PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT.

Range Determination of Design Parameters by Qualitative Reasoning and its Application to Electronic Circuits

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

Central Research Laboratory, Hitachi, Ltd.
Higashi-Koigakubo, Kokubunji, Tokyo 185, Japan
ohki@crl.hitachi.co.jp

Abstract

There are numerous applications of qualitative reasoning to diverse fields of engineering. The main application has been to diagnosis, but there are a few applications to design. We show a new application to design, suggesting valid ranges for design parameters; this application follows the step of structure determination. The application does not provide more innovative design, but it is one of the important steps of design. To implement it, we use an envisioning mechanism, which determines all possible behaviors of a system through 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 the envisioning process, and (3) calculates the ranges of those design parameters that give the preferable behaviors.

We built a design-support system Desq (Design support 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. The Desq system can deal with quantities qualitatively and quantitatively, like Qupras. Accordingly, we may someday be able to determine the quantitative ranges, if the parameters can be expressed quantitatively. Quantitative ranges are preferable to qualitative values, to support the determination of design parameters.

1 Introduction

Recently, many expert systems have been used in the diverse fields of engineering. However, several problems still exist. One is the difficulty of building knowledge bases from the experience of human experts. The other is that these expert systems cannot deal with unimaginable situations [Mizoguchi 87]. Reasoning methods using deep knowledge, which is the fundamental knowledge of a domain, are expected to solve these problems. One reasoning method is qualitative reasoning [Bobrow 84]. Qualitative reasoning determines dynamic behaviors, which are the states of a dynamic system and its state changes, using deep knowledge of the dynamic 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 application has been to 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 for design parameters; it follows the step of structure determination. This application is not considered to be more innovative than the previous applications to design [Murthy 87, Williams 90], but it is one of the important steps of design [Chandrasekaran 90].

The key to design support is applying an envisioning mechanism, which predicts the behaviors of the dynamic system, to those design parameters whose values are undefined. If the envisioning is performed on condition that the design parameters whose values a designer wants to determine are undefined, all possible behaviors under the undefined design parameters can be predicted by the envisioning process. Some hypotheses are made to obtain each behavior. The main reason why hypotheses are made is that conditions written in the definitions of objects and physical rules cannot be evaluated because the design parameters are undefined. Among the obtained possible behaviors, more than one behavior desired by the designer is expected to exist. The designer can select the behaviors which he/she exactly prefers. Although the designer may not know the values of the design parameters, he/she knows the desired behavior. The values of the undefined parameters can be calculated from the hypotheses made to obtain the desired behavior.

To sum up, the method of determining valid ranges for design parameters offers the following:

(1) Performs envisioning for design parameters whose values are initially undefined,

(2) Selects preferable behaviors from possible behaviors found by the envisioning process, and

(3) Calculates the ranges of those design parameters that give the preferable behaviors.

We used a qualitative reasoning system Qupras (Qualitative physical reasoning system) [Ohki 86, Ohki 88, Ohki 92] to construct a decision support system Desq (Design support system based on qualitative reasoning) that suggests valid ranges for design parameters.

Qupras, using knowledge about physical rules and objects after being given an initial state, determines the followings:

(1) Relations between objects that are components of physical systems.

(2) The subsequent states of the system following a transition.

We extended Qupras to construct Desq as follows:

(1) Envisioning

In Qupras, if a condition of a physical rule or an object cannot be evaluated, Qupras asks the user 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 the envisioning process, constraints related to some constant parameters become stronger because conditions in the definitions of physical rules and objects are hypothesized. The constraints propagate to the subsequent states.

(4) Parallel constraint solving

Qupras uses a combined constraint solver consisting 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 combined constraint solver was heavy, so we converted it to KL1 to speed up processing.

Desq can deal with quantities qualitatively and quantitatively like Qupras. Accordingly, we may someday be able to get quantitative ranges, if the parameters can be given as quantitative values. Quantitative ranges may be preferable for decision support. The usual qualitative reasoning like [Kuipers 84] gives qualitative ranges.

Section 2 shows how Desq suggests ranges for design parameters, Section 3 describes the system organization of Desq, Section 4 shows an example of Desq suggesting the value of a resistor in a DTL circuit, Section 5 describes related works and Section 6 summarizes the paper.

