Efficient Design and Implementation of DFA Based Pattern Matching on Hardware

Aakanksha Pandey[1], Dr. Nilay Khare[2] and Akhtar Rasool[3]

[1] M Tech Information Security, Maulana Azad National Institute of Technology Bhopal, Madhya Pradesh 462003, India

[2] Faculty of Computer Science Engineering, Maulana Azad National Institute of Technology Bhopal, Madhya Pradesh 462003, India

[3] Faculty of Computer Science Engineering, Maulana Azad National Institute of Technology Bhopal, Madhya Pradesh 462003, India

Abstract

Pattern matching is a crucial task in several critical network services such as intrusion detection. In this paper we present an efficient implementation of the DFA with optimized area and optimized memory by the introduction of state minimization algorithm. By using minimized DFA the clock frequency reduces to 40% of the original and the area also reduces to 30%. This optimized architecture of DFA is simulated and synthesized using VHDL on the Xilinx ISE 12.4..

Keywords: *String Matching, DFA, VHDL.*

1. Introduction

With the increased amount of data transferred by network the amount of malicious packet also increased therefore it is necessary to protect the network against malicious attack. Intrusion Detection Systems (IDS) are emerging as one of the most promising way of providing protection to systems on the network against these malicious attacks. Intrusion Detection System continuously monitors the network traffic for suspicious pattern and informs the administrator to take proper action. String matching is the heart of IDS. String matching matches each incoming packet against some stored patterns and identify the suspicious activity. The pattern matching can be implemented in both software and hardware. The main motivation of implementing it into the hardware is the performance gap. Hardware provides efficient and flexible way of implementation. FPGA (field Programmable Gate Array) provides flexibility and FPGA based pattern matching increase the performance of software based system by 600x for large patterns.

A deterministic finite automaton (DFA) is a simple language recognition device. It can be seen as a machine working to give an indication about strings which are given

in input or it can be given a mathematical definition and provide string matching. Most of the papers deal with the string matching but none of them was able to present a method which is fast as well as having optimized area .The main problem of string matching is the area efficiency and memory optimization. This paper deals with the use of minimized DFA for pattern matching with reduced memory requirement and optimized area. Our method used as a optimization that reduce the number of transition, memory size and area in DFA.

The rest of the paper is organized as follows section 1 is the introduction part section 2 describe the related work in this field and section 3 presents the background information of the work, next section 4 deals with the implementation and result and last section 5 is the conclusion part.

2. Related Work

In the past few years numerous hardware based pattern matching solution have been proposed. The main techniques are CAM based architecture [5,9,12].This architecture uses discrete comparator results higher throughput with increased area and low efficiency, other technique is hash function[2,6,11,12] that is used to compress the string set find probable match and reduce the total number of comparison , other one is regular expression and finite automata based pattern matching [1,2,3,4,5,11,13] results low throughput with increase the area of implementation .the main aim of this paper is to

reduce the area of implementation and resource used by applying the state minimization algorithm.

Ioannis et al[10] has given the CAM based architecture uses discrete comparator for pattern matching in which the frequency of the pattern matched get increased but the comparator required to implement the model increases with number of patterns so they uses decoded CAM architecture for better performance and to reduce area density and pipelined CAM to increase processing speed they conclude that pipelined DCAM is the best choice for hardware implementation of pattern matching.

Recently Dhanpriya et al[11] have designed word split hash algorithm in which on the basis of sub hash the pattern is matched .So the malicious packet is detected at the initial stage if so. This architecture reduces the total number of comparison and also reduces the execution time.

Sidhu and Prasanna[14] mapped the NFA into an FPGA results the modest throughput with large area so Karuppiah and Rajaram[1] recently mapped the regular expression into DFA which reduces the number of states used results the area efficiency.

