CNNA-94 Third IEEE International Workshop on Cellular Neural Networks and their Applications
Rome, Italy, December 18-21,1994

Analog Combinatorics and Cellular Automata - Key Algorithms and Layout Design

Péter L. Venetianer[†‡], Péter Szolgay[†], Kenneth R. Crounse[†], Tamás Roska[‡] and Leon O. Chua[‡]

[†] Analogical and Neural Computer Laboratory, Computer and Automation Inst, Hungarian Academy of Sciences, H-1518 Budapest, Hungary
tel: 36-1-2698263 fax: 36-1-2698264 e-mail: venetian@sunserv.sztaki.hu

[‡] Department of Electrical Engineering and Computer Sciences, ERL, University of California Berkeley, Berkeley, CA 94720, USA

Abstract - *This paper demonstrates how certain logic and combinatorial tasks can be solved using CNNs. The most important application generalizes a shortest path algorithm to design the layout of printed circuit boards. Besides, it is shown how cellular automata can be simulated on CNN, and tasks, such as sorting, parity analysis, histogram calculation of black-and-white images, and computing minimum Hamming distance are also solved.*

1 Introduction

The Cellular Neural Network (CNN) [1, 2, 3] is a paradigm for locally connected, nonlinear, analog, dynamic computing arrays. In a recent paper [6] it was shown that the game of life algorithm can be realized by appropriate analog templates (locally interactive weight patterns). Therefore, any Turing-machine can be realized by CNN. The enormous implied capability has been evidenced by the development of many applications [3, and its references].

Here we will show some interesting examples of CNN templates which perform logical operations. We show, theoretically, how simple analog templates can generate logic functions for determining the time evolution of an arbitrary cellular automata. As a practical application, we describe the use of the analogic CNN Universal Machine [4] (CNNUM - a stored-program analogic microprocessor) to perform certain logic and combinatorial tasks, as opposed to the standard analog ones. Some of these applications, especially the layout design, outperforms traditional solutions, while others can serve as subroutines of complex analogic algorithms.

In many CNN applications the two saturation values (e.g. +1 and -1) are enough for detection tasks. However, in other cases the two states corresponding to the saturation regions $|v_x| \geq 1$ of the CNN might not be sufficient to represent the different states of

0-7803-2070-0/94/$4.00 © 1994 IEEE

a model. In such cases it is straightforward to map some states to values or regions in the $(-1, +1)$ interval of the dynamics. However, if simulating a multistep procedure, it is also important that the different cells of the network reach their new states at the same time. A possible solution to this problem is to discretize the time in the state equation of the CNN. We shall call this network a piecewise-linear unity-gain discrete-time cellular neural network (pwuDTCNN, not to be confused with the original DTCNN [5] which has a threshold-type output equation).

Section 2 shows how an arbitrary cellular automata can be simulated with a pwuDTCNN; in Section 3 some useful combinatorial tasks are solved; in Section 4 the minimum Hamming distance is computed; and finally, Section 5 describes how the layout of a printed circuit board can be designed.

2 Realization of cellular automata

Cellular automata [7] are a class of dynamical systems which are discrete in space, state, and time. Each cell holds a state taking on a value from a finite set. The state values evolve in time according to a state transition rule which gives the next state as a function of the current states of a cell and its neighbors. For our purposes, we assume that the transition rule is space invariant. The neighborhood is usually defined by a radius, r, for which any cell inside this radius is in the neighborhood.

If the set of possible states for a cell is binary and the neighborhood is $r = 1$, we will call the cellular automata *first order*. The two-step approach used in [6] can be used to show that the pwuDTCNN with arbitrary template nonlinearities can be used to implement any first order cellular automata. In the first step, the binary current states in every cell neighborhood of the cellular automata are encoded into a unique integer by the standard method using powers of two. In the second step these integers are used to index into a truth table to determine the next state of the cellular automata at each cell. Since any binary function can be represented by a truth table, this allows any binary function of the neighborhood to be implemented.

