International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May 2018 www.irjet.net p-ISSN: 2395-0072

Design of an Efficient Hardware Searching algorithm

Ashish Jadhao[1], Vanita Agarwal[2]
[1]*M. Tech, Electronics and Telecommunication Department, College of Engineering, Pune, India*
[2]*Assistant Professor, Electronics and Telecommunication Department, College of Engineering, Pune, India*

Abstract - *Searching algorithms has been widely investigated to increase its speed. Also, Cellular automata (CA) with evolutionary rules has been a significant area of research. Speed is a very important factor in any search algorithms. Many researchers have concluded that Searching with the help of hardware is always fast than software. Therefore, we propose a hardware implementation of a searching algorithm. This algorithm has been designed with the help of Cellular Automata (CA). The Rule induction for cellular automata is done by using Learning from examples module 2 (LEM2) algorithm. It is a Rough set approach; Rough set approach has also been used for updating the CA rules. This algorithm has been implemented on Xilinx Artix 7 FPGA.*

Key Words*: *Cellular Automata(CA), Learning from examples module 2(LEM2)

1. INTRODUCTION

A cellular automaton is a mathematical model or a system of cells with some specific characteristics. Cellular Automata (CA) are discrete mathematical models. They are the systems that have self-reproduction and chaotic behaviour and are determined by simple rules. The Cellular Automata have following characteristics:

- Key elements of a Cellular Automata are grids, states and neighbourhoods.
- Each cell lives on a grid.
- Every cell has a state or value having finite state capabilities.
- Every cell has its neighbourhood. It can be described in multiple ways, generally it is an array of adjacent cells.
- A new state of a cell is a function of value of that state and its adjacent cells at the previous time instance.
  Cell state at time t= f (cell neighbourhood at time t-1)
- In Elementary Cellular Automata, we have three cells, each with state of 0 or 1 as shown in fig.1. Black states are denoted as '1' and white represents '0'.

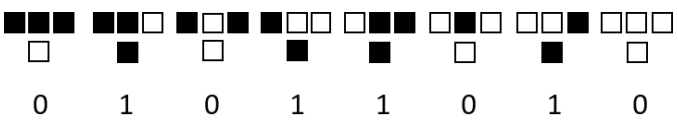

Fig. 1: Rule 90 of cellular automata

Fig. 1 shows an example of Cellular Automata using Rule 90. Rule 90 is given by the modulo-2 addition of both the neighbourhoods.

The evolution of a cell depends on its own value and those of its nearest neighbours. The local rule is the function of the cells within the neighbourhood [5] and expressed as

$$T(x) = \beta_{i-1}x^{i-1} * \beta_i x^i * \beta_{i+1}x^{i+1} \quad \text{..... (1)}$$

'*'- denotes any binary operation

In Elementary CA, we can generate total 256 rules. Each rule gives different solution on next time instance.

The total configuration of a CA is specified by the values of total number of cells [5] and represented by a characteristic equation

$$A^{(t)}(x) = \sum_{i=0}^{N-1} a_i^{(t)} x^i \quad \text{..... (2)}$$

where, the value of cell i is the coefficient of x_i

Such behaviour of Cellular automata can be used in Genetic algorithms and Data mining methods. This has been a wise topic in Artificial Intelligence. Therefore, Researchers have studied the behaviour of CA and has been widely used in some applications. Also, Hardware implementation will increase its speed and efficiency [6]. Therefore, we have also this algorithm on hardware.

This paper explains about the FPGA implementation of a searching algorithm using Cellular Automata and a Rough set approach for generating the rules of Cellular Automata. The next section explains about the methodology to find the Rule set. The complete LEM2 algorithm with pseudocode and steps to solve the algorithm. The third section gives the results given by LEM2 algorithm. The fourth section explains the conclusion and future work.

2. METHODOLOGY

In general, the problem of CA identification is to find the cell neighborhood and updating rule based on the different input data set, to solve this type of problem, the input dataset should be represented in the form of a decision table [2]. Decision table is a set of observations and cell states. Thus, rough set approach is used to identify rules and to update rule of CA [2].