2 Method of determining 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 only to determine the new values of the components. This is common for electronic circuits. We apply qualitative reasoning to the design decisions.

The key process used to determine design parameters is envisioning. Our 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 in the definitions 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 case 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, Desq finds several possible behaviors of a device.

The characteristics of this approach are as follows:

(1) Only deep knowledge is used to determine 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,

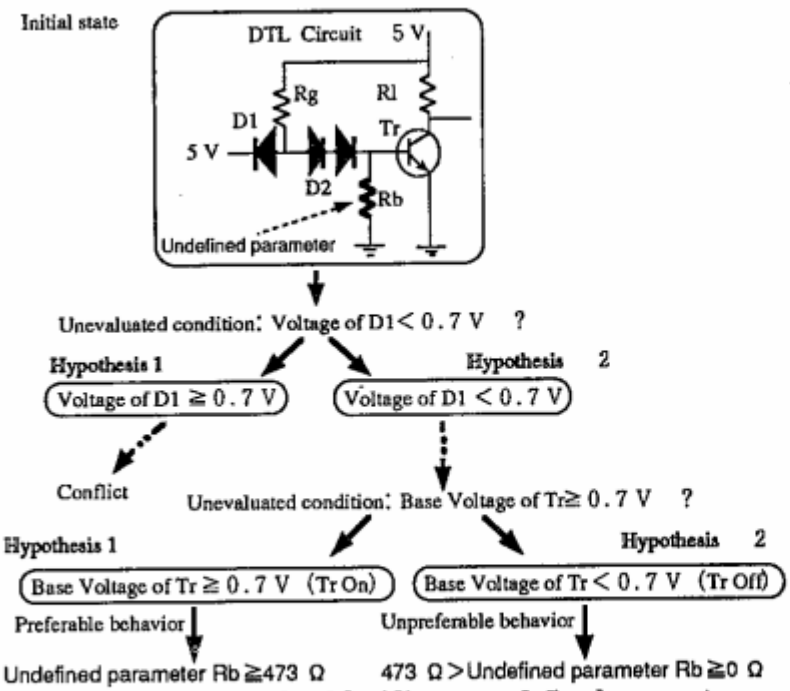

Figure 1 An example of deciding an undefined parameter

SPICE, he/she need not simulate values outside the ranges.

Figure 1 shows an example of suggesting ranges for a design parameter. This example illustrates the determination of 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 in the definitions of physical rules and objects. If they are satisfied, the 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 first condition is whether the voltage of 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, another state is detected where it is not known whether or not the 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 not preferable. Thus, the resistance Rb must be larger than 473 ohms. This proves that Desq can deal with quantities qualitatively and quantitatively.

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

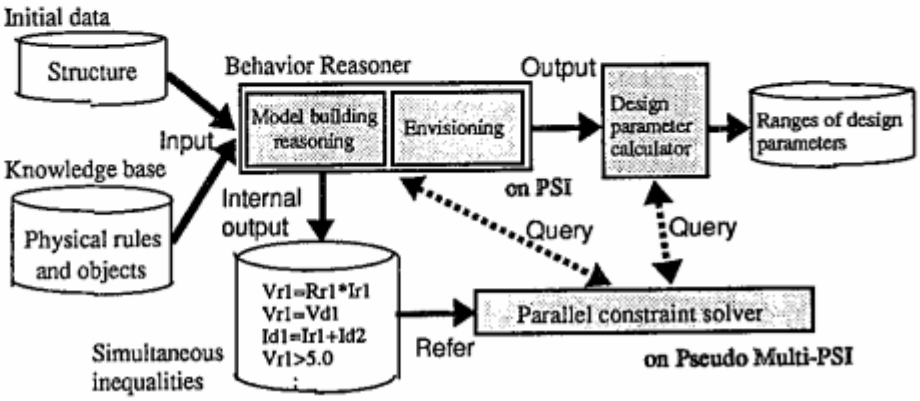

Figure 2 System organization

the same way as that in 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 the definitions of physical rules and objects are checked by the parallel constraint solver. If the conditions are satisfied, the inequalities in the consequences of the 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.

