The Journal of Mathematics and Computer Science

Available online at
http://www.TJMCS.com

The Journal of Mathematics and Computer Science Vol. 4 No.2 (2012) 264 - 269

The Implementation of Cellular Automata with Non-identical Rule on Serial Base

Nazanin Moarefi[1], Ali Yarahmadi[*2]

[1] Young Researchers Club, Islamic Azad University, Doroud Branch, Doroud, Iran,
nazaninmoarefi@gmail.com
[2] Faculty of Engineering-Department of Computer, Islamic Azad University,
Boroujerd Branch, Boroujerd, Iran,
alyarahmadi@gmail.com

Received: January 2012, Revised: April 2012
Online Publication: June 2012

ABSTRACT

A cellular automaton is a calculation method which strongly needs high-speed processing of data. In cellular automata, a very large volume of data must be processed in a short time, so that the results of cellular automata can be used.

Different methods can be used to implement cellular automata, among which the implementation of cellular automata on serial bases is one of the simplest ones. Two types of implementing cellular automata on a serial base were studied in this paper.

In a method of implementation, the whole cells of automata the use the same rule; and in another method, the rule related to cells is different depending on the location. Then, the speed of implementation of the two methods is compared: it was shown that the speed of operations in the serial implementation is independent of cellular automata rule.

Keywords: Cellular automata, Implementation of cellular automata.

INTRODUCTION

One of the applications that require data processing is cellular automata, in which a very large volume of data must be processed in a short time, and the results of the processing

[*] Corresponding Author

must be reapplied to the automaton so that the results produced by automata can be used after a sufficient number of processing operations was carried out [2,1].
Cellular automata, in fact, are the computational models that were first studied by John Von Neumann in 1940 [3]. The main purpose of these automata was to design the self-replicating systems that are computationally complete.
There are various methods for implementation of cellular automata, the simplest of which is implementation of cellular automata on the serial base and the use of a processor to perform the operation.
Since only one processor is used in this method to calculate the new value for the cells of cellular automata, it is expected that if a serial base is to be used for implementing cellular automata, the rule for each cell will be ineffective in the time required to calculate the new value of the cell and the speed of operation will be independent of the rule for each cell.
In this paper, implementing cellular automata on the serial base was examined. It has been shown that if we put cellular automata on the base, the speed of operations is independent of the rule for each cell. Then, the hardware implementation has been made for cellular automata with both similar and non-identical rules for cells on the serial base, and speed of each method of implementation was compared with other methods for cellular automata.

Characteristics of serial base

One of the simplest methods already used for the implementation of cellular automata is serial methods, in which only one processor is used to calculate the new values of the cells, and the operation is done only with using one processor.
The use of one processor causes a queue of cells that intend to calculate new values for themselves; and only one cell can use the processor to calculate its new value at any moment [4]. In this case, it has only one processor cell to compute its new value, which provides the processor to other cells so that these cells can use the processor and calculate their new values.
Since a single cell can use the processor in this method at any moment, the overall speed of calculation is much slower than the parallel methods [5].
If the speed of the processor responsible for performing calculation in serial implementations can be raised, a new value for each cell can be calculated faster, and the overall rate of cellular automaton rises and improves [6]. Although the use of high-speed processor increases the speed of operation, much speedup still cannot be reached due to the large queue of cells waiting for calculating the new value.
Since it has only one cell in each processor, the speed of operations is expected to be independent of the cell rule.
In this paper, the accuracy of the above assumption is examined by the hardware implementation of cellular automata for cellular automata with the same rules and non-identical rules for the cells on the serial base.

Implementation of cellular automata on the serial base

In this method, a processor, called PE, which is designed for the calculation to obtain the new value of each cell, is used. The number of processors is assumed to be only “1”, and the number of cells in cellular automata is assumed to be NC.
To implement PE processor, VHDL instructions are used; and the test bench related to implementation of cellular automata is written using Verilog. In this method, a series of

memory is used to store values in automata cells so that a box of memory is considered for each cell of cellular automata.
Values in each cell are located in "current state" memory; and new values will be located in "next state" memory after the data of the cell are processed by PE. Then, the amount of the "next state" memory will be replaced in the "current state" memory. See Figure 1.

