TR-698

Decision of Design Parameters using Qualitative Reasoning and its Application to Electronic Circuits

by
M. Ohki, E. Oohira, H. Shinjo &
M. Abe (Hitachi)

October, 1991

Decision of Design Parameters using Qualitative Reasoning and its Application to Electronic Circuits

Masaru Ohki, Eiji Oohira, Hiroshi Shinjo, and Masahiro Abe

Central Research Laboratory, Hitachi, Ltd.
Higasi-Koigakubo, Kokubunji, Tokyo 185, Japan

Abstract

There are many applications of qualitative reasoning to engineering fields. The main application field is diagnosis, and there are also several applications to design. We show a new application to design, supporting decisions by suggesting valid ranges of design parameters after designing structures. We use the envisioning mechanism, which determines all possible behaviors of a system using qualitative reasoning. Our method: (1) performs envisioning with design parameters whose values are initially undefined, (2) selects preferable behaviors from all possible behaviors found by envisioning, (3) calculates the ranges of the design parameters that give the preferable behaviors. We built a design-supporting system Desq (Design supporting system based on qualitative reasoning) by improving an earlier qualitative reasoning system Qupras (Qualitative physical reasoning system). We added three new features: envisioning, calculating the undefined parameters, and propagating new constraints on constant parameters. Desq can deal with quantities qualitatively and quantitatively, like Qupras. Accordingly, we may be able to find not qualitative but quantitative ranges, if the parameters can be expressed quantitatively. Quantitative ranges are preferable to qualitative ones for supporting the decision of design parameters.

1. Introduction

Recently, many expert systems have been used in engineering fields. However, several problems still exists. One is the difficulty in building knowledge bases from the experience of human experts. The other is that these expert systems cannot deal with unimagined situations [Mizoguchi 87]. Reasoning using deep knowledge, which is the fundamental knowledge of a domain, is expected to solve these problems. One type of reasoning is qualitative reasoning [Bobrow 84]. Qualitative reasoning determines the dynamic behaviors, which are the states of a dynamic system and state changes, using deep knowledge of the system. Another feature of qualitative reasoning is that it can deal with quantities qualitatively. So far, there have been many applications of qualitative reasoning to engineering [Nishida 88a, Nishida 88b, Nishida 91]. The main field is diagnosis [Yamaguchi 87, Ohwada 88], but recently there have also been applications to design [Murthy 87, Williams 90].

In this paper, we show a new application to design that supports decisions by suggesting valid ranges of design parameters after designing structures. This application is not more innovative than previous applications to design [Murthy 87, Williams 90], but it is one of the important steps of design [Chandrasekaran 90]. Envisioning predicts all possible behaviors of a dynamic system. The valid ranges of design parameters are found as follows:

(1) Perform envisioning with design parameters whose values are initially undefined,
(2) Select preferable behaviors from possible behaviors found by envisioning,
(3) Calculate the ranges of the design parameters that give the preferable behaviors.

We used a qualitative reasoning system Qupras (Qualitative physical reasoning system) [Ohki 86, Ohki 88, Ohki 91] to make a decision support system Desq (Design supporting system based on qualitative reasoning) which suggests valid ranges of design parameters. Using knowledge about physical rules and objects after being given an initial state, Qupras determines:

(1) Relations among objects that are components of physical systems.
(2) The next states of the system following a transition.

We extended Qupras to make Desq as follows:

(1) Envisioning
In Qupras, if a condition of a physical rule or an object cannot be evaluated, Qupras asks users to specify the condition. We extended Qupras to allow it to continue assuming an unevaluated condition.
(2) Calculating the undefined parameters
After envisioning all possible behaviors, Desq calculates the ranges of the undefined design parameters that give the behavior specified by the designer.
(3) Propagation of new constraints on constants

In envisioning, constraints related to some constant parameters become stronger because conditions in physical rules and objects may be hypothesized in envisioning. The constraints propagate to the next states.

(4) Parallel constraint solving

Qupras uses a combined constraint solver which consists of three basic constraint solvers: a Supinf method constraint solver, an Interval method constraint solver, and a Groebner base method constraint solver, all written in ESP. The processing load of the constraint solvers was heavy, so we converted them in KL1 to speed them up.

Desq can deal with quantities qualitatively and quantitatively like Qupras. Accordingly, we may be able to get not qualitative but quantitative ranges, if parameters can be given as quantitative values. Quantitative ranges may be preferable for the decision support. The usual qualitative reasoning like [Kuipers 84] only gives qualitative ranges.

Section 2 shows how Desq suggests ranges of design parameters, Section 3 describes the system organization of Desq, Section 4 shows an example suggesting the value of a resistor in a DTL circuit, and Section 5 summarizes the paper.

