Algorithms for calculation of electrical circuit state

Alexander Timofeev[1], Pavel Ushakov
[1] Siedlce University of Natural Sciences and Humanities
Institute of Computer Science
Sienkiewicza St. 51, 08-110 Siedlce, Poland

Abstract. Algorithms of calculation of a state of an electric circuit with passive components are described. Algorithms are intended for usage in system of simulation of the mixed electronic devices. Each component of a circuit is presented by program model of high complexity. In this article the resistor model is described. Electric voltages in circuit nodes are calculated by method of summation of volt-ampere characteristics. Component models control iterations at calculation of circuit nodes voltage. Results of calculation of voltage in R-2R ladder of the digital-to-analog converter are presented.

Keywords. Simulation, electrical circuit, resistor model, R-2R ladder.

1 Introduction

Electronic devices contain quite often both digital and analog units. At simulation of such devices the behavior of digital components is imitated and separately simultaneous equations describing analog units are solved.

Analog unit constitutes often a small part of the electronic device. In that case it is represented favourable [1] to use simulation system, in which each part of the mixed electronic device is equally presented.

2 Calculation of the device nodes voltage

Analog and digital components interact between themselves in device nodes. Output signals of analog and digital components can be one-type, if equivalent current generators Jk and equivalent conductivities Gk are matched to contacts of components. Data about currents Jk and conductivities Gk allow to calculate voltage U in node by nodal potentials method: $U = \sum Jk / \sum Gk$.

The described method cannot be applied, if the node is connected to a voltage source because conductivity of a source is equal to zero.

It is possible to bypass the specified exception if to describe contacts by volt--ampere characteristics.

Usage of volt-ampere characteristic and also piecewise-linear approximation allows to describe signals in devices not only with linear, but also with nonlinear characteristics.

On fig. 1 the voltage source characteristic, the resistor characteristic, input and output characteristics of a logic component are presented as examples of volt--ampere characteristics. On presented characteristics the sense of current flowing out of node is accepted as positive.

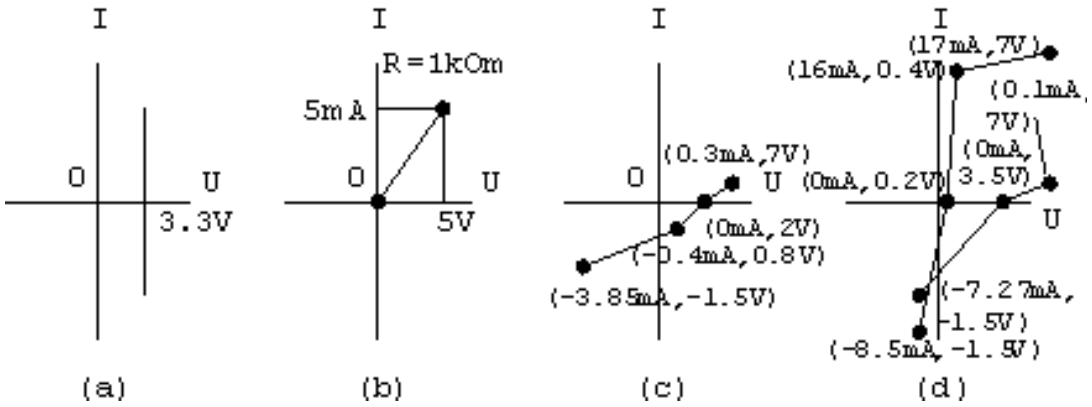

Figure 1. Examples of volt-ampere characteristics: the voltage source characteristic (a), the resistor characteristic (b), input (c) and output (d) characteristics of a logic component

It is possible to present volt-ampere characteristics in modeling programs as sequences of points (I, U) in space "current-voltage" and to store these points in dynamic arrays. The minimum length of such sequence is equaled to two points.

In case of usage of the volt-ampere characteristic the voltage calculation in node should be based on Kirchhoff's law: $\sum Ik = 0$. In the calculation process it is necessary to define voltage at which the sum of the currents flowing into node and flowing out of node is equaled to zero.

On fig. 2 the graphic solution of the problem of voltage calculation on an input of a logic component is showed.