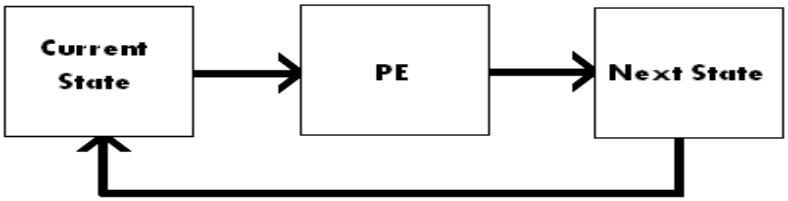

Fig 1: Memory structure in the implementation of cellular automata on the serial base

In this implementation, the calculations are done at positive edge of clock signal; replacement operations will occur for the new values of cells after the new values of all cells are calculated in the first negative edge of clock signal. Each PE processor has six input pin and one output pin.
Three of the input pins are related to the amount of the cells on which processing must be done; and two other input pins are related to the clock and reset. In addition, the other input pin is also considered to determine the rule of cell, which varies depending on the value of zero or one of the pin. The output pin, which transfers the result of processing of three cells to outside of PE, is called "result". In this method of implementation, only one PE processor is used for all the cells, and PE is implemented using VHDL. The reset has an asynchronous structure in the processor, which its output pin will be zero upon activation.
PE computes the new value for the middle cell when it receives the value of three cells side by side in the one-dimensional cellular automata. To calculate the new value of the middle cell, it uses the automaton rule which is determined by the amount of rule pin. Depending on the amount of rule pin, the rule which is used by PE can be identified.If the amount of this pin is equal to "zero", the rule 90 will be used; and if it is equal to "one", the rule 150 will be used to calculate the new value of the middle cell. Figure 2 shows an example of PE simulation; and Figure 3, VHDL instructions related to the implementation of PE.

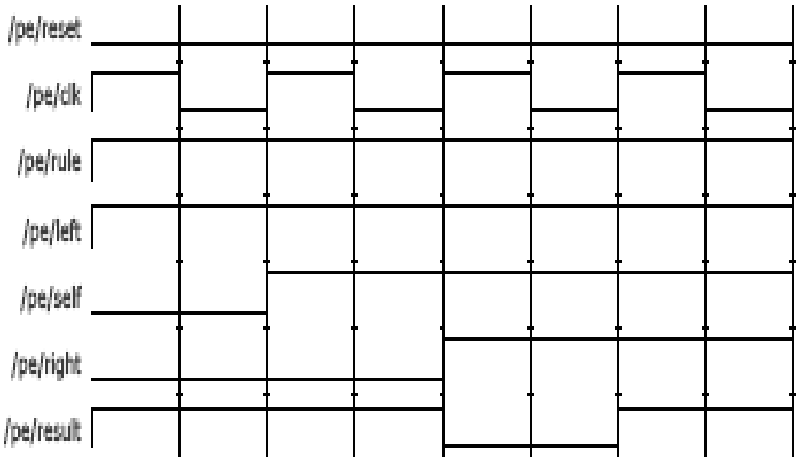

Fig 2: An example of PE simulation

```
entity PE is
   port(
   Left    : in bit;      --left cell value
   Self    : in bit;      --current cell value
   Right   : in bit;       --right cell value
   Result  : out bit;     --result of process
   Clk     : in bit;      --clock
   Reset   : in bit;       --reset pin
   Rule    : in bit       --rule of cell
   );
end PE;
--if reset then current cell value = 0
--else for order=1 current cell value = rule 150
--and for order=0 current cell value = rule 90
architecture Struct of PE is
   begin
   process (Left,Right,Clk,Reset,Rule)
   begin
    if (Reset='1')then
        Result<='0';
    elsif Clk'event and Clk='1' and Clk'last_value='0' and Rule='1' then
       Result<=(Left xor Self)xor Right;
    elsif Clk'event and Clk='1' and Clk'last_value='0' and Rule='0' then
       Result<=Left xor Right;
   end if;
   end process;
end  Struct:
```

Fig 3: VHDL Code related to the PE implementation

A processor called PE is used to simulate the serial form of cellular automata. In this simulation, NC different cells are used, and the calculation will be performed MR times.

In this method, cells are numbered from zero to NC-1; and values of the related cell and two adjacent cells are added to PE to calculate new values for each cell. As was said, a PE is used for all cells.

Depending on the value of rule pin and according to the rule which is considered for cellular automata, PE processor will calculate the new value for the related cell, and put the result in the "next state" memory.

The new value for all cells is calculated at the positive edge of clock signal. After the new value was calculated for all cells, new values for each cell are placed within the "next state" memory; and in the first negative edge of clock signal, these values will be replaced in "current state". See Figure 1.