2. Method of deciding design parameters

In design, there are many cases in which a designer does not directly design a new device, but changes or improves an old device. Sometimes designers only change parameters of components in a device to satisfy the requirements. The designer, in such cases, knows the structure of the device, and needs to determine the new values of the components. This is common in electronic circuits. We apply qualitative reasoning to the design decisions.

The key to deciding design parameters is envisioning. The method is as described in Section 1:

(1) All possible behaviors of a device are found by envisioning, with design parameters whose values are initially undefined.

(2) Designers select preferable behaviors from these possible behaviors.

(3) The ranges of the design parameters that give the preferable behaviors are calculated using a parallel constraint solver.

If a condition of a physical rule or an object cannot be evaluated, Desq hypothesizes one case where the condition is valid and another where it is not valid, and separately searches each cases to find all possible behaviors. This method is called envisioning, and is the same as [Kuipers 84]. If a contradiction is detected, the reasoning is abandoned. If no contradiction is detected, the reasoning is valid. Finally, we get several valid reasonings, and Desq finds several possible behaviors of a device.

The characteristics of this approach are as follows:
(1) Only deep knowledge is used to decide design parameters.
(2) All possible behaviors with regard to undefined design parameters are found. Such information may be used in safety design or danger estimation.
(3) Ranges of design parameters giving preferable behaviors are found. If a designer uses numerical CAD systems, for example, SPICE, he/she need not simulate values outside the ranges.

Figure 1 shows an example for suggesting ranges of a design parameter. This example is designing a resistance value in a DTL circuit. The designer inputs the DTL structure and the parameters of the components except for the resistance Rb.

Desq checks the conditions of physical rules and objects. If they are satisfied, equations in their consequences are sent to the parallel constraint solvers. But, it is not known what state the diode D1 is in, because the resistance Rb is undefined. The condition is whether the voltage over D1 is lower than 0.7 volts. Desq hypothesizes two cases; in the first the condition is not satisfied, and in the second it is. The first hypothesis is abandoned because the parallel constraint solver detects a conflict with the other equations. In the second hypothesis, no conflict is detected. After some more hypotheses are made, it is not known whether a condition giving the state of the transistor Tr is satisfied. Desq similarly hypothesizes this condition. Finally, Desq finds two possible behaviors for the initial data. Then, Desq calculates the resistance Rb. The resistance must be larger than 473 ohms to give the desired behavior, where the circuit acts as a NOT circuit because the transistor is "on". If the resistance is smaller than 473 ohms, the circuit shows another behavior which is unpreferable. Thus the resistance Rb must be larger than 473 ohms. This shows the advantages that Desq deal with quantities qualitatively and quantitatively.

