An Automatic Circuit Diagram Reader with Loop-Structure-Based Symbol Recognition

AKIO OKAZAKI, TAKASHI KONDO, KAZUHIRO MORI, SHOU TSUNEKAWA, AND EIJI KAWAMOTO

Abstract—**A high-performance logic circuit diagram reader has been developed for VLSI-CAD data input. Almost all logic circuit symbols include one or more loop structures. This paper first describes an efficient method for recognition of these loop-structured symbols. The proposed method consists of two processes: symbol segmentation and symbol identification. In particular, symbol identification is achieved by a powerful hybrid method, which uses heuristics to mediate between template matching and feature extraction. The entire symbol recognition process is carried out under a decision-tree control strategy. This can make readjustment, such as the improvement of recognition accuracy or addition of new symbols, very easy. Finally, the entire recognition system for circuit diagrams is briefly explained, including character string recognition and connecting line analysis.**

Experimental results show that symbol recognition accuracy is more than 95 percent and that a Japanese Industrial Standard (JIS) A1-size (594 mm × 841 mm) drawing can be processed within 30 min. To evaluate a method, more than 800 drawings were read automatically. A reduction of about 50 percent in input time was obtained, compared to conventional interactive entry.

Index Terms—**Circuit diagram reader, decision tree, hybrid pattern recognition, line drawing interpretation, pipeline architecture, real-time image processing, rule-based pattern recognition, structural pattern recognition.**

I. Introduction

While CAD systems have come into wide use to speed up engineering design work, the initial design work is still being carried out on drawing paper by experienced engineers. As the design information in the drawings has become more complicated in terms of amount and content, it has been pointed out that the initial data input from drawings is becoming a bottleneck in these CAD systems. Recently, to overcome the bottleneck, automatic CAD data input using pattern recognition techniques has been studied [1]-[8].

Fig. 1 shows a time comparison between an automatic reading method and an interactive (digitizing) method. The former can shorten the initial input time, although the drawing time may increase slightly due to the need for careful drawing for automatic reading. Furthermore, error correction time decreases as input accuracy improves. A characteristic of the automatic reading method is the need for reject modification, in which symbols rejected by the reader are modified with an interactive schematic editor in the same way that error correction is achieved. However, this additional time is very small because not only is the number of rejections small, but also all the locations of the rejected symbols are known. All in all, the automatic reading method can be much more effective than the interactive method.

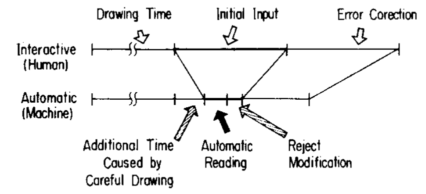


Fig. 1. Time comparison between automatic reading and interactive entry.

There are two major problems in developing such an automatic reader for hand-drawn diagrams.

One problem is the symbol pattern recognition performance. A drawing contains many symbol patterns. For example, the drawing shown in Fig. 2 includes about 400 symbols. Moreover, the symbols are made up of similar patterns, such as those shown in Fig. 3. Thus, a skillful symbol recognition method is required if information is to be input faster than by the interactive entry method.

Another problem is the ease of system ''readjustment.'' In the course of development, it is often necessary to improve the system's recognition accuracy or to add new symbols.

Some conventional systems make use of severe restrictions on drawing rules to achieve high performance. Although some systems achieve high performance with reasonable drawing rules, they can be improved only by rewriting complicated programs.

Considering these problems, a high-performance logic circuit diagram reader has been developed. In circuit diagrams, there are two types of logic circuit symbols. The overwhelming majority of symbols is of the first type, consisting of at least one closed symbol, or ''loop.'' We call these ''loop symbols.'' The second type, ''loop-free symbols,'' includes only a few examples (GROUND, CONDENSER, AND RESISTOR). Thus, special consideration must be taken in regard to loop-symbol recognition.

Manuscript received May 15, 1987; revised July 30, 1987.

A. Okazaki and S. Tsunekawa are with the Information Systems Laboratory, Toshiba Research and Development Center, 1 Komukai Toshiba-cho, Saiwai-ku, Kawasaki, 210 Japan.

T. Kondo and K. Mori are with the Man-Machine Interface Department, Information and Communication Systems Laboratory, Toshiba Research and Development Center, 1 Komukai Toshiba-cho, Saiwai-ku, Kawasaki, 210 Japan.

E. Kawamoto is with the Department of CAD & CAT Engineering, Integrated Circuits Division, Toshiba Semiconductor Inc., Sunnyvale, CA 94086.

IEEE Log Number 8820105.