3. Background

3.1 Regular Expression

It is the most common way to represent the pattern to match. Full regular expressions are composed of two types of characters .The special characters (like the * from the filename analogy) are called metacharacters**,** while everything else are called literal. Literal text acting as the words and metacharacters as the grammar. The words are combined with grammar according to a set of rules to create an expression which generate patterns. Some metacharacters are *,+,?,|,Repetition is specified with *, for
zero or more, +, for one or more, and ?, for zero or one, Alternation is specified with |. In regular expression if Σ is an alphabet, then Σ+ denotes the set of all finite strings of symbols in Σ. Any subset of Σ+ is a language over Σ.Example of regular expression is

{^ (yes|YES|Yes)$}

This matches exactly “yes”, “Yes”, or “YES”. Regular expressions have been used in a variety of practical applications to specify regular languages in a perspicuous way. The problem of deciding whether a given string belongs to the language denoted by a particular regular expression can be implemented efficiently using finite automata. A regular expression is used for pattern matching that matches one or more string of characters. Regular expression is generated for every string in the rule set and nondeterministic / deterministic finite automata are generated that examines the one byte input at a time.

3.2 Nondeterministic Finite Automata

An NFA is represented formally by a 5-tuple, (Q, Σ, Δ, q0, F), consisting of a finite set of states Q ,a finite set of input symbols Σ, a transition relation $\Delta : Q \times \Sigma \rightarrow P(Q)$,an initial
(or start) state $q0 \in Q$,a set of states F distinguished as accepting (or final) states $F \subseteq Q$. Here, P(Q) denotes the power set of Q. Let w = a1a2 ... an be a word over the alphabet Σ. The automaton M accepts the word w if a sequence of states, r0,r1, ..., rn, exists in Q with the following conditions:$r0 = q0, ri+1 \in \Delta(ri, ai+1)$, for i = 0, ..., n−1,$rn \in F$.

3.3 Deterministic Finite Automata

A deterministic finite automata is similar to the Non Deterministic finite automata the only difference is in transition function ($\delta : Q \times \Sigma \rightarrow Q$) where Q is the only one state instead of power set of Q. Let w = a1a2 ... an be a
string over the alphabet Σ. The automata M accepts the string w if a sequence of states, r0,r1, ..., rn, exists in Q with the following conditions:$r0 = q0, ri+1 = \delta(ri, ai+1)$(for i = 0,..., n−1)and $rn \in F$.

DFA differ substantially from NFA in the way they process data.An essential property of DFA is that at any given point od time only one state is active ie for each input symbol a single state needs to be processed .In contrast , an NFA can have multiple active states at the same time which all need to be processed when the next input symbol is read.

3.4 State minimization Algorithm

Algorithm for Minimizing Number of States in DFA:
1. Remove states that are not reachable.
2. Group all non-final states together as indistinguishable
3. Group all final states together as indistinguishable
4. Repeat till no more states are distinguishable
(a) Apply symbol to a group and split group if states are Distinguishable

A state $s1 \in Q$ is said to be inaccessible or unreachable if there exists no string
w in Σ* such that $\delta(s,w)=s1(s1 \notin (s2|w \in \Sigma^*, \delta(s1,w)=s2\})$

Two states s1 and s2 are indistinguishable if for all w ∈Σ*
δ (s2,w) ∈ F) ⇒ δ*(s1,w) ∈ F
δ* (s1,w) ∉ F) ⇒ (s2,w) ∉ F

4.Implementation and Results

The state machine bubble diagram in the below fig 1 shows
the operation of a four-state machine that reacts to a single input and matches all the patterns having at least one a.

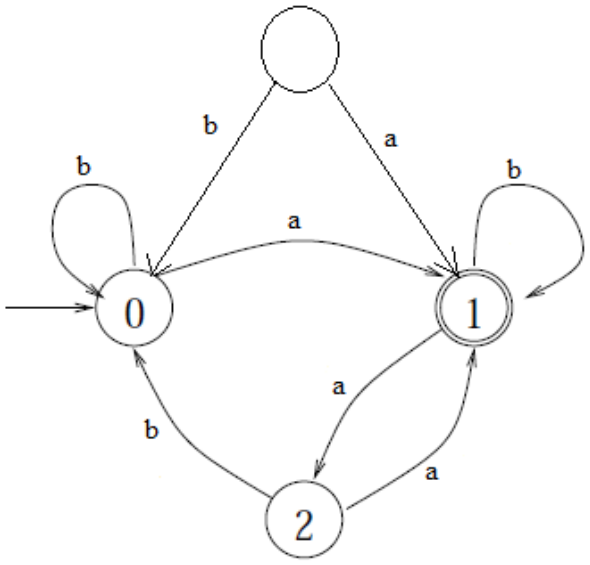

Fig 1 DFA1 for the pattern having at least one a.