3 Algorithm of voltage calculation by volt-ampere characteristics

On the basis of an example fig. 2 can be offered the following algorithm of voltage calculation by volt-ampere characteristics of the components connected to nodes.

At first it is necessary to check up, whether the contacts which are connected to node and subject to summation, characteristics "a voltage source" possess. If such characteristics there is more than one it is necessary to inform about it to the user and to finish simulation.

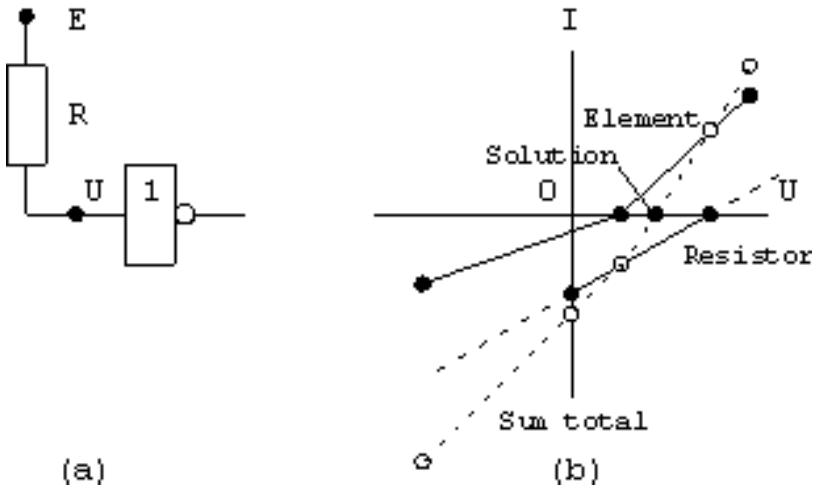

Figure 2. The graphic solution of a problem of voltage U calculation on an input of a logic component: a – circuit, b - volt-ampere characteristics

Clearly that in case among characteristics "the voltage source" is found, the node voltage is equaled to voltage of a source.

At a following stage of algorithm it is necessary to start formation of a resultant volt-ampere characteristic. It is represented favourable to summarize the two first volt-ampere characteristics, and then to add in one volt-ampere characteristic.

Each volt-ampere characteristic is presented by sequence of points (I, U) in which the inclination angle of a linear segment of the characteristic varies. Points (I, U) of different volt-ampere characteristics can have not conterminous co-ordinates U.

By summation of volt-ampere characteristics it is possible to apply strategy of increase of number of points (I, U) in a resultant volt-ampere characteristic in relation to number of points (I, U) in the summands. All unequal co-ordinates U in two summable volt-ampere characteristics will generate corresponding points (I, U) in a resultant characteristic. Such points (I, U) are shown on fig. 2, b.

On fig. 2, b it is visible that on intervals where the given volt-ampere characteristic is not defined, extrapolation is required.

At the algorithm final stage it is necessary to view points (I, U) a resultant volt-ampere characteristic and to find an interval, on which borders points (I, U) have co-ordinates I with different sign. Knowing two such points (-I1, U1) and (I2, U2) we find node voltage: $U = (U1 * I2 + U2 * I1)/(I2 + I1)$.

If at view of a resultant volt-ampere characteristic the interval, on which borders the sign co-ordinates I varies, is not found, it is necessary to extrapolate the left or right extreme interval of the characteristic before crossing with axis U.

4 Representation of connections of components by cyclic lists

In simulation system it is favourable to represent inputs and outputs of the components connected to nodes by cyclic list structures.

Component contact can match structure s_joint containing two member:

inf - the pointer to the dynamic array containing points of the volt-ampere characteristic;

next - the pointer to the next component of the list.

The fig. 3 shows how connections in the simulation system Amethyst [2] using cyclic lists are presented.