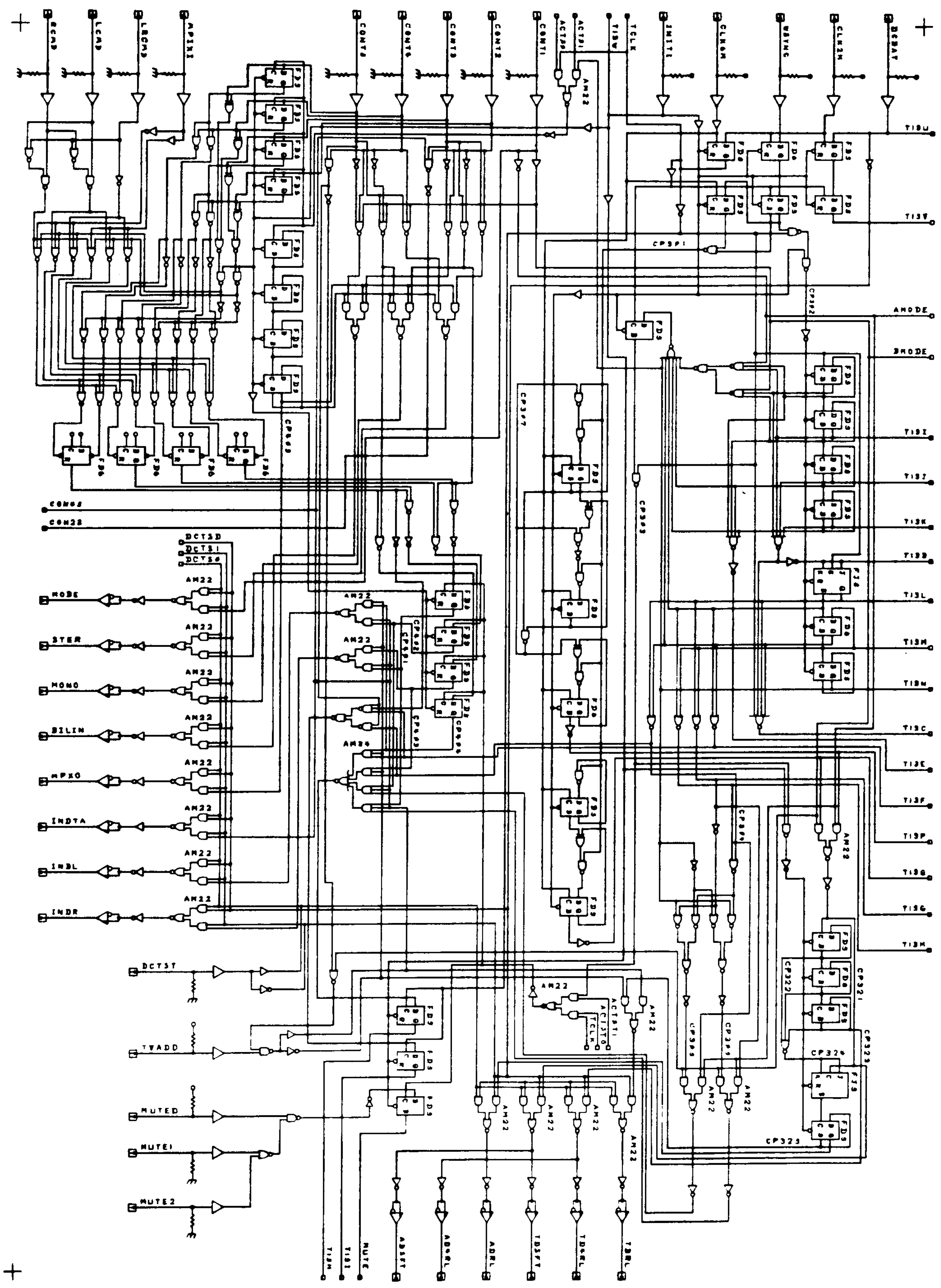


Fig. 2. Logic circuit diagram example.

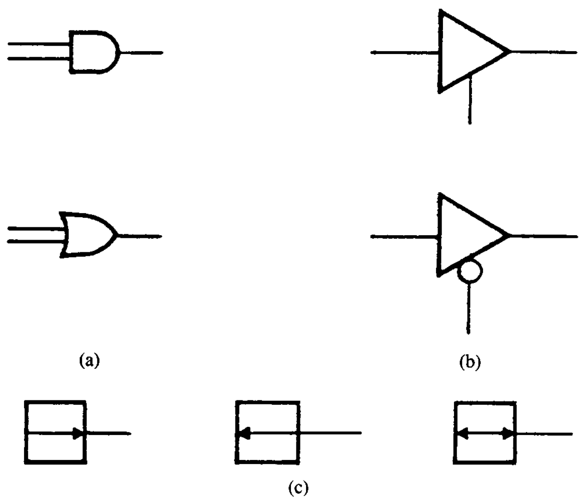


Fig. 3. Examples of similar logic symbols.

A loop-structure-based method for loop-symbol recognition, considering the two problems mentioned above, is described in Section II. In Section III, we exemplify typical drawing rules and user requirements. In Section IV, an implementation based on these is discussed. Finally, the effectiveness of the method is shown by experimental results in Section V.

II. Loop-Symbol Recognition

In this section, an approach to loop-symbol recognition is described.

Symbol patterns dealt with here have the following properties.

1) Symbols are composed of one or more primitive loops. A loop is a drawing forming a closed symbol, like a triangle, rectangle, diamond, etc. A primitive loop is defined as a loop which cannot be decomposed into further loops.

2) The size, direction, and position of symbols are not fixed: there is some variation in size, eight directions are allowed, and position is free.

3) Symbols are connected to each other with connecting lines and are drawn very densely in a diagram sheet.

4) Many symbols are similar to each other in shape, as shown in Fig. 3.

5) The addition of new symbols may be requested by users.

A natural approach to the recognition of such patterns is to detect primitive loops as the key to analysis. A particular problem for this approach is eliminating from consideration erroneous small loops, which are apparent loops (as defined above) formed when the lines connecting the symbols intersect, but which are not themselves symbols. A topology-based approach has been proposed in [4]. However, [4] does not deal sufficiently with the patterns described in 1), 4), and 5) above and does not discuss erroneous loops at all.

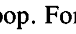
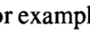
A solution for loop-symbol recognition after detecting a candidate loop is described below. It is a two-stage recognition procedure. In the first stage, the legitimacy of a candidate loop is ascertained, and the relevant region of the diagram is delimited. Henceforth, this recognition procedure will be called "symbol segmentation." In the second stage, the internal analysis of the delimited region is carried out. This recongition will be called "symbol identification." Both symbol segmentation and identification are controlled by decision trees. This strategy has been adopted for coping with readjustment, i.e., the addition of new symbols or improvement of recognition accuracy.

A. Loop-Structure-Based Symbol Segmentation

An individual loop can be found by applying the technique of component labeling [9, p. 347] to the background. The goal of the first processing stage is to isolate a "minimum region for analysis" (MRA), given one candidate loop which may be an instance of the symbol. The MRA can be defined as the minimum region sufficient for symbol identification (that is, a region including all the components of the symbol).

The strategy for calculating the MRA is described below, after the definitions of some useful concepts (see Fig. 4).

a) The set of symbols used is predefined, and individual symbols are drawn using a template. Hence, their size is constant. Four orientations, plus mirror images, are permitted. Thus, an individual symbol may have up to eight actual realizations.

b) Except for loop-free symbols, all symbols are associated with a characteristic loop. For example, , , and all have as their characteristic loop.

c) All symbols have an associated "symbol window," which is defined as the rectangle which exactly fits over the symbol.

d) The base point of a symbol is at the center of its characteristic loop.

e) We define a "characteristic window" for a given characteristic loop as the amalgamation of all the symbol windows of symbols containing that loop, centered on their base points.

f) Symbols having the same characteristic window are grouped into "intermediate categories."

Because the symbols are predefined, it follows that all these features are known *a priori*. In our experiment, 93 loop symbols were grouped into 14 intermediate categories.

