Electronic Schematic Recognition

Donald Bailey[1], Andrew Norman[2], and Giovanni Moretti[2]

[1]Physics Department and [2]Computer Science Department
Massey University
Palmerston North

E-mail: D.G.Bailey@massey.ac.nz, G.Moretti@massey.ac.nz

We demonstrate the feasibility of using image analysis for automatically converting from a scanned electronic circuit schematic to a netlist of the components and their connections. A preprocessing stage removes non-components from the image (component label and values). The image is then segmented to produce a set of graphical primitives (lines, circles, and arrows). The recognition stage uses rule-based templates to match primitives to components. At present, the software will only recognise a limited range of components, although the rule-based approach allows the system to be extended to handle a wider range of component types or styles. The final stage constructs a netlist of components and their connectivity. Netlists are frequently used to represent electronic circuits for simulation.

Keywords: electronic schematics, circuit recognition, image analysis, circuit netlists.

1. NEED FOR SCHEMATIC RECOGNITION

Circuit diagrams, or schematics, are a standard means of representing electronic circuits. While this format is excellent for conveying much of the relevant circuit details in visual form to scientists, engineers, and hobbyists, it is not able to be readily manipulated or processed by computers.

Having a circuit in computer useable form is required for such applications as automated layout of printed circuit boards, simulating the electrical characteristics of the circuit, and maintaining a database of circuits or circuit modules. In all of these, a suitable representation of the circuit is as a netlist, that is as a list of components and their interconnections.

This project is examines the feasibility of using image analysis for automatically obtaining netlists from circuit schematics in printed form. This process involves recognising the various electronic component symbols within the schematic and determining their connectivity.

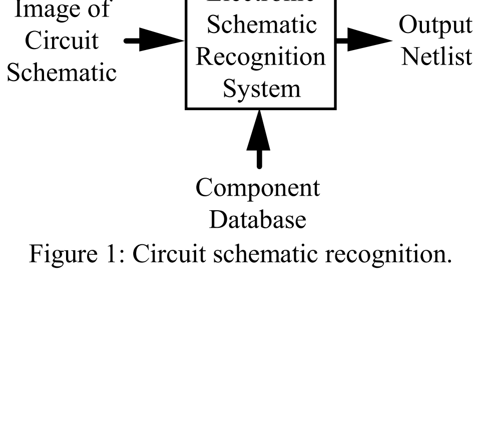
Image of Circuit Schematic
Electronic Schematic Recognition System
Output Netlist
Component Database

Figure 1: Circuit schematic recognition.

2. RELATED WORK

This sort of problem has been considered in various forms for just about as long as computers have been used for storing and manipulating documents. Very early work was concerned with efficient representation and processing of line drawings in pictorial form [1]. Much of the more recent document processing work has concentrated on distinguishing between mixed text and figures in documents, for example [2], with optical character recognition of the text parts now being a reasonably standard operation. There has been some work done on interpreting engineering drawings, for example [3] and [4], but relatively little on processing electronic circuit diagrams.

Most of the work on recognising electronic circuit diagrams as concentrated on logic diagrams. There, the image is broken into graphical primitives which are then grouped together to form gates, flip flops, and other logic symbols. Tudhope et al [5] used arcs, circles, and lines as the primitives, and a recognised the components using a series of production rules. Fahn et al [6] considered the segments between the junction of three or more lines, and used a picture description language to match the segments forming a component. Okazaki et al [7] recognised logic gates based on the characteristic shape of the closed loops representing the symbol. The closed loops were classified based on their projections.

Bley [8] segmented electrical schematics using a picture graph approach. The picture graph is then parsed to identify dominant horizontal, vertical and diagonal lines. Collections of lines conforming to predefined patterns are then clustered together to give circuit symbols using production rules.