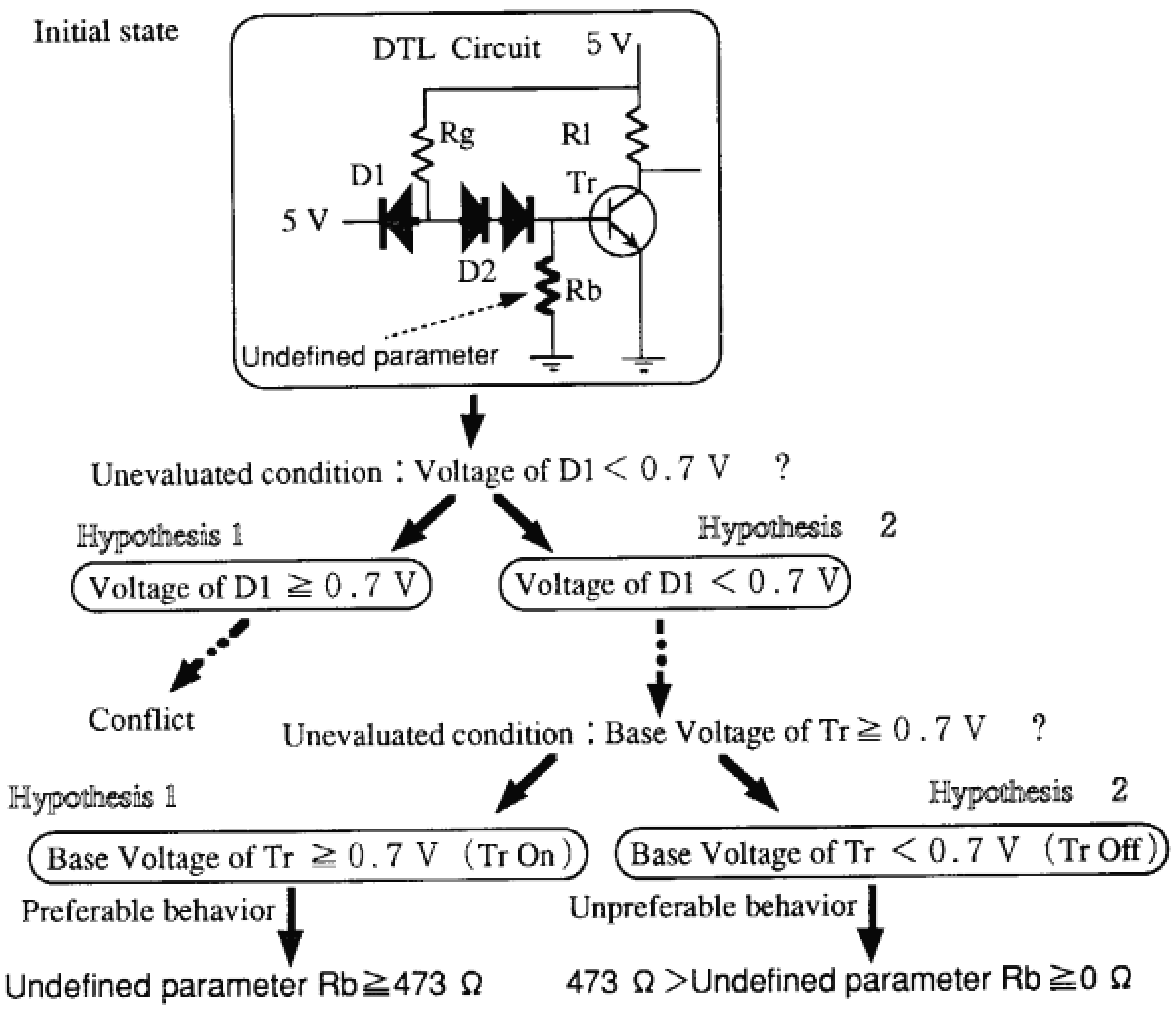
Initial state
DTL Circuit
5 V
Rg
Rl
D1
Tr
5 V
D2
Rb
Undefined parameter
Unevaluated condition : Voltage of D1 < 0.7 V ?
Hypothesis 1
Voltage of D1 ≧ 0.7 V
Hypothesis 2
Voltage of D1 < 0.7 V
Conflict
Unevaluated condition : Base Voltage of Tr ≧ 0.7 V ?
Hypothesis 1
Base Voltage of Tr ≧ 0.7 V (Tr On)
Hypothesis 2
Base Voltage of Tr < 0.7 V (Tr Off)
Preferable behavior
Unpreferable behavior
Undefined parameter Rb≧473 Ω
473 Ω >Undefined parameter Rb≧0 Ω

Figure 1 An example of deciding an undefined parameter

3. System organization

This section describes the system organization of Desq. Figure 2 shows that. Desq mainly consists of three subsystems:

(1) Behavior reasoner

This subsystem is based on Qupras. It determines all possible behaviors.

(2) Design parameter calculator

This subsystem calculates ranges of design parameters.

(3) Parallel constraint solver

This subsystem is written in KL1, and is executed on PIM, Multi-PSI, or Pseudo Multi-PSI.

When the designer specifies initial data, the behavior reasoner builds its model corresponding to the initial state, by evaluating conditions of physical rules and objects. The physical rules and objects are stored in the knowledge base. The model in Desq uses simultaneous inequalities in the same way as Qupras. Simultaneous inequalities are passed to the parallel constraint solver to check the consistency and store them. If an inconsistency is detected, the reasoning process is abandoned. Conditions in physical rules and objects are checked by the parallel constraint solver. If the conditions are satisfied, the inequalities in the consequences of physical rules and objects are added to the model in the parallel constraint solver. If a condition cannot be evaluated by the parallel constraint solver, envisioning is performed. Finally, when all possible behaviors are found, the design parameter calculator deduces the ranges of design parameters that give preferable behaviors.