The strategy for computing MRA's, given a candidate loop, devolves into the problem of determining the intermediate category for a candidate loop. Note that we operate here only on the detected loop, not on an entire symbol. Also, it should be recalled that the candidate loop may not even be part of a symbol.

The procedure to obtain an MRA, given a candidate loop, is as follows.

Step 1: Extract the eight features shown below for the candidate loop.

- *F1:* loop area;

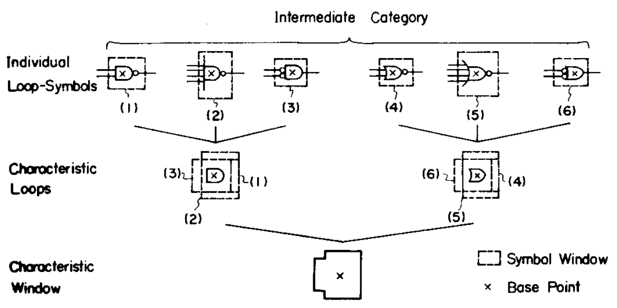


Fig. 4. Concepts used in calculation of MRA.

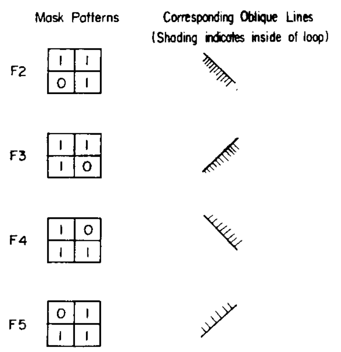

Mask Patterns
Corresponding Oblique Lines (Shading indicates inside of loop)
F2
F3
F4
F5

Fig. 5. Four mask patterns.

- *F2–F5:* number of occurrences of each of four mask patterns (Fig. 5), which approximates the length of the corresponding oblique line;
- *F6:* number of occurrences of three-branch points [9, p. 370] (it is assumed that the drawing is thinned [9, p. 394]); and
- *F7, F8:* x-length and y-length of the rectangle window, which just fits over the candidate loop (not the same as symbol window above).

Step 2: Determine the intermediate category of the candidate loop on the basis of F1–F8 (the direction of the loop is calculated on the basis of F2–F5 at the same time) using a decision-tree control strategy.

Step 3: The result of Step 2 gives the MRA by *a priori* definition.

B. Loop-Structure-Based Hybrid Symbol Identification Mediated by Heuristics

The goal of the second stage of processing is to calculate the exact symbol type, given an MRA and an intermediate category. The intermediate category information is useful because the goal can be limited. A hybrid method based on loop structure has been designed to obtain better identification performance (Fig. 6).

A simple method for the identification of symbols is template matching. Loop-structure-based identification by template matching is carried out as follows.

a) Prepare templates only for individual loop patterns (primitive templates), which correspond to the templates used by designers for drawing symbols on the sheets. For example, ⊃ and ○ are prepared for ⊃○ as primitve templates. Primitive templates are divided into two groups. One is a set of commonly used radical symbols, and the other is a set of auxiliary symbols, as shown in Fig. 7. In our experiment, 34 templates were prepared: 31 for radical symbols and 3 for auxiliary symbols.

b) Execute template matching in a selective way by interpreting the decision tree B in Fig. 6. The decision tree B describes a multistage matching procedure, each stage of which contains the following knowledge: i) the primitive template name to be matched, ii) the localized region where the matching operation is carried out, and iii) the minimum matching value for thresholding, whether the template symbol is successfully matched or not. The location where the highest matching resultant value is obtained is considered to be the location of the template symbol. The first template to be matched depends on the intermediate category. Depending on the result of thresholding by the value of iii), the content of the next template matching is determined. For example, after successful template matching for a radical symbol, possible auxiliary symbols and their location are known, and further template matching can thus be guided. However, in some cases, unique identification is difficult. For example, the difference in matching values among the symbols in Fig. 3(c) is so small that it is quite difficult to decide on a reasonable threshold value to discriminate among them. Therefore, the output is the three or fewer most probable symbol types above the threshold value and their orientations.

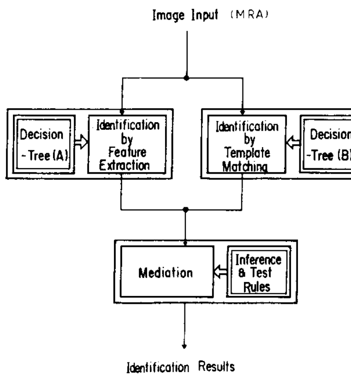


Fig. 6. Hybrid symbol identification mediated by heuristics.

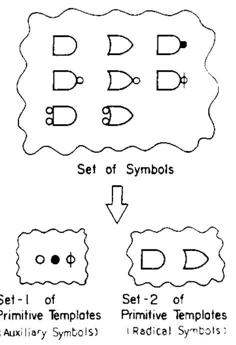

Set of Symbols
Set-1 of Primitive Templates (Auxiliary Symbols)
Set-2 of Primitive Templates (Radical Symbols)

Fig. 7. Primitive template set for matching.

The template-matching strategy described above has merits for the identification of structural symbols. For example, it can easily discriminate between the symbols in Fig. 3(b) because they consist of two primitive templates. Furthermore, it has the ability to withstand noise. However, it has the demerit described above. Therefore, as an alternative to template matching, another classification method, feature extraction, was considered.

An outline of identification by feature extraction is as follows.

a) Extract the following features from the MRA (Fig. 8).

• Extract the number of connecting lines (input, output, and control). The number is obtained by counting the connected components which enter the region as calculated from the location of the characteristic loop and intermediate category name.

• Extract the number of peaks and their geometrical information (location and height) in X-projection and Y-projection of the original image [9, p. 382].

• Extract the number of clusters and their geometrical information (location, height, width, and area) in X-projection and Y-projection for the image when filtered by hole filling. Hole filling can be accomplished by applying the component labeling technique.

• Extract several local features of the filled-hole image. For example, the area of a certain range (corresponding to the right-hand half of the characteristic loop in Fig. 8) in X-projection (if the direction of the symbol is from left to right) is calculated for discriminating between AND (⊐) and OR (▷) symbols.

b) Identify the symbol in the MRA from the above features. The identification is carried out by interpreting decision tree A in Fig. 6, which contains knowledge about the features needed and the threshold values. If a symbol pattern is segmented, preserving topology, and is not smeared with noise, unique identification is possible by selecting the appropriate threshold values. Therefore, the output is a single symbol type and its orientation.