The set of literals are Σ=(a,b),Q=(0,1,2,3),q0=(0),F=(1). As you can see in fig 1 DFA1 have only one unreachable state ie state1 since there is no such state from where state3 can be reached and state (0,2) is equivalent or indistinguishable to state(0) so state(0) can be merge and can call it as state(0,2). As you can see from the schematic fig 2 of DFA1, XST has used two flip flops for implementing the state machine and from Table 1 we can see that this schematic needs 3 macrocels,4 product term,4 function block,2 registers and 5 pins.

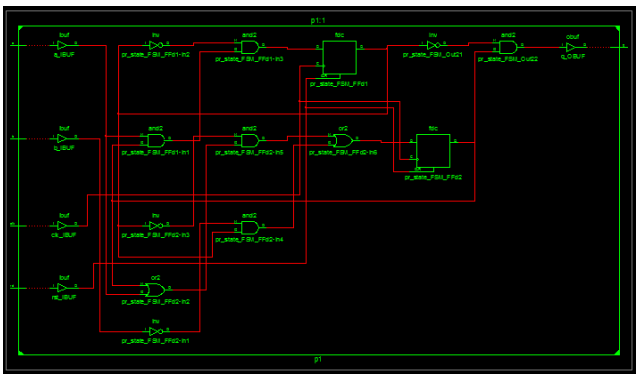

Fig2 technology schematic1 of the DFA1 .

After applying the state minimization algorithm to the fig 1
the minimized DFA is DFA2(fig3).

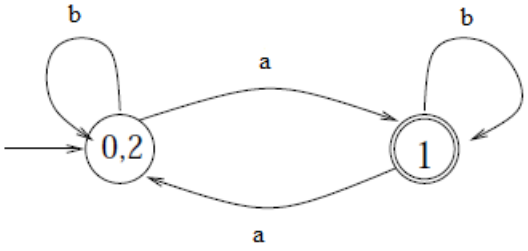

Fig 3:DFA2,minimized DFA of fig1.

In fig 3 The set of literals are Σ=(a,b),Q=((0,2),1),q0=(0,2),F=(1). As you can see that state 3 has been removed because this was the only unreachable state and indistinguishable states are also combined and form one state (0,2) instead of two different state 0 and 2. The technology schematic is shown in fig 4.

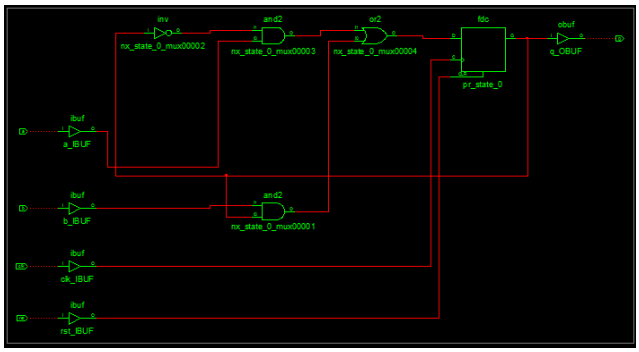

Fig4 technology schematic2 of the DFA2.

As you can see from the schematic of DFA, XST has used one flip flop for implementing the state machine whereas the schematic of DFA1 require 2 flip flops. And the other resource summary of the technology schematic as we cansee in table 2 the macrocels required is only 1 ,product term is 2,the functional blocks are 3,registers used is 1 and the total number of pins required are 5 .

	DFA		Minimized DFA	
	Used/Total	%	Used/Total	%
Macrocels	3/36	8%	1/36	3%
Product Term	4/180	2%	2/180	1%
Function Block	4/108	4%	3/108	3%
Registers	2/36	6%	1/36	3%
Pins	5/34	15%	5/34	15%

Table1:Resource summary of DFA and Minimized DFA.

Comparing the results of resources of DFA1 and DFA2 we can see that the macrocels required is 75% less for minimized DFA, product term used is 50 % less, Function block required are 5%less ,registers used are 50 % less and pins used are is same as the original DFA so overall we can conclude that the total area is very less as compare to the original DFA.