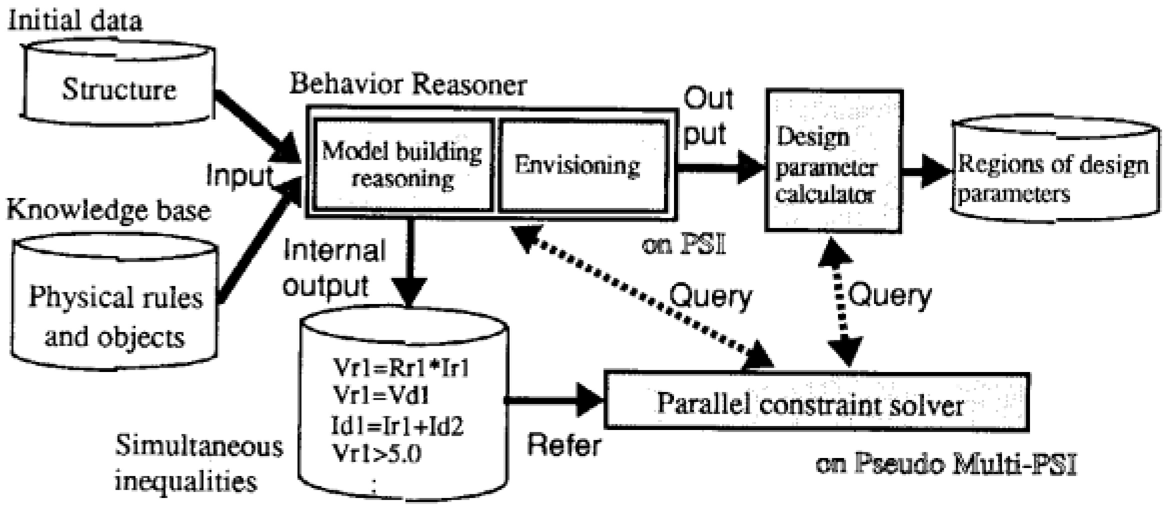

Figure 2 System organization

3.1 Behavior reasoner

Next, we describe Qupras and explain the reasoning in the system.

3.2.1 Qupras Outline

Qupras is a qualitative reasoning system using knowledge from a physics or engineering textbook. Qupras has the following characteristics:

(1) Qupras has three primitive representations: physical rules (laws of physics), objects and events.
(2) Qupras determines dynamic behaviors of a system by building all equations for the system using knowledge of physical rules, objects, and events. Users need not enter all the equations of the system.
(3) Qupras deals with equations which describe basic laws of physics qualitatively and quantitatively.
(4) Qupras does not need quantity spaces to be given in advance. It finds the quantity spaces for itself during reasoning.
(5) Objects in Qupras can inherit definitions of their super objects. Thus, physical rules can be defined generally by specifying the definitions of object classes with super objects.

Qupras is similar to QPT [Forbus 84], but does not use influence. The representations describing relations of values in Qupras are only equations. Qupras aims to represent physical laws given in physics textbooks and engineering textbooks. Physical laws are generally described not using influences in the textbooks, but using equations. Therefore Qupras uses only equations.

The representation of objects mainly consists of existential conditions and relations. The existential conditions correspond to conditions for the existence of the objects. Objects satisfying these conditions are called active objects. The relations are expressed as relative equations which include physical variables (hereafter physical quantities are referred to as physical variables). If existence conditions are satisfied, its relations become known as relative equations that hold for physical variables of the objects specified in the physical rule.

The representation of physical rules mainly consists of objects, applied conditions, and relations. The objects are those necessary to apply a physical rule. The representations of applied conditions and relations are similar to those of objects. Applied conditions are those required to activate a physical rule, and relations correspond to the physical laws. Physical rules whose necessary objects are activated and whose conditions are satisfied are called active physical rules. If a given physical rule is active, its relations become known as in the case of objects.

Qualitative reasoning in Qupras consists of two forms of reasoning: propagation reasoning and prediction reasoning. Propagation reasoning determines the state of the physical system at a given moment (or during a given time interval). Prediction reasoning determines the physical variables that change with time, and predicts their values at the next given point in time. Then, propagation reasoning determines the next states of the physical system using the results of prediction reasoning.

3.2.2 Behavior Reasoner

The behavior reasoner is not much different to that of Qupras. The two features below are additions to Qupras.

(1) Envisioning

In Qupras, if conditions of physical rules and objects cannot be evaluated, Qupras asks users to specify the conditions. It is possible for Desq to continue to reason in such situations by assuming unevaluated conditions.

(2) Propagation of new constraints on constants

There are two types of parameters (quantities), constant and variable. In envisioning, constraints related to some constant parameters become stronger by hypothesizing some conditions in physical rules and objects. The constraints propagate to the next states.

Before the reasoning, all initial relations of the objects defined in the initial state are set as the known relations, which are used to evaluate conditions of objects and physical rules. Initial relations are mainly used to set the initial values of physical variables. If there is not an explicit change to an initial relation, the initial relation is held. An example of an explicit change is the prediction of the next value in the prediction reasoning.

Propagation reasoning finds active objects and physical rules whose conditions are satisfied by the known relations. If a contradiction is detected, the propagation reasoning is stopped. If no condition of physical rules and objects can be evaluated, the reasoning process is split by the envisioning mechanism into one process hypothesizing that the condition is satisfied and other hypothesizing that it is not.

Prediction reasoning first finds the physical variables changing with time from the known relations that result from propagation reasoning. Then, it searches for the new values or the new intervals of the changing variables at the next specified time or during the next time interval. Desq updates the variables according to the sought values or intervals in the same way as Qupras. The updated values are used as the initial relations at the beginning of the next propagation reasoning.

