Circuit Recognition Using Netlist

Srikanth Sridar and Krishnan Subramanian
Student, Department of Electronics & Communication
Meenakshi Sundararajan Engineering College
Chennai, India
srika91@gmail.com

Abstract— **The existing software for circuit design and simulation require the user to be skilled either with good programming ability or 'pick and paste' model. To remove this barrier of programming knowledge, we propose a simulation model where a circuit drawn on a paper will be simulated. The circuit drawn on the paper will be fed to the computer using a scanner/camera. The image is de-noised and the nodes in the circuit are detected. All the characters, numbers and symbols alone are stored in a separate image which is used for optical character recognition. After node detection and character recognition, a netlist is compiled which is used for simulation. Applications of this simulation model include smart teaching systems, tablet app and with more research, a great deal of components including transistors and ICs can be simulated.**

Keywords—Circuit Recognition; Circuit Simulation; Netlist; Pseudonodes

I. INTRODUCTION

Most software aimed at circuit design and simulation requires the user to have a good command over them in order to use them efficiently. The simulation model proposed here does not require the user to be well versed in any particular programming language or software. Moreover, in "pick and drop" software, selecting each component is time consuming and most importantly, connecting wires in the circuit becomes a tedious task as the circuit complexity increases. The clear advantage of this simulation model is that the user only has to draw his circuit. In general, all the components in electrical and electronic circuits can be represented as a combination of resistor (R), inductor (L) and capacitor (C). Hence, currently we have only included RLC circuits with DC supply. Considering that offline circuit recognition is relatively a lesser researched field as compared to online, we believe, this paper will serve as a base paper for our further research.

II. RELATED WORK

With regard to circuit recognition, the current research trends follow online circuit recognition which uses a digital input device/interface [1]–[7]. Many online recognition systems make use of the sketches or strokes as useful information, which is not available in the case of offline. Also, a significant research work has been performed in the digital circuit recognition [3]–[5]. Offline digital circuit recognition (four basic gates) has been performed by Kumaravel Jagasivamani [7]. Offline circuit recognition, however, still remains a lesser researched field. Ravi Palakodety and Vijay

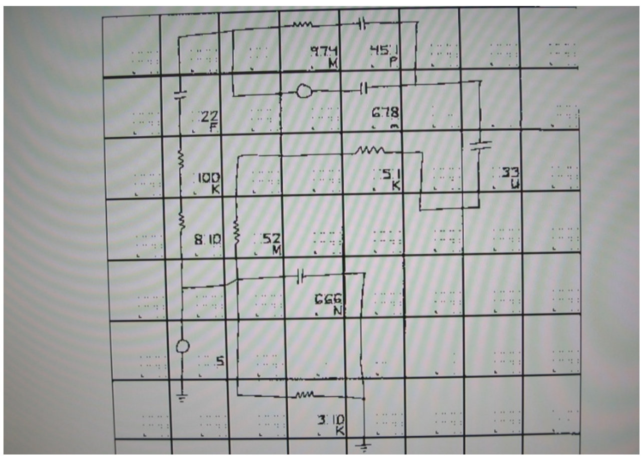


Fig. 1. 8x8 Grid View

Shah performed recognition of hand drawn circuits using finite state machine. The user draws the circuit in an 8x8 grid paper with each component centered in a block and values of the components written within a 10x8 block of lower right of the grid block. They also created the SPICE netlist and computer generated components replaced the hand drawn circuit [8].

A. Okazaki, T. Kondo, K. Mori, S. Tsunekawa and E. Kawamoto determined logic gates from the shape of the loop structured symbols [9]. Donald Bailey, Andrew Norman and Giovanni Moretti recognized printed circuits based on chain codes .The chains formed by the chain codes which are representation of the segments of the component symbols and interconnecting lines are used to recognize the components based on graphical primitives and regrouping them by predefined component definitions [10]. B.Edwards and V.Chandran used a nearest neighbor classifier which uses 35 features from each component, consisting of geometrical features, polygonal approximations and Hu invariant moments along with other features [11]. Most of the research related to offline circuit recognition involves identifying a set of patterns for the components and to create a grammar or a set of definitions for the components. Our approach uses the well developed optical character recognition to identify the component from units which gives the user sufficient liberty to draw the components without too much emphasis on their absolute shapes.