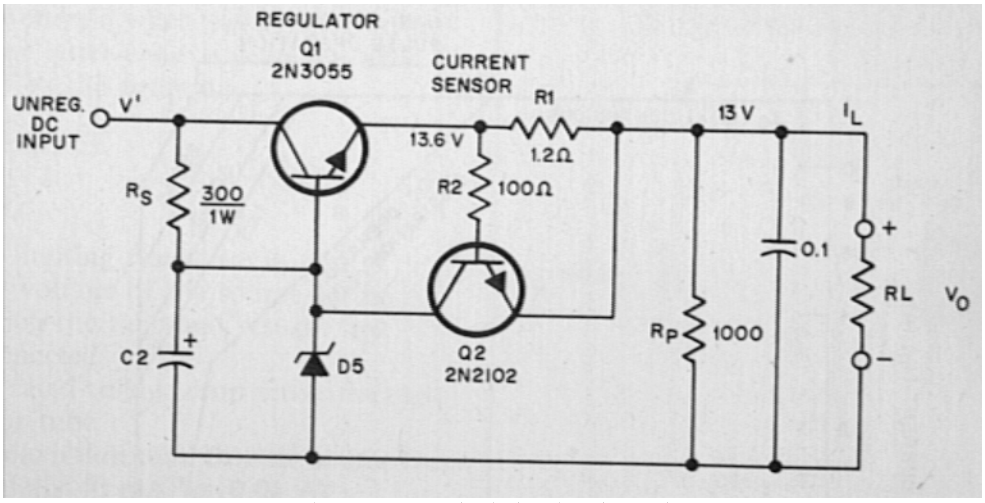

Figure 2. A typical circuit schematic.

3. OUR APPROACH

The circuit schematic needs to be scanned into the computer before it is processed. For many of our initial trials, we used a video camera and frame grabber. This worked well for small circuits, but had inadequate resolution for larger diagrams. For these, a flatbed scanner is more appropriate. The resolution required depends primarily on the thickness of the lines within the circuit schematic. There should be at least 2 pixels across each line to prevent them from becoming broken in subsequent processing. This also allows any text labels to be recognised and processed reliably.

3.1 Preprocessing

The scanned image is represented as an array of 8-bit pixels (figure 2). A significant proportion of the image conveys no useful information (the empty space between the components and interconnecting wires). Preprocessing reduces the volume of data by arranging it into a more useful compact form.

3.1.1 Background removal. The background of the image may not be completely uniform because of variations in lighting. A background image is formed by using a maximum filter [9], selecting the maximum pixel value within a scanned 15x15 window. This has the effect of replacing the dark inked areas within the image with adjacent background pixels. The original image is subtracted from this to remove the background, leaving the lines and text.

3.1.2 Segmentation. The image is then thresholded using a single global threshold level to segment the lines and text from the plain paper. After thresholding, the image is binary with the white pixels representing the information and the black pixels representing the background.

3.1.3 Chain code. Chain coding provides a convenient and compact method of representing binary line drawings [1]. It works by coding the boundary pixels between the black and white regions within the image as a chain of direction codes from one pixel to the next around the boundary. This process is shown in figure 3. Each boundary produces a separate chain. By recording the location of the starting pixel with each chain, the binary image is able to be reconstructed exactly from the chain codes.

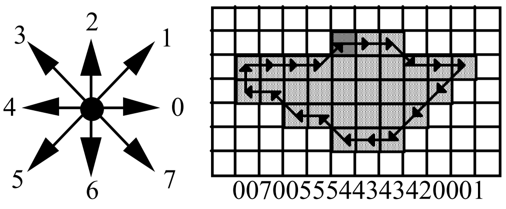

Figure 3. Chain coding a boundary

3.1.4 Separate the text and drawing. Text is used within circuit diagrams to label components (Q1, C2, RL), to provide component values (100**Ω**, 1W) or part numbers (2N3055, 2N2102), to annotate the circuit (13.6V, I_L, V_O) and describe its operation (current sensor, regulator). While conveying important information, this text is not directly related to the component symbols. The processing is simplified by separating the text from the rest of the image [10]. The simplest approach is to use an area threshold. All chains which have an area less than this threshold are considered to be text, and are separated for separate processing. Figure 4 shows the results of chain coding and sorting the chains into text and circuit parts. Using a simple area threshold may also remove parts of components, for example polarity dots of transformers, arrows adjacent to LED, and the insides of terminal nodes. Where necessary, any of these symbols not recognised as characters may be reclassified as component parts.