4.1 Simulation Result

Fig 5 presents the simulation result for pattern "bba" and fig 6 is the simulated waveform for pattern"bbabb".The graph can be easily interpreted.The first ccolumn shows the signal names it also shows the mode(direction)of the signals(the inward arrow shows the input and the outward arrow shows the output).the second column has the value of each signal in the position where the vertical cursor is placed(in fig 5 the cursor is at 710 ns and in this position the value of the output signal is 1 and all other are 0 similarly if fig 6 the cursor is at position 700 ns and in this position the value of the input signal b and output signal is 1 and all other are 0.The third column shows the simulation proper. The simulation result is same for the DFA1 and DFA2 .

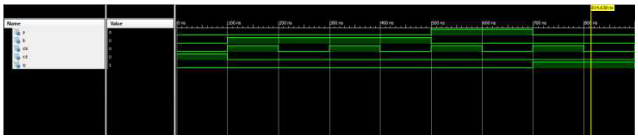

Fig5:Simulated Waveform of pattern "bba"

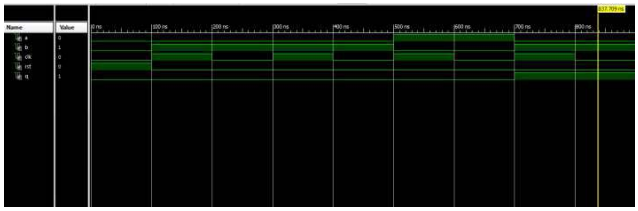

Fig6:Simulated Waveform of pattern "bbabb"

4.2 Performance Analysis

Table 2 presents the performance summary with the comparison of Deterministic Finite Automata and improvement of the DFA with state minimization. We can see the difference of different clock period difference of DFA and minimized DFA is 6.5 ns,the clock frequency is 46.6 Mz high for the minimized DFA,6.5 ns more clock to setup time required for minimized DFA,to simulate the apttren "bba " DFA1 it require 4.39 ns and minimized DFA require only 3.10 similarly memory usage for the same pattern is also less in case of minimized DFA.

-Analysis-	DFA	Minimized DFA
Min clock period	15.50 ns	9 ns
Max clock frequency	64.156Mz	111.11Mz
Clock to setup	15.50 ns	9 ns
CPU time to completion(for "bba")	4.39 sec	3.10 sec
Memory Usage(for' bba')	17236 KB	17186 KB

Table 2:Performance summary

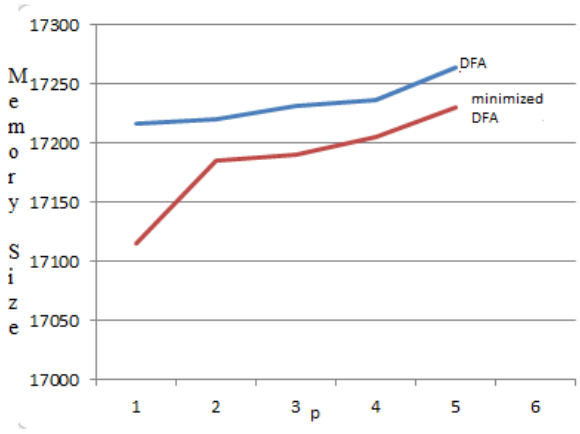

Fig 7: memory size for the different value of p(simulation result)

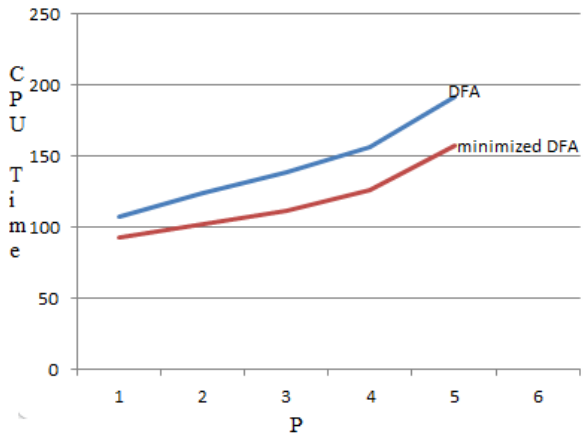

Fig 8: CPU time(Y-axis in ms) for the different value of p (Xaxis): simulation result

Fig 7 and fig 8 shows the comparison graph for the DFA with the minimized DFA Fig 7 shows the memory size in KB for the different value of p (p is the pattern size) simulated is Xilinx 12.4 so you can compare memory usage in KB . The smallest pattern (p=1)requires 17216KB
memory usage in case of DFA and 17122 KB memory usage in case of minimized DFA. similarly fig 8 illustrate the CPU simulation time in ms for the different size of pattern p.The smallest pattern (p=1)requires 108 ms in case of DFA and 93 ms in case of minimized DFA.

