APARATURA
BADAWCZA I DYDAKTYCZNA

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 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 nonlinear 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

```
.TASK title of the task 1
CIRCUIT BLOCK
.CMD
COMMAND BLOCK
.END end of the task 1
```

```
.TASK title of the task 2
CIRCUIT BLOCK
.CMD
COMMAND BLOCK
.END end of the task 2
```

…next task

Figure 1: Input task

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

```
.MODEL     model_name
.OPTI      name=value[, ...];
.EXTERNAL  external_vars [, ...];
.INTERNAL  internal_vars [, ...];
.FLOW      flow [, ...];
.GROUP     group_id: node_or_flow[, ...];
.INPUT     id( node_or_flow [, pin_minus] ) : type_id par;
.OUTPUT    id( node_plus [ pin_minus, flow [,flow] ] ) :
        ↪ type_id par;
.PARAM     TypeId par1, par2, ...;                # element
        ↪ parameters
.COMMON    TypeId par1, par2, ...;                # model
        ↪ parameters
.STATE     TypeId par1, par2, ...;                # state
        ↪ variables
.VAR       TypeId var, ... ;                      # model
        ↪ variables
.MEM       TypeId var, ... ;                      # element
        ↪ variables
.VAR       TypeId var, ... ;                      # local
        ↪ variables
.BEGIN
# MODEL BODY
...
.END
```

The meaning of the individual directives in the definition of the model is as follows:

.OPTI set model options,

.EXTERNAL declaration of external variables (nodes) to connect an element,

.INTERNAL declaration of internal variables (nodes) of the model,

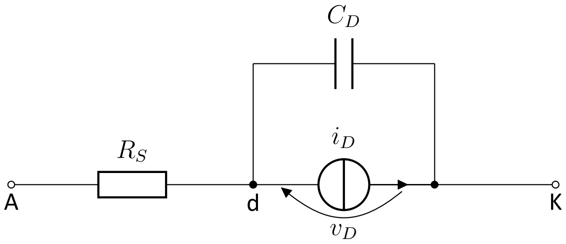

Figure 2: Model of the diode.

.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: *TypeId* 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):

Table 1: Diode parameters

element parameters		
AREA	area factor	1
model parameters		
IS	saturation current	1e-14 A
N	emission coefficient	1
BV	reverse breakdown *knee* voltage	infinite
IBV	reverse breakdown *knee* current	1e-3
NBV	reverse breakdown ideality factor	1
TT	transient time	0sec
CJO	zero-bias junction capacitance	0pF
VJ	p-n potential	1V
M	p-n grading coefficient	0.5
FC	forward-bias depletion capacitance coefficient	0.5
EG	band gap voltage	1.11eV
XTI	IS temperature exponent	3
TNOM	nominal temperature	300.15K

$v_D \leq -BV_T$

$$i_D = -IS_T \cdot \left[\exp\left(\frac{-BV_T - v_D}{V_T} \right) - 1 - \frac{BV_T}{V_T} \right]$$

$v_D = BV_T$

$$i_D = IBV$$

$-BV_T < v_D < -5 \cdot N \cdot V_T$

$$i_D = -IS_T + GMIN \cdot v_D$$

$v_D \geq -5 \cdot N \cdot V_T$

$$i_D = -IS_T \cdot \left[\exp\left(\frac{v_D}{N \cdot V_T} \right) - 1 \right]$$

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

$$C_D = C_j + C_d \tag{1}$$

where:
$v_D \leq VJ \cdot FC$

$$C_j = CJO_T \cdot \left(1 - \frac{v_D}{VJ_T} \right)^{-M}$$

$v_D > VJ \cdot FC$

$$C_j = \frac{CJO_T}{(1 - FC)^{(M+1)}} \left(1 - FC\,(M+1) + M\,\frac{v_D}{VJ_T} \right)$$

$$C_d = IS_T\,\frac{TT}{N \cdot V_T}\,\exp\left(\frac{v_D}{N \cdot V_T} \right)$$

