ISSN: 0975-8585

Research Journal of Pharmaceutical, Biological and Chemical Sciences

Simulation Of Non Deterministic Cellular Automata By Boolean Circuits.

Nisha VM*, and Sajidha SA.

*School of Computer Science and Engineering *VIT University, Chennai Campus
School of Computer Science and Engineering VIT University, Chennai Campus

ABSTRACT

Cellular automaton is a computational model used for parallel language recognition, where the language is a pattern. This paper proposes a new mechanism to incorporate non-determinism in cellular automata. We compare cellular automata and the uniform family of boolean circuits due to their inherent parallelism. A tessellation automaton, which is used as a variant of cellular automata is simulated using a modified Ladner Fischer circuit. A one-way tessellation automata is defined which acts as a non-deterministic time varying cellular automata, which is simulated into Boolean circuits to establish a relationship between them. Modifications are made to the original Ladner Fischer circuit for the simulation.

Keywords: Tessellation automata, Simulation, Ladner Fischer Circuit, Non-determinism.

**Corresponding author:*

INTRODUCTION

There are different parallel computation models of which Boolean circuits and Cellular automata are known for massive parallelism. Cellular automaton is a collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells. The rules are then applied iteratively for as many time steps as desired. In this paper, we will focus on one-dimensional cellular automata with the simplest communication pattern. In one-dimensional cellular automata [1] there are two possible states (labeled 0 and 1) and the rule to determine the state of a cell in the next generation depends only on the current state of the cell and its two immediate neighbors. On the line of cells, the information flow is either two-way or one-way, according to the connection of the cells to its neighbors - to both its left and right neighbors or simply to its left neighbor or right neighbor. We differentiate two-way cellular automata (CA) with one way cellular automata (OCA) [2], where the information flow is in one direction.

We define tessellation automaton [3] as an infinite array of similar finite-state machines, where every machine can receive information directly from finite variety of neighboring machines, where every machine is connected to its neighbors in a uniform way throughout the array. Each machine will modify its state only at discrete time steps as a function of the states of the machines in the uniformly- defined finite set of neighboring machines. This function may be changed from time step to time step, but it is identical for each machine at any given time step. We only consider the one-way model of the tessellation automata, where each cell is communicating only to its predecessor for the simulation. OCA has properties that are not shared by CA. In an OCA the state sequence of cell c only depends on the initial state and the previous state at previous time step. The computation is of sequential nature.

We use the sequential feature to simulate One-Way tessellation Automata (OTA) by Boolean circuits. Ladner Fischer Circuit [4] can be used to simulate any rational transducer by using circuits of smaller depth.
Non-determinism[5] is introduced in cellular automata, to increase the computational power of cellular automata by providing multiple transitions from every state at each time step. The transitions can be selected based on various parameters[6], at each time configurations. Nondeterministic cellular automata exhibit more computational power[7] as a language recognizer.

A non-uniform alternating cellular automata (ACA) [8] is time equivalent to circuit families[9]. A relationship exists between unbounded fan-in circuit family and non-uniform cellular automata. Also a relation exists between uniform ACA and constant fan-in family of circuits.

MATERIALS AND METHODS

DEFINITIONS

A tessellation automata is defined as a four tuple(S, #,F, δ_{nd}),where S is the finite set of states and m S, F is the accepting state(s),# is the quiescent state with a quiescent property of $\delta($ #,#$)=$ # and δ_{nd} is a nondeterministic transition function which takes parameters as input symbols.

$\delta_{nd}(u_1,u_2) \rightarrow 2^m$, where u is the input symbol which is represented as states. The states are stored as machines when compared to Cellular Automata where states are cells. Details of Tessellation Automata are extensively [7]. Tessellation automata are a perfect example which can be used as a machine model that can be converted to Boolean circuit model. The nature of tessellation automata is similar to the act of multiplexer and we can combine these ideas to simulate non determinism in Boolean circuits.

PROPOSED METHODOLOGY

We propose a One-way tessellation automata simulation in a modified Ladner Fischer Circuit with the help of the circuit in Fig. 2. The layered approach[4] from has been used to simulate multiple machines in Boolean circuits. We found out that a polynomial time simulation is possible from Boolean circuit to equivalent cellular automata. So there exists a relationship between Boolean circuits and cellular automata.

