CORE Metadata, citation and similar papers at core.ac.uk
Provided by Brunel University Research Archive

# Transient Analysis and Synthesis of Linear Circuits using Constraint Logic Programming

**Archana Shankar, David Gilbert, Michael Jampel**

{shankar,drg,jampel}@cs.city.ac.uk

**Department of Computer Science, City University**

**London, UK**

In this paper describes the design of a transient analysis program for linear circuits and its implementation in a Constraint Logic Programming language, CLP(R). The transient analysis program parses the input circuit description into a network graph, analyses its semantic correctness and then performs the transient analysis. The test results show that the program is at least 97% accurate when run at two decimal places. We have also compared the performance of our program with a commercial package implemented in an imperative language. The advantages of implementing the analysis program in a CLP language include: quick construction and ease of maintenance. We also report on the synthesis of generation of a circuit with given transient characteristics.

## 1. INTRODUCTION

Electrical circuit analysis is done in order to allow the designer to verify his design and to predict the response of the system under varying conditions of load and excitation.

A circuit can be analysed at different points in time. Whenever a switch is closed in a circuit, the voltages and currents in the circuit take some time to settle down to their final values; the components of voltages and currents that die down are called as *transients*. *Transient analysis* of a circuit is done *at the time of switching* to study the effects of the transients as well as to determine the time taken by the system to settle down.

The behaviour of the circuit as a function of time is studied under transient analysis. The inductors in the circuit are replaced by their equivalent current sources and resistances, and the capacitors in the circuit are replaced by their equivalent voltages sources and resistances. The circuit voltages and currents are calculated at the time of switching (usually at $t = 0$); this is the initial condition solution. The voltages across the capacitors and the currents across the

Inductors are used to calculate the circuit voltages and currents at each time step; this is done repeatedly for a designated amount of time and the results are then plotted. Constraints provide an elegant means of stating relationship among objects that should be maintained by the underlying system.

## 2. BACKGROUND

Electrical *circuits* consist of electrically (and sometimes magnetically) connected building blocks. Examples of building blocks are resistors, inductors, capacitors, voltage sources and current sources. The calculation of voltages and currents in a circuit and its response to input excitations form the basis for analysing a network. The mathematical tools involved in analysis are the basic voltage-current relationships for the circuit elements, rules for generating simultaneous equations, and solution procedures for these equations. The tools involved in analysis are the basic voltage-current relationships for the circuit elements, rules for generating simultaneous equations, and solution procedures for these equations.

Circuits in which radiated electromagnetic energy may be assumed negligible are called **lumped circuits** and they are obtained by interconnecting **lumped component circuit elements**. The lumped component circuit elements are idealised models of physical elements. **The circuit analysed is therefore not the physical circuit, but its mathematical model.**

The node method consists of summing the currents at each node in the network with unknown voltages at the nodes and solving the resultant equations for the voltages.

The basic relations for circuit analysis are the first two laws of Kirchoff :

1. The sum of all currents entering a node must equal the sum of all currents leaving it (current law or KCL).

2. The sum of all voltages in a given loop must be equal to zero (voltage law or KVL).

One or the other of the above laws is applied to every independent node or independent loop of the network in analysis. The voltage-current relationships for passive elements in a linear circuit are characterised by $i = y.v$ with $y$ = 1/R for a resistor, 1/L for an inductor, or C for a capacitor where R is the resistance, L is the inductance, C is the capacitance, and $y$ is the admittance; in the AC analysis, $i$ and $v$ are complex valued phasor representations for the current and voltage.

## 3. ANALYSIS OF A NETWORK

The elements that are used in this analysis is a minimum subset of elements that are present in a network ; those used in our program comprise *passive elements* - resistance (R), inductance (L), capacitance (C), and *active elements* (Sources) - independent voltage sources (V) and independent current sources (I).

In our analysis we will consider only accompanied voltage sources as unaccompanied sources present difficulties when forming the branch admittance matrices. Since the accompanying resistance R is zero, the admittance (1/R), becomes infinite and the current through the branch becomes indeterminate.