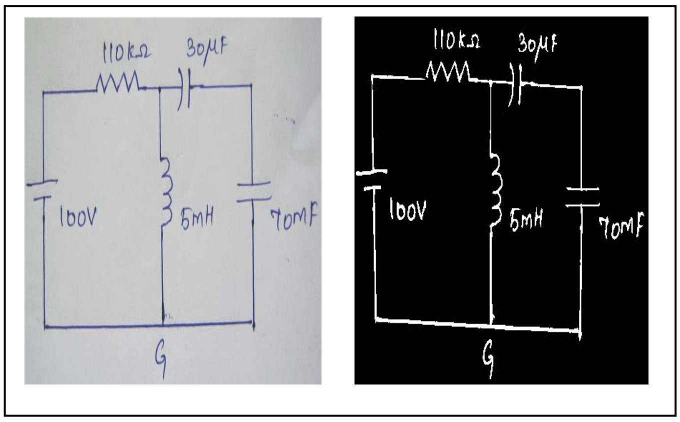


Fig. 2. RGB and Black and White Images of Circuit

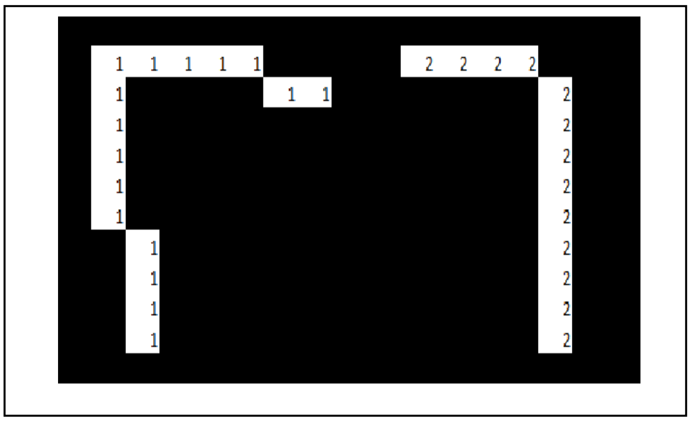


Fig. 3. Connected component analysis

III. Preprocessing

The circuit diagram is drawn on a neat white un-ruled paper. We used a 10 Mega Pixel camera to take the image of the circuit, under good lighting conditions. Noise reduction is performed and conversion of the input RGB image to gray image using Otsu's method [12]. The image is then dilated with the structuring element square with its width equal to two. The image is then thinned for single pixel thickness.

The connectivity of the components can be described using a netlist. To create a netlist, the nodes in the circuit, the values of the components and the node to which the components are connected, are required. The values are separated from the circuit using the area threshold. It is assumed that the total number of pixels in each character is always less than the segments of a circuit. The image with characters alone can be later used for optical character recognition and the image with circuit alone is used to extract the connection information of each component to respective nodes for creating the netlist.

IV. Recognizing Components

To create the netlist, the components have to be identified. One approach to identify the components is to find the patterns that are unique to each component. However, we identify the components using units of the components. For example, H denotes Henry - the unit of inductor, R for resistance - the unit of resistor, etc. This approach reduces the problem of identifying components to character recognition, which is relatively, a developed field.

V. Connected Component Analysis

In the subsequent sections we will look into various steps to construct the netlist. In most of the methods discussed, we have used connected component analysis. Connected component provides a convenient way to separate multiple objects in an image. Two pixels in an image are said to be connected if a path can be drawn between the two pixels such that the path contains the same image value. Identifying such groups of connected pixel objects is determined. This is usually done by finding the neighbors of a pixel. The procedure is as follows [13]:

1. Start from the top left corner towards right, then top to bottom looking for objects.
2. If an object is identified, then it is labeled. If another object is found but not connected to the previous one then label it with different label name and similarly for other pixels.
3. To find the connected regions neighbors of the object are identified. If it has neighbors (say 8 connected), label the current pixel with the same label as neighbor.
4. Repeat the steps for the entire image.