A one-way tessellation automaton (OTA) is a one-dimensional array of identical finite automata (machines) numbered 1, 2, etc. from left to right, and working synchronously at discrete time steps. Each machine depends on its left neighbor and takes on a value from a finite set S, the set of states. At every step, the state of each cell is computed according to a transition function δ_{nd} which is selected from a set of transition functions

$$\delta_{nd} = \{\delta_1, \delta_2, \delta_3, \delta_4\}$$

Involving its own state and the state of its left neighbor. Formally denoting <m, t > the state of the machine 'm' at time t, we have <m, t> = δ_{nd}(<m − 1, t – 1>, <m, t – 1>). Because the first machine 1 has no left neighbor, we use a special state ♯ not in S as a border state: <1, t> = δ_{nd} (♯, <1, t – 1>). As language recognizer, we have to specify two subsets of S: the input alphabet Σ and the set of accepting states S_{accept}. The input mode is parallel. At initial time 1, the i^{th} bit of the input word k = $x_1 \cdots x_n$ is fed to the cell i: < i, 1> = x_i. The output machine where the result of the computation is displayed, is the machine numbered by |w| the size of the input. That is the first machine which can get all the information of the input k. An OTA accepts a word k, if on input k the output machine enters an accepting state at some time t. Let f be a function from N into N. An OTA accepts a language L in time f, if it accepts exactly the words k ∈ L of length |k| at time t ≤f(|k|).

Let us emphasize how each cell c of an OTA (S, δ_{nd}) behaves as a finite transducer. This finite transducer constitutes the set of states, the input alphabet as well as the output alphabet is S, the transition function as well as the output function is δ_{nd} and the initial state is <m,1> the state of the machine m at initial time 1. On input word <m − 1,1>⋯<m − 1,t>, the machine m starting in state <m , 1>, successively (enters in states and) outputs symbols <m , 2>… <m , t + 1>. Besides, we can interpret the partial runs of the cell using the maps δ_u defined as follows. For any u = $u_1 \cdots u_n \in S^*$, the map $\delta_u : S \to S$ is obtained by applying the transition function δ on the successive symbols of u: $\delta_u(s) = \delta(u_n, \delta(u_{n-1}, \ldots, \delta(u_2, \delta(u_1, s)) \ldots))$. Observe that these t of these maps F = {δ_u: u ∈S^*} is finite, as S is finite. The behavior of the OTA (S, δ_{nd}) can be expressed in terms of these maps δ_u. Let u = <m − 1,t_1><m − 1,t_1 + 1>⋯<m − 1,t_2 − 1> be the states sequence of the machine m − 1 at consecutive steps t_1, t_1+1,...,t_2−1:we have <m,t_2>=δ_u(<m ,t_1>).

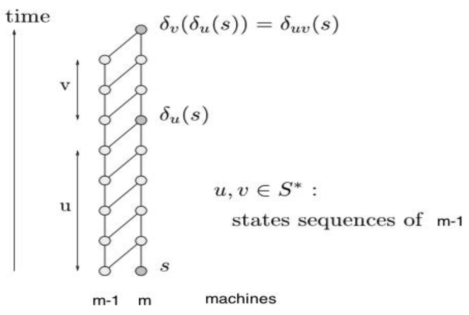

Fig. 1 . State sequence diagram

Now the OTA acts as a finite transducer. The mapping can be done to Boolean Circuits now. The δ_{nd} function can be mapped into circuits by exploiting the non-deterministic behavior. According to the rules of tessellation automata. At every state of the machine the transition is selected from a set of global transitions. But each state of the same configuration C_i will have the same transition function. The corresponding circuit is shown in Fig. 2.

We use four transition functions in our example of $\delta_{nd.}$ Let's' and 'a' be two states of 'm' and 'm-1' of time t. The transition function δ_{nd} is mapped into Boolean circuit by using four gates for the four transitions and then using a multiplexer for selecting anyone of the transition. Based on the values of's' and 'a' any one of the four transition function can be selected at the output line of the multiplexer, ie. $\delta_{n..}$

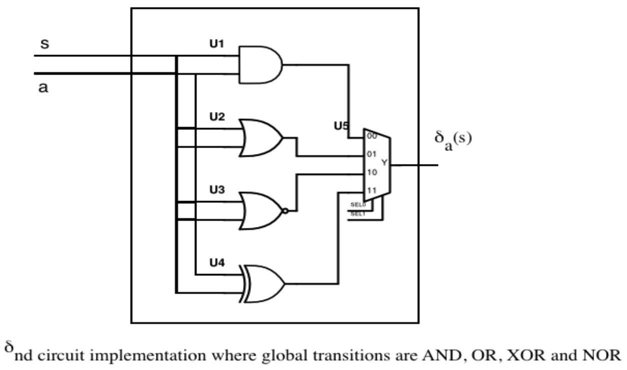
s
a
U1
U2
U3
U4
U5
δa(s)
δnd circuit implementation where global transitions are AND, OR, XOR and NOR

Fig. 2. δ_{nd} implementation in circuit

Now as we have defined $\delta_{nd.}$ we can transform the working of a single cell into Ladner Fischer circuit. In, Ladner and Fischer [4] have designed a small boolean circuit to simulate any finite state transducer. This circuit will be the key component to realize the simulation of OTA by Boolean circuits. Precisely, this circuit can simulate the work of any one machine of an OTA.