If a segmented symbol pattern does not preserve the correct topology or is severely smeared with noise, this leads to misidentification or rejection. For example, if the auxiliary symbol Φ of the symbol type ⊐Φ was smeared, and so looked like Þ, the identification result would possibly be the symbol type ⊐∘ . Although the feature extraction method is quite sensitive to noise, it can be quite possible to predict the correct result from the results of template matching, which is robust against the noise. Heuristics based on knowledge of both the merits and demerits of template matching and feature extraction are used for mediation. Heuristics are represented by two types of production rules, namely, inference rules and test rules, which are applied hierarchically.

The mediation procedure is as follows.

Step 1: If at least one of the outputs from the two methods is rejection, then the image is rejected.

Step 2: If there is a match between them, return this as a candidate result. In the case where the output of template matching is not a single symbol, there is a match, if the result of feature extraction matches one of the results of template matching.

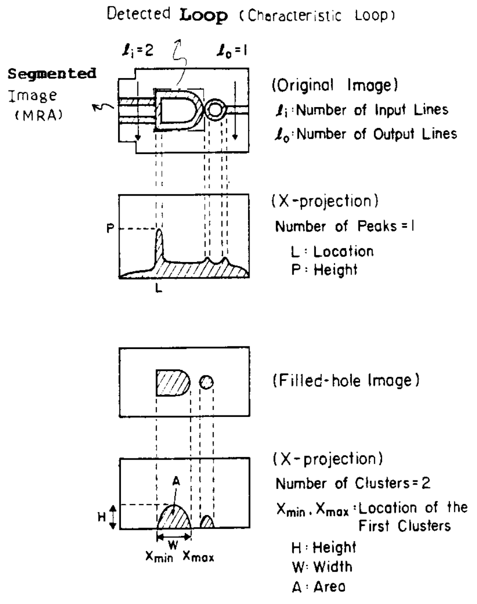


Fig. 8. Features used in feature extraction method.

Step 3: If there is no match, apply inference rules.

a) If this is successful, return this as a candidate result.

b) If this fails, reject the input image.

Step 4: Apply test rules to the candidate result. Return the result of these rules as a final result.

The format of the inference rules is $(A, B) \rightarrow C$; that is, C is inferred from A and B. Here, A is the identification result by feature extraction, and B is the one by template matching. If the template matching output is not a single symbol, inference rules permit a certain priority to be applied. The format of the test rules is $C \rightarrow P$. The test rules are provided to check whether a candidate C is acceptable or not by calling the test routine P. Test rules are applied to reject misrecognitions, while inference rules work to reduce reject error. In our experiment, 42 inference rules and 17 test rules were implemented. Fig. 9 shows examples of them.

Although hybrid pattern recognition has been studied [10], no reports on mediation by heuristics represented by production rules have been found.

C. Decision-Tree Control Strategy

Knowledge engineering techniques were taken into consideration in order to realize ease of readjustment. The main idea was to separate effectively an object-oriented part from the analysis process and to formalize it as knowledge data. A decision tree was adopted for the representation of knowledge because it is both more transparent and efficient.

Fig. 10 shows the analysis process block diagram. Here, an interpreter carries out the following operations.

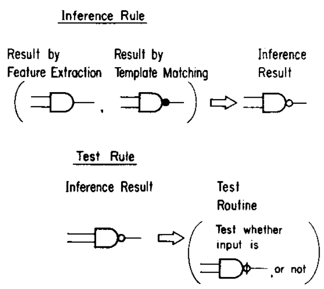


Fig. 9. Heuristic production-rule examples.

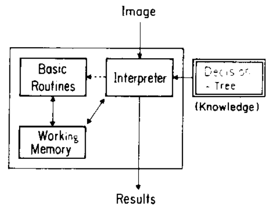


Fig. 10. Decision-tree control strategy.

1) Activate image processing routines or feature extraction routines (basic routines) and obtain some information when it is needed.

2) Make some decision by referring to some information in the working memory.

3) Output recognition results, when the interpreter is guided to a certain goal.

A tool which supports the interactive design of a decision tree has been developed [8]. A translator compiles a decision tree, designed by a tree editor, into table data composed of cells, as shown in Fig. 11. This support system can drastically shorten development time. Many decision-tree approaches to pattern recognition have been reported [1], [12]. However, no reports have been found on systematic approaches including an interpreter and a support system.

III. Drawing Rules and Specification

The greatest difficulty involved in introducing an automatic reading method is in deciding the drawing rules. LSI designers request that additional drawing rules over and above the conventional ones should be as few as possible. In this section, the drawing rules adopted to ensure good recognition accuracy and to keep the drawing time from increasing are described. Following this, the reader specifications are shown.

A. Drawing Rules

Reasonable drawing rules were decided upon as shown below.

1) A template is used to draw logic symbols. Eight symbol orientations are allowed: symbols rotated 90 degrees and their reflections.

2) A ruler is used to draw line segments.

3) Free-hand characters are written in the OCR font. Their size is less than 4 mm × 4 mm. Character strings can be written in two directions: left to right and bottom to top.

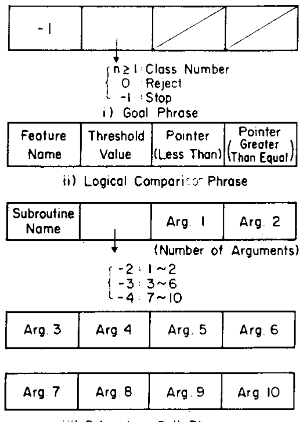


Fig. 11. Table structure of decision-tree.

4) The diameter of dots to branch points is bigger than 1 mm.

5) The minimum line segment length is 2 mm.

6) The minimum clearance between elements is 1 mm, except that the clearance between line segments is 2 mm.

Rules 1) and 2) correspond to accepted conventions. Rules 3)-6) are modifications to improve recognition accuracy.

B. Reader Specifications

The automatic reader specifications are as follow.

1) The Drawing sheet size is up to Japanese Industrial Standard (JIS) A1 (594 mm × 841 mm).

2) An A1-size drawing can be read within 30 min.