3.3 Design parameter calculator

The method of calculating the design parameters is simple. After finding all possible behaviors, the designer specifies which design parameters to calculate. Then, the upper and lower values of the specified parameters are calculated by the parallel constraint solver.

3.4 Parallel constraint solver

The parallel constraint solver tests whether the conditions in the definitions of the objects and physical rules are proven by the known relations obtained from active objects and active physical rules, and from initial relations.

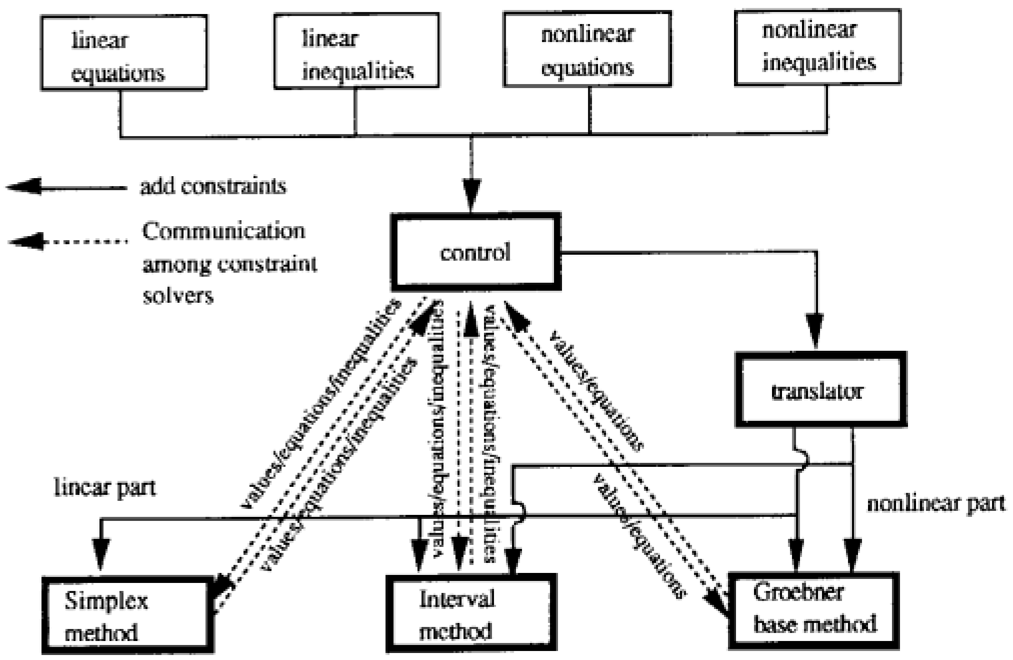

Figure 3 Combined constraint solver

We want to solve nonlinear simultaneous inequalities to test the conditions in the objects, physical rules, and events. More than one algorithm is used to build the combined constraint solver, because we do not know of any efficient algorithms for nonlinear simultaneous inequalities. We connected the three solvers as shown in Figure 3. The combined constraint solver consists of the following three parts:

(1) Nonlinear inequality solver based on the interval method [Simmons 86],
(2) Linear inequality solver based on the Simplex method [Konno 87],
(3) Nonlinear simultaneous equation solver based on the Groebner base method [Aiba 88].

If any constraint solver finds new results, the results are passed to other constraint solvers. This combined constraint solver can solve broader equations than each individual solver can. However, its results are not always valid, because it cannot solve all nonlinear simultaneous inequalities.

The reason that we can get quantitative ranges is that the combined constraint solver can process quantities quantitatively as well as quantitatively.

3. Example

3.1 Description of Model

We show another example of the operator. We use a DTL circuit the same as in Figure 1. In this example, however, the input voltage and the resistance Rb are undefined.

The initial data is shown in Figure 4. The "objects" field specifies components and their classes in the DTL circuit. The "initial_relations" field specifies the relations holding in the initial state. For example, "connect(t2!Rg, t1!D1, t1!D2)" specifies that the terminal t2 of the resistor Rg, the terminal t1 of the diode D1, and the terminal t1 of the diode D2 are connected. "!" is a symbol specifying a part. "t2!Rg" means the terminal t2 which is a part of Rg. Rb is specified as a resistor in the "objects" definition. "@" is specifies parameters. "resistance@Rl" means the resistance value of Rb. "resistance@Rl = 6000.0" specifies that Rl is 6000.0 ohms. The resistance Rb is constrained to be positive, and the input voltage is constrained to be between 0.0 and 10.0 volts. Both values are undefined, and Rb is a design parameter.

