PRIMENA ML DESIGNER NA UNAPREĐENJE FUNKCIONALNE BEZBEDNOSTI MEDICINSKIH UREĐAJA

IMPROVING FUNCTIONAL SAFETY OF MEDICAL DEVICES BY USING MLDESIGNER

Volker Zerbe, *University of Applied Sciences Erfurt, Germany*
Sebastian Niller, *Ilmenau University of Technology, Germany*
Miroslav Božić, Darko Todorović, and Goran S. Đorđević,
University of Niš, Faculty of electronic engineering,
Aleksandra Medvedeva 14, P.O. Box 73, 18000 Niš, Serbia

Sadržaj – *Savremeni medicinski uređaji moraju da ispune stroge zahteve na bezbednost, pre svega zbog mogućnosti da povrede čoveka u svom okruženju.Na primer, akcija bezuslovnog prekida rada mora uređaj da dovede u bezbedno stanje za najkraće moguće vreme tako da pacijent ne bude povređen. Pored toga, takva akcija mora biti izvedena na potpuno pouzdan način. Uz pomoć okruženja kakvo je MLDesigner moguće je analizirati bezbednost sistema još u ranim fazama projektovanja. Ovakav pristup podrazumeva system orijentisan događajima kao i adekvatan domen projektovanja. Pogodnim primerima testiranja može se celokupan sistem testirati i analizirati. Razvijeni model potom ispunjavaće sve zahteve koji su postavljeni na početku projektovanja, održavši osnovnu funkciju – potpunu pouzdanost pri ostvarenju funkcije bezbednosti. U ovom postupku, prvo se razvija VHDL kod. Ovaj kod se može simulirati i testirati pre primene u samom sistemu.U ovom radu daje se primer dodataka MLDesigner-a, tako da se VHDL opis sistema bezbednosti može posmatrati u kontekstu modela celokupnog sistema. Integracijom specifikacija niskog nivoa složenosti i zahteva u izvršne apstraktne modele mogućnost greške može biti značajno umanjena tokom procesa projektovanja. Ova metodologija obezbeđuje značajna unapređenja funkcionalne bezbednosti. Sledeći ovaj tok procesa projektovanja razvili smo i testirali konkretan sistem bezbednosti jednog medicinskog uređaja.*

Abstract - *Medical devices are a subject of strict safety regulations due to possible damage to the patients and operators. For example, an emergency stop action has to guide the medical device into a safe state, so that the patient is protected from a collision. Such action must be performed with absolute certainty in function. With the help of the system design tool ML Designer the medical device as an overall system is modeled already in the early design phases. This approach concerns an event-oriented system thus asking for the DE-domain (DE - discrete event) modeling by MLDesigner. By suitable test scenarios the overall system can be simulated and validated. The developed model will have the specifications of the design and the implementation of the safety system. First we develop the VHDL code. This code can be again simulated and validated before the system is implemented. The paper also gives an extension of MLDesigner, so that now the VHDL description of the safety system is in the context of the modeled overall system. By integration of lower level specifications into executable abstract models the fault liability can be decreased significantly through complete design process. This methodology ensures substantial improvement of the functional safety of medical devices. Based on that design flow we developed and validated a safety board supervising the standard event-driven components to be used in medical devices.*

1. INTRODUCTION

Modern medical systems implement an increasing number of features at the same time of complying with rising requirements. The resulting complexity of a system is handled by integration of digital electronics and software to handle communication of subsystems. A product development demands collaboration and interaction of different engineering disciplines, all using their specific languages, notations, design methods and tools. A typical medical system consists of, but is not limited to, mechanical, electrical, electronic and software subsystems. The growing number of subsystems and the complexity caused by their interaction in a product makes it impossible to foresee all planned and unwanted effects (that could cause failures) by a development engineer focused on one field. Therefore, the simulation models have to be employed to analyze and optimize a design at each stage of a design flow. Engineers of different disciplines, to cooperate in design of heterogeneous systems, need a unified and comprehensive simulation of a system as a whole using a single environmental model. There are different ways to achieve this. One way is to use a single general simulator that provides special libraries or extensions for every field of engineering incorporated into the design. Another way would be to link several simulators customized to distinct fields of engineering to co-simulate a whole system. In this paper we present a link between the simulators MLDesigner and GHDL, to support a seamless design flow of digital systems, [1].

2. MLDESIGNER

"MLDesigner is a software system for the design of missions, systems, products, and chips", [2]. Systems are modeled by hierarchical block diagrams, connectivity and program source code forming an executable model (Fig. 1). Each block has a distinct type, defined by shape, input and output ports.