In general, cellular automata can have more than two possible states per cell. It may be possible to make an argument similar to the first-order case for implementing higher order cellular automata with pwuDTCNN, but the state table increases exponentially with the number of states making it quite impractical. However, if the high-order cellular automata is of a special form, e.g. totalistic, it may be possible to easily design a simple nonlinearity.

3 Analog combinatorics

CNNs outperform traditional digital solutions in lot of applications. Sometimes this is not the case, but it is still worth implementing tasks in CNN, because the problems might occur as part of more complex analogic CNN algorithms and in such cases it is very important to save the time of AD/DA conversions and to be able to solve these subproblems on the CNN. This section contains such image processing and combinatorial tasks that can be solved using pwuDTCNN templates:

- Calculating the histogram of black-and-white images, e.g. shifting black pixels to the left of each line (Fig. 1).
- Sorting values in the [-1,1] interval in a 1D image (Fig. 2).

- Determining the parity of an image, e.g. whether the number of black pixels is even or odd in a line (Fig. 3).
- Majority vote-taker, deciding whether there are more black or white pixels in a line (Fig. 4)

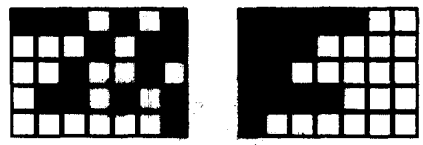

Figure 1: The histogram (right) of a black-and-white image (left)

Figure 2: Sorting values in ascending order (input-left, output-right)

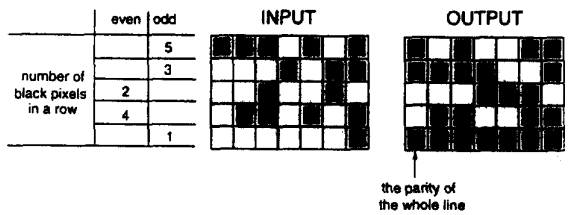

Figure 3: Determining the parity of each line, black standing for 1, white for 0

4 Computing minimum Hamming distance

In the theory of information processes it is a common problem that, given a code received on a noisy channel and the set of legal code words, we have to determine the code word nearest in some metric to the received one. In the case of binary codes the Hamming distance is the most common choice to measure the distance. The Hamming distance of two binary strings is the number of differing bits, e.g. $d(01001, 11011) = 2$. The nearest code word can be computed with a 4-step pwuDTCNN algorithm. In the first step, the input is compared to all legal code words, then the number of differences are counted, afterwards the minimum of these differences is computed, and finally the legal code word(s) having this minimum distance are selected. The whole algorithm requires $(m + n + const)$ steps, where m is the number of legal code words, and n is their length (Fig 5).

5 Layout design with CNN

In this section it will be shown how the layout of a printed circuit board (PCB) can be designed using a cellular neural network universal machine. A 2-layer model is assumed where one layer contains only vertical while the other only horizontal wires. The problem is: given a set of equipotential nodes of a PCB, interconnect them with the minimum

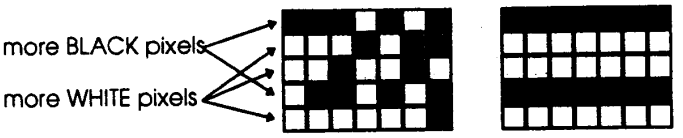

Figure 4: Majority vote-taker (left-input, right-output)

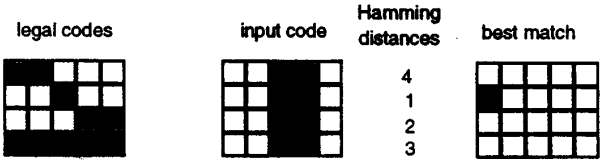

Figure 5: Computing the minimum Hamming distance

wire-length and minimum number of layer crossings, and not crossing any of the already drawn wires. The algorithm terminates in approximately $6 * wirelength * \tau$ time, where τ is the settling time of the analog transient (e.g. around 100ns). Similar methods can be used for IC mask design.