5. Conclusion

In this paper minimized DFA is implemented for pattern matching which results the reduced area, better performance, less number of resources. In general DFA may require up to 2n states but after minimization the equivalent DFA require n states. Number of resources also reduced up to 40%.So the implementation in hardware with state minimization is very apparent. The minimized DFA is very much efficient than the original DFA. The future scope of the work is to apply some technique to process multiple literals of the pattern parallel at the same time.

Acknowledgments

I would like to express my gratitude to my supervisor Dr. Nilay Khare and Co-supervisor Akhtar rasool, for their guidance, encouragement and support during my post graduate study.

References

[1] A. Babu Karuppiah, Dr. S Rajaram "Deterministic Finite Automata for Pattern Matching in FPGA for intrusion Detection" in International Conference on Computer and Electrical Technoogy-ICCCET 2011,18th & 19th March,2011."
[2] Jan Kastil, Jan Korenek Hardware Accelerated Pattern Matching Based onDeterministic Finite Automata with Perfect Hashing,IEEE 2010,p-149-152.
[3] Kai Wang, Yaxuan Q, Yibo Xue, Jun L Reorganized and Compact DFA for Efficient Regular Expression Matching,IEEE communication society
[4] Hiroki Nakahara,Tsutomu Sasao and Munehiro matsuura "A Regular Expression Matching Using Non-Deterministic Finite Automata" in IEEE 2010.
[5]Ivano Bonesana,Marco Paolieri,Marco D. Santambrogio "An adaptable FPGA based system for regular expression Matching" in IEEE 2008.
[6] Mother Aldwairi,Thomas Conte,Paul Franzon "Configurable string Matching Hardware for Speeding up Intrusion detection" inACM SIGARCH Computer Architecture News in,Vol. 33,No. 1, March 2005.
[7]Ashok kumar Tummala and Parimal Patel"Distributed IDS using Reconfigurable Hardware" in IEEE 2007.
[8]Hoang Le and Viktor K. Prasanna Ming Hsieh Department of Electrical Engineering University of Southern California Los Angeles, CA 90089, USA A Memory-Efficient and Modular Approach for String Matching on FPGAs ,2010
[9]M. Dhanapriya,C. Vasanthanayaki "Hardware Based Pattern Matching Technique for Packet Inspection of High Speed Network" International Conference on "Control,Automation,Communication and energy Consevation- 2009 4th -6th june 2009.
[10] Ioannis Sourdis, Dionisios N.Pnevmatikatos, and Stamatis Vassiladis," Scalable Multigigabit Pattern Matching for Packet Inspection," in Proc. IEEE Symp. Field program. Custom Comput. Feb. 2008.
[11] B. L. Hutchings and R. Franklin and D. Carver "Scalable hardware implementation usonf Finite Automata"Department of Electrical and Computer Engineering.
[12] J. Hasan, S. Cadambi, V. Jakkula and S. Chakradhar, "Chisel: A Storage-efficient, Collision-free Hash-based Network Processing Architecture," 33rd International Symposium on Computer Architecture,p.203-215.
[13]Reetinder Sidhu,Vikot K. Prasanna "Fast Regular Expression Matching using FPGAs"9th Annual Symposium IEEE2001.

Aakanksha Pandey is a research scholar student of M. Tech., Maulana Azad National Institute of Technology, Bhopal, Madhya Pradesh, India.

Dr Nilay Khare is a Head of the department of computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal, Madhya Pradesh, India.

Akhtar Rasool is a Faculty of computer Science and Engineering Department, Maulana Azad National Institute of Technology, Bhopal, Madhya Pradesh, India.