3) The elements applicable for use on the drawings are

• 96 different logic symbols, whose typical size is 4 mm × 7 mm;

• alphanumeric character strings in the OCR font;

• rectangles with small circles (negation), whose sizes can arbitrarily range from 10 mm × 15 mm to 100 mm × 100 mm; and

• lines composed of horizontal and vertical line segments.

4) Recognition accuracy is more than 95 percent, and the misrecognition error ratio is less than 1 percent.

The misrecognition error should be minimized, even if it causes an increase in reject errors. The reason is that the cost of correcting misrecognition errors is much greater than the reject error correcting cost. Rule 4) was introduced in order to satisfy this requirement.

IV. Implementation

Fig. 12 shows a part of a logic circuit diagram. A diagram is composed of five kinds of components. They are 1) characters, 2) loop symbols (NAND, etc.), 3) loop-free symbols (GROUND, CONDENSER, RESISTOR), 4) connecting lines, and 5) functional rectangles, with small circles indicating negation.

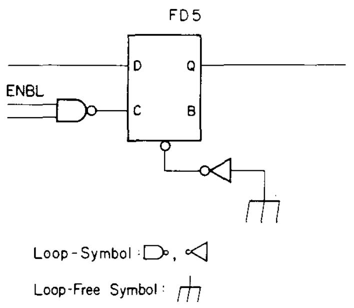


Fig. 12. Part of a logic circuit diagram.

Characters and loop symbols can be easily segmented by connected component analysis of the input image and its negative image. Once they are separated, individual component recognition can be carried out in parallel. Functional rectangles can also be relatively easily segmented by connected component analysis of the negative image. However, because their sizes are large and are not fixed, only rectangle detection is accomplished in the same way as loop-symbol detection. Further fine recognition is carried out by line analysis. On the other hand, it is quite difficult to distinguish loop-free symbols from connecting lines. Our approach to this problem is to search first key feature points, such as end points and corner points, which are relatively easy to detect, and to extract structural features by line tracing from those key points. Fig. 13 shows the flow of control of these processes.

Specially designed image processing hardware, which employs pipeline and multiprocessor techniques to speed up drawing processing, has been implemented in the automatic reader. A hardware block diagram is shown in Fig. 14. The module numbers in Fig. 14 correspond to those in Fig. 13.

Preprocessing, component separation, and loop-symbol detection are implemented by wired-logic hardware. The other processes are executed for four CPU's: two microprocessors (module 3) and two Toshiba TOSBAC 7/20E minicomputers (modules 4, 5, 6, 7). Loop-free symbol and rectangle recognition, line analysis, data management, and system control are carried out on the same CPU (TOSBAC 7/20E).

The individual modules are briefly explained here.

A. Image Input

The input device is a drum scanner. The specifications are as follow: 1) the maximum input rate is 10 lines/s; 2) the resolution is 10 lines/mm; and 3) the output data density is 256 levels/pixel.

B. Preprocessing [9], [13], [14]

After thresholding (conversion from a 256-level image to a bilevel image), thinning and noise filtering are accomplished using 4 × 4 pixel logical filtering. A special gate-array LSI has been developed for this purpose. These processes are constructed with about 30 pipeline stages (30 LSI's linearly concatenated are used). Then, local

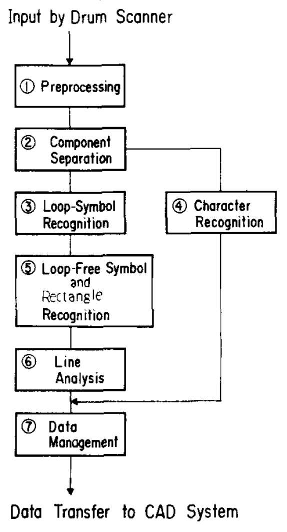


Fig. 13. Circuit diagram recognition flow.

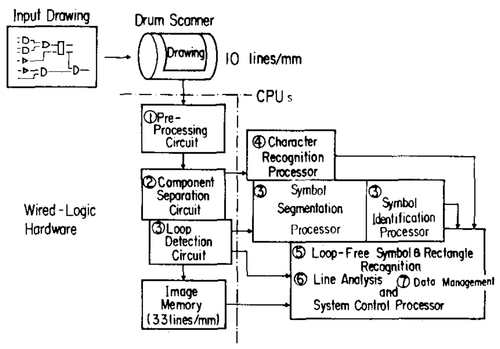

Input Drawing
Drum Scanner
Drawing
10 lines/mm
CPUs
①Pre-Processing Circuit
②Component Separation Circuit
③Loop Detection Circuit
Image Memory (33lines/mm)
Wired-Logic Hardware
④Character Recognition Processor
③ Symbol Segmentation Processor
③ Symbol Identification Processor
⑤ Loop-Free Symbol & Rectangle Recognition
⑥ Line Analysis and ⑦ Data Management
System Control Processor

Fig. 14. Hardware block diagram of automatic reader.

feature extractions are carried out in order to extract end points, corners, and branch points, end points and branch points by 3 × 3 pixel logical filtering and corners by 9 × 9 pixel special neighbor processing. Such local features are appropriately coded as pixel labels. The labeled image data are transferred to the next process.

C. Component Separation

Characters are separated from the input image by connected component analysis [6]. As mentioned in Section III-A, the maximum character size is 4 mm × 4 mm. Here, the character image size is defined as 48 × 48 pixels. Images of this size are separated one by one from the original image so as to enclose suitably each whole character. Then, the data are stored in the character buffer memory (30 kbytes). The image window location is also saved in the buffer memory, attached to the character image data.

D. Loop-Symbol Recognition

The two stages of symbol recognition, described in Section II, have been implemented in a pipeline architecture [15], [16]. Fig. 15 shows the decision-tree-controlled symbol recognition pipeline architecture.

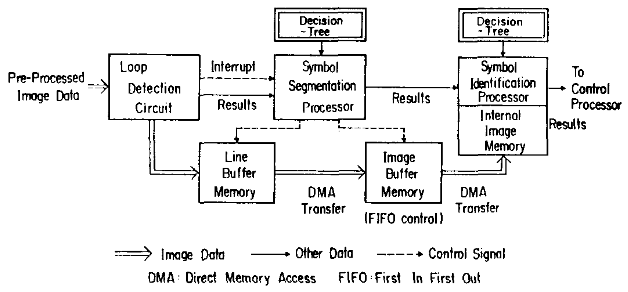


Fig. 15. Decision-tree-controlled symbol recognition pipeline architecture.