An example of the connected component analysis is shown in figure 3, where two objects are determined labeled as 1 and 2.Here 8-connected neighbors is used which is typical in case of hand-drawn lines.

VI. Extracting Wires

The connection information of components can be extracted from the wires alone without components. As our approach identifies components from the domain knowledge of electronics, the components are separated from the circuit. This is performed by raster scanning the entire image to get horizontal and vertical wires. It is also assumed that the circuit connections are made only using horizontal and vertical wires only. The wires drawn by the user are not absolutely vertical and horizontal in general. They might be little jagged even if they are perceived straight. An obvious way to extract the wires is Hough transform. Applying Hough transform to such jagged lines leads to formation of multiple lines instead of a single line. One can increase the threshold of gap that Hough transform allows to get a complete line. But as a consequence this could make a line crossing the resistor if the resistor is drawn very closely, thereby shorting the resistor. So to get a net alignment of wires, the jagged lines in the image are removed by joining the first and last points of each wire using the connected component analysis.

VII. Capacitor Removal

The capacitors do not exhibit any distinct property that can be exploited for their removal as they appear very similar to wires and hence have a significant interference in the node identification stages. Even the capacitors satisfy the conditions used for identifying the pseudonodes (left node, right node, up node and down node explained in node classification) that

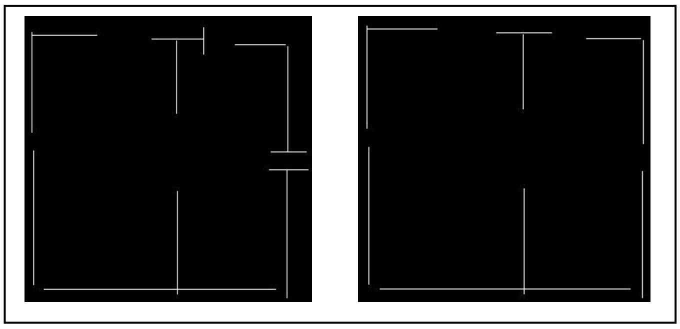


Fig. 4. Circuit with capacitor and capacitor removed from circuit

lead to a complication in identifying the nodes correctly. Such interference is not desired and therefore the capacitor should be removed. As the capacitor is hand-drawn, the only property of parallel plates is lost, as, either plate might get removed during the straightening due to threshold. This implies that the method for removal of capacitor should be a more generalized one that can remove the capacitor irrespective of the way it is reconstructed.

The gap between the capacitor plates can be used as a threshold. As long as the capacitors in the circuit satisfy the given threshold, they will be removed successfully. Also, the capacitor removal should be done before the wire alignment process because, due to threshold, there is a very good chance that the capacitor plates might get connected after which their removal becomes cumbersome.

VIII. Alignment Of Wires

In most cases, there will be some misalignment in the wires of the circuit. There are several types of misalignment that are to be dealt with. During the straightening process, there arise unwanted gaps between wires, which are originally connected. This is because the slightly tilted lines in the input circuit image are made absolutely vertical or horizontal as applicable. Therefore, a void is formed along the angle titled and this needs to be filled. Since these wires are reconstructed based on their end points, a single wire might actually exist as two wires with the end point of the halves not present along the same column or row. And in the previous filling part, a bridge like small strip of wire would have connected the two halves which needs to be removed and one of the halves should be made to align along the same row or column as the other half. This process completes the wire alignment part.