3.1 Behavior reasoner

3.1.1 Qupras Outline

Qupras is a qualitative reasoning system that uses knowledge from physics and engineering textbooks. Qupras has the following characteristics:

(1) Qupras has three primitive representations: physical rules (laws of physics), objects and events.

(2) Qupras determines the dynamic behaviors of a system by building all equations for the system using knowledge of physical rules, objects and events. The user need not enter all the equations of the system.

(3) Qupras deals with equations that describe basic laws of physics qualitatively and quantitatively.

(4) Qupras does not require quantity spaces to be given in advance. It finds the quantity spaces for itself during reasoning.

(5) Objects in Qupras can inherit definitions from 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 laws of physics given in physics textbooks and engineering textbooks. Laws of physics are generally described not by using influences in the textbooks, but by using equations. Therefore, Qupras uses only equations.

The representation of objects mainly consists of existential conditions and relations. Existential conditions correspond to conditions needed for the objects to exist. 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 existential conditions are satisfied, their relations become known as relative equations that hold for physical variables of the objects specified in the physical rule definition.

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 laws of physics. 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 involves 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. The propagation reasoning also determines the subsequent states of the physical system using the results from the prediction reasoning.

3.1.2 Behavior Reasoner

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

(1) Envisioning

In Qupras, if conditions of physical rules and objects cannot be evaluated, Qupras asks the user 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, the constraints related to some constant parameters become stronger by hypothesizing some conditions in the definitions of physical rules and objects. The constraints propagate to the subsequent states.

Before the reasoning, all initial relations of the objects defined in the initial state are set as known relations, which are used to evaluate the conditions of objects and physical rules. Initial relations are mainly used to set the initial values of the physical variables. If there is no 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.

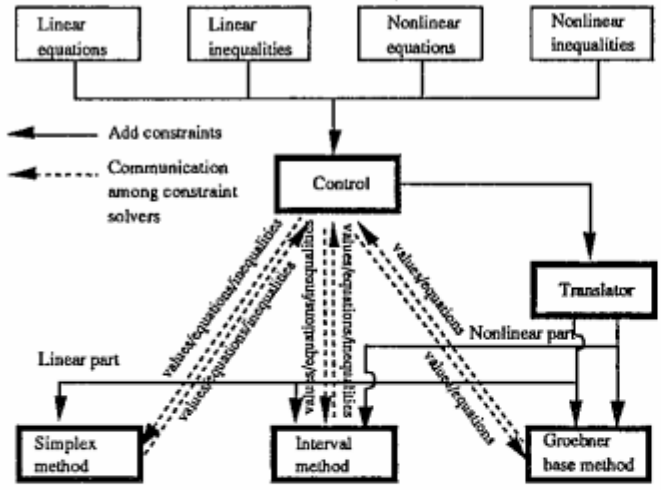

Figure 3 Combined constraint solver

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 two process: 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 the 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.

3.2 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.

3.3 Parallel constraint solver

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

We want to solve nonlinear simultaneous inequalities to test the conditions in the definitions of objects, physical rules and events. More than one algorithm is used to build the combined constraint solver, because we do not know of any single efficient algorithm 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], and

(3) Nonlinear simultaneous equation solver based on the Groebner base method [Aiba 88].

If any one of the three constraint solvers finds new results, the results are passed on to the other constraint solvers by the control parts. 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 why we can get quantitative ranges is that the combined constraint solver can process quantities quantitatively as well as qualitatively.

4 Example

4.1 Description of Model

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

```
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

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. The "!" is a symbol specifying a part. The "t2!Rg" expresses the terminal t2 which is one part of Rg. Rb is specified as a resistor in the "objects" definition. The "@" indicates a parameter. The "resistance@Rl" represents the resistance value of Rl. The "resistance@Rl = 6000.0" specifies that Rl is 6000.0 ohms. The resistance Rb is constrained to be positive, and the input voltage, which is the voltage of the terminal t2 in the diode D1, 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 the diode inherits the properties of the two_terminal_device, i.e., it has two parts, both of which are terminals. Each terminal has two attributes "v" for voltage and "i" for current. The diode has an initial

```
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

relation, which specifies the voltage difference between its terminals. The diode also has two states: one is the "on" state where the voltage difference is greater than 0.7, and the other is the "off" state where the voltage difference is less than 0.7. If the diode is in the "on" state, it behaves like a conductor. In the "off" state, it behaves like a resistor. A transistor is defined like a diode, but it has three states, "off", "on" and "saturated" (In the example of Figure 1, we used a transistor model with two states, "off" and "on").

