Simul'Elec, a Delphi written simulator for power Electrical Engineering, using VHDL-AMS modeling

Fabien Legrand
ALGO'TECH INFORMATIQUE
Bidart – France
contact@algotech.fr

Maria Azurmendi, Fernando Martin, Luis Fontán, Senior Member, IEEE
CEIT and TECNUN-Dept. of Electronics and Communications
San Sebastián – Spain
mazurmendi@ceit.es

Jean-Jacques Charlot
ELECTRONIC SOFTWARE
Setif, Algeria
jean-jacques.charlot@wanadoo.fr

Nadine Couture
LIPSI-ESTIA
Bidart – France
n.couture@estia.fr

ABSTRACT.

Few simulators are designed for power electrical engineering and they should be usually used by high skilled users. Simul'Elec is an electrical simulator, part of an electrical CAD software. This simulator is designed with the focus of giving access to simulation to electrical CAD users usually not very familiar with simulation tools. It provides a graphical modeling interface. This friendly simulator written in Delphi integrates graphical user interface to generate VHDL-AMS models and a compiler to simulate its.

1 INTRODUCTION.

If simulation is a complete and unavoidable part of the design process in electronics, it's not really the same with electrical engineering. Some simulators can be used by electrical engineers but are usually designed for power electronics, automation or for educative purpose. Most of these simulators are separate tools, distinct from the CAD software available for electrical engineering.

Another issue is that electrical simulators are usually developed for system design and requires a good knowledge about the behavior of each modeled component and also about the simulator working.
System maintenance and evolution, usually not made by designers, could be successfully improved by electrical simulation. For this, simulator must be able to provide result without needing modeling skills or advanced knowledge about the working of the simulator calculation core. A particular attention must be paid to the used modeling tools and language to find the best compromise between a friendly way to model and the accuracy of the models. Previous works demonstrated the interest of using VHDL-AMS for power electrical engineering components modeling [1]. Then, our solution is based on a friendly graphical user interface to produce a VHDL-AMS model, allowing afterward to use the capabilities of this standard modeling language to define more complex behaviors.
The rest of the paper is organized as follows: Section 2 briefly presents our simulator and the context when using simulation and CAD software in electrical engineering. In section 3, we present the graphical user interface created for the model edition that produces VHDL-AMS models. The principles of translation of GUI data to VHDL-AMS are detailed into section 4. Section 5 gives some details about the VHDL-AMS compiler necessary to the simulator to integrate the models.

2 SIMUL'ELEC, AN EVENT DRIVEN ELECTRICAL SIMULATOR.

Simul'Elec is an electrical simulator integrated to the Electrical CAD Software PackElecBuilder edited by Algo'Tech Informatique [2]. Simul'Elec simulates the diagrams drawn into the CAD environment without needing a specific editor. This simulator is able to calculate voltage and current values in every point of the circuit using the nodal analysis [3] as Berkeley University's Spice Simulator. Simul'Elec is also able to evaluate the logical state of components as relays, circuit breakers, multiple contacts using a event driven modeling [4]. It can be used to simulate electrical installation of buildings, special machines, etc.

The CAD software and the simulator are fully developed using Delphi® [5] development environment. Before 2004, all the models were built in models, written in Delphi and compiled with the simulator. The user was not able to create its own models.

Since 2004, Algo'Tech Informatique leads the FRESH project from the 6th FP (European R&D Framework Program). The aim of this project is to develop a suite of tools for avionics electrical engineering, improve the existing tools, including the simulator.

978-1-4244-1567-0/07/$25.00 © 2007 IEEE

One of the major improvements about simulation is allowing user to create custom models, including behavioral modeling.

The focus of this simulator is to give fast and coherent result, with a reasonable accuracy but without needing modeling skills or simulator special knowledge. Indeed, the simulator is one tool among a range of electrical CAD tools. The potential users are electrical technician or engineers for the industrial side, student and teachers for the educational side. Considering the use context of Simul'Elec and the approach of users more practical than scientific, all the simulator's tools, analysis modes and models are designed to favor a friendly use.