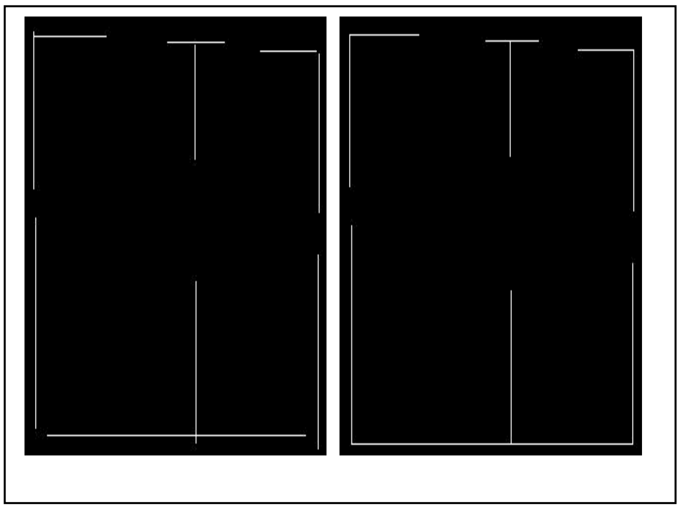


Fig. 5. Misaligned and Aligned Circuits

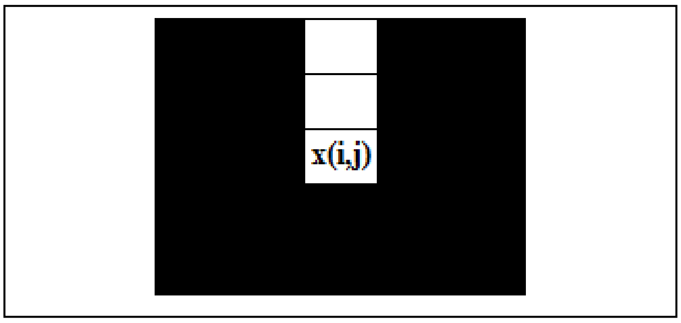


Fig. 6. Node Classification-Up node

IX. Node Classification

The nodes present in the circuit are to be identified to find which component is connected to it and its respective value. For this purpose, four types of pseudonodes extracted from the aligned circuit, namely, left node, right node, up node and down node are used. The left node and right node refer to the left and right terminals of the horizontal component, respectively. Similarly the up and down nodes refer to the top and bottom terminals of the vertical components, respectively. The procedure for identifying upnode is given as follows:

```
if (x(i,j) and x(i-1,j) are not null and other terms in the 3x3
matrix centred at x(i,j) are null) //depicted in figure 6
{
        up_node=[i j] //store the position of the upnode;
}
if (x(i,j) and x(i,j-1) are not null and other terms in the 3x3
matrix centred at x(i,j) are null) //depicted in figure 7
{
        left_node=[i j] //store the leftnode;
}
```

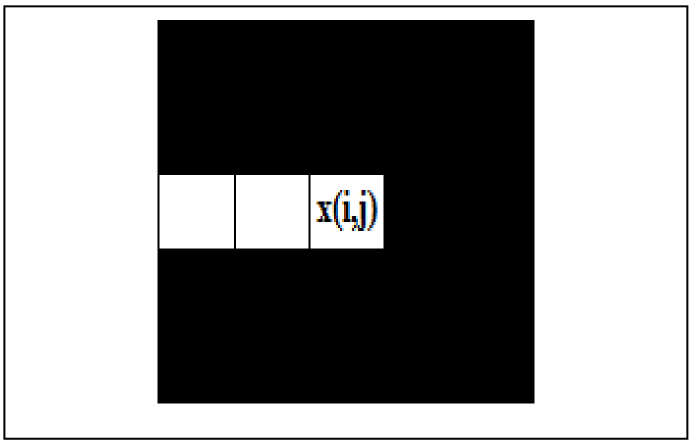


Fig. 7. Node Classification-Left node

The right nodes and the down nodes can be found in a similar manner. The left and the corresponding right node coordinates, as identified by the process above, are stored separately and the same goes for the up node and the corresponding down node coordinates. The convention being followed is that, for a horizontal component the values are written right above the component and for a vertical component, the values are written to the very right side of that component. This convention eliminates the ambiguity that arises in tracking values especially, when two components are connected in parallel.