There are only few theoretically exact layout algorithms. The Lee model [9] is perhaps the most well known one, providing a solid, theorem based solution, also taking into account the different technological requirements.

In the procedure of designing the layout of a printed circuit board, a 2-layer model is assumed, where one layer contains only vertical, while the other only horizontal wires. This model can easily be extended to more than two layers. A traditional processor is controlling the algorithm, using a CNNUM as a slave to execute most computing. The processor reads the PCB description file containing the size of the board, the location of the devices and the lists of equipotential nodes. It determines, using heuristics, the order of the design, e.g. which set of equipotential nodes to interconnect first. In our algorithm, first the longest wires are drawn. One major step of the algorithm deals with a set of equipotential nodes. These nodes are sent to the CNNUM which interconnects them with the shortest path, taking into account that restrictions might apply to the position of the wires. These restrictions are represented in the form of fixed state maps - a concept extensively used in the algorithm. It means that some cells do not change throughout the transient. Such restricted locations are the pins of devices and already existing wires which cannot be crossed by later wires. The main steps of the algorithm are:

1. Generate initial fixed state maps with the digital processor containing the pins of the components.
2. Interconnect a set of equipotential nodes. This is the key element of the algorithm which is detailed in the next paragraph.
3. Update fixed state maps by the recently drawn wires (simple logic operation).
4. If more equipotential nodes left, goto step 2.

The key element of this procedure is interconnecting equipotential nodes with the shortest path and the minimum number of layer crossings. This is done by the well-known Dijkstra algorithm [8]. It operates in two step. Assume that we want to interconnect two

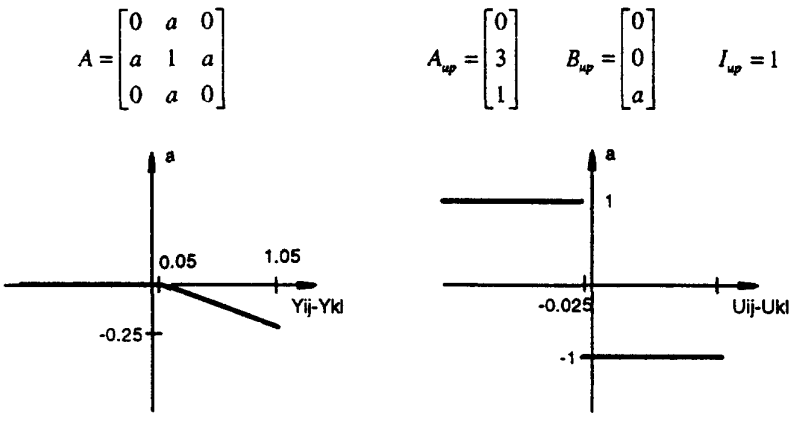

Figure 6: The templates of the shortest path finding algorithm: (a) explores all possible paths from a white source point against black background applying restricted locations to the fixed state map (b) select the shortest path using the result of (a) as input and the black target point(s) as initial state, using the (b) template and its 3 variants (left, down right) cyclically

nodes called source and target. In the first step of the algorithm a wave-like propagation is initiated from the given source point, marking each cell with a value a unit greater than the mark of the cell from which it was reached. In other words, this step is a breadth-first search, exploring all routes starting from the source, and marking each cell with a value corresponding to its distance from the source. In the second step the shortest path has to be selected. This can easily be done based on the marks of the previous step. It follows from the method how the cells were marked, that each cell, except for the source, has exactly one smaller neighbor (a cell with a smaller mark). If we move from a cell to this smaller neighbor, we get closer to the source. So, starting from an arbitrary point, and always moving towards a smaller neighbor, we will finally reach the source on the shortest path. The above algorithm can also be used if having more than two points, finding the shortest path to one selected point (source) from all others (targets). The templates of the shortest path finding algorithm are shown in Fig reff:layouttem.

A PCB layout designed with the CNN is shown in Fig 7.

