Dero 4 simulator as a didactic tool

DR ENG. PAWEŁ PLASKURA
FACULTY OF SOCIAL SCIENCES, JAN KOCHANOWSKI UNIVERSITY, BRANCH IN PIOTRKÓW TRYBUNALSKI

Keywords: didactic tools, microsystem simulator, web-based systems, virtual laboratories, distance learning

ABSTRACT:
The work presents the Dero simulator with a behavioral model description language. Behavioral description enables modeling and simulation of physical phenomena. Thanks to these features, the Dero can be used in the didactic process as a modeling and simulation tool. The interface through websites has been integrated with the Quela learning process management platform. The system allows the creation of virtual laboratories.

Symulator Dero 4 jako narzędzie dydaktyczne

Słowa kluczowe: narzędzia dydaktyczne, symulator mikrosystemów, systemy zdalne, wirtualne laboratoria, nauczanie zdalne

STRESZCZENIE:
W pracy przedstawiono symulator Dero z behawioralnym językiem opisu modelu. Opis behawioralny umożliwia modelowanie i symulację zjawisk fizycznych. Dzięki tym funkcjom symulator Dero może być wykorzystywany w procesie dydaktycznym jako narzędzie do modelowania i symulacji. Interfejs graficzny za pośrednictwem stron internetowych został zintegrowany z platformą zarządzania procesem nauczania Quela. System umożliwia tworzenie wirtualnych laboratoriów.
1 INTRODUCTION

The very rapid development of electronics since the 1970s has led to the development of advanced methods, algorithms and simulation programs for electronic circuits [6, 11, 12, 17, 21]. Simulation programs of analog electronic circuits have become the basic tool for designers of electronic circuits [8, 20]. In other areas, progress was not so fast. Since the beginning of the 1990s, a more frequent problem was the simulation of systems from different environments. This led to the development of the analogy [22], which allows simulation of systems from different environments. This principle made it possible to simulate microsystems [23, 25] (e.g. electrical-mechanical systems). The basis of this type of simulation system is the behavioral modeling [7] which enables a description of the system using mathematical formulas (mathematical equations). Behavioral models can be easily used in the simulation process. A number of behavioral languages have been developed, including VHDL-AMS [1], EMDL [5, 13], MDL [16] and others.

Due to the high demand for computing power, simulation systems were not previously available to an ordinary user. Currently, the situation has changed. High computing power and the access to the computer network allows access to such systems. For years, there has been a tendency to create simulation systems in the cloud. This approach has a number of advantages. The system can be accessed from many places in the world. The user does not have to install the software, which is sometimes a complicated process. Access via websites enables the use of systems on most computer and mobile devices. This allows the creation of virtual laboratories and their use in the didactic process. It reduces the costs of education or makes experiments possible at all. Simulations allow reducing the time of the experiment. The specificity of computational algorithms in simulation systems requires knowledge of mathematics and understanding of the operation of such programs.

The aim of the work is to present the Dero simulator and user interface DeroWWW in the form of the website. The system gives the opportunity to create a virtual laboratory.

2 DERO SIMULATOR

The Dero simulator presented in the work was developed by the author. In a sense, it is a continuation of the Optima simulator project [5, 19, 20], which was developed in the Institute of Electronic Systems at Warsaw University of Technology. The model description language used in the Dero simulator is a modified and extended EMDL [13] language of the Optima simulator [5]. The Dero simulator [3, 16] is a new project. It was written in C++ [24] using object-oriented programming. It uses a number of techniques to improve the reliability of the entire system [14]. A number of algorithms have been developed and implemented in the simulator [16]. The Dero simulator has many unique solutions unparalleled in practice. It has MDL behavioral model description language. The derivative calculation module for MDL code has greatly simplified the creation of MDL models. Many programming techniques have been developed and implemented to detect errors during program execution. This system allows detection of errors in the program code and in the MDL model code. The extensive system of error detection allows detection of errors in the program code, numeric errors, and errors in the MDL model code. This is especially important because of the implementation of iterative algorithms. The program is adapted to simulate large circuits. Sparse matrix method was implemented. Program messages can be easily translated into other languages. For advanced users, a syntax file has been created for the Vim editor which enables very efficient work with program files [2].

The simulator is developed in the Linux [9, 18] system environment and distributed in the form of *.deb packages. Dero is a stand-alone program run from the command line. Thanks to this, it can be easily integrated with other programs.

The program allows analysis of linear and non-linear systems. It enables DC, time and frequency analysis as well as DC characteristics analysis. Selected versions of the program allow analysis of mixed signal types (analog-digital) and event-driven iterated timing analysis.

The program can be used as a tool for automatic design. The use of deterministic optimization was used here. Optimization involves selection of such system parameters to make the modified system
meet the imposed design requirements. The main distinguishing feature of this simulator from other simulators is the behavioral model description language (MDL) which has the ability to simulate microsystems by use of analogy [22]. The MDL has the ability to define: network variables, data types, inputs and outputs, numeric value multipliers and constants. The MDL is relatively simple and intuitive model description language comparing to other simulators.