Before that, we need to induce rules for CA which can be done by several algorithms. One of them is LEM2 (Learning from examples module 2). Among other rule induction algorithms, this one gives better results [1]. It is

a local rule induction algorithm [1]. It means the search is a set of attribute-value pairs.

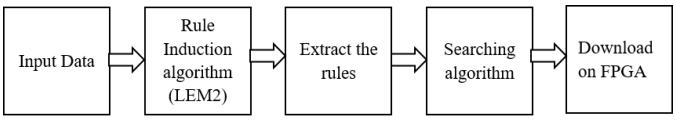

Fig. 2: Block diagram of complete process

The input data should in the tabular form with decision attributes. Suppose, there are four or five attributes which defines the decision attribute, which means the value of decision attribute is decided by the pairs of values of four or five input attribute. LEM2 algorithm gives rule set which is enough to define the decision. Local covering will give reduced set of attribute-value pairs. Here, the Reduct is the local covering of input data set. That set is enough to define the decision attribute-value. In a searching, instead of checking all attribute-value pairs for a output we can check reduced set of attribute-value pairs which are conditional attributes. In this way, we can reduce the time complexity and hardware utilization. For different input dataset, there are different rule sets. It is based on the number of attributes and their values.

Table - 1: Dataset for defining the rules

Case	Attributes				Decision Flu
	A1	A2	A3	A4	
1	V	Y	Y	N	Y
2	H	Y	N	Y	Y
3	N	N	N	N	N
4	N	Y	Y	Y	Y
5	H	N	Y	N	Y
6	H	N	N	N	N
7	N	N	Y	N	N

From the above table, some terms are defined as follows:
V= very high
H= high
N= normal (for attribute A1 only)
Y= yes
N= no (for attributes other than A1)

For the example given in Table 1, the LEM2 algorithm and symbols used in the algorithm are explained further.
Let U be a set of all observations and cell state values. A set of m is a set of all cases from U such that for attribute a have value v i.e. m = (a, v). Let X be a set of decision-value pair (d, v). Set X depends on a set M of attribute-value pairs m = (a, v) only if it satisfies the following equation

$$\emptyset \neq [M] = \bigcap_{m \in M} [m] \subseteq X \quad \ldots\ldots (3)$$

Set M is a minimal set of X's if and only if X depends on M and there is no real subset M' of M should exists such that X also depends on M'. Let Γ be a set of attribute-value pairs. Then Γ is a local covering of X if and only if the following two conditions are satisfied:
(1) Each member M of Γ is a minimal of X,
(2) $\bigcup_{m \in \Gamma}[M] = X$, and
Γ is minimal, i.e., Γ has the smallest possible number of members.
The following is the pseudocode for the LEM2 algorithm [1]:

(Input: set X,
Output: single local covering Γ of set X);
begin
Z: = X;
Γ: = Φ;
while Z ≠ Φ;
begin
M: = Φ;
M(Z): = {m|[M] ∩ Z ≠ Φ}
while M = Φ or [M] ⊄ X
begin
Select a pair m ϵ M(Z) such that |[m] ∩ Z| is maximum;
if a tie occurs, select a pair m ϵ M(Z) with the smallest cardinality of [m]; cardinality means number of cases present in that set.
If another tie occurs, select first pair;
M: = M ∪ {m};
Z: = [m] ∩ Z;
(Z): = {m|[M] ∩ Z ≠ Φ;
(Z): = M(Z) – M;
end {while}
for each m ϵ M do
if [M – {m}] ⊆ X then M: = M – {m};
Γ: = Γ ∪ {M};
Z: = X – $\bigcup_{M \in \Gamma}[M]$;

end {while};
each M ε Γ do
if $\bigcup_{S \in \Gamma - \{M\}}[S] = X$ then Γ: =Γ –{M};
end {procedure}

1. The first step is to compute the elementary set of all attribute with their values. These blocks contain set of cases which has specific attribute and its value. There can be multiple sets as one attribute can have more than one value.
2. Second step is to select the Concept for which rule should be induced. Concept is the set of all cases denoted by same decision value [1]. After selecting concept, compute the set of attribute-value pairs of those cases.
3. Next step to compare the sets of steps 1 and 2 and the intersection of these sets should be maximum. If the cardinality of more than one set is same, then the next criteria is the size of attribute-value pairs. The size should be smaller. By satisfying these two conditions, the resultant set will be the first Reduct of the given concept.
4. Check if all the cases of concept are covered or not, if not, then follow the step 2 and step 3 until all the cases could not cover.