The voltage-current relationships for the passive elements lead to the matrix equation

$\mathbf{I}_e = \mathbf{Y}_e.\mathbf{V}_e$

where $\mathbf{Y}_e$ is a diagonal matrix of M rows and M columns if the network consists solely of resistors, inductors and capacitors.

### 3.1. THE INDUCTOR MODEL

The voltage-current relationship for an inductor of value L is

$$i(t) = \frac{1}{L}\int_{t=0}^{t} v(t)dt + i(0)$$

where i(0) is the current in the inductor at time t=0. The voltage values are calculated at equally spaced time intervals $t_0$, $t_1$ ... $t_{k-1}$, $t_k$

The integral can be replaced by trapezoid approximation

$$\int_{t=0}^{t} v(t)dt \approx \frac{1}{2}\Delta t(v_{k-1} + v_k)$$

with Δt being the time step. The current is now a linear relationship

$i_k = i_{k-1} + ½\, \Delta t\, (v_{k-1} + v_k)$

The last equation is of the form

i = I + g.v

The value of g will not change during the entire calculation. However, the value of I must be readjusted at every new time value.

## 3.2. THE CAPACITOR MODEL

The voltage-current relationship for a capacitor C is

$i(t) = C dv(t)/dt$

The current at time $t = t_k$ is

$i_k = C\, dv/dt \,|_{t=tk}$

The slope of the voltage vs. time curve may again be approximated. Once such approximation is

$dv/dt \,|_{t=tk} \approx 1/\Delta t\, (v_k - v_{k-1})$

Hence the capacitor current is approximated by

$i_k = C/\Delta t\, (v_k - v_{k-1})$

The last equation is of a conductance of value C/Δt in series with a voltage source $v_{k-1}$ .

Since each branch contains only one component, the element currents $\mathbf{I}_e$ and element voltages $\mathbf{V}_e$ may be separated as follows:

$$\mathbf{I}_e = \begin{bmatrix} \mathbf{I}_L \\ \mathbf{I}_R \\ \mathbf{I}_C \end{bmatrix} \qquad \mathbf{V}_{\mathbf{e}} = \begin{bmatrix} \mathbf{V}_L \\ \mathbf{V}_R \\ \mathbf{V}_C \end{bmatrix}$$

where $\mathbf{I}_L$, $\mathbf{V}_L$ refer to the currents and voltages across the inductors in the network; $\mathbf{I}_R$, $\mathbf{V}_R$ refer to those across the resistors in the network and $\mathbf{I}_C$, $\mathbf{V}_C$ refer to those across the capacitors in the network.

This derivation assumes that the branches in the network graph are numbered such that the inductors are in a group of branches followed by the resistors and then by branches containing the capacitors.

The currents and voltages of the various elements are related as follows:

$\mathbf{I}_R = [1/R]\, \mathbf{V}_R$

$\mathbf{I}_C = [C]\, 1/\Delta t\, (\mathbf{V}_{Ci} - \mathbf{V}_{Ci-1})$

$\mathbf{I}_{Lk} = [L]^{-1}\Delta t/2\, \mathbf{V}_{Lk} + [L]^{-1}\Delta t/2\, \mathbf{V}_{Lk-1} + \mathbf{I}_{Lk-1}$

Hence the element currents at time $t_k$ are given by

$I_{ek} = \mathbf{Y}_e . \mathbf{V}_{ek} + \mathbf{I}_{pk-1}$

The vector $\mathbf{I}_{pk-1}$ contains all the past history of the inductors and capacitors in the circuit.

$$\mathbf{I}_{ek} = \begin{bmatrix} (\Delta t/2)[L]^{-1} & 0 & 0 \\ 0 & [1/R] & 0 \\ 0 & 0 & (1/\Delta t)[C] \end{bmatrix} \begin{bmatrix} \mathbf{V}_{LK} \\ \mathbf{V}_{RK} \\ \mathbf{V}_{CK} \end{bmatrix} + \begin{bmatrix} (\Delta t/2)\,[L]^{-1} * \mathbf{V}_{LK} + \mathbf{I}_{LK-1} \\ 0 \\ -(1/\Delta t)[C] * \mathbf{V}_{ck-1} \end{bmatrix}$$

At time *t* = 0, the vector $\mathbf{I}_{pk-1}$ does not exist. For this, the initial condition, the inductor is replaced by a very small conductance in parallel with the current source denoted by $I_0$, the initial current across the inductor. The capacitor is replaced by a small resistance in series with the voltage source $V_0$, the initial voltage across the capacitor. Then, the capacitor voltages and inductor currents are solved, and these form the basis for calculating the first $\mathbf{I}_p$, so that normal circuit equations may be formed for $t = \Delta t$.