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

Listing 2: Model of the diode (library d/ds.mdl)

```
# Revision : 1.6
#
# SPICE like diode model
#
.MODEL DS
.OPTI       LIMU=OFF,CLUB=8,CLLB=2;
.EXTERNAL A,K; # EXTERNAL NODES
.INTERNAL AI;  # INTERNAL NODES
# INPUT
.INPUT      V(AI,K):REAL V_D;
# INPUTS
.OUTPUT     J(AI,K):REAL I_D;
.OUTPUT     C(AI,K):REAL C_D;          # J(AI,K):
.OUTPUT     R(A,AI):REAL R_S;
# ELEMENT PARAMETERS
.PARAM   INT   LIM_ON "turn on/off input limit
      ↪ "=0;
```

```
.PARAM   INT   DEBUG   "debug printout
        ↪ "=0;
.PARAM   REAL TEMP    "temperature
        ↪ "=300.15;
.PARAM   REAL AREA    "area factor (1)
        ↪ "=1;
# MODEL PARAMETERS
.COMMON REAL IS      "saturation current (1e-14 A)
        ↪                "=1E-14;
.COMMON REAL RS      "resistance (Ohm)
        ↪ "=1e-9;
.COMMON REAL N       "emission coefficient (1)
        ↪ "=1;
.COMMON REAL BV      "reverse breakdown knee voltage (infinite)
        ↪ "=100;
.COMMON REAL IBV     "reverse breakdown knee current (1e-10)
        ↪ "=1E-3;
.COMMON REAL NBV     "reverse breakdown ideality factor (1)
        ↪ "=1.0;
.COMMON REAL TT      "transient time (0 sec)
        ↪ "=0;
.COMMON REAL CJO     "zero-bias junction capacitance (0 pF)
        ↪ "=1e-6;
.COMMON REAL VJ      "p-n potential (1 V)
        ↪ "=1;
.COMMON REAL M       "p-n grading coefficient (0.5)
        ↪ "=0.5;
.COMMON REAL FC "forward-bias depletion capacitance coefficient
        ↪ "=0.5;
.COMMON REAL EG      "bangap voltage (barrier height)  (1.11 eV)
        ↪ "=1.11;
.COMMON REAL XTI     "IS temperature exponent (3)
        ↪ "=3;
.COMMON REAL TNOM    "nominal temperature (300.15K)
        ↪ "=300.15;
# PREPROCESSED VARS
.MEM     REAL TEMP_OLD=-1;
.MEM     REAL EG0,REAL RAT,REAL EGT,REAL TOL,REAL IT,REAL IBVV;
.MEM     REAL IST,REAL NVT,REAL VT,REAL BVV,REAL BVTN;
.MEM     REAL CJT,REAL VJT,REAL FCVJ,REAL FC1,REAL FC2,REAL FC3,
        ↪ REAL VCRIT;
.VAR     REAL CJUN;
.VAR     REAL CDIFF;
.VAR     REAL GD;
.VAR     REAL VDTEMP;
.BEGIN

IF( TEMP != TEMP_OLD ){
# TEMPERATURE CHANGED - FIRST CALL
  TEMP_OLD  = TEMP;

# CHECKING VALUES
  IF((AREA<=0)||(IS<=0)||(N<1)) { ASSERT(1); }
  IF((EG<0)||(TNOM<=0)||(XTI<0)){ ASSERT(2); }
  IF((BV<0)||(IBV<=0)||(NBV<1)) { ASSERT(3); }

# PREPROCESSING
  IST=IS*AREA;
  VJT=VJ;
  CJT=CJO*AREA;
  EG0=1.16-7.02E-4*(TNOM*TNOM)/(TNOM+1108);
  VT=(KBOLTZ/QELE)*TEMP;
  NVT=N*VT;
  VCRIT=NVT*LOG(NVT/(SQRT(2)*IST));
  BVTN=NBV*VT;
  RAT=TEMP/TNOM   ;

  ASSERT ( NVT ==0 );
  ASSERT ( VJ  ==0 );
  ASSERT ( VJT ==0 );
  ASSERT ( BVTN==0 );
  ASSERT ( IST ==0 );

  IF(RAT!=1){
    EGT=1.16-7.02E-4*(TEMP*TEMP)/(TEMP+1108);
    IST=IST*(RAT^(XTI/N))*EXP((RAT-1)*EG/NVT);
    VJT=VJT*RAT-3*VT*LOG(RAT)-EG0*RAT+EGT;
    CJT=CJT*(1+M*(4E-4*(TEMP-TNOM)+(1-VJT/VJ)));
  }
  FCVJ=FC*VJT;
  FC1=(1-FC)^(1+M);
  FC2=1-FC*(1+M);
  FC3=M/VJT;
  BVV=BV;
  TOL=1E-4*IBV;
  IF(BV){
    BVV=BV-BVTN*LOG(1+IBV/IST);
    IT=0;
    IBVV=0;
    WHILE((ABS(IBVV-IBV)>TOL)&&(IT<=25)){
      BVV=BV-BVTN*LOG(IBV/IST+1-BVV/BVTN);
      IBVV=IST*(EXP((BV-BVV)/BVTN)-1+BVV/BVTN);
      IT=IT+1;
    }
  }
}

IF( OPT_NRME == 0 && NR <= 1 && DC_ANALYSIS ){
# AUTOMATIC START POINT
    V_D=VCRIT;
}ELSE{
  IF( LIM_ON ){
#  LIMIT INPUT VALUE INSIDE MODEL (LIM_ON!=0)
    IF((BVV!=0)&&V_D<MIN(0,-BVV+10.0*NVT)){
      VDTEMP= -V_D-BVV;
      (VDTEMP)=PNJLIM(VDTEMP,(-(V_D_+BVV)),NVT,VCRIT);
      V_D=-VDTEMP-BVV;
    }ELSE{
      (V_D)=PNJLIM(V_D,V_D_,NVT,VCRIT);
    }
  }
}

# OUTPUTS CALCULATION BLOCK
R_S = 0; # RESISTANCE RSef
IF ( RS != 0 ){
  R_S = RS/AREA;
}

IF(V_D>(-5*NVT)){
# NORMAL AND SUBTHRESHOLD REGION (I)
    I_D=IST*(EXPO(V_D/NVT)-1);
}ELSE{
  IF((BVV!=0)&&(V_D<(-BVV))){
# BREAKDOWN REGION (III)
    I_D=-IST*((EXPO(-(BVV+V_D)/(VT*NBV))-1)+BVV/(VT*NBV));
  }ELSE{
# CUTOFF REGION (II)
    I_D=-IST;
  }
}

# CAPACITANCE  C_D = D_dif + D_jun
C_D = 0;
IF(V_D>FCVJ){
  CJUN=(FC2+V_D*FC3)*CJT/FC1;
}ELSE{
  CJUN=CJT/((1-V_D/VJT)^M);
}
GD=(TT*IST/NVT)*EXPO(V_D/NVT);
CDIFF=TT*GD;
C_D=CDIFF+CJUN;

.END
```