3 GRAPHICAL USER INTERFACE FOR MODEL EDITION.

The user needs to define his own models, then a set of graphical tools have been designed for this. The purpose of these tools is to avoid the user to write model, allowing defining a whole model graphically. As the simulator is based on the nodal analysis, it's only able to simulate circuits made of primitives: basic components as ideal voltage and current sources, resistances, capacitors, etc. Then every model has to be described as a circuit made of primitives and other models made also of primitives, as we can define a macro model with SPICE simulator.

The different steps of graphical model definition are:

- Drawing the model circuit and its external connections.
- Voltage and current characteristic values, for display purpose at different simulation time.
- Model parameters.
- Relation between model parameter and the inner model parameters.

And finally, the model will be saved into the computer memory.

3.1 Drawing of the circuit model.

The user draws a circuit with the same interface he uses to draw electrical diagrams. It's a key point with an ergonomic point of view in order to respect the continuity and the homogeneity for the user interaction. He takes the components he wants from the symbol library. A model is automatically associated to each component of the circuit. By using drawing tools, the user adds wires to link to components.

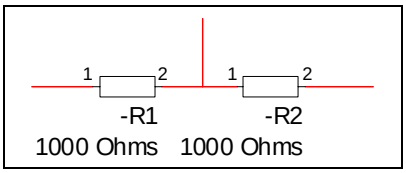

Figure 1: Variable resistance without connections.

Finally, he add connection points one some wire to indicate the input / output between the model and the outside. The model should have the same number of connections as the component it is about to model.

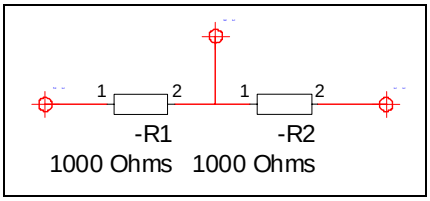

Figure 2: Variable resistance with external connections.

3.2 Voltage and current characteristics.

Some measurement tools as voltmeters and amperemeters can be set on the circuit to indicate to the simulator what are the most interesting internal current or voltage to display. This is not necessary for modeling and simulation. It allows only the user to see some internal current and voltage at different simulation times.

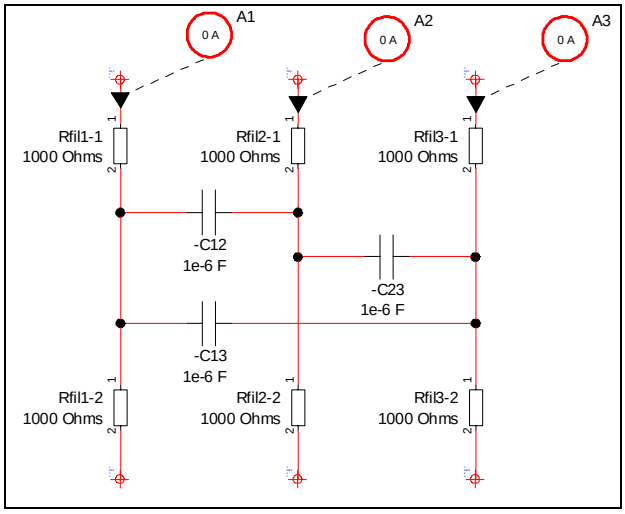

Figure 3: Model of 3 wires high voltage line.

The model of 3 wires high voltage line of Figure 3 has capacitor between each vertical wire to model capacitive effects. Here, we don't want to display internal voltage and current values because the model component are not actual components but are use to model some behaviors. The 3 amperemeters (A1, A2 and A3) indicate to the simulator what are the most pertinent values.
The measurement tools have not effect on simulation calculation, only on results display.