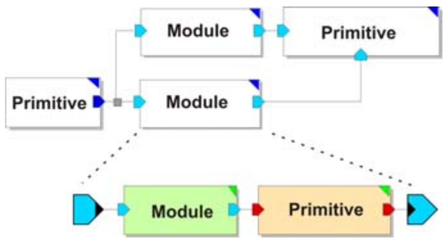

Fig. 1. MLDesigner modeling elements

A module is a block consisting of modeling elements and their connectivity. It is used to group elements related within a system, to name them, and to hide complexity. Module blocks can be used to build subsystems that are instanced multiple times. Modules must not contain instances of themselves directly or indirectly in most MLDesigner domains.

A primitive block contains the Ptolemy language source code to model functionalities. While primitives are usually not a composition of other modeling elements an MLDesigner system contrary to a primitive is not part of any other modeling block but contains all other elements of a design. MLDesigner systems act like test-benches of other simulation tools; they are an element necessary to start a simulation run.

Primitives and modules communicate to each other through ports on their boundaries. Ports are of type input, output or input/output and do have a specific data type to communicate over relations they connect through.

Another way of primitives interacting is by shared elements: memories, events and resources. Shared elements are described more in detail in [2].

Parameters are constant values to be set prior to the start of a simulation run and can be read by methods in the Ptolemy language source code of primitives.

Each system, module and primitive belongs to an MLDesigner domain. Selectable targets and schedulers of that domain manage execution (simulation or code generation) of the modeling element, thus are responsible for the simulation semantics of a model.

3. GHDL

GHDL (G Hardware Design Language, where G has no meaning) is a free VHDL (Very High Speed Integrated Circuit Hardware Description Language) simulator developed by Tristan Gingold [3] under the GPL (General Public License). It is a language frontend in GCC (Gnu Compiler Collection), programmed in Ada. GHDL analyses, elaborates and compiles an executable simulator out of VHDL input sources. The compiled simulator, when running, simulates the design and writes a trace file of its signal values. VHDL language features to input and output files and to print or report messages.

The front-end parse a VHDL source and generates an intermediate internal representation of the source which is then passed to the code generator backend of the GCC. GHDL features several methods to interface the compiled simulator, the simulator linking between MLDesigner and GHDL here has been implemented using the VPI interface that is stacked on top of GHDL's PLI/programming language interface AVHPI.

GHDL further is able to print sources with references as HTML easily.

4. VHDLS DOMAIN

In the following section the implementation of the VHDLS domain in MLDesigner is described. This domain communicates to a GHDL compiled simulator for designing digital circuits.

To represent the structure of a VHDL design in MLDesigner, a relation of structural elements in VHDL and structural elements in MLDesigner is proposed in Table 1.

A VHDL test-bench is an element of a design necessary to start a simulation. It is usually composed of the model under design and a model of its environment that generates inputs to the system and analyses outputs from the system. A VHDL test-bench corresponds to an MLDesigner system.

VHDL	**MLD VHDLS domain**	**MLD MML**
Test-bench	MLDesigner System	Model
Port	Port	Port
Component	Module	Entity
Component	Primitive	Primitive

Table 1: Mapping VHDL structure to MLDesigner (extract).

A port in a VHDL model is mapped to a port in an MLDesigner model. Ports in VHDL can be of type input (in), output (out) or they can be of bidirectional type (in-out). Only the input and output port types are mapped, bi-directional typed ports are currently unsupported. Typical uses of in-out ports are data busses of processors. An intermediate solution to interface in-out ports in the VHDLS domain is to direct the data of an in-out typed port inside a VHDL model to one input and one output port of a wrapping both, the entity and architecture.

Only the std_logic and std_logic_vector data types are supported on ports, they are mapped to MLDesigner ports of data type string. This limitation imposes no problem since these two types are used at the boundary of most VHDL designs to describe electronic values on pins of an IC. The std_logic_vector data type is also well suited for this task because it allows defining the bit order of its values.

MLDesigner distinct between modules and primitives. Modules are solely used to model structure of a design; primitives are used to describe the function. VHDL has no such distinction, components contain code to describe behavior and function, but may also contain further component instances connected to signals that footprint into the code (signals connected to a component instance can be read and driven by functional code of the enclosing component). VHDL component instances are mapped to MLDesigner modules.

Generic constants in VHDL can be mapped to MLDesigner parameters. The mapping of generic constants to parameters is not implemented yet.

VHDL signals and connections set up by the VHDL port map language construct are mapped to MLDesigner relations (not all VHDL language connection constructs can be converted to MLDesigner port relations at this time).

5. STRUCTURAL CONVERSION

Now it is described the methods developed to derive the structure of an existing VHDL design and to convert the derived structure to a hierarchical MLDesigner representation in the VHDLS domain.