3.2 Operational amplifier ua741

Figure 3 shows the schematic of the operational amplifier (*ua741*) application. The input task describing the amplifier is shown in the Listing 3. The task begins on line 2. The MDL models are loaded on lines 3..13. The model lines are (*.MODEL* directives) placed on lines 17..19. The lines of models refer to the model. The declaration of the subcircuit line *XO1* is placed on line 26. The subcircuit line *XO1* refers to the subcircuit definition of *O741* (included on line 15).

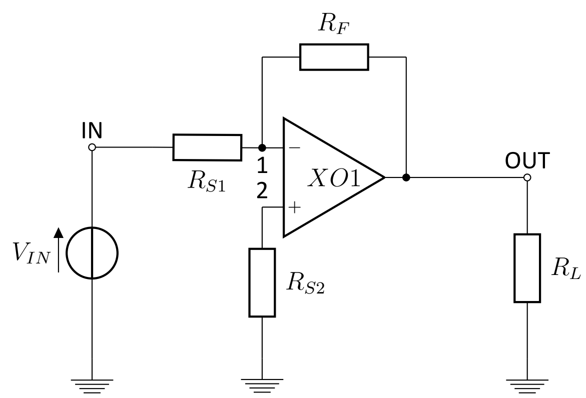

Figure 3: The amplifier circuit.