3.3 Model parameters.

Once the previous steps are done, the model parameters must be defined. These parameters are the key values allowing characterizing every model. Usually, these values

are used to define the parameters of the inner components of the model. Each parameter has these characteristics:

- Name (unique for a given model).
- Description (long text).
- Type of data (integer, float, boolean, text, etc).
- Unit.
- Default value.
- Range of validity.

Among these values, some are not necessary to simulate but are used by the graphical interface.

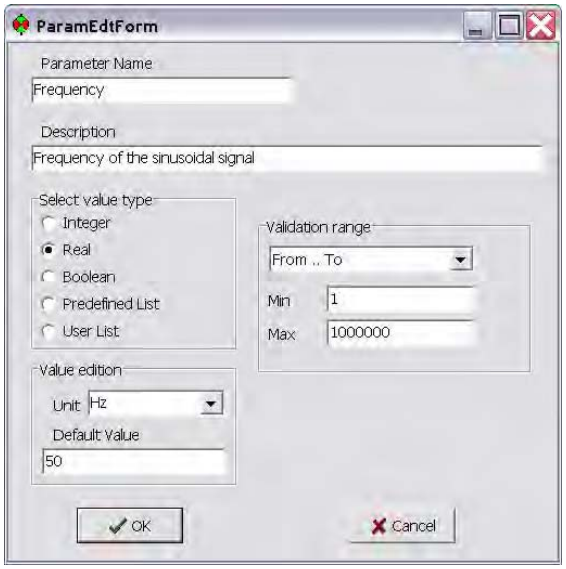

Figure 4: Definition of the parameter "frequency" of a voltage sinusoidal source.

3.4 Association between model parameters and inner component parameters.

Every inner component of the model has parameters. With the graphical interface, the user can associate a formula for each component parameter that is a mathematical expression including constants and model parameters.

For example, a model of variable resistance is composed of two resistances R1, R2 in series. The model parameters are the global resistance Rtotal and the cursor position PosCur.

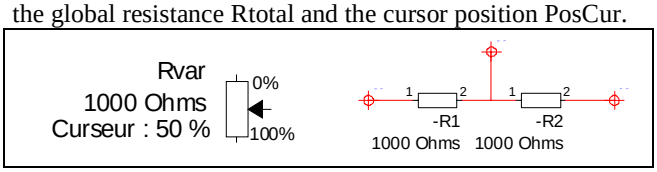

Figure 5: Symbol and model of a variable resistance.

The next figure shows the interface editing the value of the first resistance. Each (outer) parameter of the model, is identified by a label as "P1".

The formula to define the value of the inner parameter uses these labels (P1) and not the name of the outer parameter (Rtotal).

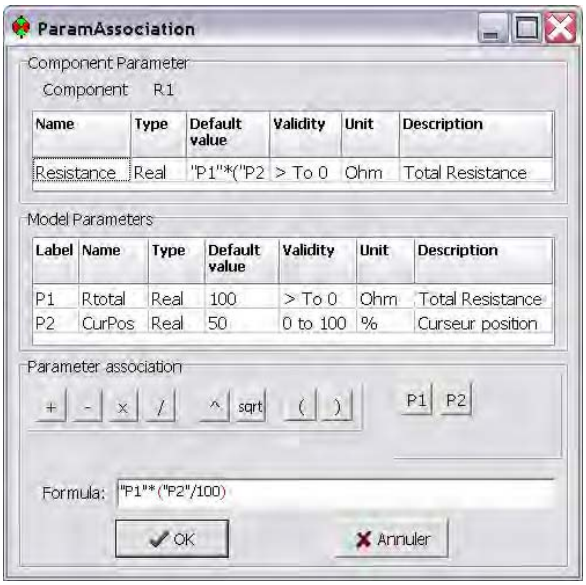

Figure 6 :Inner parameter definition.

Using these formulas, the two inner resistances will be calculated by the simulator.