Figure 5 shows the definition of a diode. Its super object is a two_terminal_device, so a diode inherits the properties of a two_terminal_device, i.e. it has two parts, which are both terminals. Each terminal has two attributes "v" for voltage and "i" for current. A diode has an initial relation, which specifies the voltage difference between its terminals. A diode also has two states. One is the "on" state where the voltage difference is greater than 0.7. The other is the "off" state where the voltage difference is less than 0.7. If the diode is in "on", it behaves a conductor. In the "off" state, it behaves a resistor. A transistor is defined like a diode, but it has three states, "off", "on", "saturated" (In the example of Figure 1, we used a model of a transistor which has two states, "off" and "on").

```
initial_state dtl
 objects
  Rl-resistor ;
  Rg-resistor ;
  Rb-resistor ;
  Tr-transistor ;
  D1-diode ;
  D2-diode2 ;
 initial_relations
  connect(t1!Rl,t1!Rg) ;
  connect(t2!Rg,t1!D1,t1!D2) ;
  connect(t2!D3,t1!Rb,tb!Tr) ;
  connect(t2!Rl,tc!Tr) ;
  resistance@Rl=6000.0 ;
  resistance@Rg=2000.0 ;
  resistance@Rb >= 0.0;
  v@t1!Rl = 5.0 ;
  v@t2!D1 >= 0.0 ;
  v@t2!D1 =< 10.0 ;
  v@te!Tr = 0.0 ;
  v@t2!Rb = 0.0 ;
end.
```

Figure 4 Initial state for DTL circuit

Figure 6 shows a definition of a physical rule. The rule shows Kirchhoff's law when the terminals t1 of three two_terminal_devices are connected. It is assumed that the current into t1 of a two_terminal_device flows to the terminal t2. In fact, three two_terminal_devices can be connected eight ways according to which terminal is connected. In this rule, the equations describing physical rules are known.

```
object terminal:Terminal
  attributes
    v ;
    i ;
end.

object two_terminal_device:TTD
  parts_of
    t1-terminal ;
    t2-terminal ;
end.

object diode:Di
  supers
    two_terminal_device;
  attributes
     v ;
     i ;
     resistance-constant ;
  initial_relations
      v@Di=v@t1!Di-v@t2!Di ;
  state on
    conditions
       v@Di >= 0.7 ;
    relations
       v@Di= 0.7 ;
       i@Di >= 0.0 ;
  state off
    condition
       v@Di < 0.7 ;
    relations
       resistance@Di=100000.0 ;
       v@Di=resistance@Di*i@Di ;
end.
```

Figure 5 Definition of diode

```
physics three_connect_1
 objects
  TTD1 - two_terminal_device ;
  TTD2 - two_terminal_device ;
  TTD3 - two_terminal_device ;
  T1-terminal partname t1 part_of TTD1 ;
  T2-terminal partname t1 part_of TTD2 ;
  T3-terminal partname t1 part_of TTD3 ;
 conditions
  connect(T1,T2,T3);
 relations
  v@T1 = v@T2 ;
  v@T2 = v@T3 ;
  i@T1 + i@T2 + i@T3 = 0 ;
end.
```

Figure 6 Definition of physics

3.2 Results

Table 1 shows all behaviors of the DTL circuit obtained by envisioning. The state columns show the states of the diode, the diode2 and the transistor. The next columns show the ranges of the input voltage (volts), of the resistance of Rb (ohms), and of the output voltage (volts). Envisioning finds nine states. Because the input voltage and the resistance of Rb were undefined, the conditions of the two diodes and the transistor could not be evaluated. Desq hypothesizes both cases, and searches all paths. Figure 7 shows the relationship between the resistance and the input voltage. The reason that the ranges in Table 1 overlap is that the models of the diode and the transistor are approximated models.

Table 1 Result of DTL circuit

State	Range of input	Range of resistance value	Range of Output
1 ON-ON-SAT	1.40081 ~ 1.5381	486.16 ~ infinite	0.2
2 ON-ON-ON	1.4 ~ 1.40081	482.75 ~ infinite	0.2~5.0
3 ON-ON-OFF	0.7 ~ 1.4	0 ~ 233,567	4.94
4 ON-OFF-ON	0 ~ 1.4007	100,000 ~ infinite	0.842~5.0
5 ON-OFF-OFF	0 ~ 1.4	0 ~ 233,567	4.94
6 OFF-ON-SAT	1.40081 ~ 10.0	460.9 ~ infinite	0.2
7 OFF-ON-ON	1.4 ~ 10.0	457.8 ~ 488.53	0.2~5.0
8 OFF-ON-OFF	0.7 ~ 10.0	0 ~ 484.1	4.94
9 OFF-OFF- *	Conflict		

A designer can decide by investigating Figure 7 the resistance Rb for the DTL to behave as a NOT circuit. It is desired that Rb should be greater than about 0.5 k ohms and less than about 100 k ohms so that the DTL circuit can output a low voltage (nearly 0 volts) when the input is greater than 1.5 volts, and the output is high (nearly 5 volts) when the input is less than about 1.5 volts. The range shows the area enclosed by dotted lines in Figure 7.