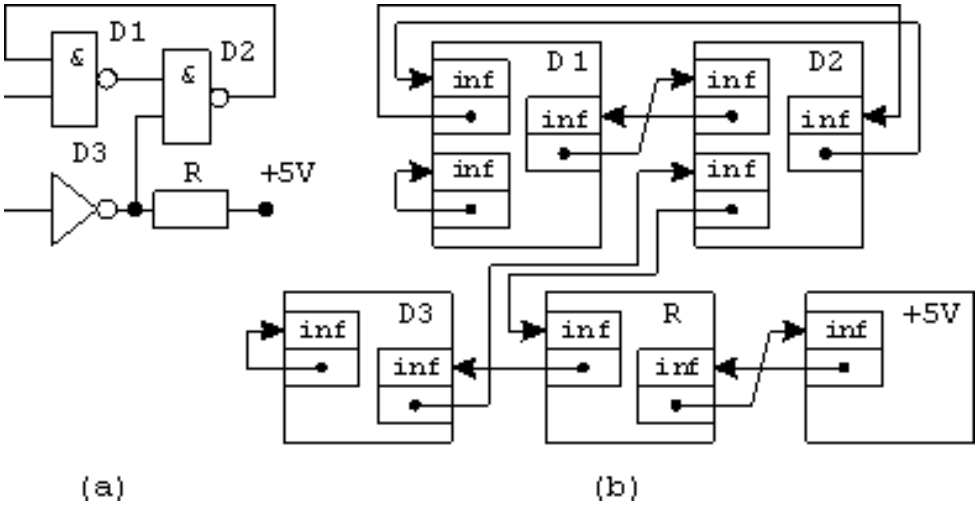

Figure 3. Example of representation of connections of components by cyclic lists: a - the device circuit; b - cyclic lists

If component contact is not connected, the member next of the structure s_joint points to the element of the list with this member, that is "on itself".

In system Amethyst the process of calculation of a new state of model of the device is divided on two phases. In a phase 1 the model of each component has the right to change value of the member inf for each contact, that is to point in this member to other volt-ampere characteristic. And also changes in dynamic array to which the member inf points can be made.

In the phase 2 the model of each component views cyclic lists and calculates voltage value on the contacts by a method of summation of the volt-ampere characteristics specified in members inf.

Possibility of connection of the components belonging to different levels of hierarchy concerns to advantages of representation of connections of components by cyclic lists.

The model of a composite component, that is the component consisting of subitems, should contain models of contacts. The contact model is object of type s_joint with the zero member inf: inf=NULL.

The fig. 4 illustrates usage of models of contacts for connection of two composite components. On fig. 4, a two composite components, in which components D1 are connected to corresponding contacts, are shown.

The fig. 4, b shows, how pointers next in models of contacts were switched and the common list uniting both composite components was formed.

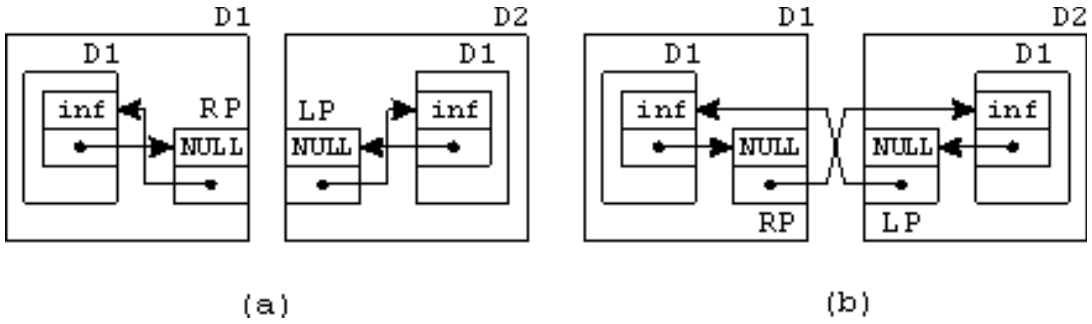

Figure 4. Connection of two composite components:
a - components are standalone; b - components are connected

Procedure of connection of contacts is based on operation of join of two lists:

```
void fConnectTwoPins(s_joint* ppin1,s_joint* ppin2)
{
 s_joint* ptmp=ppin1->next;
 ppin1->next=ppin2->next;
 ppin2->next=ptmp;
}
```

5 Resistor model

Two contacts of the resistor are described by volt-ampere characteristics (fig. 5), position of each of which depends on voltage on opposite contact.