R1 = Rtotal * CurPos / 100 (1)
R2 = Rtotat * (1-CurPos / 100) (2)

3.5 Model backup.

After all these steps done by the user, it is necessary to save all the information about the model. The circuit drawn at the first step is saved into a CAD file, only for later display purpose. No model information will be used from this drawing at simulation time. Every model information is saved into a single VHDL-AMS model file.

3.6 What about more complicated models.

It appears that the graphical model edition is quite friendly and easy to use and does not require specific modeling skills. This method is sufficient for purely structural models but does not allow defining more complicated behavior. For example, a circuit breaker can be modeled by an ideal switch in series with a resistance, but the rules of its opening following the value of the current cannot easily be defined graphically.
This last aspect, managed by the event manager of the simulator will be defined by some addition into the model file obtained using the graphical interface.

4 GRAPHICAL MODEL TO VHDL-AMS TRANSLATION.

The model made using the graphical interface needs to be saved into a file to be used later by the simulator. Different solutions could be used, such as using binary files or custom text file format (based on XML for example), Delphi script or a standard modeling language.
The last solution has been chosen, using VHDL-AMS as modeling language.

4.1 Choice of VHDL-AMS.

It is important to provide a friendly and easy way to allow end user of the simulator to change some features of the electrical models, without the necessity of a deep knowledge of the internal structure of the simulator. For this purpose, the electrical models defined previously in Delphi language, can also be described in a more intuitive and standard textual modeling language. The language chosen is VHDL-AMS: Analog and Mixed-Signal extension to the Very high speed integrated circuit Hardware Description Language (VHDL). One of the main characteristics of this language and an important reason for this choice is its behavioral modeling capability for discrete, continuous and mixed systems. The continuous systems are described using differential algebraic equations. VHDL-AMS also provides mixed-discipline modeling, so different domains such as electrical and thermal could be described and simulated in a single entity, which is very interesting thinking in possible future improvements of the simulator.

4.2 The translator.

A translator was developed to obtain VHDL-AMS files from the model data structure of the simulator. After the translation and the possible changes introduced by the user to modify some aspects of the components, these files are checked and compiled to be understandable by the simulator. Basic electric components are defined in a behavioral architecture and stored in a library. More complex models are defined from these basic components according to a structural architecture of VHDL-AMS files. The final target is to complete a model library containing VHDL-AMS files describing all the electric components required by the simulator.

Generated VHDL-AMS files contains some comment lines that are inserted by the translator to give useful information to the simulator interface, such as:

- The different sections of these files.
- Parameter and quantity names.
- Parameter values, description, data type, range or possible values, units, etc.

Listing 1: Variable resistance model.

```
LIBRARY IEEE;
LIBRARY Composants_elementaires;
USE IEEE.ELECTRICAL_SYSTEMS.ALL;
USE IEEE.MATH_REAL.ALL;

ENTITY Potentiometre IS
   GENERIC (
            --@Name=Rtotal@Description=Total Resistance
            --@Data Type=1@NumberRange=1@MaxValue=0
            --@MinValue=0@MaxLength=0@ListPosition=-1
            --@PossibleValues=@Unit=Ohm@Value=100@
            Rtotal : REAL := 100.0;
            --@Name=CurPos@Description=Curseur position (percentage)
            --@Data Type=1@NumberRange=5@MaxValue=100
            --@MinValue=0@MaxLength=0@ListPosition=-1
            --@PossibleValues=@Unit=%@Value=50@
            CurPos : REAL := 50.0
   );
   PORT(  TERMINAL EXT1, EXT2, EXT3: ELECTRICAL );
END ENTITY Potentiometre;

--ARCHITECTURE DECLARATION
ARCHITECTURE Struct OF Potentiometre IS
   -- intermediate values for resistance calculation
   QUANTITY Comp_R1_Param_Resistance : REAL;
   QUANTITY Comp_R2_Param_Resistance : REAL;

BEGIN
   Comp_R1_Param_Resistance == Rtotal*(CurPos/100);
   Comp_R2_Param_Resistance == Rtotal*(1-CurPos/100);

   R1 : ENTITY Composants_elementaires.RESISTANCE(Behav)
            GENERIC MAP( Resistance => Comp_R1_Param_Resistance )
            PORT MAP( EXT1 => EXT1, EXT2 => EXT2 );
   R2 : ENTITY Composants_elementaires.RESISTANCE(Behav)
            GENERIC MAP( Resistance => Comp_R2_Param_Resistance )
            PORT MAP( EXT1 => EXT2, EXT2 => EXT3 );
END ARCHITECTURE Struct;
```