3.2 Extract Circuit Primitives

The chains represent the boundaries of the component symbols and interconnecting lines. These need to be regrouped into individual circuit symbols and interconnecting wires. Our approach is to break the chains into a set of graphical primitives which are then grouped to form the components. The graphical

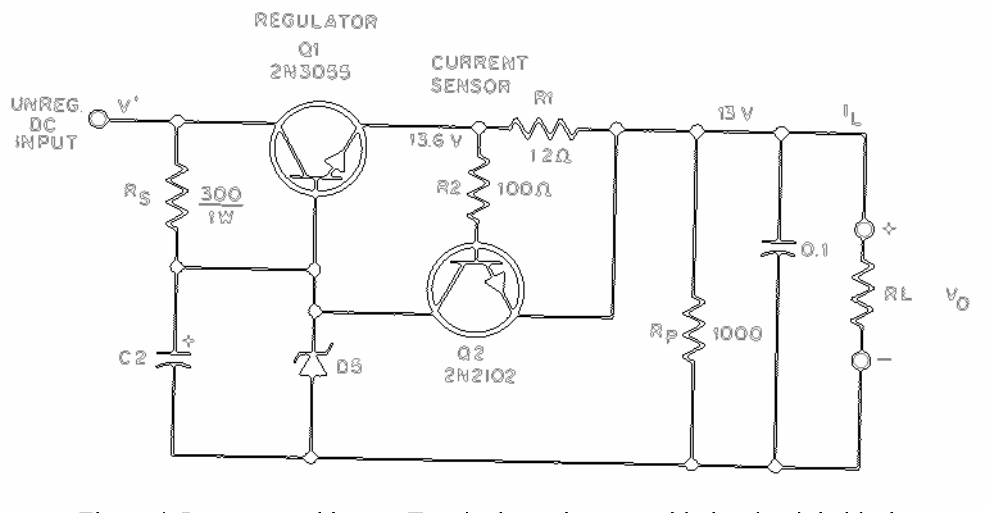

Figure 4. Preprocessed image. Text is shown in grey, with the circuit in black.

primitives that we use are line segments, circles, and arrowheads.

3.2.1 Vectorisation. The first step of this process is to break the chain into a series of approximately linear sections. First, each chain is split into two segments at convenient corners (the points closest to and furthest from the origin are easy to locate). Each segment is then successively split into two segments until all the resulting segments are straight to within a given tolerance. This process is performed as follows. For a given segment, a straight line between its two end points is considered. The point on the chain that deviates the most from this line is found. If the deviation is less than the allowed tolerance the segment is considered straight, otherwise the segment is split at the point of maximum deviation. This is repeated recursively until all of the segments are straight. A tolerance is necessary to prevent diagonal lines from being broken into very short segments, and to provide some noise immunity. We found a tolerance of 1.5 pixels to be about optimal.

3.2.2 Circle identification. Many components or circuit parts have circles associated with the symbol. These include transistors, input/output (I/O) nodes, and junctions. Any circles or circular arcs within the image will be broken into a series of line segments by the vectorisation stage. Reconstructing these into circles simplifies the subsequent component recognition stage.

By scanning through the list of segments, for each pair of adjacent segments the radius and central position of the potential arc are calculated. Arcs with improbable radius or and inappropriate angle between the segments are eliminated. Once all possible arcs have been identified, they are grouped into circles based on centre location and radius. Those circles which are represented by a significant proportion of their perimeter are retained, while the others revert to their linear segments. The linear segments making up the circle are deleted, being replaced by the circle primitive.

Any segments completely inside a circle are likely to form part of the same component as the circle. These segments are therefore associated with the circle to simplify the search when the component is identified.

3.2.3 Vector optimisation. All lines within the image will produce a pair of parallel segments since the chain coding process encodes the boundary on each side of the line. This could be avoided by first thinning the line to a single pixel thickness before chain coding [3], however, the thinning step would cause problems with solid regions such as diodes or arrows within transistors. The vector optimisation step replaces each parallel pair with a single segment.

Long lines may be broken into shorter segments because of noise or scanning nonlinearities. Such segments are rejoined if they are collinear to within 5 degrees.

3.2.4 Identification of arrows. To make the identification of transistors and diodes easier, it is necessary to detect arrow heads. Arrows consist of a group of three segments forming an acute isosceles triangle. The set of segments is searched to find all such triangular features. Where the triangle is equilateral, the base of the arrow head is defined as the segment which has another segment closest to its centre. This identification is important, because many of the arrows are equilateral triangles.