2.1 Input language

The accepted description is a list of connected elements via nodes (netlist). Each element is described by its model. Nodes are connected to inputs and outputs of component models. Information about changes in the values of network variables (signals) is available at the inputs of the model. Outputs represent the values of corresponding system functions depending on inputs values, time and time derivatives. Subcircuits can be inserted into the main circuit many times. The program allows defining nested subcircuits. The input file describes the circuit in the form of a list of connected elements (netlist). It is divided into circuit description section and command section as shown in the Figure 1. Several tasks can occur in the input stream (or file). The single task starts with the keyword .TASK and ends with the keyword .END. The single task is divided into two main parts:

1. Circuit description, which may include definitions of data types, units, variables, functions, models, subcircuits, and declarations of model lines, elements, subcircuits.
2. Command block starting with the directive .CMD. The circuit description is implemented using a list of elements connected via nodes. The model defines two kinds of parameters: individual and common. The individual parameters are associated with the element. The common parameters are stored in the model line. Each element refers to the model through the model line.

2.2 Model definition

The .MODEL directive allows defining an element model. Definition of the model (Listing 1) consists of two parts: header and model body. The header defines item and model parameters, variables, controls, and Model output. The element is attached to the system via external nodes defined by the directive .EXTERNAL. The internal nodes are defined by the directive .INTERNAL. Branch variables of the outputs must be defined (directive .FLOW). Values of the output variables and their derivatives relative to the controls are calculated in the model body.

Listing 1: Structure of the model definition

```
1 .MODEL model_name
2 .OPTI name=value[, ...];
3 .EXTERNAL external_vars[, ...];
4 .INTERNAL internal_vars[, ...];
5 .FLOW flow[, ...];
6 .GROUP group_id: node_or_flow[, ...];
7 .INPUT id(node_or_flow[, pin_minus]): type_id par;
8 .OUTPUT id(node_plus[, pin_minus, flow[, flow]]): type_id par;
9 .PARAM TypeId par1, par2, ...;
10 .COMMON TypeId par1, par2, ...;
11 .STATE TypeId par1, par2, ...;
12 .VAR TyepId var1, var2, ...;
13 .MEM TyepId var1, var2, ...;
14 .VAR TyepId var1, var2, ...;
15 BEGIN
16 # MODEL BODY
17 ...
18 END
```
\.FLOW declaration of network variables that are not network nodes (e.g. load variables, branch currents)
\.PARAM list of individual element parameters,
\.COMMON list of common model parameters (set in model line),
\.VAR list of model variables reset before calling model code.
\.MEM lists the model variables that are stored between model code calls,
where: Typeld is a data type identifier, parX is the name of the parameter.

3 RESULTS AND DISCUSSION

3.1 Example of the diode model

The diode model DS is fully compatible with the SPICE 2G6 model \[10\]. The model is shown in the Figure 2. The model parameters are listed in the Table 1. Equations of the diode model for the operating ranges are shown below (where $T$ is the temperature):

\[
v_D \leq -BVT
\]
\[
i_D = -I ST \cdot \left[ \exp \left( \frac{-BVT - v_D}{V_T} \right) - 1 \right] - \frac{BV_T}{V_T}
\]
\[
v_D = BV_T
\]
\[
i_D = I BV
\]
\[-BVT < v_D < -5 \cdot N \cdot V_T
\]
\[
\frac{i_D}{-IS_T + GMIN \cdot v_D}
\]
\[
v_D \geq -5 \cdot N \cdot V_T
\]
\[
\frac{i_D}{-IS_T \cdot \left[ \exp \left( \frac{v_D}{N \cdot V_T} \right) - 1 \right]}
\]

The $C_D$ capacity model is the sum of the junction and diffusion capacities (1).

\[
C_D = C_j + C_d
\]

where:

\[
v_D \leq VJ \cdot FC
\]
\[
C_j = \frac{CJOT}{\left( 1 - \frac{v_D}{VJ} \right)^{-M}}
\]
\[
v_D > VJ \cdot FC
\]
\[
C_d = IS_T \cdot \frac{TT}{N \cdot V_T} \cdot \exp \left( \frac{v_D}{N \cdot V_T} \right)
\]

The MDL code of the model is shown in the Listing 2.

\[
\begin{array}{l}
\text{Listing 2: Model of the diode (library d/ds.mdl)}
\end{array}
\]
Dero 4 simulator as a didactic tool

REAL TNOM " nominal temperature (300.15K)
REAL FC " forward
REAL TT " transient time (0 sec)
REAL NBV " reverse breakdown ideality factor (1)
REAL IBV " reverse breakdown knee current (1e-10)
REAL RS " resistance (Ohm)
REAL IS " saturation current (1e-10)
REAL VT " transient time (0 sec)
REAL NBV " reverse breakdown ideality factor (1)
REAL I S " saturation current (1e-10)
REAL JS " reverse breakdown knee current (1e-10)
REAL RS " resistance (Ohm)
REAL IS " saturation current (1e-10)