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

```
1 # Revision : 1.6
2 .TASK "Operational Amplifier ua741"
3 .LIB "types.mdl";
```

```
4  .LIB "units.mdl";
5  .LIB "vars.mdl";
6  .LIB "bjt/pnjlim.mdl";
7  .LIB "math/one.mdl";
8  .LIB "math/pulse.mdl";
9
10 .LIB "src/v.mdl";
11 .LIB "rlc/r.mdl";
12 .LIB "rlc/c.mdl";
13 .LIB "bjt/bjt2.mdl";
14
15 .INC "ua741.sub"
16
17 .MODEL R      R;
18 .MODEL C      C;
19 .MODEL V      VT  V1=0   V2=5,TD=5mS,TR=1mS,PW=7mS,TF=1mS,FREQ=0;
20
21 V.VIN  IN   0              DC=-1;
22 R.RS1  IN   1              R =1kOhm;
23 R.RS2  2    0              R =0.5kOhm;
24
25 #          - + out 0
26 O741:XO1   1   2   OUT   0;
27 C.CL   OUT  1               C=0.1uF;
28 R.RF   1     OUT            R=1kOhm;
29 R.RL   OUT   0              R=100kOhm;
30 #
31 .CMD
32 .PRINT ADD TR("IN") TR("OUT") PAR(HN) PAR(RLTE) PAR(ORDER);
33 .PRINT ADD OP("IN") OP("OUT") PAR(NR);
34 .OP   TITLE="OP - ua741";
35 .TR   STEP=1mS TMIN=0mS TMAX=20mS HMAX=10mS   TITLE="TRAN - ua741
       ↪ ";
36 .END
```

XO1 is attached to the main circuit via nodes: 1, 2, OUT, 0. The remaining elements are placed on lines 21..23 and 27..29. 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 ($R_{1..11}$) 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.

```
1  # Revision : 1.6
2  .SUBCKT O741 1 2 OUT zero
3  .MODEL C    C;
4  .MODEL R    R;
5  .MODEL QNL  BJT2_E TYPE= 1;
6  .MODEL QPL  BJT2_E TYPE=-1;
7  .MODEL V    SRCV_E TD=20E-9 TR=9E-9 PW=30E-9 TF=12E-9 FREQ=0;
8
9  QNL.Q1     3      2      4;
10 QNL.Q2     3      1      5;
11 QPL.Q3     7      6      4;
12 QPL.Q4     8      6      5;
13 QNL.Q5     7      9      10;
14 QNL.Q6     8      9      11;
15 QNL.Q7     VCC    7      9;
16 QNL.Q8     6      15     12;
17 QNL.Q9     15     15     VEE;
18 QPL.Q10    3      3      VCC;
19 QPL.Q11    6      3      VCC;
20 QPL.Q12    17     17     VCC;
21 QPL.Q14    22     17     VCC;
22 QNL.Q15    22     22     21;
23 QNL.Q16    22     21     20;
24 QNL.Q17    13     13     VEE;
25 QNL.Q18    VCC    8      14;
26 QNL.Q19    20     14     18;
27 QNL.Q20    22     23     OUT;
28 QPL.Q21    13     25     OUT;
29 QNL.Q22    VCC    22     23;
30 QPL.Q23    VEE    20     25;
31
32 R.R1    10   VEE  R=1e3;
33 R.R2    9    VEE  R=50e3;
34 R.R3    11   VEE  R=1e3;
35 R.R4    12   VEE  R=3e3;
36 R.R5    15   17   R=39e3;
37 R.R6    21   20   R=40e3;
38 R.R7    14   VEE  R=50e3;
39 R.R8    18   VEE  R=50;
40 R.R9    OUT  25   R=25;
41 R.R10   23   OUT  R=50;
42 R.R11   13   VEE  R=50e3;
43
44 C.COMP  22   8  C=30pF
45
46 V.VCC     VCC      zero   DC=15;
47 V.VEE     VEE      zero   DC=-15;
48
49 .END
```

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

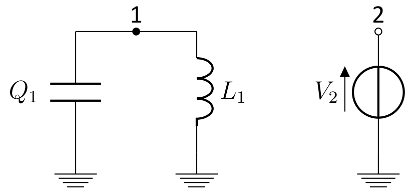

Figure 4: Controlled charge.