References

[1] L.O.Chua and L.Yang, "Cellular neural networks: Theory", IEEE Trans. on Circuits and Systems, Vol.35, pp.1257-1272, 1988.

[2] L.O.Chua and L.Yang,"Cellular neural networks: Applications", ibid., pp.1273-1290.

[3] L.O.Chua and T.Roska, "The CNN Paradigm", IEEE Trans. on Circuits and Systems-I, Vol.40, pp.147-156, 1993.

[4] T.Roska and L.O.Chua, "The CNN Universal Machine: An Analogic Array Computer". IEEE Trans. on Circuits and Systems-I, Vol.40, pp.163-173, 1993.

[5] H.Harrer and J.A.Nossek, "Discrete time cellular neural networks", Int.J.Circuit Theory and Applications, Vol.20, pp.453-468, 1992.
[6] L.O.Chua, T.Roska and P.L.Venetianer, "The CNN is as Universal as the Turing Machine", IEEE Trans. on Circuits and Systems-I, Vol.40, pp.289-291, 1993.
[7] T.Toffoli and N.Margulos, "Cellular Automata Machines: a new environment for modeling", Cambridge, Mass., MIT Press, 1987.
[8] E.W.Dijkstra, "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik, Vol.1, pp.269-271, 1959.
[9] C.Y.Lee, "An algorithm for path connections and its applications", IRE Trans on EC, Vol. EC-10, pp.346-365, 1961.

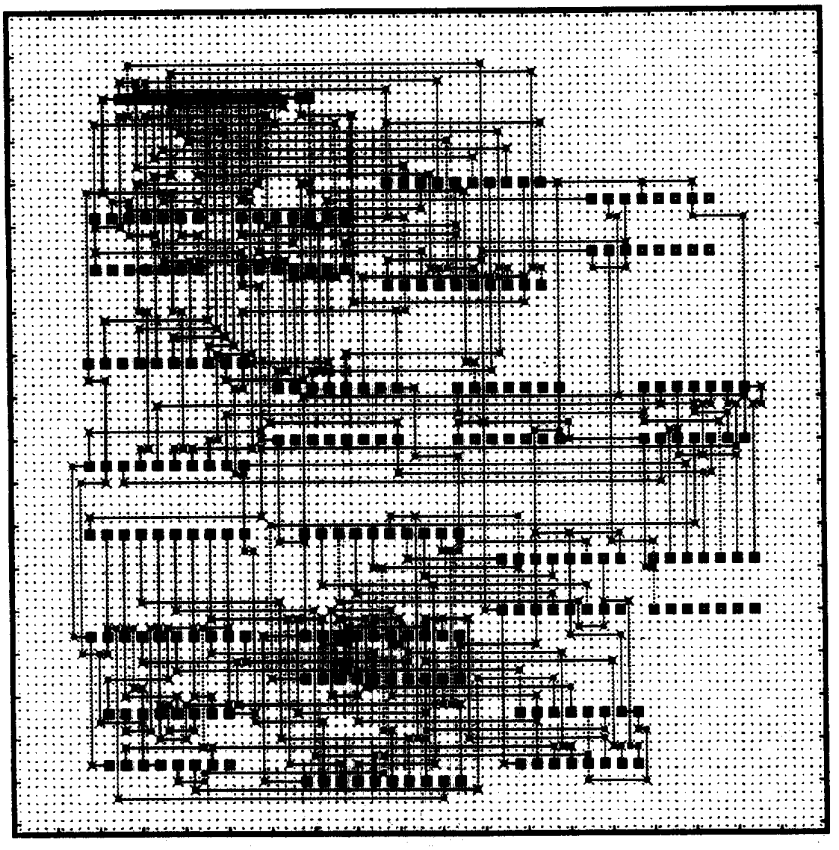

Figure 7: Layout design with CNN: black squares represent the pins of the componenets: grey squares are layer crossings, the horizontal and vertical wires are on different layers, being superimposed on the figure