## 4. DESIGN

The transient analysis program is implemented as an analytical tool (an interpreter) to simulate the behaviour of an electrical circuit. Nodal analysis of the network (i.e. calculation of the currents and voltages at different nodes (location) in the network) is carried out and the results are output to the user. Nodal analysis is based on the algebraic manipulations of matrices.

Therefore any input given by the user has to be suitably transformed before the actual analysis can take place. In this program, one of the inputs is a file containing a description of the circuit topology and the other inputs are the analysis time, the nodes at which output voltages should be calculated and the file to which CLP(R) should write the output. The transformations take place in the parser and the analyser, the voltages and currents are calculated in the interpreter, and the results are displayed as output.

The interpreter is the process which does the actual transient analysis of the given circuit. It requires as inputs, the circuit topology the time scale for which the simulation should run and the desired output from the program. The circuit topology is described using a Circuit Definition Language defined by us for this purpose. A parser checks the inputs and reports any syntactic errors back to the user and the output from the parser is in the form of a network graph.

The network graph is then passed on to the analyser. The analyser performs a semantic analysis of the network, reporting any errors back to the user. It also converts the network graph into matrices and passes them on to the interpreter. The interpreter has access to CLP(R) and to a purpose built library implementing various matrix operations as an abstract data type. The interpreter calculates and outputs the dynamic behaviour of the circuit (i.e. the change of node voltages over a period of time as specified by the user).

Apart from the circuit description, the user has to input other information such as the time step, the start time and end time of analysis, the output nodes which are of interest as well as a file name to which CLP(R) can write the output.