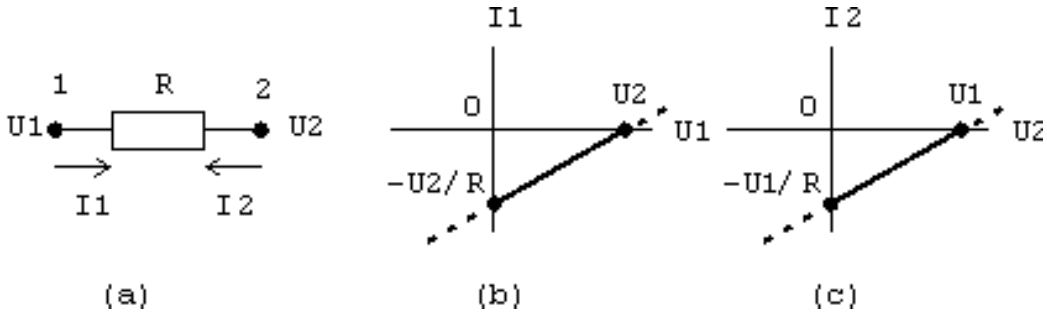

Figure 5. Volt-ampere characteristics (b, c) of the resistor (a)

The volt-ampere characteristic from contact 1 can be described by sequence of points (I,U): $(-U2/R,0),(0,U2)$
and from contact 2 - by sequence: $(-U1/R,0),(0,U1)$.

Such form of characteristics generates difficulties in a case when voltage U2 or U1 is equaled to zero. In case of zero voltage segments of straight lines (fig. 5, b and 5, c) are subtended in a point (0,0).

It is better to enter supplementary parameter Umax - voltage bigger than each of possible in calculations. Using for the description of a straight line two points: one on axes I, another - on line Umax, we have two volt-ampere characteristics:

- from contact 1: $(-U2/R,0),((U\max-U2)/R,U\max)$

- from contact 2: $(-U1/R,0),((U\max-U1)/R,U\max)$

The vector of a state of the resistor model should contain internal variables U1 and U2. Values of the specified variables set volt-ampere characteristics on resistor contacts at calculation of a new state of model in a phase 1.

In a phase 2 the model of the resistor counts new values of variables U1 and U2. In calculation of each of new values the previous value of voltage on opposite contact is used.

Interdependence of voltage on resistor contacts generates iterative character of process of calculation. To operate iterations, it is necessary to compare new and previous values of variables U1 and U2. Value in system Amethyst are compared to the set absolute error.

The inquiry about iteration is realized in system Amethyst as follows. Each model operates time of the nearest event in device model. In case of need of repetition of calculation the model establishes time of the nearest event equal to current modeling time.

The resistor model is realized in the program in language C ++ as class C_Resistor defined as follows:

```
#pragma pack(push,8)
struct s_Resistor
{ //---state---------
  float U1,U2; __int32 iterat;
  s_Resistor(){U1=0.0; U2=0.0; iterat=0;}
};
struct sp_Resistor
{//---parameters---
  float R;
  float dU; //absolute error
  sp_Resistor(){ R=1.0;    dU=0.00001;}
};
#pragma pack(pop)
class C_Resistor : public C_baseComp
{
  static float Umax;
public:
  C_Resistor(); __fastcall ~C_Resistor();
private:
   dynFloatArray* p1Ship;
  dynFloatArray* p2Ship;
public:
  bool fSet();
  bool f1();
  bool f2();
  char* fGetClassName();
};
```

Function f1 is called in a phase 1 and simulates output of signals. Function is realized as follows:

```
bool C_Resistor::f1()
{//----design output value---
  s_Resistor* ps=(s_Resistor*)m_ps;
  sp_Resistor* pp=(sp_Resistor*)m_pp;
```

```
  //-- left pin VAC: (-U2/R,0),((Umax-U2)/R,Umax)
  float J21=-(ps->U2/pp->R);
  float J22=(this->Umax - ps->U2)/pp->R;
  this->p1Ship->fSetAt(0,J21);
  this->p1Ship->fSetAt(1,0.0);
  this->p1Ship->fSetAt(2,J22);
  this->p1Ship->fSetAt(3,this->Umax);
  //--- right pin VAC: (-U1/R,0),((Umax-U1)/R,Umax)
  float J11=-(ps->U1/pp->R);
  float J12=(this->Umax - ps->U1)/pp->R;
  this->p2Ship->fSetAt(0,J11);
  this->p2Ship->fSetAt(1,0.0);
  this->p2Ship->fSetAt(2,J12);
  this->p2Ship->fSetAt(3,this->Umax);
  return true;
}
```

