha blending
  - Sub-pixel investigation
  - Special rendering
- Route to hardware

Our longer term research will be to apply the virtual prototyping of embedded systems to the entire system environment.

The ViPERS environment successfully demonstrated the viability of linking virtual prototyping to a SoC methodology. At present the linking can be applied at every level but simulation might fail to be in real-time for low level implementations. Further research will focus on the simulation of complex systems and RTL implementations, and therefore deal with concurrency, time constraints, scheduling and performance issues; emulation might be used to speed simulation with low level models.

The ViPERS environment and the use of UML-EFMs successfully increased the capability of separating user interfaces from behaviour; this allows testing different interfaces without having to modify the functional description of the prototype.

It is intended by the ViPERS team to further explore the link between UML and SystemC for platform based designs. Companies such as Rational are predicting an imminent development of a UML.SystemC profile [Sardini, 2002]. It is included in the future work the development and design of other EFMs at different abstraction levels. In particular the ViPERS team will implement Transaction Level models (TLM) and Register Transfer Level (RTL) models, perhaps with mixed HDL languages. TLMs will be produced to prove how a substantially detailed model can still produce fast simulation in the ViPERS environment, while RTL models will prove the link of the methodology down to synthesisable code.

Future work includes the integration of the simulation in virtual environments [Lister et al., 2002] in order to enhance the experience and obtain a more natural interaction of the user with pervasive electronic devices. Graphical models (2 and 3 dimensional) will be placed in their natural environments, where users will be able to navigate through and interact with the devices and view the changes not only through device outputs but also through the changes in the virtual environment (e.g. the light in the oven being switched on).

Acknowledgments

This work was funded as part of the ESPRIT framework 5 VIPERS project IST-2000-30023. We are grateful to our project colleagues for their constructive interaction, particularly Javier Mendigutxia of IKERLAN SA. The cooperation of our colleagues Teresa Riesgo and Eduardo de la Torre of Universidad Politécnica de Madrid and Sabastian Pantoja of Celestica Valencia is acknowledged. System TM is a trademark of the Open SystemC Initiative. CoCentric® System Studio is a registered trademark of Synopsys, Inc. Windows XP®, Visual Studio® and Visual Basic® are a registered trademark of Microsoft Cor-

poration. Rational® Rose Real Time Studio is a product of IBM® Kylix TM is a trademark of Borland® Software Corporation

References

Altia, Inc., `http://www.altia.com` [last accessed 29/07/04]

Cybelius Software, `http://www.cybelius.com` [last accessed 29/07/04]

Douglass, B.P. *Doing Hard Time: Developing Real-Time Systems using UML, Objects, Frameworks and Patterns*, Addison-Wesley, 0201498375, 1999.

Douglass, B.P.*Real-Time UML Second Edition, Developing Efficient Objects for Embedded Systems*, Addison-Wesley, 0201657848, 1999.

International Technology Roadmap for Semiconductors (ITRS), 2003 Edition.

Kimura, I. and Verlag, S. *Product Development with Mathematical Modeling, Rapid Prototyping, and Virtual Prototyping*, ISBN 3-8322-0896-8, Chapter 1, June 2002.

Lister, P.F. Newbury, P.F. Watten, P.L. Senkoro, L. Dountsis, A. Midha, M. Banerjee, I. Trignano, I. and White, M.*Virtual Reality in Electronic Systems*, Proceedings of 5th International Conference on Business Information Systems, Poznan, Poland, April 2002. Pp. 390-394.

Lister, P.F. Watten, P.L. Lewis, M.R. Newbury, P.F. White, M. Bassett,M.C. Jackson, B.J.C. and Trignano, V. *Electronic Simulation for Virtual Reality: Virtual Prototyping*, Theory and Practice of Computer Graphics 2004 (TPCG04), Southampton, UK, June 2004.

Lister, P.F. Watten, P.L. Newbury, P.F. Bassett, M.C. Jackson, B.J.C. and Trignano, V. *Virtual Reality for Electronic Product Development of Hand Held Devices*, Design Automation and Test in Europe (DATE'04), Paris, February 2004.

Object Management Group, *UML profile for Schedulability, Performance, and Time*, OMG document ptc/03-02-03, Needham MA, 2002.

Object Management Group, *Unified Modelling Language (UML)* – Version 1.5, OMG document formal/2003-03-01, Needham MA, 2003.

Open SystemC Initiative. See `http://www.systemc.org/` [last accessed 29/07/04]

Preece, J. Rogers, Y. and Sharp, H. *Interaction Design, beyond human-computer interaction*, John Wiley and Sons, Inc. ISBN 0-471-49278-7, 2002.

RAPID virtual prototyping tools, e-SIM LTD, `http://www.e-sim.com/` [last accessed 29/07/04]

Rational Unified Process® for Systems Engineering, `http://www.rational.com/` [last accessed 29/07/04]

Sardini, A. *SoC Design with UML and SystemC*, European SystemC, 6.Users Group Meeting, Lago Maggiore, October 2002.

Selic, B. and Rumbaugh, J. *Using UML for modelling Complex Real-Time Systems*, white paper, rational (Object Time), march 1998.

Trignano, V. Bassett, M.C. Watten, P.L. and Lister, P.F. *Extending SystemC for high-level multi-platform SoC simulations*, IEE Postgraduate Colloquium on System-on-Chip Design, Test and Technology, September 2, 2003, Cardiff University.

Vanderperren, Y. Sonck, G. van Oostende, P. Pauwels, M. Dehaene, W. and Moore, T. *A Design Methodology for the Development of a Complex System-on-Chip using UML and Executable System Models*, Forum on Specification and Design Languages (FDL'02), Marseille, France, September 2002.

VIPERS Project references and web pages, `http://www.upmdie.upm.es/projects/vipers/` [last accessed 29/06/04]