4. Conclusion

We have described a method to suggest ranges of design parameters using qualitative reasoning, and implemented the method in Desq. The ranges obtained are quantitative, because our system deals with quantities quantitatively as well as qualitatively.

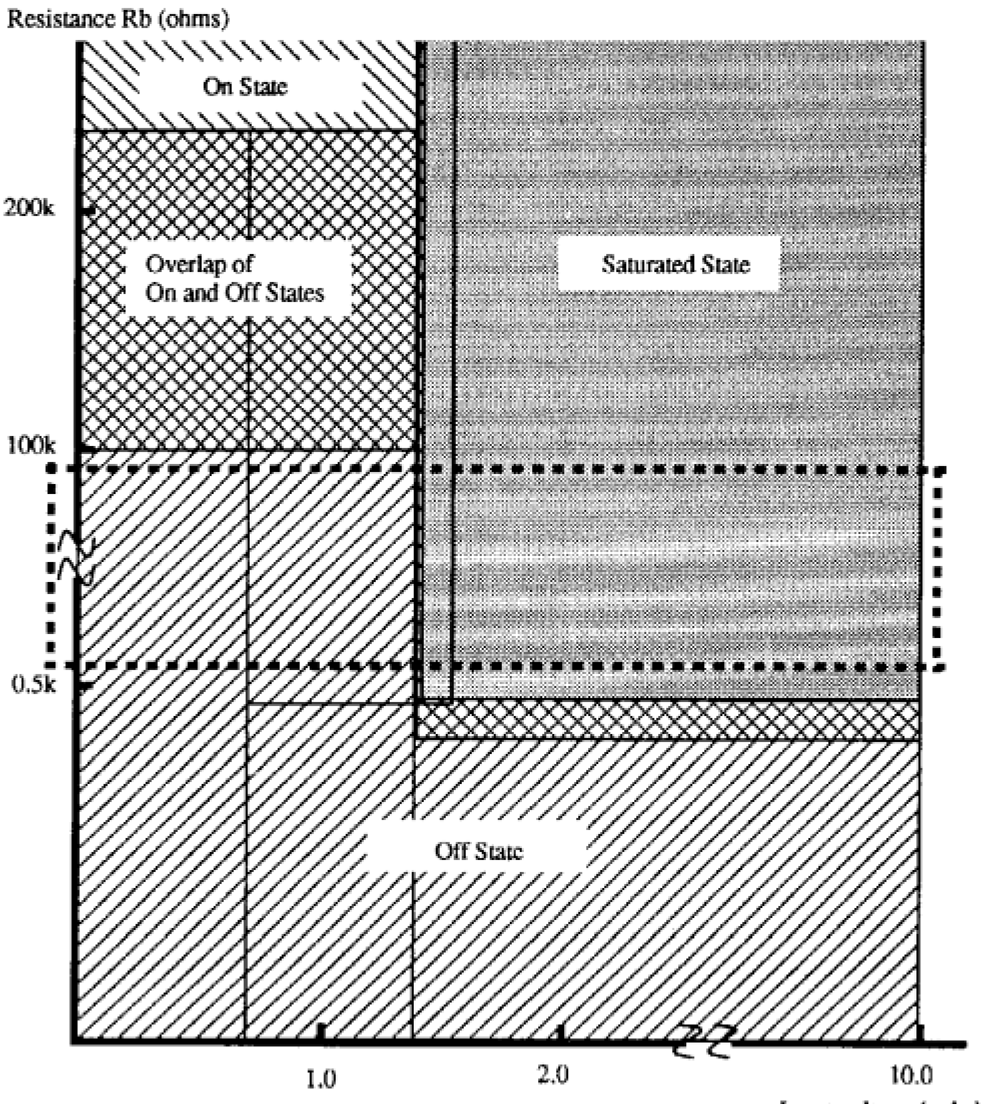

Figure 7 Relationship between Resistance and Input voltage

We are currently working on how to find ranges of design parameters of circuits that change with time, for example a Schmitt trigger circuit. In such a case, we need to propagate new constraints on constant parameters. Moreover, we are investigating the load balancing of the parallel constraint solver to speed up it.