Function f2 is called in a phase 2 and simulates signal reception. Function has the following implementation:

```
bool C_Resistor::f2()
{
  s_Resistor* ps=(s_Resistor*)m_ps;
  sp_Resistor* pp=(sp_Resistor*)m_pp;
  bool flagIteration=false;
  float actualU1,actualU2;
  //--- left pin ---
  s_joint* pleft=(s_joint*)this->apLJoints->Items[0];
  if (fDesignU(pleft,&actualU1)==false)
  { char* pclassname=fGetClassName();
    if (fDialogToContinue(pclassname,"f2()") == false)
      return false;
  }
  //--- right pin ---
  s_joint* pright=(s_joint*)this->apRJoints->Items[0];
  if (fDesignU(pright,&actualU2)==false)
  { char* pclassname=fGetClassName();
    if (fDialogToContinue(pclassname,"f2()") == false)
      return false;
  }
  float delta1=ps->U1 - actualU1;
if (delta1>0 && delta1>pp->dU ||
      delta1<0 && (-delta1)>pp->dU)
  { flagIteration=true;
  }
  else
  { float delta2=ps->U2 - actualU2;
    if (delta2>0 && delta2>pp->dU ||
        delta2<0 && (-delta2)>pp->dU)
    { flagIteration=true;
    }
  }
  if (flagIteration)
  { a_pTime->e=a_pTime->t;
```

```
    ps->iterat++;
  }
  else ps->iterat=0;
  ps->U1=actualU1; ps->U2=actualU2;
  return true;
}
```

It is possible to see a state of model of the resistor in the process of simulation in a state window (fig. 6).

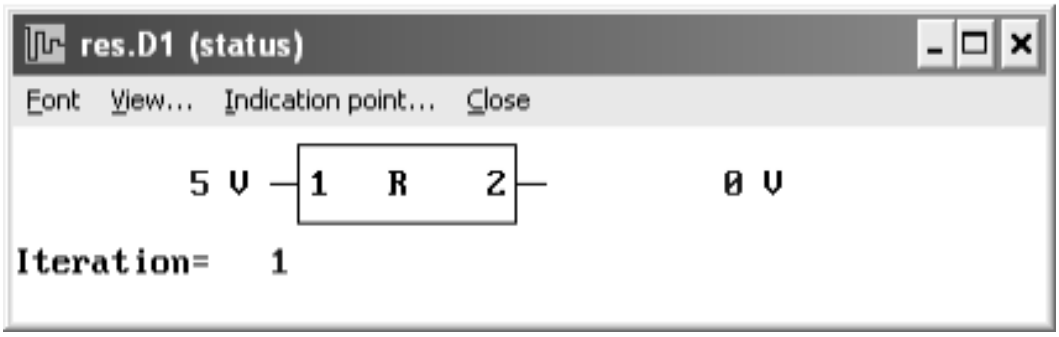

Figure 6. Window of a state of the resistor model

6 R-2R ladder

Calculation of voltage in a network of passive components becomes complicated that characteristics of the passive components having more of one contact depend in strong degree on these voltage.

We will consider arising problems on an example of calculation of voltage in the digital-to-analogue converter on the basis of R-2R ladder.

The simplified circuit of the converter is presented on fig. 7. Resistors 2R and pairs of electronic keys in each bit of the converter are united on the circuit in elements-keys Kj. Elements-keys are operated from inputs Xj. The element-key exit is described by one of two volt-ampere characteristics, presented on fig. 7, b.