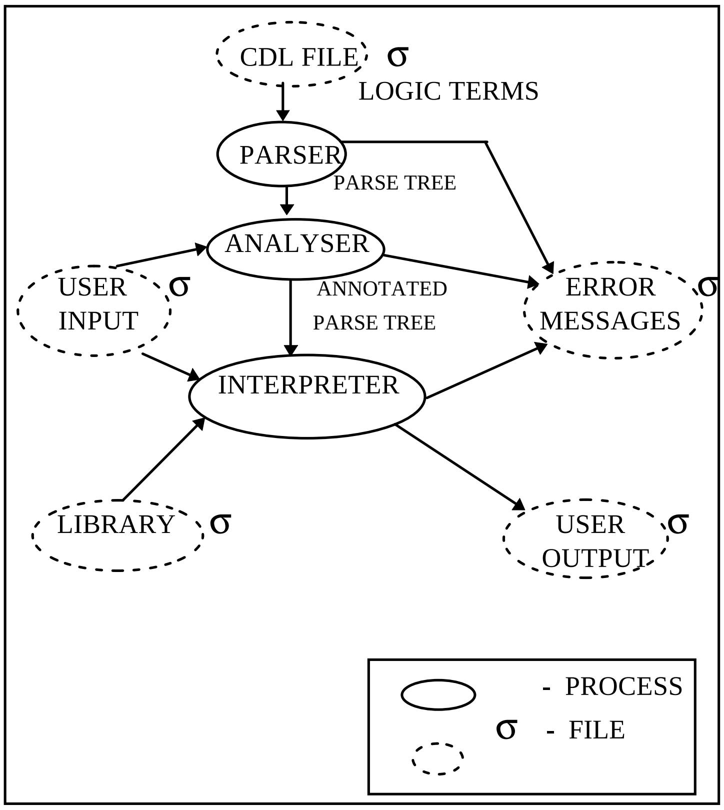


Figure 1 Transient analysis program structure

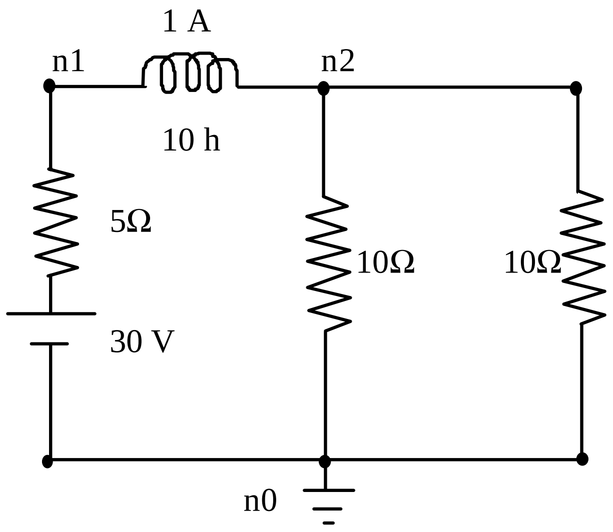
## 4.1. CIRCUIT DEFINITION LANGUAGE

The circuit details must be presented in a specific manner and in this program, this is done using a language called the circuit definition language which will be defined in this section.

The language has to represent in a unique way all the circuit elements and has to distinguish between the active elements such as the current source and the voltage source (the drivers), and the passive elements such as the inductors, the capacitors and the resistors.

The connection of each element in the network is indicated by the two nodes to which it is connected. A node is represented by the letter n followed by a digit. Numbering of the nodes starts from 0. In electrical engineering, voltages are measured as the potential difference between two any two nodes. When one of such nodes is at zero potential it is called as the ground and is represented as n(0). All voltages are measured relative to the ground. An example is:

```
circuit(  [voltage_source(v1,30,5,n(0),n(1))],
                    [inductor(l1,10,1,n(2),n(1)),
        resistor(r1,10,n(0),n(2)),
        resistor(r2,10,n(2),n(0))] ).
```



### 4.2. PARSER

The parser reads the input file and converts it into a network graph. The network graph we use is represented by two sets: the set of independent nodes in the circuit and set of branches or arcs in the circuit. Each branch in the circuit is given a number by the parser. This number is used later on by the analyser to form the network matrices and then by the interpreter to manipulate the matrices. The branches containing inductors in the circuit are numbered first, followed by the resistors, the capacitors, the voltage sources and finally the current sources. The output of the parser is a positionally ordered set of arcs and an ordered set of nodes. The parse is not unique for the arcs as the ordering in the set of arcs depends on the ordering in the input file. However, the time taken to parse the file does not depend on the ordering of the input file.

### 4.3. ANALYSER

The analyser transforms the inputs i.e. the set of arcs and the set of nodes into matrices. In addition it also does a semantic analysis of the network, reporting errors back to the user. The matrices form the foundation for the nodal analysis of the network and so the analyser plays an important part in the program.

The outputs of the analyser are the incidence matrix (A), the admittance matrices (Yb and Yb0), the independent current matrix (Ig) and the independent voltage matrix (Vg). The product of the incidence matrix and the branch current matrix represents the network graph at the end of the analysis process. In the semantic analysis, the analyser checks to see if the circuit is connected properly and detects open circuits. The analysis is done by checking that the number of elements in the arc list is greater than the number of elements in the node list. If the node list is the larger list, the incidence matrix is used to locate which of the branches are open. The results of the semantic analysis are output to the user.

### 4.4. SEMANTIC ANALYSIS OF THE NETWORK

The circuit definition file is context sensitive due to the inclusion of the nodes in the language. Therefore, the file has to be checked to ensure that it is correct semantically. In semantic analysis, the network is checked to ensure that it is connected properly i.e. the branches are all connected to one another and there are no open circuits. The incidence matrix and the network graph are used in the check. In a properly connected circuit, the number of nodes is always less than or equal to the number of elements. Hence, the first check is made on the number of arcs and the number of nodes in the network graph. If the nodes are greater in number than the arcs, it implies that the circuit is not properly connected. If the network is open, the incidence matrix is used to determine which of the branches are open. This can be found out from the fact that every node has at least two connections to it. In other words, every row in the incidence matrix should have more than one non-zero column in it. If a row has just one entry that is non-zero, then that node is the open node.