Figure 6 shows the 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 in eight ways depending on how the terminals are connected.

Table 1 All behaviors of DTL circuit

State	Range of input	Range of resistance value	Range of output
1 ON-ON-SAT	1.40081 ~ 1.5381	486.16 ~ infinity	0.2
2 ON-ON-ON	1.4 ~ 1.40081	482.75 ~ infinity	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 ~ infinity	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 ~ infinity	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		

4.2 Results

Table 1 shows all behaviors of the DTL circuit obtained by envisioning. The state column indicates the states of the diode, the diode2 and the transistor. The following columns show the range of the input voltage (volts), the range of the resistance Rb (ohms), and the range of the output voltage (volts). As is shown, the envisioning found nine states. Because the input voltage and the resistance Rb were undefined, the conditions of the two diodes and the transistor could not be

evaluated. So, Desq was used to hypothesize both cases, and to search all paths. Figure 7 shows the relationship between the resistance and the input voltage. The reason why the ranges in Table 1 overlap is because the models of the diodes and the transistor are approximate models.

A designer can decide, by looking at Figure 7, the resistance Rb for the DTL circuit to behave as a NOT circuit. It is desired for Rb to 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, or can output a high voltage (nearly 5 volts) when the input is less than about 1.5 volts. The range is shown by the area enclosed by the dotted lines in Figure 7.

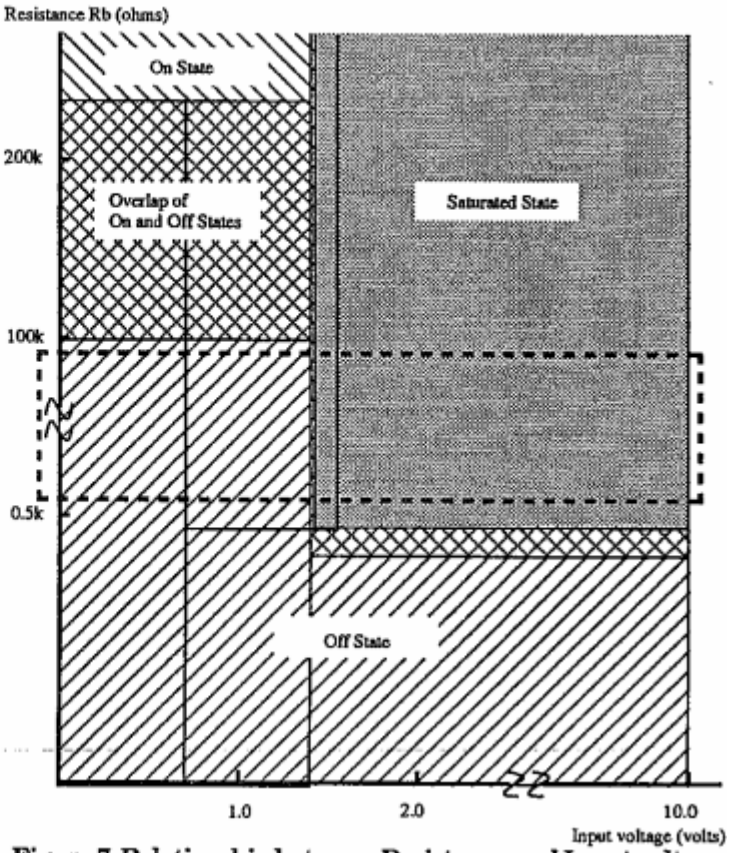

Figure 7 Relationship between Resistance and Input voltage

5. Related Works

Desq does not suggest structures of devices like the methods of [Murthy 87] and [Williams 90]. Rather, it suggests the ranges of design parameters for preferable behaviors. The suggestion is also useful, because determining values of design parameters is one of the important steps of design [Chandrasekaran 90].