These steps will give the local covering and rule set is defined by these local covering. Those rule set will contain attribute-value pairs. Hence, for searching of a decision, we need not check all the attributes. Just search these attributes and their values. Every rule set follows a specific pattern. That's why, we can use cellular automata for specific pattern. We can just map those rules set to CA rules.

LEM2 algorithm gives better results. Also, CA can give pattern for searching algorithm. CA can be used in pattern searching and many searching algorithms.

As we know, Cellular Automata consists of cell states, therefore the generated rules can be labeled as a one state. Based on number of reducts induced by this algorithm, the dimension of CA is determined. If number of states is less than three, then 1-D CA which is also known as Elementary CA has been used, otherwise 2-D CA is used. In our example, two reducts have been generated which can be labeled as two states. Hence, we have Elementary CA in Searching algorithm. In Searching algorithm, instead of looking for whole set of attributes and their values, now we are checking these reducts for the rest of searching and it reduces time complexity and hardware utilization.

3. RESULTS

For the given dataset in Table 1, the rule set generated by LEM2 is,
For the concept (flu, Y), the rules generated are (A2, Y) and (A1, H) & (A3, Y).
We are getting the same result by solving theoretically and by using coding method. The following shows the Synthesis Report for code of LEM2 algorithm.
The operating clock frequency on FPGA is 100MHz.
Following is the timing analysis of the code on hardware.

Table - 2: Timing (ns)

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	6.23	1.25

Table - 3: Latency (clock cycles)

Latency		Interval	
Min	Max	Min	Max
169	345	169	345

Following is the usage of hardware requirements for this code.

Table - 4: Utilization Estimates

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	142
FIFO	-	-	-	-
Instance	0	-	299	2323
Memory	0	-	832	53
Multiplexer	-	-	-	878
Register	-	-	106	-
Total	0	-	1237	3396
Available	780	740	269200	129000
Utilization (%)	0	0	~0	2

4. CONCLUSION

This paper presented the hardware implementation of LEM2 algorithm and Cellular automata in searching technique. Artix7 FPGA was used for the proposed work. From the results, it is shown that total utilization LUTs is just 2%. Although, this synthesis is for one input dataset. It may change for different inputs as each input can have different number of attributes and values, but this algorithm will generate better results as compared to others. Future work includes defining the states of Cellular Automata using the generated rules.

ACKNOWLEDGEMENT

Authors would like to thank Prof. A. B. Patki for his valuable inputs and timely suggestions which helped us in timely completion of our proposed work. We would also like to thank all the VLSI lab members of COE, Pune for their constant support during this period.

REFERENCES

[1]Jerzy W. Grzymala-Busses, "RULE INDUCTION", University of Kansas

[2]Bartlomiej Placzek, "Neighborhood selection and rules identification for cellular automata: a rough set approach"

[3]KEN-ICHI MAEDA and CHIAKI SAKAMA, "Identifying Cellular Automata Rules" , Journal of Cellular Automata, Vol.2, pp 1-20

[4]Xianfang Sun, Paul L. Rosin, and Ralph R. Martin, "Fast Rule Identification and Neighborhood Selection for Cellular Automata", IEEE Transactions on systems, Man and Cybernetics, Vol. 41, No. 3, June 2011.

[5]Olivier Martin, Andrew M. Odlyzko and Stephen Wolfram, "Algebraic Properties of Cellular Automata", Communication in Mathematical Physics 93, 219-258,1984

[6]F.K. Hanna and A.K. Misra, "Hardware realization of binary search algorithm," IEEPROC, Vol. 127, Pt. E, No.4, JULY 1980

[7]Robert S. Boyer, J Strother Moore, "A Fast String Search Algorithm," Communications of ACM, Vol. 20, Number 10, October 1977

[8]Hossam E. Mostafa, Ahmed I. Khadragi, Yasser Y. Hanafi, "Hardware Implementation Of Genetic Algorithm On FPGA," 21th National Radio Science Conference (Nrsc2004)(NTI) March 16-18,2004