Following is a list of terms used in VHDL and its grammar.

- A design in VHDL is described by one or more design files.
- Each design file contains one or more design units.
- A design unit can contain any number of entity declarations and architecture bodies (among other language constructions).

An entity declaration can contain a formal port clause in its entity header. Port names, types and data types of an entity to be used in the conversion are derived out of the formal port clause:

```
entity_declaration ::=
    entity identifier is
        entity_header
        entity_declarative_part
        [   begin
            entity_stateent_part    ]
    end   [   entity   ] [   entity_simple_name   ] ;

    entity_header ::=
        [   formal_generic_clause   ]
        [   formal_port_clause   ]
```

An architecture body has a name identifier, and refers to the entity it implements by its entity name. The architecture body contains an architecture declarative part and an architecture statement part. The architecture declarative part can have component declarations to be instantiated in the architecture statement part.

```
architecture_body ::=
    architecture identifier of entity_name is
        architecture_declarative_part
    begin
        architecture_statement_part
    end  [ architecture  ] [  architecture_simple_name ] ;

architecture_declarative_part ::=
    { block_declarative_item }

Block_declarative_item ::=
        subprogram_ declaration
    |   subprogram_body
    |   type_declaration
    |   subtype_declaration
```

A component declaration has a name identifier that refers to the name of its entity and a local port clause that has to redeclare ports of the formal port clause of its entity.

```
component_declaration ::=
    component identifier   [   is   ]
        [ local_generic_clause ]
        [ local_port_clause ]
    end component   [   component_simple_name ] ;
```

An architecture statement part can contain component instantiation statements that in turn contain port mappings from the instantiated components ports to ports of the instantiating entity or to signals declared locally in the architecture.

```
Architecture_statement_part ::=
    {   concurrent_statement   }

concurrent_statement ::=
        block_statement
    |   process_statement
    |   concurrent_procedure_call_statement
    |   concurrent_assertion_statement
    |   concurrent_signal_assignment_statement
    |   component_instantiation_statement
    |   generate_statement

component_instantiation_statement ::=
    instantiation_label :
        instantiated_unit
            [ generic_map_aspect ]
            [ port_map_aspect ] ;
```

Instead of developing a new parser to extract the structure of a VHDL design, the GHDL has been extended to export its internal VHDL structure representation, called IIR, to XML format after parsing.

An --xml command line switch has been added that outputs an XML representation of the design. A patch extending GHDL will be available at [4]. XML is a widespread language to describe and exchange structured data. Most of the today programming languages feature libraries to handle XML. By generating XML out of VHDL it is possible to use these high level programming languages to operate on VHDL designs without having to develop a parser in the first place. For example, if inner nodes of such an XML tree representation contain tags that represent parsed non-terminals, obtaining the names of all component instantiation statements of a distinct architecture in a design file becomes a simple operation. Prior to this work, XML representations of VHDL designs like HDML have been defined [5, 6].

The GHDL parser front-end supports sources of several VHDL standard revisions produces good error messages and is able to parse large input files. GHDL represents the model internally following the IIR defined in [7]. Many nodes in the IIR are named after terminals and non-terminals of the VHDL grammar.

Using the --xml command line switch, for each design file of a design GHDL produces one XML file that contains the nodes of the GHDL IIR after parsing.

6. GHDL XML TO MLDesigner MML CONVERSION

An MLDesigner model is described by a set of MML files that contain hierarchy, connectivity, parameters and a set of Ptolemy language files that contain the function of primitives. The MLDesigner MML file format is an XML dialect developed as a variant of the MoML file format in [8] and analyzed as part of [9]. To represent structure of a VHDL model in MLDesigner, the IIR XML files generated by GHDL as shown in the previous section are transformed to MML files. There are many ways for this task, XSLT being one prominent technique often used. This work uses the HXT [10] library to carry out the transformation. HXT is a Haskell library for the purpose of processing XML. Although it has support for XSLT, a more general way is employed for the conversion. First the XML output of GHDL is parsed by HXT resulting in an XmlTree DOM, which is a tree representation of the XML files with generic nodes.

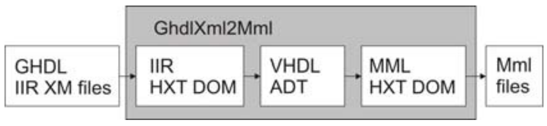

Fig. 2. XML to MML Conversion

A coupling of the SDF and/or the DE domain to the VHDLS domain has been developed. An important synchronous VHDL model appears for example as a SDF wormhole to a module of the VHDLS domain. To co-simulate and synchronize a VHDL model as part of a model in the SDF domain, the VHDLS target forks a process in its setup method using the UNIX fork system call, then replaces the new MLDesigner instance by the GHDL simulator using the Unix execl system call [11]. The location of the GHDL simulator is given by the GHDLSimulatorPath and GHDLSimulator target parameters.