Figure 5 shows the graphical primitives (circles, lines and arrows) that were extracted from the example image.

3.3 Component recognition

The most complex stage is recognising the components from the primitives. We used a series of rules to define the configuration of primitives making up each component. This approach was chosen because it allows the rules to be modified or extended (to recognise new components) without changing the recognition software. The rules are stored as a text file containing multiple component definitions.

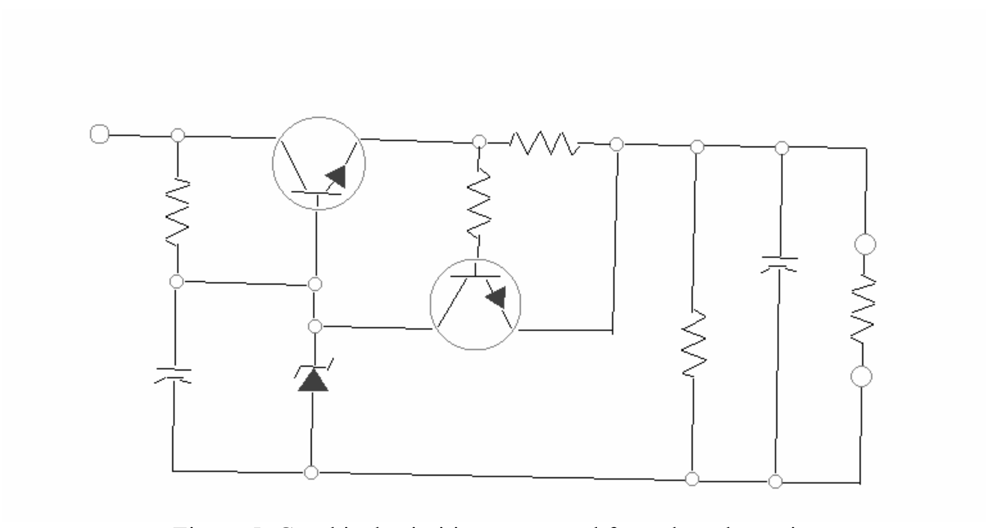
Figure 5. Graphical primitives extracted from the schematic.

3.3.1 Component definitions. Each component definition starts with the name of the component in square brackets, followed by a series of rules and an output definition (as illustrated in figure 6).

[Diode]	*Component name*
ARROW (B,P)	*Critical points*
ARROW (P) MIDLINE 90	*Rule statement*
EMIT B P	*Output definition*

Figure 6: An example of a component definition

Any given component may have multiple component definitions, where each separate definition can be used to recognise the component independently of the others. This allows variations in components to be recognised, for example a resistor may be represented by a rectangular box, or as a series of 7 angled lines.

Rules within the set define the following:

- length of a line
- relationships between pairs of lines
- radius of a circle
- number of connections to a circle
- direction of an arrow within a circle
- relationship between an arrow and a line
- relationship between a line and a component

3.3.2 Critical points. Every component in the image has a series of critical points, whether they are the ends of a resistor, or the terminals of a transistor. Many of the rules have a mechanism for extracting these critical points, so that they may be output with the component identifier. During the recognition stage, the critical points are simply positions within the input image.

3.3.3 Rule search. Each component definition is checked in turn. When searching for a component, each rule in turn is matched to the remaining primitives. If a match is found, those primitives are added to the component and the next rule is examined. If any of the rules is unable to be matched for the current component, the search backtracks to check for alternative matches to previous rules. When the component definition is completely satisfied, the component is given a unique identifier, and is added to the component output list. After all components matching that definition have been found, the search proceeds with the next component definition. The search continues until every component definition has been examined, or there are no primitives remaining to be matched.

3.3.4 Construct the netlist. The component recognition phase includes component definitions for junctions and interconnecting wires. These are removed and replaced by netlist node numbers in the following manner:

1. Two wires connected together are replaced by a single wire.
2. Two junctions connected by a wire are combined together, and the wire eliminated.
3. A junction connected to an I/O node is eliminated, being replaced by the I/O node.
4. A wire between a junction and a component is eliminated, being replaced directly by the junction.
5. A wire between two components is replaced by a junction.