This approach may be regarded as one application of constraint satisfaction problem solving. There are several papers that deal with electronic circuits as examples, using constraint satisfaction problem solving [Sussman 80, Heintze 86, Mozetic 91]. Sussman and Steele's system cannot suggest ranges for design parameters, because their system uses only equations. Heintze, Michaylov and Stuckey's work using CLP(R) to design electronic circuits is the most similar to Desq, but Desq is different from Heintze's work for the following points:

(1) Knowledge on objects and laws of physics is more declarative for Desq.

(2) Desq can design ranges of design parameters (of devices) that change with time.

(3) Desq can deal with nonlinear inequalities, and Desq can solve nonlinear inequalities in some cases.

In Mozetic and Holzbaur's work, numerical and qualitative models are used. In their view, our approach uses numerical models rather than qualitative models. But, if a constraint solver is used to solve inequalities, it is possible to use both numerical and qualitative calculations.

6. Conclusion

We have described a method of suggesting ranges for 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. In an example utilizing the DTL circuit, Desq suggested that the range of a resistance (Rb in Figure 1) should be greater than about 0.5 k ohms and less than about 100 k ohms to work the DTL circuit as a NOT circuit. If the designer wishes for a more detailed design, for example, to minimize the response time by performing numerical calculation, he/she need not calculate outside the range, and thus can save on the calculation cost, which is greater for direct numerical calculation (outside range).

However, there are some possibilities that Desq cannot suggest valid ranges or the best ranges for design parameters. This is because of the following:

(1) The ability to solve nonlinear inequalities in Consort is short

Desq may suggest invalid or weak ranges because Consort cannot perfectly solve nonlinear inequalities. But, almost all results can be obtained by performing more

detailed analysis using numerical analysis systems, for example, SPICE.

(2) Inexact definitions are used

It may be difficult to describe the definitions of physical rules and objects. This is because from inexact definitions, inexact results may be obtained.

(3) The ability to analyze circuits is short

The current Desq cannot analyze positive feedback. If there are any positive feedbacks in a circuit, Desq may return wrong results.

The example in this paper does not change with time. We are currently working on how to determine ranges of design parameters (of circuits) that change with time, for example, a Schmidt 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 it up.

7. Acknowledgements

This research was supported by the 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 of performing this research in the Fifth Generation Computer Systems Project. We also wish to thank Dr. Nitta, Mr. Ichiyoshi and Mr. Sakane (Current address: Nippon Steel Corporation) in the seventh research laboratory for their many comments and discussions regarding this study, and Dr. Aiba, Mr. Kawagishi, Mr. Menjyu and Mr. Terasaki in the fourth research laboratory of the ICOT for allowing us to use their GDCC and Simplex programs, and helping us to implement our parallel constraint solver. And we wish to thank Prof. Nishida of Kyoto University for his discussions, Mr. Kawaguti, Miss Toki and Miss Isokawa of Hitachi Information Systems for their help in the implementation of our system, and Mr. Masuda and Mr. Yokomizo of Hitachi Central Research Laboratory for their suggestions on designing electric circuits using qualitative reasoning.

References

[Aiba 88] Aiba, A., Sakai, K., Sato Y., and Hawley, D. J.: Constraint Logic Programming Language CAL, pp. 263-276, Proc. of FGCS, ICOT, Tokyo, 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, ICOT, Tokyo, 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.

[Mozetic 91] Mozetic, I. and Holzbaur, C. : Integrating Numerical and Qualitative Models within Constraint Logic Programming, Proc. of the 1991 International Symposium on Logic Programming, pp. 678-693, 1991.

[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., 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 92] Ohki, M., Sakane, J., Sawamoto, K., and Fujii, Y.: Enhanced Qualitative Physical Reasoning System: Qupras, New Generation Computing, 10, 1992 (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.

[Sussman 80] Sussman, G. and Steele, G. : CONSTRAINTS - A Language for Expressing Almost-Hierachical Descriptions, Artificial Intelligence, 14, pp. 1-39, 1980.

[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.