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.

```
1  # Revision : 1.6
2  .TASK "Controlled charge (schematic: charge-sch.eps)"
3
4  .LIB "types.mdl"
5  .LIB "units.mdl"
6  .LIB "math/pulse.mdl"
7  .LIB "src/v.mdl"
8  .LIB "rlc/r.mdl"
9  .LIB "rlc/l.mdl"
10 .LIB "rlc/c.mdl"
11
12 .MODEL QQ;
13 .OPTI        LIMU=OFF,CLUB=5,CLLB=5;
14 .COMMON      REAL UG=1;
15 .PARAM       REAL C=1;
16 .EXTERNAL    N1,N2,N3,N4;
17 .FLOW        QQ;
18 .OUTPUT      Q(N1,N2,QQ):REAL Q;
19 .INPUT       V(N1,N2):REAL U;
20 .INPUT       V(N3,N4):REAL US;
21 .BEGIN
22
23 Q=C*(1+US/UG)*U;
24
25 .END
26
27 .MODEL L      L;
28 .MODEL R      R;
29 .MODEL Q      QQ;
30 .MODEL V      SRCV_E TDSI=10 VSI=0.99 FREQ=0.03 V1=0 V2=0;
31
32 L.L2     1 0          L=1;
33 Q.CP1    1 0 IN 0     C=1;
34 V.V2     IN  0        DC=0;
35
36 .CMD
37 .OPTI REXG=2;
38 .PRINT ADD TR(1)  TR(IN)  TR(CP1.QQ)  PAR(HN)  PAR(HN) PAR(RLTE
       ↪ )  # PAR(ORDER);
39 .IC  CP1.QQ=1;
40 .OPTI PIV=1, WDCT=1, HMIN=1e-9, HINI=1E-5, POST=0, ERTR=1e-5,
       ↪ RELT=1e-7, ALLP=1;
41 .TRAN STEP=0.1S TMIN=0.1S TMAX=41S HMAX=0.2S UIC TITLE="TR
       ↪ analysis";
42 .END
```

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

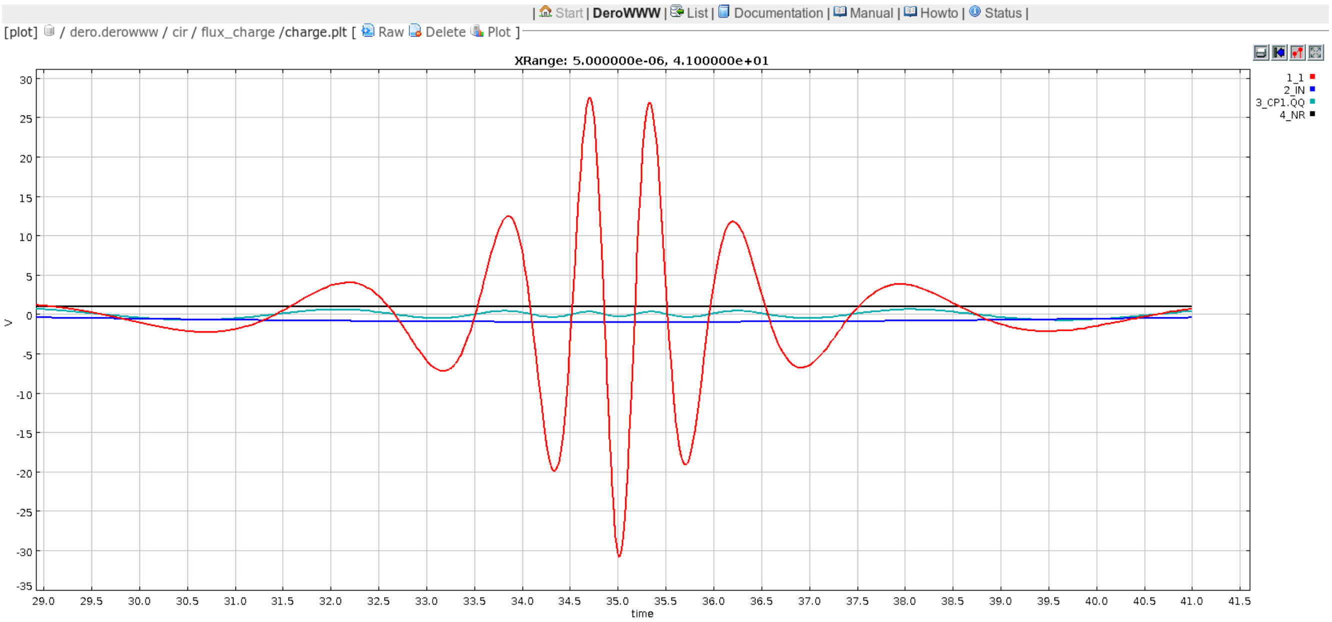

Figure 5: Simulation results of the circuit in the Figure 4.

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

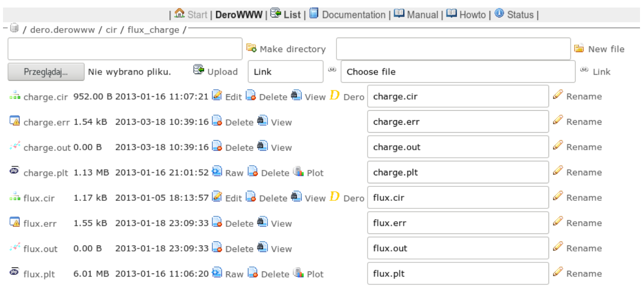

Figure 6: Directory containing tasks and results for the Dero simulator.