### 4.5. INTERPRETER

The interpreter is the process that performs the actual transient analysis. The matrices from the analyser and the time input by the user are the inputs to this process. The interpreter steps through the analysis, one time step at a time from the initial time $T_{ini}$ to the final time $T_{fin}$ calculating the currents and voltages required by the next iteration. The time step is represented by $\delta t$. The past history of the capacitors and the inductors are represented by the matrix Ip. At each iteration, the output voltages and currents are accumulated and at the end of the analysis, are sent to the output process. The interpreter makes use of the operations defined in the matrix library to calculate the voltages and currents. The output process is used to output the results of the analysis. The node voltages of interest to the user are by default printed on to the screen for each time step. A copy of the output is also written to an output file at the request of the user.

### 4.6. MATRIX LIBRARY

To implement the abstract data type of matrices, operations such as multiplication, addition, subtraction, transpose, determinant and inverse have been defined. The definition of matrix inverse is: if A is the matrix to be inverted, I is a unit matrix, and AxB = I , then B is the inverse of A. This can be implemented very elegantly and economically in CLP(R) by making use of the multi-directionality of logic programs. Since matrix multiplication has already been defined, it can be used to find the inverse and a separate predicate need not be defined. This is also very efficient.

## 5. TESTING

The transient analysis program was tested using the examples given in Staudhammer [STAU75]. For the analysis, the inductors and capacitors in the circuits are replaced by their equivalent models as discussed in Section 2. The calculations start with the determination of the initial conditions existing in the network, prior to the application of the time-iterations. The initial conditions should be consistent to gain a meaningful result: for example, if we replace a 0.5 farad capacitor by two 1 farad capacitors, we would expect the initial voltage across the two separate capacitors to be equal.

As an example, let us consider the RL circuit of Fig. 3-2. The inductor connected between node 1 and node 2 has an initial current of 1 ampere flowing through it. The solution to this network is given by the following equations in [STAU75]:

$$V_1(t) = 15 + 10e^{-T/\tau}$$

$$V_2(t) = 15 - 10e^{-T/\tau}$$

with $\tau = L/R_T = 1$ since, $R_T$, the total resistance that exists in the current loop of the network is equal to 10 ohms.

The analysis was carried out with an initial time of 0, a time step of 0.1, and a final time of 5 seconds. Table 1 shows a comparison between the expected results (V1 and V2) and the actual results obtained (n(1) and n(2) represent the voltages at node 1 and node 2 respectively) for selected values, presented at time intervals of 0.5 seconds for reasons of clarity.

| Time (sec) | V1 (volts) | n(1) (volts) | V2 (volts) | n(2) (volts) |
|---|---|---|---|---|

| Time (sec) | V1 (volts) | n(1) (volts) | V2 (volts) | n(2) (volts) |
|---|---|---|---|---|
| 0.0 | 25.00 | 25.00 | 5.00 | 5.00 |
| 0.5 | 21.06 | 21.06 | 8.94 | 8.94 |
| 1.0 | 18.68 | 18.68 | 11.32 | 11.32 |
| 1.5 | 17.23 | 17.23 | 12.77 | 12.77 |
| 2.0 | 16.35 | 16.35 | 13.65 | 13.65 |
| 2.5 | 15.82 | 15.82 | 14.18 | 14.18 |
| 3.0 | 15.50 | 15.50 | 14.50 | 14.50 |
| 3.5 | 15.30 | 15.30 | 14.70 | 14.70 |
| 4.0 | 15.18 | 15.18 | 14.82 | 14.82 |
| 4.5 | 15.11 | 15.11 | 14.90 | 14.89 |
| 5.0 | 15.07 | 15.07 | 14.93 | 14.93 |

**Table 1- Comparison of expected results with actual results**

Comparison of the expected results with the actual results shows that the program is 99.99% accurate at 2 decimal places; the only discrepancy occurring when time is equal to 4.5 seconds. See [SHAN95] for detailed results of this and other examples, where the minimum accuracy is 97%.