Preprocessed image data are input to the loop-symbol detection circuit. The loop detection circuit detects candidate loops by connected component analysis for a negative input image. First, it extracts the features F1–F6 (see above). Then, it eliminates erroneous loops (i.e., loops composed of connecting lines) on the basis of F1–F6, that is, by looking up the table (read-only memory) in which data on slice levels for each of F1–F6 are stored.

The symbol segmentation processor (processor A) receives the features F1–F6 and locations of the detected loop (x-start, x-end, y-start, and y-end) on an interrupt from the detection circuit. The rectangle size of the detected loop is calculated by processor A, that is, x-length (F7: x-end minus x-start) and y-length (F8: y-end minus y-start). On the basis of the eight features (F1–F8), processor A classifies the detected loop into 1 of 14 intermediate categories or rejects it as an erroneous loop. Most erroneous loops are eliminated in this stage. If there are some exceptions, such loops are removed in the stage of symbol identification.

If processor A does not reject the detected loop (this means the detected loop is recognized as a characteristic loop), it determines the orientation of the intermediate category (if the orientation is defined *a priori*). Then, processor A calculates an MRA from three pieces of information: the loop location, the intermediate category, and the orientation, in the manner described in Section II-A. Finally, processor A commands the direct memory access (DMA) controller to transfer the MRA image from the line buffer memory to the image buffer memory, which is operated as a queue (first-in first-out).

As the number of loops which are dealt with by processor A can be reduced by the detection circuit, processor A is able to carry out the symbol segmentation in real time. The symbol classification processor (processor B) receives the intermediate category identification and its orientation from processor A. Processor B commands the DMA controller to move an MRA image from the image queue to its internal image memory. Then, processor B analyzes the moved image as described in Section II-B and outputs a final identification result to the system control processor.

E. Character Recognition

Separated character images, with their window locations, are read one by one from a character image queue during drum scanner input. Image recognition of individual characters can be accomplished by the conventional OCR technique [17]. However, for string recognition, it was necessary to develop a different technique for the automatic reader. Thus, the main effort was directed towards recognizing strings, given only the basic assumptions that characters are written either from left to right or from bottom to top, as mentioned in Section III-A.

The string recognition strategy is as follows.

1) Each character image is recognized twice, assuming two possible orientations. At each time, the probability is calculated for each recognition result. Data concerning all their window locations and recognition results are temporarily stored in the main memory until all characters written in the drawing have been recognized in this way.

2) String analysis is carried out on the basis of both the two-way recognition result and location information. As a special case, the character "1" (numeral one) and the special symbol "−" (minus) would be difficult to distinguish in isolation since their orientation cannot be known. However, since "minus" rarely occurs on its own, the problem does not arise. Characters whose components are separated, such as ":", "=", etc., are not allowed. Designers are requested to use "Ø"rather than "0" (zero) in order to remove confusion between the numeral "zero" and the letter "oh." Using the proposed approach, high recognition accuracy for strings can be obtained, even if strings are written very densely and close together.

The loop-symbol recognition and character-string recognition results are sent to the system control processor, which saves them on a floppy disk. Meanwhile, after loop detection, dots representing connections between lines are extracted by 11 × 11 pixel neighbor processing. New labels representing dots are assigned to the corresponding pixels, adding to the labels of preprocessing results. Then, the labeled image data are reduced to 1/3 in both x-length and y-length by converting 3 × 3 pixels to 1 pixel, whose label is decided from several different labels assigned to 3 × 3 pixels according to the preset precedence. The reduced image data are stored in the image memory, where edge points, branch points, corners, and connection dots are represented by assigning appropriate values to them. Loop-free symbol and rectangle recognition and line analysis are carried out on the reduced image data because these processes do not need a high resolution.

The system control processor reads the floppy disk and masks the corresponding loop symbols on the reduced image memory by assigning codes to the pixels belonging to each loop symbol.

F. Loop-Free Symbol and Rectangle Recognition

Loop-free symbols and rectangles are recognized by line tracing from key feature points: end points and corner points. A key feature point is defined for each loop-free symbol: an end point for a GROUND or CONDENSER symbol and a corner point for a RESISTOR symbol or functional rectangle. For example, a key feature point for a CONDENSER symbol is sequentially detected by the raster scan search.

After detecting such points, line tracing from this feature point is carried out by extracting other feature points in a depth-first search manner. Fig. 16 shows a simple GROUND symbol line tracing example. The extracted structural features, such as branch points, corner points, end points, and their connection, are sequentially matched against the predefined dictionary, where loop-free symbol types are described in the form of graph-like representations [18, p. 216]. The matching results are sent to the system control program if successful matching is accomplished. Masking on the reduced image is accomplished by the system control program as explained above.

G. Line Analysis

The reduced image is analyzed to determine connecting line segments. Labeled pixels, corresponding to recognized symbols, are never accessed. Line tracing starts from an end point or a three-branch point, which are searched for by the raster scan mode. Line tracing is continued until an end point, a three-branch point, a labeled pixel corresponding to the recognized symbol, or a connection dot (see above) is found. If a four-branch point is found, the middle path is selected from among the three possible paths for the continued line tracing. A pair of coordinates for the start point and the end point of line tracing are recorded in a net list. The accessed pixels are marked in order to prevent further line tracing. The resultant net list is saved on a floppy disk.

H. Data Management

Finally, a data management for the recognized results is executed by a data management program. That is, all processing results are read from the floppy disk and arranged in their appropriate formats. Then, data transmission to the CAD system is carried out. A special protocol is defined for effective data communication.

Postprocessing by the host computer is explained next. The data are temporarily stored in a file on the host computer. Global symbol recognition is performed by a CAD interface program. The automatic reader recognizes primitive elements: symbols, character strings, functional rectangles, and lines. On the other hand, the interface program converts primitive elements into combinative patterns, such as complex gates, which are composed of several primitive symbols and interconnecting lines, a gate with a character string, and a flip-flop/macro gate, which consists of a functional rectangle and several character strings. A conversion table is prepared to define these combinations of symbols, rectangles, and character strings. Although the number of primitive elements is limited, the users can freely define new kinds of gate symbols.