This method does not suggest structures of devices like the methods of [Murthy 87] and [Williams 90]. Rather, it suggests ranges of design parameters for preferable behaviors. This approach may be regarded as one application of constraint satisfaction problem solving. In fact, it is similar to using CLP(R) to design of electronic circuits in that it uses constraint solvers [Heintze 86]. However, several features of our method are different:

(1) Knowledge on objects and physical laws is more declarative.
(2) Desq can design devices which change with time.
(3) Desq deals with nonlinear inequalities.

The method is not dependent on a domain of electronic circuits. If we wish to apply the method to another domain, we need only change knowledge base stored definitions for objects and physical rules.

5. Acknowledgements

This research is supported by ICOT (Institute of New Generation Computer Technology). We wish to express our thanks to Dr. Fuchi, Director of the ICOT Research Center, who provided us with the opportunity to perform this research with the Fifth Generation Computer Systems Project. We also wish to thank Dr. Nitta, Mr. Ichiyoshi, and Mr. Sakane in the Seventh Research Laboratory of ICOT for their technical advice regarding this study.

References

[Aiba 88] Aiba, A., Sakai, K., Sato Y., and Hawley, D. J.: Constraint Logic Programming Language CAL, Proc. of FGCS, 1988.

[Bobrow 84] Bobrow, D. G.: Special Volume on Qualitative Reasoning about Physical Systems, Artificial Intelligence, 24, 1984.

[Chandrasekaran 90] Chandrasekaran, B.: Design Problem Solving: A Task Analysis, AI Magazine, pp. 59-71, 1990.

[Forbus 84] Forbus, K. D.: Qualitative Process Theory, Artificial Intelligence, 24, pp. 85-168, 1984.

[Hawley 91] Hawley, D. J.: The Concurrent Constraint Language GDCC and Its Parallel Constraint Solver, Proc. of KL1 Programming Workshop '91, pp. 155-165, 1991.

[Heintze 86] Heintze, N., Michaylov, S., and Stuckey P.: CLP(R) and some Electrical Engineering Problems, Proc. of the Fourth International Conference of Logical Programming, pp.675-703, 1986.

[Konno 87] Konno, H.: Linear Programming, Nikka-Girren, 1987 (in Japanese).

[Kuipers 84] Kuipers, B.: Commonsense Reasoning about Causality: Deriving Behavior from Structure, Artificial Intelligence, 24, pp. 169-203, 1984.

[Mizoguchi 87] Mizoguchi, R.: Foundation of expert systems, Expert system - theory and application, Nikkei-McGraw-Hill, pp. 15, 1987 (in Japanese).

[Murthy 87] Murthy, S. and Addanki, S.: PROMPT : An Innovative Design Tool, Proc. of AAAI-87, pp.637-642, (1987).

[Nishida 88a] Nishida, T.: Recent Trend of Studies with Respect to Qualitative Reasoning (I) Progress of Fundamental Technology, pp. 1009-1022, 1988 (in Japanese).

[Nishida 88b] Nishida, T.: Recent Trend of Studies with Respect to Qualitative Reasoning (II) New Research Area and Application, pp. 1322-1333, 1988 (in Japanese).

[Nishida 91] Nishida, T.: Qualitative Reasoning and its Application to Intelligent Problem Solving, pp. 105-117, 1991 (in Japanese).

[Ohki 86] Ohki, M. and Furukawa, K.: Toward Qualitative Reasoning, Proc. of Symposium of Japan Recognition Soc. in 1986, or ICOT-TR 221, 1986.

[Ohki 88] Ohki, M., Fujii, Y., and Furukawa, K.: Qualitative Reasoning based on Physical Laws, Trans. Inf. Proc. Soc. Japan, 29, pp. 694-702, 1988 (in Japanese).

[Ohki 91] Ohki, M., Sakane, J., Sawamoto, K., and Fujii, Y.: Enhanced Qualitative Physical Reasoning System: Qupras, New Generation Computing, 10, 1991 (to appear).

[Ohwada 88] Ohwada, H., Mizoguchi, F., and Kitazawa, Y.: A Method for Developing Diagnostic Systems based on Qualitative Simulation, J. of Japanese Soc. for Artif. Intel., 3, pp. 617-626, 1988 (in Japanese).

[Simmons 86] Simmons, S.: Commonsense Arithmetic Reasoning, Proc. of AAAI-86, pp. 118-128, 1986.

[Yamaguchi 87] Yamaguchi, T., Mizoguchi, R., Taoka, N., Kodaka, H., Nomura, Y., and Kakusho, O: Basic Design of Knowledge Compiler Based on Deep Knowledge, J. of Japanese Soc. for Artif. Intel., 2, pp. 333-340, 1987 (in Japanese).

[Williams 90] Williams B. C.: Interaction-based Invention: Designing Novel Devices from First Principles, Proc. of AAAI-90, pp.349-356, (1990).