7. MEDICAL DEVICES

So far the subject was the development of a VHDLS Domain for the tool MLDesigner. The VHDLS Domain takes over thereby the coupling between MLDesigner and the GHDL simulator.

To improve the functional safety of medical devices means:

- to implement a few number of emergency stop function, but also,
- to design a fault free system.

The only solution is, to develop the system model based. Means, first an overall model is to be specified even with MLDesigner. System requirements are transformed in a formal model. The figure below shows the top level MLDesigner Model of the developed medical device. Based on this executable specification simulations can be done right now in the early design phases, [12]. The simulation of the system allows a validation, hence **faults can be found in the early design stages**.

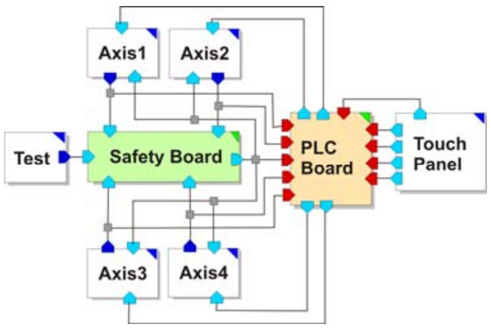

Fig. 3. Top level MLDesigner Model of the developed medical device

With that formal model we have now a specification for designing the safety board. One of the logic functions of the safety board is the emergency stop.

y = (x_0x_1 v x_2x_3 v x_4x_5 v x_6 v x_7 v x_8)x_test

This equation is easy and can be specified in VHDL, simulated and implemented. Figure 4 shows the final implementation.

But the point is that the safety board must work under all conditions being embedded as the top priority subsystem in the medical device. Does this work? With the help of the explained tool linking between the GHDL simulator and MLDesigner the in VHDL specified emergency stop functions could be validated.

Fig. 4. The Hardware implementation of the safety board

We could proof successfully during all design phases that the emergency board works as well as the final implemented system.

8. CONCLUSIONS

A link between simulators of design in the standardized and widely used hardware description language VHDL and MLDesigner has been developed to support the design of digital systems. Integrating a VHDL simulator into MLDesigner enables design flows using a single environmental model or test-bench in specification, development, integration, test and evaluation. These design steps have been carried out and validated in a short explained small example. A tool coupling has been used to simulate the whole medical devices with the synthesized the Safety board. Since a vast amount of new processor architectures are now released along with their hardware description, the tool coupling can be seen as one step further into the direction of unified hardware/software co-simulation and verification based on MLDesigner framework.

ACKNOWLEDGEMENT

This paper is supported in part by the Project Grant III44004 (2011-2014) financed by Ministry of Education and Science Republic of Serbia.

REFERENCES

[1] Niller, S., *A Model-Based Design Flow for Digital Systems in MLDesigner.* Diploma thesis, Ilmenau University of Technology, 2010.

[2] MLDesign GmbH, *MLDesigner Documentation Version 2.7.* 2007.

[3] Tristan Gingold, *GHDL Guide.* 2010.

[4] Niller, S., Patch to extend GHDL version 0.29 with XML output of its IIR. www.stud.tu-ilmenau.de/~seni-in/ghdl-0.29-xmlpatch, 2010

[5] Reshadi, M. H.; Gorji-Ara, B. and Navabi, Z., *HDML: Compiled VHDL in XML.* 2000.

[6] Zamfirescu, A., Zhao, Z. *HXML. A New Approach to managing Hardware Information.*

[7] Willis, J., et. al. *Internal Intermediate Representation (IIR) Specification Version 4.6.* (Trial Implementation Draft of 2/3/99), 1999.

[8] Hauguth, M., *Entwurf eines XML-basierten Dokumentenformats für Blockdiagramme.* Diplomarbeit, Technische Universität Ilmenau 2000.

[9] Kornemann, S., *Ansätze zur Bearbeitung von MML-Modell Beschreibungen mit Eclipse*. 2009.

[10] Schmidt U., et. al., HXT Haskell XML Toolbox. http://www.fh-wedel.de/si/HXmlToolbox/index.html, 2010.

[11] Stevens, W. R., *Advanced Programming in the UNIX Environment.* Addison Wesley 1993.

[12] Zerbe, V., Backhaus, M., *Model based Design of an Efficient CORDIC Algorithm Implementation.* INDEL 2010, Banja Luka.