X. MAPPING CHARACTERS TO RESPECTIVE NODES

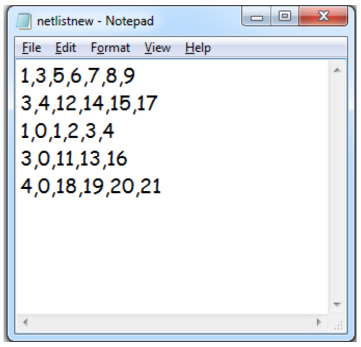
In the character image that has been extracted, a connected component analysis is performed to determine the number of components in the image. For each component (character) a centroid is determined and this centroid value, along with the component number is stored in an array. This array is used to map with the left nodes and the up nodes by finding the Euclidean distance between the centroid and the left node and up nodes. Once the first centroid is mapped to a node, the successive centroids are obtained using the same Euclidean distance method which is below a threshold. All these component numbers and their respective pseudo nodes are together taken as an array, which is used in character recognition, where in, the object numbers, are replaced by characters/numerals.

XI. MERGING NODES

The purpose of pseudo nodes is to trace the values for every component. Naturally, the exact nodes in the circuit differ from the pseudo nodes, in that same node can be seen as different node for the components connected to it. To find the correct nodes these pseudo nodes are merged. As wires are just the connection between nodes and the terminals of the components it can be seen that each wire itself acts as a node. So the node coordinates that are present in a particular wire is assigned that wires label value (the label value if obtained by numbering the wires column-wise). Labeling all the wires depending on their connectivity forms the real nodes from the pseudo nodes.

XII. CREATING NETLIST

The initial netlist formed contains a series of numbers. The first two columns in the netlist represent the nodes and the remaining values are the component number derived from the connected component analysis of the character image. The formation of a netlist without the ground node is, however, incomplete. To find the ground node, the user has to write a “G” instead of the conventional ground symbol near the ground node. The node closer to ‘G’ will be replaced by a “0” in the netlist. The character image is used for character recognition and the characters thus recognized are used to replace the component numbers in the initially formed netlist.



```
1,3,5,6,7,8,9
3,4,12,14,15,17
1,0,1,2,3,4
3,0,11,13,16
4,0,18,19,20,21
```

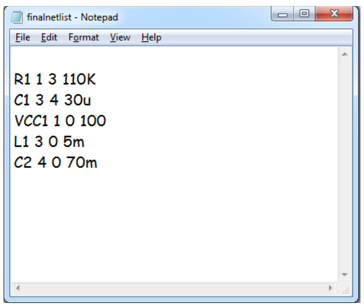
Fig. 8. Initial Netlist

XIII. OPTICAL CHARACTER RECOGNITION

Optical character recognition for the characters has been divided depending on the components. For example if the component is a resistor the possible values such as M for mega and K for kilo ohm are taken into account. Similarly µ for capacitor is considered. This reduces the number of classes to perform character recognition using machine learning techniques. The initial classes are the units of the four components, namely, V for voltage supply, **Ω** for resistor, F for capacitor and H for inductor. Thus only four classes are used instead of combining with all the numbers and various letters. The number of classes can be increased depending on the number of components. We have used wavelet transform as a feature for component classification and image coordinates as feature for the numbers and letters. As optical character recognition techniques are well developed using classifiers like support vector machine, high recognition rate can be achieved.

XIV. CONCLUSION

Currently our paper has been centered upon RLC circuit simulation. The number of components that can be simulated can be expanded. The logic used for preprocessing involves the heavy use of image independent thresholds. Such usage of extensive thresholds will limit the functionality of this model. Therefore, algorithms with less or no dependency on thresholds can be framed for such processes, increasing the efficiency of the system. While we have taken up the offline circuit



```
R1 1 3 110K
C1 3 4 30u
VCC1 1 0 100
L1 3 0 5m
C2 4 0 70m
```

Fig. 9. Netlist after character recognition

recognition and simulation as a field, the applications of this paper is not limited to the field of electronics. It can be used in smart teaching systems (e.g.: a mathematical equation may be recognized and its corresponding graph can be generated). With more developments in research, the number of components that can be simulated can be expanded and attempts can be made to simulate block diagrams and circuits printed in a book.