(* *.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

[1] IEEE standard VHDL (integrated with VHDL-AMS changes). Technical report, 345 East 47th Street, New York, NY 10017, USA, Aug 1998.

[2] Vim manpage, 2016. URL `http://andothersmanpages.debian.net/cgi-bin/man.cgi?query=vim`.

[3] Dero web page, 2017. URL `http://dero.aiva.pl`.

[4] Quela web page, 2017. URL `http://quela.aiva.pl`.

[5] Bukat D., Ogrodzki J. *OPTIMA v.2.0: uniwersalny analizator układów elektronicznych*. WNT, Warszawa, 1995. ISBN 83-204-1816-X.

[6] Chua L., Lin P. *Komputerowa analiza układów elektronicznych*. WNT, Warszawa, 1981.

[7] Deutch J., Newton A. Data-flow based behavioral-level simulation and synthesis. *Proc. IEEE ICCAD*, Sept 1983.

[8] Herniter M. *Schematic Capture with MicroSim PSpice*. Prentice Hall, Upper Saddle River, NJ, 1998.

[9] Linux. Linux.org, 2016. URL `https://www.linux.org/`.

[10] Nagel L. *SPICE2 a computer program to simulate semiconductor circuits*. Electronic Research Laboratory College Engineering University of California, Berkeley 94720, May 1980.

[11] Ogrodzki J. *Circuit simulation methods and algorithms*. CRC Press, Boca Raton, USA, 1994.

[12] Ogrodzki J. *Komputerowa analiza układów elektronicznych*. PWN, Warszawa, 1995.

[13] Ogrodzki J., Bukat D. Compact modelling in circuit simulation: the general purpose analyzer OPTIMA 3. *ISCAS 94 PROCEEDINGS*, pages 383–386, 1994.

[14] Plaskura P. *Dydaktyka informatyki*, volume 10 of *Piotrkowskie Studia Pedagogiczne*, chapter Niezawodność i koszty tworzenia oprogramowania, pages 97–120. Naukowe Wydawnictwo Piotrkowskie, Piotrków Trybunalski, 2003.

[15] Plaskura P. Quela 2 - podręcznik użytkownika. 2012.

[16] Plaskura P. *Symulator mikrosystemów Dero v4. Metody i algorytmy obliczeniowe, modelowanie behawioralne, przykłady. (Microsystems simulator Dero v4. Computational methods and algorithms, behavioral modeling, examples.)*. AIVA, 2013. ISBN 978-83-937245-1-2.

[17] Plaskura P. *Zaawansowane metody symulacji układów elektronicznych. Metody i algorytmy obliczeniowe. (Advanced methods of electronic circuit simulation. Computational methods and algorithms.)*. AIVA, 2013. ISBN 978-83-937245-0-5.

[18] Plaskura P. *System operacyjny Linux*. AIVA, Piotrków Trybunalski, 2016. ISBN 978-83-937245-2-9. URL `http://epub.aiva.pl/?isbn=978-83-937245-2-9`.

[19] Plaskura P., Ogrodzki J. Circuit simulator OPTIMA v4 - data structure oriented at model processing. *ECCTD 97 Proc.*, 1997.

[20] Plaskura P., Ogrodzki J. *OPTIMA v4 User Manual*. Institute of Electronic Systems, Warsaw University of Technology, Warsaw, 1999.

[21] Saleh R., Jou S.„ Newton A. *Mixed-mode simulation and analog multilevel simulation*. Kluwer Academic Publishers, Norwell, Massachusetts 02061, USA, 1994. ISBN 0-7923-9473-9.

[22] Senturia S. Cad challenges for microsensors, microactuators and microsystems. *Proc. of the IEEE*, 86:1611–1626, 1998.

[23] Senturia S. Simulation and design of microsystems: a 10-year perspective. *Sensors and Actuators*, A67:1–7, 1998.

[24] Stroustrup B. *Projektowanie i rozwój języka C++*. WNT, Warszawa, 1996.

[25] van Duyn D. Modeling and simulation of solid-state transducers: the thermal and electrical energy domain. *Sensors and Actuators*, A41:268–274, Jan 1994.