Additionally, the interface program performs coordinate transformation to rectify location error caused by carelessly setting the sheet on the drum scanner. A linear coordinate transformation is accomplished on the basis of the extracted reference marks, which are recognized as isolated cross lines by the automatic reader.

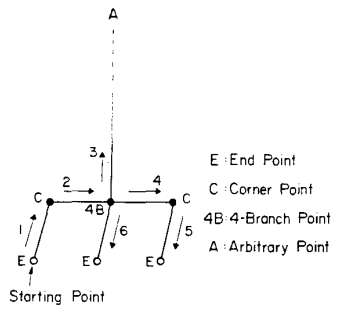


Fig. 16. Line tracing example in loop-free symbol recognition.

Once the data from the automatic reader are converted into the CAD data format by the interface program, many tools are available. Modification of recognition results can be accomplished using an interactive schematic editor. Then, a fair copy can be output using an on-line plotter, if necessary.

This processing hierarchy for the reader and the interface program enables users to define their own gate symbols and make the entry system extendable [7].

V. Experimental Results

An automatic logic circuit diagram reader has been developed using the proposed symbol recognition techniques.

A test drawing in which all symbols appeared at least once in each direction was designed and drawn in accordance with the drawing rules mentioned in Section III. The reader recognized all the symbols correctly. Then, ten practical logic circuit diagrams, which were drawn by different designers, were selected for performance tests. Although the designers were requested to follow the drawing rules for the automatic reader, they were not always able to do so correctly. Test diagram 3 is shown in Fig. 2.

Table I shows the recognition results, which correspond to the following performance figures.

1) The symbol recognition rate was 96.5 percent. (The misrecognition ratio was 0.7 percent.)

2) The character recognition rate was 91.7 percent.

3) The processing time was within 30 min for A1-size drawings: 17 min for loop-symbol recognition and character recognition (pipeline processing time), 10 min for loop-free symbol and rectangle recognition and line analysis (postprocessing time), and 3 min for data management.

Most recognition errors occurred because the designers did not follow the drawing rules for the automatic reader. The other cause of error was failure due to thresholding or noise filtering. Consequently, hybrid symbol identification mediated by heuristics was also satisfactorily applied. Heuristic rules improved the loop-symbol recognition accuracy by about 5 percent.

The reader transferred the recognition results to the host computer immediately through a 9600 bit/s synchronous line. The average transfer data size was about 17 kbytes.

TABLE I
EXAMPLE OF LOGIC SYMBOL RECOGNITION RESULTS (PERCENT)

Test No.	Sheet* Size	Symbol Number	Correct Recognition	Mis-Take	Reject
1	A1	256	92.9	2.0	5.1
2	A1	326	97.9	0	2.1
3	A1	417	97.4	0.2	2.4
4	A1	159	99.4	0	0.6
5	A1	244	95.5	0	4.5
6	A1	181	93.3	1.7	5.0
7	A1	471	99.0	0.4	0.6
8	A1	178	93.8	1·1	5·1
9	A2	97	100	0	0
10	A2	159	99.4	0	0.6
Total		2488	96.5	0.7	2.8

* JIS A1-Size : 594 mm × 841 mm
JIS A2-Size : 210 mm × 297 mm

Small improvements were subsequently made. Symbol set expansion was easily achieved by the support system. Additionally, a character template was newly prepared for the users who were not used to writing OCR font characters. This template made character recognition accuracy very stable.

In total, 851 drawings of various JIS sizes have been read by the automatic reader up to now for the purpose of evaluating the automatic reading method, viz. 67 for the A1 size (594 mm × 841 mm), 328 for the A2 size (420 mm × 594 mm), 352 for the A3 size (297 mm × 420 mm), and 104 for the A4 size (210 mm × 297 mm). As a result, 79 gate-array LSI's have already been developed. As can be seen, A2- and A3-size drawings were most frequently processed because of their design efficiency. The automatic reader can read A2-size drawings within 20 min. According to statistical results, error correction/rejection modification for symbols, characters, and lines was needed at about 29 places, on average, for A2-size drawings. The details of error correction/reject modification were as follow: symbols, 18.1 percent; functional rectangles, 1.0 percent; characters, 41.8 percent; and connecting lines, 39.1 percent. In total, it took about 30 min for the schematic entry of an A2-size drawing, compared to about 1 h required with conventional interactive entry.

VI. CONCLUSIONS

A logic circuit diagram reader has been developed in order to speed up initial CAD data input. Our view point is that, although several line drawing editors have been developed for direct circuit design, skillful design work is still carried out on drawing paper. Thus, pattern recognition techniques have had to be applied to the automatic input of large-size hand-drawn logic circuit diagrams. A two-stage symbol recognition method based on loop-structure analysis had been proposed, paying special attention to the symbol recognition accuracy and ease of system readjustment. Although the method was developed initially for logic symbols, the underlying concepts, especially those of hybrid identification mediated by heuristics and the decision-tree control strategy, could be widely applied. It should be pointed out that the automatic reading and line drawing editing are not in competition with one another and that they can be used in combination—for example, automatic reading for rough schematic entry and then line drawing editing for refinement.

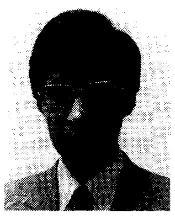
ACKNOWLEDGMENT

The authors wish to thank Dr. K. Yoshida and Dr. Y. Nakamura for their encouragement and support. The authors are also very grateful to Dr. M. Kidode for his constructive discussions and useful suggestions and to Dr. H. Somers for assistance in preparing the manuscript.