For indicative purposes, we have coded the logic in MATLAB. We assumed specific values for resistor, inductor capacitor and dc voltage supply and with the same value for each component, we made 25 circuits with varying complexity. As far as component recognition is concerned, since we have only four classes, namely, Ω, H, F and V which are quite distinct, almost all the components have been recognized. However, for the creation of final netlist, which involves identifying nodes, connections and mapping values to respective nodes, 80% of the netlists created, correctly describing the circuits. We are also working further on this field to add more components and increase the usability of this simulation model.

References

[1] Feng Gui-Huan, Sun Zhengxing, Christian Viard-Gaudin, "Hand-drawn Electric Circuit Diagram Understanding Using 2D Dynamic Programming", 11th International Conference on Frontiers in Handwriting Recognition (ICFHR 2008), Concordia University, Montreal, Quebec, Canada, August 13-15, 2008.

[2] Leslie Gennari, Levent Burak Kara, Thomas F. Stahovich, Kenji Shimada, "Combining geometry and domain knowledge to interpret hand-drawn diagrams", Computers & Graphics, Volume 29, Issue 4, Pages 547-562, ISSN 0097-8493, August 2005.

[3] Shizhong Lia, Menghua Duan, "Sketch recognition via string kernel", Natural Computation (ICNC), 2012 Eighth International Conference on , vol., no., pp.101,105, 29-31 May 2012.

[4] Shane W. Zamora, Eyrún A. Eyjólfsdóttir, "CircuitBoard: Sketch-Based Circuit Design and Analysis", Proceedings of International Conference on Intelligent User Interfaces (IUI) Workshop on Sketch Recognition, February 2009.

[5] David Johnston, "Sketch Recognition of Digital Logical Circuits", M.S. Thesis, Dept. Comput. Sci. Eng., Univ. California, San Diego,CA, March 2013.

[6] A. K. Mishra, J. A. Eichel, P. W. Fieguth, and D. A. Clausi, "VizDraw: A Platform to Convert Online Hand-Drawn Graphics into Computer Graphics", In the proceedings of the 6th International Conference on Image Analysis and Recognition (ICIAR '09), Mohamed Kamel and Aurélio Campilho (Eds.). Springer-Verlag, Berlin, Heidelberg, 377-386, 2009.

[7] Kumaravel Jagasivamani, "Recognition of digital logic circuit diagrams", unpublished.

[8] Ravi Palakodety and Vijay Shah, "Hand-Drawn Circuit Recognition", Massachusetts Institute of Technology, May 2005.

[9] A. Okazaki, T. Kondo, K. Mori, S. Tsunekawa and E. Kawamoto, "An Automatic Circuit Diagram Reader with Loop Structure-Based Symbol Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol-10,pp:331-341, 1988.

[10] Donald Bailey, Andrew Norman, and Giovanni Moretti, "Electronic Schematic Recognition", Massey University.

[11] Edwards,B. and Chandran,V., 'Machine recognition of hand drawn circuit diagrams' ICASSP'00 Vol-.6, pp: 3618 – 3621, 2000.

[12] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, pp. 62-66, 1979.

[13] H. Motameni, M. Norouzi, M. Jahandar. and A. Hatami, Labeling method in Steganography, Proceedings of world academy of science, engineering and technology, Volume 24, pp.349-354, October 2007.

[14] Arica, Nafiz , Yarman-Vural, Fatoş T., "An Overview of Character Recognition Focused on Off-Line Handwriting", IEEE transactions on systems,man and cybernetics—Part C-Applications and review, Vol. 31, NO. 2, pp: 216 – 233, 2001.

[15] Lécun, Yann ; Bottou, Léon ; Bengio, Yoshua ; Haffner, Patrick "Gradient based learning applied to document recognition", Proceedings of the IEEE Vol: 86,pp:2278 – 2324, 1998.

[16] Mozaffari, Saeed ; Faez, Karim ; Kanan, Hamidreza Rashidy, "Feature Comparison between Fractal Codes and Wavelet Transform in handwritten Alphanumeric Recognition Using SVM Classifier", ICPR 2004 Vol.2,pp:331 – 334, 2004.