The cells are numbered from zero, and are initialized in such a way that the values of all cells (except cell NC-1) are assumed to be zero; and only the value of the cell NC-1 is equal to one. Whenever a new value is calculated for the cells, one unit is added to a counter, and the processing operation is terminated when the value of the above counter is equal to MR.

Method of comparison

In this paper, it is first assumed that all cells of cellular automata use a fixed rule and that cellular automata are implemented on a serial base. Then, it is assumed that the cellular automata rule is different for the cells that are in even and odd positions, and implementation has been done again.

In this implementation, the cells that are in even and odd positions use rule 90 and rule 150, respectively. In both simulations, the end cells are intermittently neighbored together. ModelSim 6 software was used for simulation, some results of which are presented in tables 1 and 2 and Figures 4 and 5.

Table 1. Implementation for automaton with 50 cells

Time in Similar Method	Time in Dissimilar Method	Number of Repetition	Number of Cells
26030	26030	25	50
51030	51030	50	50
101030	101030	100	50

Table 2. Implementation for automaton with 10,000 cells

Time in Similar Method	Time in Dissimilar Method	Number of Repetition	Number of Cells
1000200030	1000200030	5000	5000
2000200030	2000200030	10000	10000
4000200030	4000200030	20000	20000

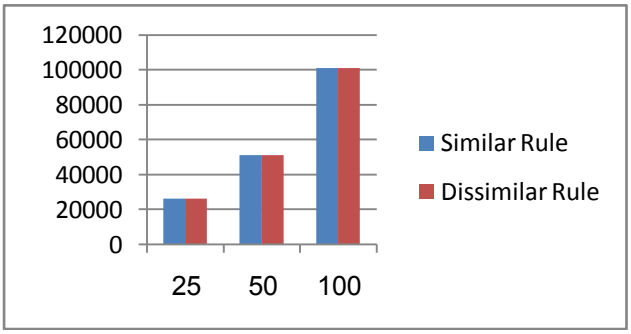

Fig 4: Comparison of the speed of implementation for 50 cells.

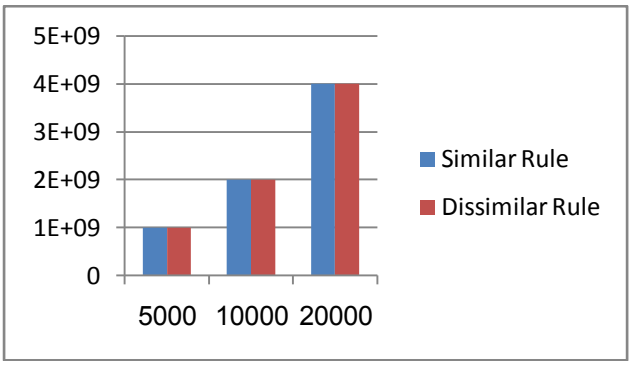

Fig 5: Comparison of the speed of implementation for 10 000 cells

CONCLUSION

In this paper, implementation of cellular automata on serial base was studied both with the same and different rules for cells. Considering the tables and charts in this paper, it can be seen that the speed of operation is independent of the cell rule if the serial bases are used to implement cellular automata.

REFERENCE

[1] M.Mosleh, S. Setayeshi, M. Mehdi Lotfinejad, and A.Mirshekari, , "FPGA implementation of a linear systolic arrays for speech recognition based on HMM," The 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, vol. 3, pp. 75 - 78, April 2010.

[2] M.Mosleh, S. Setayeshi, and A. M. Rahmani, "A synergy between HMM-GA based on stochastic cellular automata to accelerate speech recognition," in Magnetism, Vol. 6, No. 18 pp.1304-1311, 2009.

[3] Y.BarYam, "Dynamics of complex systems," United States of America: Addison-Wesley, 1997.

[4] N.Ganguly, K.Sikdar Biplab, A. Deutsch, G. Canright, and P. Pal Chaudhuri, "A survey on cellular automata," Technical Report, Center for High Performance Computing, Dresden University of Technology, 2003.

[5] J.L. Guisado, F. Fernández de Vega, F. Jiménez-Morales and K.A. Iskra," Parallel Implementation of a Cellular Automaton Model for the Simulation of Laser Dynamics," Lecture Notes in Computer Science, 2006, Volume 3993/2006, 281-288, DOI:10.1007/11758532_39

[6] H. T. Kung, "Why systolic architectures?," IEEE Computer Society Press Los Alamitos, CA, USA, 1982.(ISSN:0018-9162)