The test results show that CLP(R) can be used to perform transient analysis of circuits with accuracy and ease. The response time of CLP(R) is very fast. For example, in the test, CLP(R) had to solve 700 equations and this was done within 2 seconds.

## 6. BACKWARD EVALUATION

One of the main benefits of implementing the transient analysis program in CLP(R) is that in addition to the analysis, we can use the same program for design. For example for the circuit in Fig 3-2, we can query the program as to what should be the value of the voltage source and the initial current across the inductor so that the node voltages at n1 and n2 are 25 and 5 respectively.

```
circuit([voltage_source(v1,V,5,n(0),n(1))],
        [inductor(l1,10,I,n(2),n(1)),
        resistor(r1,10,n(0),n(2)),
        resistor(r2,10,n(2),n(0))]).
```

The program comes out with the answer `V = 30` and `I = 1`. To accomplish the same result in an imperative language either a different program has to be written or the simulation has to be run several times with different values of V and I and the answer to the question found by trial and error. We note that due to the fact that the transient analysis maps inductors and capacitors to resistors, the generation of a circuit given an expected transient behaviour requires the provision of a skeleton circuit which describes what type of components are in it. Our method then is able to generate values for the skeleton components.

Currently we are extending the system to permit the description of transient behaviour as constraints within ranges rather than absolute values. At the same time we are enhancing our Circuit Definition Language to permit the description of circuits using constraints in the form of ranges and relationships between values. In this way we permit the synthesis of circuits with constrained values from a required constrained description of its behaviour. This will permit circuit designers to select the tolerances of circuit components, and thus potentially to ensure that circuit components used have only the required level of tolerance, thus potentially saving manufacturing costs.

## 7. COMPARATIVE TEST

The time taken by the transient analysis program to evaluate a 13 element circuit was compared against the time taken by PSPICE to evaluate the same circuit. A larger example was used in order to get better precision in comparison. See [SHAN95] for full details.
PSPICE is designed to be the micro computer version of SPICE, the acronym for Simulation Program with Integrated Circuit Emphasis, developed by the University of Berkeley, California.

PSPICE took about 5 seconds to analyse the circuit while our program took about 16 seconds. The tests were run on a 486 DX machine and the time taken by the two programs is the total time and not CPU time alone. The reasons for the slower response time could be as follows: firstly our program being a prototype, is not optimised for efficiency and speed while PSPICE is an optimised commercial package. Secondly, our program is not compiled, but is interpreted by the CLP(R) interpreter while PSPICE is an compiled executable. There could also be a penalty for calculating inverses at each time step. Since our matrix library is a standalone module, it can be replaced by other faster more efficient library routines.

## 8. RELATED WORK

CLP(R) has been used to in the design and analysis of circuits by [HEIN92] and also used by [FATT94] to model dynamic systems in general.

In [HEIN92], CLP(R) was used for the analysis of circuits such as RLC circuits, transistor circuit design etc. Steady state, or static analysis, is done on RLC circuits containing voltage sources, current sources, resistors, inductors and capacitors.
A general purpose approach has also been adopted by [FATT94] to model dynamic systems using bond graphs and CLP(R). The structure of the dynamic system is described using a bond graph language. The circuit is analysed using state space analysis and the resulting differential equations are solved using a relational differential equations solver implemented in CLP(R).
In [HONG94] a solver for differential equations described, included an example program to solve simple relationships over electrical circuits. We believe that this system could provide a good implementation platform for extensions to our work.

## 9. FURTHER WORK

Our work has demonstrated that CLP(R) and its multi-directional capabilities can be used in a particular application domain which is generally considered to be more suitable for imperative languages. We concentrated on the DC analysis of linear RLC networks and as it stands can be used as a teaching aid for undergraduates. To make it a commercially viable package the following have to be added:

- Providing an integrated windows environment for the program
- Addition of further circuit elements to the basic set considered here.
- Adding constraints to the circuit definition language (the circuit description) used in the program.