This process eliminates all interconnecting wires, and replaces interconnected junctions with a single junction or I/O node. The final step is to assign a node number to each junction, and output the netlist. The netlist for this example is shown in table 1.

Table 1: The resulting netlist.

I/O nodes:	I1	1
	I2	2
	I3	3
Resistors:	R1	1,4
	R2	5,6
	R3	5,2
	R4	2,3
	R5	2,3
Capacitors:	C1	4,3
	C2	2,3
Transistors:	T1	4,1,5
	T2	6,4,2
Zener diodes:	Z1	3,4

4. EXTENSIONS

In this preliminary study, we have made no attempt to process the text associated with the schematic. For the netlist to be complete, it needs to be augmented with component values and part descriptions. This process is complex as it not only requires an optical character recognition stage but also recognising whether an item of text is associated with a single component (a component label, R3; a component modifier, + on electrolytic capacitor; a component value 1.2kΩ), a group of components, or is a stand-alone annotation. A rule of associating text to the nearest component would work for most schematics, although where components are drawn close together, this could lead to errors.

The current system only has rules for recognising the most common components. To be useful, it needs to be extended to recognise a wider range of components. This can be accomplished by adding further component definitions to the rule file.

The current rule set is not sufficient to be able to recognise some classes of component. Rules are required for combining symbols or components based on proximity. This would allow, for example, discrimination between related components such as standard diodes, light-emitting diodes, and photodiodes.

The rules are very specific in what they allow. However, there are a variety of different standards of representing many components. For example, a variety of different transistor symbol styles is shown in figure 7. To recognise all of the variations, one approach would be to have different component definitions for each standard. An alternative is to have rules which allow the optional inclusion of graphical primitives.

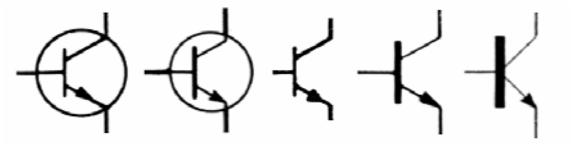
Figure 7. Transistor symbols from a range of sources.

5. CONCLUSIONS

Our preliminary study has demonstrated the feasibility of automatically constructing a basic netlist from an electronic circuit schematic. Further work is required to augment the netlist with component values, and to extend the range of components that can be recognised.

6. REFERENCES

[1] H. Freeman, Computer Processing of Line Drawing Images, ACM Computing Surveys, **6**, 57-97 1974.

[2] P.J. Bones, T.C. Griffin, C.M. Carey-Smith, A Projection-based Algorithm for Document Image Segmentation, Proceedings of the 5th NZ Image Processing Workshop, 87-93, 1990.

[3] C.C. Han, K.C. Fan, Skeleton Generation of Engineering Drawings via Contour Matching, Pattern Recognition, **27** (2) 262-275, 1994.

[4] Y.H. Yu, A. Samal, S. Seth, Isolating Symbols from Connection Lines in a Class of Engineering Drawings, Pattern Recognition, **27** (3) 391-404, 1994.

[5] D.S. Tudhope, J.V. Oldfield, A High-Level Recognizer for Schematic Diagrams, IEEE Computer Graphics and Applications **3** (3) 33-40, 1983.

[6] C.S. Fahn, J.F. Wang, J.Y. Lee, A Topology-Based Component Extractor for Understanding Electronic Circuit Diagrams, Computer Vision Graphics and Image Processing **44** (2) 119-138, 1988.

[7] A. Okazaki, T. Kondo, K. Mori, S. Tsunekawa, E. Kawamoto, An Automatic Circuit Diagram Reader with Loop-Structure-Based Symbol Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, **10** (3) 331-341, 1988.

[8] H. Bley, Segmentation and Preprocessing of Electrical Schematics Using Picture Graphs, **28** (3) 271-288, 1984.

[9] Y. Nakagawa and A. Rosenfeld, A Note on the Use of Local MIN and MAX Operations in Digital Picture Processing, IEEE Transactions on Systems, Man and Cybernetics, **8**, 632-635, 1978.

[10] R. Kasturi, S.T. Bow, W. El-Masri, J. Shah, J.R. Gattiker, and U.B. Mokate, A System for Interpretation of Line Drawings, IEEE Transactions on Pattern Analysis and Machine Intelligence, **12** (10) 978-992, 1990.