VHDL-AMS models obtained with the translator could be used, after some small changes if required, in other commercial VHDL-AMS simulators, such as Simplorer or SystemVision. Similarly, some other models obtained with these commercial simulators could be integrated in the simulator described in this paper.

4.3 The VHDL-AMS Editor.

The VHDL-AMS editor has been developed to display the source code of the model to the user. To maintain a maximum coherence between the GUI for model edition and the simulator capabilities, the user cannot modify the whole model. The entity section can be completely defined by the graphical interface. Then any change to carry to this section must be done through the graphical interface. The entity section of the VHDL-AMS model is shown but is not editable. The mandatory use of the graphical interface for entity ensures a full compatibility with the simulator. About the architecture body, only some parts can be edited, such as the formulas and names of variables, but the component

instances cannot be changed. In this way, the user can modify the behavior of the models.
The VHDL-AMS model is compiled to obtain a Delphi script model understandable by the simulator.

5 VHDL-AMS TO DELPHI COMPILER.

The VHDL-AMS generated files will be edited by the user to build, for example, event driven behavioral models. This language was also used, in other applications, to describe complex mechanical systems and their interconnections [6].

Before the existence of this range of model edition tool, all the models were written in Delphi language. Here, the purpose is to compile the VHDL-AMS model to a format understandable by the simulator, Delphi code. The closeness [7] between object oriented language such as Delphi and VHDL-AMS makes this solution applicable.

5.1 Lexical, syntax and semantic analyzers.

For checking the VHDL-AMS code, lexical, syntax and semantic analyzers have been developed. These analyzers are the components of every compiler and they are explained in the following lines:

-Lexical Analyzer: checks the source code of the input file for lexical errors. For example: if 'si' is written instead of 'is'.
-Syntax analyzer: is used to find syntax errors in the input code. For example, whether the "if" sentences have been written correctly.
-Semantic analyzer: finds in the input code semantic errors. For example, an undeclared variable is used in source code.

In the development of the lexical, syntax and semantic analyzers, we used the lex and yacc tools [8]. These tools, from a lexical, syntax and semantic specification generate the lexical analyzer, syntax analyzer and semantic analyzer.
Several types of lex and yacc compiler tools exist. To mention but a few: Flex and Bison, Jlex and Cup, TPlex and TPYacc.
The analyzers were developed with TPLex and TPYacc [9], reason being that these tools generate the analyzers in Pascal language (Delphi is Object Pascal). Thereby, making the integration of these analyzers in the simulator, developed in Delphi language, is quite easy.

5.2 VHDL-AMS compiled subset.

The developed compiler is fitted to the general structure of the source code generated by the VHDL-AMS translator.

In this structure we can find the sentences corresponding to Entity_declaration and a subset of Architecture_body [10], such as:

- quantity_declaration.
- terminal_declaration.
- simultaneous_statement.
- generic_map_aspect.
- port_map_aspect.

Actually, the limitations of the VHDL-AMS capabilities compiler reflect the simulator's own limitations. For example, when simulating current / voltage relations of a model, the simulator requires a circuit made of basic electrical components. Then the compiler is able to compile entities instantiation and mapping necessary to define the model circuit. For the same reasons, the compiler is not able to understand direct current / voltage relationship base on ACROSS and THROUGH statements. The following resistance model is a typical example of what the simulator cannot understand.