REFERENCES

[1] S. Kakumoto, Y. Fujimoto, and J. Kawasaki, "Logic diagram recognition by divide and synthesize method," in *Artificial Intelligence and Pattern Recognition in Computer Aided Design*, J. C. Latombe, Ed. Amsterdam, The Netherlands: North-Holland, 1978, pp. 457-470.
[2] H. Bunke, "Automatic interpretation of lines and text in circuit diagrams," in *Pattern Recognition Theory and Applications*, J. Kittler, K. S. Fu, and L. F. Pau, Eds. D. Reidel, 1982, pp. 297-310.
[3] S. Simizu, S. Nagata, A. Inoue, and M. Yoshida, "Logic circuit diagram processing system," in *Proc. 6th Int. Conf. Pattern Recognition*, 1982, pp. 717-719.
[4] F. C. A. Groen and R. J. van Munster, "Topology based analysis of schematic diagrams," in *Proc. 7th Int. Conf. Pattern Recognition*, 1984, pp. 1310-1312.
[5] H. Bley, "Segmentation and processing of electrical schematics using picture graphs," *Comput. Vision, Graphics, Image Processing*, vol. 28, pp. 271-288, 1984.
[6] Y. Yoshino, K. Mori, A. Okazaki, and S. Tsunekawa, "Flexible drawing reader with high-speed hierarchical processors," in *Proc. Comput. Vision Pattern Recognition*, 1983, pp. 510-514.
[7] E. Kawamoto, H. Ohta, A. Okazaki, T. Kondo, and S. Tsunekawa, "Schematic entry system equipped with an automatic reader," in *Proc. IEEE ICCAD-84*, 1984, pp. 87-89.
[8] A. Okazaki, T. Kondo, K. Mori, and S. Tsunekawa, "Knowledge-controlled pattern recognition technique for hand-drawn logic symbols," in *Proc. IEEE Workshop CAPAIDM*, 1985, pp. 524-531.
[9] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic, 1976.
[10] Y. Takebayashi, H. Shinoda, H. Asada, T. Nitta, S. Hirai, and S. Watanabe, "Telephone speech recognition using a hybrid method," in *Proc. 7th Int. Conf. Pattern Recognition*, 1984, pp. 1232-1235.
[11] G. J. Agin, "Computer vision systems for industrial inspection and assembly," *IEEE Comput.*, vol. 13, no. 5, pp. 11-20, 1980.
[12] G. R. Dattatreya and V. V. S. Sarma, "Bayesian and decision tree approaches for pattern recognition including feature measurement costs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, pp. 293-298, 1981.
[13] R. M. Haralick, "Some neighborhood operators," in *Real-Time/Parallel Computing Image Analysis*, M. Onoe, K. Preston, Jr., and A. Rosenfeld, Eds. New York: Plenum, 1981, pp. 11-34.
[14] M. Kidode, H. Asada, H. Sinoda, and S. Watanabe, "Image processing unit hardware implementation," in *Real-Time/Parallel Computing Image Analysis*, M. Onoe, K. Preston, Jr., and A. Rosenfeld, Eds. New York, Plenum, 1981, pp. 279-295.
[15] R. Suzuki and S. Yamamoto, "Real-time image processing in automated cytology," in *Real-Time Medical Image Processing*, M. Onoe, K. Preston, Jr., and A. Rosenfeld, Eds. New York: Plenum, 1980, pp. 207-219.
[16] S. Watanabe, S. Tsunekawa, Y. Okamoto, I. Sasao, and T. Tomaru, "The development of a new model cyto-prescreener for cervical cancer," in *Real-Time Medical Image Processing*, M. Onoe, K. Preston, Jr., and A. Rosenfeld, Eds. New York, Plenum, 1980, pp. 221-229.

[17] Y. Kurosawa and H. Asada, "Attributed string matching with statistical constraints for character recognition," in *Proc. 8th Int. Conf. Pattern Recognition*, 1986, pp. 1063-1067.
[18] T. Pavlidis, *Structural Pattern Recognition*. New York: Springer-Verlag, 1977.



Akio Okazaki was born in Ehime Prefecture, Japan, on April 22, 1952. He received the B.E. degree in electrical engineering, the M.E. degree in information engineering, and the Ph.D. degree in information engineering from Nagoya University, Nagoya, Japan, in 1975, 1977, and 1982, respectively.

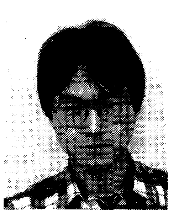
Since 1980 he has been with Toshiba Research and Development Center, Kawasaki, Japan, where he has been engaged in research and development activities in the field of line drawing processing and its applications. He is currently interested in man-machine interfaces using pattern recognition techniques, and data management of multimedia data such as text, graphic data, and images.

Dr. Okazaki is a member of the Institute of Electronics, Information, and Communication Engineers of Japan and the Information Processing Society of Japan.

Takashi Kondo was born in Osaka Prefecture, Japan, on September 1, 1948. He received the B.S. degree in mathematics from Kyoto University, Kyoto, Japan.

Since then, he has been working in the image processing field at Toshiba Research and Development Center, Kawasaki, Japan, in the Information and Communication Systems Laboratory. His major accomplishments include picture processing software libraries, an automatic inspection system for printed wiring board masks, and an image database management system. Currently, he is involved in development of image editing systems.

Mr. Kondo is a member of the Information Processing Society of Japan and the Institute of Electrical Engineers of Japan.



Kazuhiro Mori was born in Ehime Prefecture, Japan, on October 30, 1960. He graduated from the Mastuyama Technical High School, Japan, in 1979.

He has since been with the Toshiba Research and Development Center, Kawasaki, Japan, in the Information and Communication Systems Laboratory, involved in the development of computer application software. His interests are in the field of pattern recognition.

Mr. Mori is a member of the Information Processing Society of Japan.

Shou Tsunekawa was born in Nagoya, Japan, in 1945. He received the B.Eng. and M.Eng. degrees from the University of Tokyo, Tokyo, Japan, in 1969 and 1971, respectively.

He joined Toshiba Research and Development Center, Kawasaki, Japan, in 1971, and since then he has been engaged in research and development activities in the field of image recognition and its application. He is currently a Senior Researcher with the Information Systems Laboratory at Toshiba. His recent interests are in pattern recognition, image processing, and multimedia databases.

Mr. Tsunekawa is a member of the Institute of Electronics, Information, and Communication Engineers of Japan, the Japanese Society for Artificial Intelligence, and the Japan Society of Medical Electronics and Biological Engineering.

Eiji Kawamoto was born in Kanazawa, Japan, in 1952. He received the B.E. degree in electrical engineering from Kanazawa University in 1974, and the M.S. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 1976.

He joined Toshiba Corporation, Kawasaki, Japan, in 1976, and has been engaged in VLSI CAD development. He joined Toshiba Semiconductor (U.S.A.) Inc., Sunnyvale, CA, as a CAD manager in 1987.

Mr. Kawamoto is a member of the Institute of Electronics, Information, and Communication Engineers of Japan and the Information Processing Society of Japan.