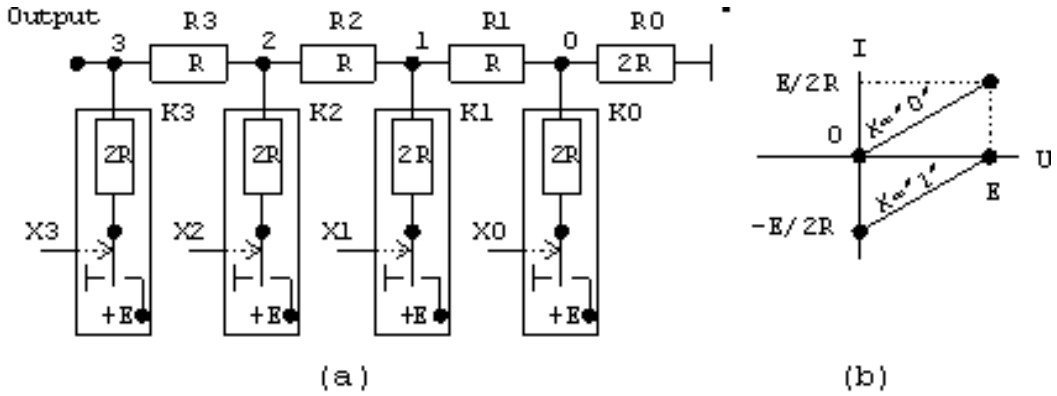

Figure 7. The converter on the basis of divider R-2R ladder:
a - the circuit; b - volt-ampere characteristics of elements-keys

Converter simulation will end successfully if iterative process of calculation of voltage in nodes 3, 2, 1, 0 converges (fig. 7).

As the basis for the conclusion about convergence the following reasoning serves. In linear circuits the rule of superposition of effects operates. Variation in one of converter nodes, for example in node 3, will extend in a considered circuit with attenuation as the variation transmission coefficient to the next node is less than one. The wave of variations will extend from node 3 to node 0, will be reflected and will return to node 3 with appreciable attenuation.

In the table below results of simulation of the converter from the moment when the escape code has varied from value "0000" on value "0001" are presented.

Calculation is fulfilled for a case when supply voltage E is equal to 8 volt, resistance R is equal to 1 kOhm, and absolute error is equal to 0.00001 V.

Iteration	Node voltage				Iteration	Node voltage			
	3	2	1	0		3	2	1	0
0	0	0	0	0	15	0,488	0,741	1,364	2,682
1	0	0	0	2	16	0,494	0,741	1,369	2,682
2	0	0	0,8	2	17	0,494	0,745	1,369	2,685
3	0	0.32	0,8	2	18	0,497	0,745	1,372	2,685
4	0,213	0,32	1,088	2,4	19	0,497	0,747	1,372	2,686
5	0,213	0,521	1,088	2,544	20	0,498	0,747	1,373	2,686
6	0,347	0,521	1,226	2,544	21	0,498	0,749	1,373	2,687
7	0,347	0,629	1,226	2,613	22	0,499	0,749	1,374	2,687
8	0,419	0,629	1,297	2,613	23	0,499	0,749	1,374	2,687
9	0,419	0,687	1,297	2,648	24	0,5	0,749	1,375	2,687
10	0,458	0,687	1,334	2,648	25 - 30	0,5	0,75	1,375	2,687
11	0,458	0,717	1,334	2,667	31 - 35	0,5	0,75	1,375	2,688
12	0,478	0,717	1,353	2,667	Theoretical values				
13	0,478	0,723	1,353	2,677		0,5	0,75	1,375	2,6875
14	0,488	0,732	1,364	2,677					

7 Conclusion

At simulation of the devices containing digital and analog blocks, application of "universal" models, in which linearized volt-ampere characteristics of inputs and outputs are used, is possible.

Volt-ampere characteristics can be allocated in dynamic arrays in the form of sequence of values of currents and the voltage corresponding to points in which the characteristic inclination varies. The pointer to a dynamic array can be a member of the structure, compared to contact of a unit of the device. If other member of structure is the pointer to the following structure the closed chain of structures (the cyclic list) contains the information on the node. This information allows each of the connected models to calculate voltage in the node.

Process of calculation of voltage (pressure) in nodes of model of the device is iterative, and models can control calculation repetition.

References

1. Мельникова С.Ю. (1995). *Разработка и исследование средств смешанного моделирования вычислительных устройств.* Дисс. на соиск. уч. ст. канд. техн. наук. Санкт-Петербургский Гос. Электротехн. Университет. Санкт-Петербург.
2. Timofeev A.O. (2004): *Computer production of programs for simulation of dynamic systems.* Proceedings of the 15th International Conference on Systems Design (7-10 September 2004, Wroclaw, Poland). Vol. 2. Wrocław, Oficyna wydawnicza Politechniki Wrocławskiej, 2004. Pp. 91-95