# MODEL PARAMETERS
COMMON REAL IS " saturation current (1e-14 A)
COMMON REAL RS " resistance (Ohm)
COMMON REAL NBV " reverse breakdown ideality factor (1)
COMMON REAL IBV " reverse breakdown knee current (1e-14)
COMMON REAL TT " transient time (0 sec)
COMMON REAL CJT " zero–bias junction capacitance (0 pF)
COMMON REAL VJ " p–n potential (1 V)
COMMON REAL M " p–n grading coefficient (0.5)
COMMON REAL FC " forward–bias depleting capacitance coefficient (100.5 S)
COMMON REAL EG " banggap voltage (barrier height) (1.11 eV)
COMMON REAL XT " 15 temperature exponent [3]
COMMON REAL TTNOM " nominal temperature (300.15K)
COMMON REAL TEMP_OLD--; COMMON REAL TEMP; COMMON REAL VT;
COMMON REAL I_S;
COMMON REAL I_D;
COMMON REAL I_BV;
COMMON REAL I_S;
COMMON REAL RS;
COMMON REAL IS;
COMMON REAL I_D;
COMMON REAL I_BV;
COMMON REAL I_S;
COMMON REAL RS;
COMMON REAL IS;

# PREPROCESSED VARS
MEM REAL TEMP_OLD--; MEM REAL TEMP; MEM REAL VT;
MEM REAL I_S;
MEM REAL I_D;
MEM REAL I_BV;
MEM REAL I_S;
MEM REAL RS;
MEM REAL IS;
MEM REAL I_D;
MEM REAL I_BV;
MEM REAL I_S;
MEM REAL RS;

# PREPROCESSING
IST=IST; VJ=VJ;
CT=CT*AREA;
EGD=16.7-0.02*EXP(-IST*NVT)/(NVT+1108);
VT=[KBIOLZ/QUELE] * TEMP;
NVT=VJ;
VCB=VT+NVT +LOG(NVT/(SORT(2)*1ST));
BVNT=NBV*VT;
RAT=TEMP/TNOM ;
ASSR=VT; BVT=VJ; VIT=IST;
IF(RAT>0) { C_D+D_JUN; }
IF(RAT<0) { D_DIF+D_JUN; }
IF(RAT=0.5) { Aƞ=0.5; }
IF(RAT<1) { BSAT; }
IF(RAT>1) { BVT; }

3.2 Operational amplifier ua741

Figure 3 shows the schematic of the operational amplifier (ua741) application. The input task describes the functional behavior of the amplifier. The subcircuit XO1 is placed on line 26. The subcircuit line XO1 refers to the subcircuit definition of O741 (included on line 15).

Listing 3: Input task describing the circuit in Figure 3.

3. REAL\n4. IN
5. (included on line 15).
6. refesh to the subcircuit definition of O741
7. (included on line 15).
8. Revision 1.6
9. TASK "Operational Amplifier ua741"
10. LIB "types.mdl"
The operational amplifier is built of discrete elements placed in the subcircuit - Listing 4. The model lines are placed (.MODEL directives) on lines 3..7. The MDL models must be pre-loaded. Lines 9..47 contain declarations of elements. The resistors ($R_{11..11}$) are placed on lines 32..42. Individual parameters ($R$) are set. Each element is described by the $R$ model through the element line $R$ (line 4).

Listing 4: The operational amplifier $ua741$ defined as subcircuit.

3.3 Controlled charge

An example of a system containing a controlled charge of $Q_1$ and linear inductance $L_1$ is shown in the Figure 4. The $Q_1$ element accumulates the charge. It is controlled by a voltage source $V_2$. It is described by the $QQ$ model. The input file is shown in the Listing 4.

Listing 5: Controlled charge circuit file.

The MDL models are loaded on lines 4..10. The QQ model definition is placed on lines 12..25. The formula for $Q$ is placed on line 23. The model lines are placed on lines 27..30. Lines 32..34 contain
declarations of elements. The transient simulation directive is placed on line 41. The results of the simulation are shown in the Figure 5.

3.4 Integration with Quela platform

DeroWWW is a web interface for the Dero simulator. It has now been integrated with the Quela platform [4, 15]. The user interface is intuitive. The system is equipped with documentation, help and a number of examples. The task list is shown in the Figure 6. The directory contains error files (*.err), output files (*.out) and format files (*.plt) for waveform visualization. The appearance of a graphical postprocessor with the results of sample analysis (Java applet) is shown in the Figure 5. It allows to scale and print waveforms. Quela platform with DeroWWW module allows creating the virtual lab. Simulator input files can be used in the didactic unit.

4 CONCLUSIONS

The article presents the Dero simulator and the user interface in the form of a website. Basic simulator features are discussed and examples of application for modeling and simulation are presented. It has been shown that creating models in MDL is relatively simple. Files describing the input tasks and circuit diagrams are provided. The method of creating a system description using netlist gives the user the knowledge on how to create such a description for the needs of simulation systems. The graphical user interface integrated with the Quela platform allows the creation of virtual laboratories. Each user with an account on the Quela platform has access to the DeroWWW simulation system. The platform allows storing input files and simulation results. The Quela platform allows you to manage the didactic process. It has an integrated e-learning platform and a didactic process management system.
REFERENCES