Let us have a look at this Ladner–Fischer circuit. See Fig. 3. In the previous section, we have stressed that the behavior of an OTA (S,δ_{nd}) can be expressed in terms of maps δu for u ∈ S∗. Precisely the machine m up to time t goes through the states <m, w> = δ<m−1,1>···<m−1,w−1>(<m, 1>) for all steps w = 2, . . . t. Actually, to simulate the work of the machine m up to time t, the Ladner–Fischer circuit will compute the maps δu for all prefixes u of <m− 1,1>···<m − 1,t – 1>. For that purpose, using a ‘‘divide and conquer’’ strategy, the circuit computes some intermediate values δv where v are sub words of <m−1,1>···<m−1,t−1>.In the Ladner–Fischer circuit, each computed value δv, where v = <m−1,i>···<m−1,j>,is carried out at depth log(j + 1 − i) and width j. Notably, there exists at most one value computed at a given depth d and a given width w. Furthermore, if we mark every position (d, w) either by ‘1’ or by ‘0’ depending on whether the circuit computes or not a value at depth d and width w, then the circuit yields the geometry of a binary counter.

The depth and the size of the Boolean circuit are of the same order than the depth and the size of the dependency graph. Indeed, the nodes of the dependency graph computes either from a binary representation of a state s ∈ S a binary representation of δs, either from two binary representations of maps δu and δv a binary representation of δv ◦ δu or from the binary representations of a map δu and a state s a binary representation of δu (s). As the set of states S and the set of maps F = {δu : u ∈ S∗} are finite, these types of nodes can be carried out by constant size Boolean circuits over {And, Or, Not}.

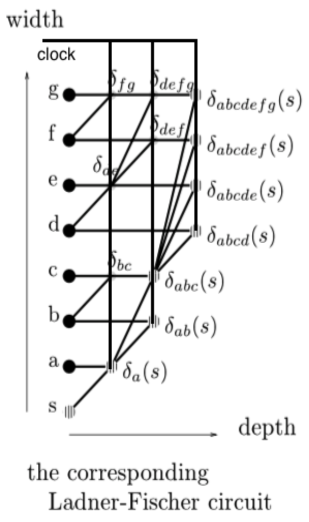

Fig.3. Corresponding Ladner Fischer circuit

In the Ladner Fischer circuit model each δ computation refers to the circuit in Fig 2. These computations are performed at each level in Ladner Fischer circuit where the computations are performed. A clock is provided to all the levels so that non determinism is performed through the multiplexer. At each depth the transition used is the same. The transition changes only at different depths.

Simulation of multiple cells can be done using the layered approach[3]. Each layer can be implemented in a parallel fashion.

RESULTS AND DISCUSSION

One-way tessellation automata are hence simulated in a modified Ladner Fischer Circuit with the help of the circuit in Fig 2. The layered approach from has been used to simulate multiple machines in Boolean circuits. We found out that a polynomial time simulation is possible from Boolean circuit to equivalent cellular automata. So there exists a relationship between Boolean circuits and cellular automata.

APPLICATION

Cellular automata are one of the best models for modeling biological cells. Biological organisms may contain dozens of organs composed of tissues containing massive amount of cells. Each cell may interact with its surrounding cells, hence treats cells as simple interacting elements. Since CA can simulate interactions of mass amount of cells, it can reproduce all common types of cell behavior. Using nondeterministic cellular automata, complex behavior of cells also can be modeled. An advantage of using CA in biology is its simplicity and ease of implementation. In addition CA reflects intrinsic individuality of the cells. Since CA can simulate to Boolean circuit, it can be shown that Boolean circuits also can be used to model biological cells.

CONCLUSION

We found out that a polynomial time simulation is possible from Boolean circuit to equivalent cellular automata. So there exists a relationship between Boolean circuits and cellular automata. Moreover the circuit complexity classes such as Nick's class and cellular automata complexity classes are related. An investigation on both the complexity classes is considered for the future work.

REFERENCES

[1] Wolfram. Cellular Automata. S.Los Alamos Science 9 (1983); 2-21.

[2] C.Dyer.One way bounded cellular automata. Information and control1980; 44: 261-281.

[3] H Yamada. Tessellation Automata. Information and Control. (1969); 4: 299—317.

[4] V.Terrier. Simulation of One Way Cellular Automata. Elsevier. Theoretical Computer Science 411 (2010); 266–276.

[5] Frank Reischle and Thomas Worsch .Simulation between alternating CA, alternating TM and circuit families. University of Karlsruche.. 9/98; Technical report.

[6] Martin Kutrib. Non-deterministic cellular automata and languages. International Journal of General Systems Vol. 41, No. 6, August 2012; 555–568.

[7] Gérard Cécé, Alain Giorgetti .Simulations over Two-Dimensional On-Line Tessellation Automata. Volume 6795 of the series Lecture Notes in Computer Science pp. 141-152.

[8] Yuri Ozhigov. Computations of nondeterministic cellular automata. Information and Computaion,148: 181-201,1999.

[9] Martin Matamala .Alternation on Cellular automata. Theoretical Computer Science. 1997;180:229-241.

[10] K. K.Kritivasan, M.Mahajan.Nondeterministic, Probabilistic, and alternating Computations on cellular array models. Theoretical Computer Science.1995;143:23-49.