Adding constraints to the input and output and implementing the program in an integrated graphics environment will be of great use to the designer since the cost of circuit components is related to the tolerances (i.e. constraints) to which they are manufactured.

## 10. SUMMARY AND CONCLUSIONS

This paper investigates the implementation of transient analysis of linear circuits in constraint logic programming, and provides a declarative alternative to the procedural approach usually adopted to model electrical circuits.

The start time, end time and time step of the analysis are user-provided, and the nodes for which the voltages and currents are calculated. The parser converts the circuit description into a network graph represented by two sets - the set of arcs and the set of nodes. The analyser checks the network graph for open connections and converts the network graph into matrices required by the interpreter. The interpreter steps through the analysis calculating the required voltages and currents and then informs the user about the results of the analysis. The tests conducted show that the accuracy of the program was 99% at two decimal places.

We have also shown that implementing the analysis package in CLP(R) enables us to use the same package for *designing* a circuit as well. Using the desired output as the constraint, the necessary inputs to the circuit were calculated. Multi-directionality of CLP(R) was also used to define the inverse of a matrix in terms of multiplication alone, thereby avoiding the task of defining a new predicate to calculate the inverse.

Future work should include additional elements such as transformers and dependent sources, thereby extending the range of circuits that can be analysed. Another important extension would be the inclusion of constraints as part of the circuit description itself.

The process of implementing the program in CLP(R) has enabled us to reason about the circuit itself as well as the analysis, thereby increasing our understanding of the problem. The declarative language used has proved to be very useful in the implementation. Backward evaluation makes the analysis tool a design tool as well. Given a known output, the program was able to deduce the values of the input excitation. The same technique was used to find the inverse of a matrix. Coding of the entire program was followed easily from the logical design. The code is readable and hence maintenance is easy. Given that our system was a interpreted prototype, its performance compares very well with a commercial package which has been implemented in a compiled imperative language.

Finally, an advantage of our approach is that we are able to generate instances of circuits from general schemas and descriptions of their required behaviour.

## 11. REFERENCES

[BOUT88] Boute Raymond T., 'Systems Semantics : Principles, Applications, and Implementation' in *ACM Transactions on Programming Languages and Systems*, Vol. 10, No. 1, January 1988, pp. 118-155.

[FATT94] El Fattah Y. and Holzbaur C., 'Constraint Logic Programming for Modelling and Simulation of Dynamic Systems', in '*Proceedings ILPS '94 workshop on Constraint Languages/Systems & their use in Problem Modelling*', Vol 1, Nov 1994.

[HONG94] Hoon Hong, 'RISC-CLP(CF) Constraint Logic Programming over Complex Functions', LPAR94

[JAFF87] Jaffar J. and Lassez J.L., 'Constraint Logic Programming', in *Proceedings of the Fourteenth ACM Principles of Programming Languages Conference*, Munich, January 1987.

[JAFF92] Jaffar J., Michaylov S., Stuckey P., and Yap R., 'The CLP(R) Language and System' in *ACM Transactions on Programming Languages and Systems*, Vol 14 No 3, July 1992, pp. 339-395.

[HEIN92] Heintze N., Michaylov S., Stuckey P., 'CLP(R) and Some Electrical Engineering Problems' *Journal of Automated Reasoning*, Vol 9, 1992, pp 231-260.

[NERI70] Nering E.D., *Linear Algebra and Matrix Theory*, John Wiley and Sons Inc., New York, 1970.

[NILS93] Nilsson J.W., and Riedel S.A., *Introduction to PSPICE,* Addison-Wesley Publishing Company Inc., 1993.

[STAU75] Staudhammer John, *Circuit Analysis by Digital Computer* Prentice-Hall Inc., 1975.

[SESH63] Seshu S., and Balabanian N., *Linear Network Analysis*, John Wiley & Sons, Inc., New York, 1963.

[SHAN95] Shankar A., Gilbert D., Jampel M., *Transient Analysis of Linear Circuits using Constraint Logic Programming*, Technical Report TCU/CS/95/17, City University Computer Science Department, London,