Listing 2: Model of a resistance.

```
ENTITY ams_resistance IS
        GENERIC (r :real:= 1.00);
        PORT (TERMINAL RIN, ROUT: Electrical);
END ams_resistance;

ARCHITECTURE res_BODY OF ams_resistance IS
  quantity vr across ir through RIN to ROUT;
BEGIN
        ir == vr / r; -- cannot be compiled
END res_BODY;
```

This limitation could be solved at compilation by transforming the current / voltage relations into an equivalent circuit. Such a work has been done with the compiler of VamSpiceDesigner [10] which transforms VHDL-AMS model into a model recognized by SPICE.

5.3 Delphi model.

In order to correct the changes made by the user, a lexical, syntactic and semantic analyzer have been developed. Finally, if the model written in VHDL-AMS language is correct, a Delphi script is generated. This Delphi model will be used in the simulator.

Today, the Delphi models obtained from the VHDL-AMS compiler still have to be compiled with the simulator source code to be used at simulation time. For each new model, a new version of the simulator has to be provided.

Scripting solution is required for runtime model integration.

6 CONCLUSION.

In order to allow Simul'Elec users to model component, a set of modeling tools have been designed. The solution used to save the created models has been carefully chosen. The easier way to implement these tools must be to build the modeling user interface on some specific language or even on Delphi programming language used to develop the simulator. Finally, we choose to use VHDL-AMS modeling language as an output of the modeling tools. Even if the implementation is more complicated, requesting translation, edition and compiling capabilities, the use of an IEEE standard carry many advantages for the current use of our simulator and future evolution.

Today, the VHDL-AMS models made with our graphical interface are successfully compiled. The compiled models are considered by the simulator, only if they are compiled with the simulator source code.
Before the end of 2007, a Delphi scripting engine will be integrated into the simulator to allow using any VHDL-AMS model coming from our interface, without compiling the model into the simulator source code.

The next evolution of the VHDL-AMS compiler will be to transform literal current / voltage relations to the equivalent circuit to allow the simulator to integrate it.

Power electrical engineering deals not only with electrical components but also with many electromechanical devices, thermal phenomenon, etc. The choice of VHDL-AMS is also justified by a will to take care, at mid term, of non-electrical behavior using its multi-technological capabilities.

REFERENCES.

[1] F. Legrand, N. Couture, H. Lévi et J.-J. Charlot, *VHDL-AMS Modeling and Library Building for Power Electrical Engineering*, ACM'04, Kawasaki, Japan, 2004

[2] Algo'tech Informatique, electrical CAD software editor, *http://www.algotech.fr/*

[3] V. Litovski, M. Zwolinski**,** *VLSI Circuit Simulation and Optimization*, Chapman & Hall, 1997 ISBN 0-412-63860-6

[4] F. Legrand, N. Couture, R. Briand, H. Lévi, *Simulation de schémas electriques ou electrotechniques par utilisation de l'analyse événementielle*, 4éme Conférence Internationale sur l'Automatisation industrielle, Montréal, 2003.

[5] Delphi integrated development environment, *http://www.codegear.com/products/delphi/*

[6] A. Suescun, J. Calleja, J. Imbernon, *Modeling of Complex Mechanical Systems in VHDL-AMS*, IEEE/ACM International Workshop on Behavioral Modeling and Simulation, Orlando (Florida), USA, 1999

[7] F. Legrand, *Modélisation de circuits électrotechniques en vue de leur simulation – Réalisation d'un simulateur*. Phd thesis, p114

[8] The Lex & Yacc Page *http://dinosaur.compilertools.net/*

[9] Turbo Pascal Lex/Yacc *http://www.musikwissenschaft.uni-mainz.de/~ag/tply/*

[10] VHDL-AMS Syntax (IEEE Std 1076.1) *http://www.iis.ee.ethz.ch/~zimmi/download/vhdlams_syntax.html*

[11] S. Jemmali, J.-J. Charlot, *VamSpiceDesigner, a hierarchical schematic design tool of multi-technological systems based on VHDL-AMS and SPICE*, MIXDES 2003