Reduction of FPGA Resources for Regular Expression Matching by Relation Similarity

Vlastimil Košař
Brno University of Technology
Bozetechova 2, 612 66,
Brno, Czech Republic
Email: ikosar@fit.vutbr.cz

Jan Kořenek
Brno University of Technology
Bozetechova 2, 612 66,
Brno, Czech Republic
Email: korenek@fit.vutbr.cz

***Abstract*—Intrusion Detection Systems have to match large sets of regular expressions to detect malicious traffic on multi-gigabit networks. Many algorithms and architectures have been proposed to accelerate pattern matching, but formal methods for reduction of Nondeterministic finite automata have not been used yet. We propose to use reduction of automata by similarity to match larger set of regular expressions in FPGA. Proposed reduction is able to decrease the number of states by more than 32% and the amount of transitions by more than 31%. The amount of look-up tables is reduced by more than 15% and the amount of flip-flops by more than 34%.**

I. Introduction

The requirements on securing networks from attacks, malicious traffic and spreading of viruses and trojan horses rises in context of steady growth of networks. Therefore importance of Intrusion Detection Systems (IDS) for network security rises. The most important part of an IDS is regular expression matching. The regular expression matching is a time critical operation for processors, therefore acceleration of regular expression matching is studied.

First algorithm for mapping regular expressions to FPGA was presented by Sindhu and Praasana in [1]. Clark et al. introduced shared character decoder in [2]. Lin et al. proposed to share prefixes, infixes and suffixes in [3]. Shared character classes were introduced together with special implementation blocks for some PCRE constructions by Sourdis et al. in [4]. These approaches introduced reduction of utilized FPGA logic by better architecture and algorithms, but formal NFA reduction has not been used.

In this paper we propose to use a relation of similarity for state reduction of NFA and mapping to FPGA by Clark approach. Reductions of state graphs are commonly studied and used in formal verification for state reduction [5], [6]. From existing reductions we have selected reduction by similarity, which was introduced by Milner [5]. NFA reductions such as [7] could also be used. Significant reduction of FPGA resources utilization was achieved by NFA reduction by similarity.

II. Simulation and Similarity

NFA created from regular expression set by Thompson construction [8] is not usually minimal. Redundancy is given by regular expressions and by relations among regular expressions in the rule set. Due to this redundancy we focus on NFA reduction. Minimization of NFA cannot be used, because it is generally PSPACE-complete [9]. In contrary reduction algorithms usually reduce state graphs in polynomial time. From existing reductions we selected reduction by similarity, which is suitable for state reduction of NFA for Snort rule set.

The similarity and simulation relation was first introduced by Milner in [5] as a means to compare programs. Definitions of simulation and similarity on state labeled graphs presented by Henzinger et al. in [6] are used (Modified for an edge labeled graph). A NFA is special case of an edge labeled graph, if we omit notation of final states.

First we define an edge labeled graph. Next we define simulation and similarity relation on this graph.

Definition 1 (Labeled Graph)**.** *An edge labeled graph* $G = (V, E, A, \langle\langle . \rangle\rangle)$ *consist of a set* V *of vertices, a set* $E \subseteq V^2$ *of edges, a set* A *of labels and a function* $\langle\langle . \rangle\rangle : E \to A$ *that maps each edge* e *to a label* $a \in A$*. We use* $post(v) = \{u | (v, u) \in E\}$ *for the successor set of the vertex* v*.*

Definition 2 (Simulation)**.** *Simulation on edge labeled graph is a binary relation* $\leq \subseteq V^2$ *on the vertex set if* $u \leq v$ *implies:*

1) *for all vertices* $\acute{u} \in post(u)$*, there is a vertex* $\acute{v} \in post(v)$ *such that* $\acute{u} \leq \acute{v}$ *and* $\langle\langle (u, \acute{u}) \rangle\rangle = \langle\langle (v, \acute{v}) \rangle\rangle$*.*

Definition 3 (Similarity)**.** *A binary relation* $\approx^S \subseteq V^2$ *on the vertex set is similarity if* $u \approx^S v$ *implies* $(u \leq v) \wedge (v \leq u)$*: The similarity relation* $\approx^S$ *is an equivalence relation and is a symmetric subset of simulation relation.*

We have used an algorithm for simulation reduction presented in [6] with time complexity $O(mn)$ where m is count of edges and n is count of vertices and space complexity $O(n^2)$. The algorithm consists of following four consecutive steps:

1) Default simulation relation, where each state simulate each other
2) While simulation relation changes do:
   a) For all possible combinations of states u and v do:
      i) Check if state v simulate state u and modify simulation relation accordingly
3) Create similarity relation

Rule set	Clark et al.		With Reduction		Reduced by	
	LUT [-]	FF [-]	LUT [-]	FF [-]	LUT [%]	FF [%]
L7 decoder	1523	821	1285	549	15.6	33.1
Snort (1)	4592	3955	3621	2788	21.1	29.5
Snort (2)	832	201	757	108	9.0	46.3
Snort (3)	851	236	719	93	15.5	60.6
Snort (4)	815	186	700	72	14.1	61.2
Snort (5)	1196	339	1038	172	13.2	49.3
Snort (6)	691	53	656	15	5.1	71.7
Snort (7)	677	99	638	58	5.8	41.4
Snort (1-7)	**9654**	**5069**	**8129**	**3306**	**15.8**	**34.8**

TABLE I
ESTIMATION OF UTILIZATION OF FPGA RESOURCES FOR VIRTEX-5

Rule set	Original ϵ-NFA		After reduction		Reduced by	
	States [-]	Trans. [-]	States [-]	Trans. [-]	States [%]	Trans. [%]
L7 decoder	791	949	547	704	30.9	25.8
Snort (1)	3800	4409	2783	3320	26.8	24.7
Snort (2)	186	226	106	127	43.0	43.8
Snort (3)	218	314	89	115	51.2	63.4
Snort (4)	182	227	69	77	62.1	66.1
Snort (5)	322	368	168	196	47.8	46.7
Snort (6)	49	72	13	16	73.5	77.8
Snort (7)	95	117	56	67	41.1	42.7
Snort (1-7)	**4852**	**5733**	**3284**	**3918**	**32.3**	**31.7**

TABLE II
STATE AND TRANSITION COUNT FOR BOTH ORIGINAL ϵ-FREE NFA (ϵ-NFA) AND REDUCED NFA

4) Join similar states

Similarity relation can be used for NFA state reduction, if we join states which are similar. Feature of this reduction is that all final states are joined into one final state. Therefore the match of a pattern can be detected, but concrete pattern cannot be determined. Better reduction results can be obtained, if the reduction will be used multiple times in forward and backward variant. [7]

III. EXPERIMENTAL RESULTS

Reduction of NFA by similarity was evaluated on several rule sets. We have used selected rule sets from the Snort and a subset of regular expressions from the L7-decoder. The Netbench framework [10] was used for implementation of the reduction and for estimation of FPGA resource utilization. We have used the Xilinx Virtex 5 FPGA architecture for the estimation of FPGA resource usage.

The measured reduction results are shown in the table II. The table contains the size of original ϵ-free NFA, the size of reduced NFA and the amount of states and transitions, which was removed by NFA reduction by similarity. The average reduction of states is 32.3% and the average reduction of transitions is 31.7% for the Snort rule sets.

We have selected the Clark et al. methodology of mapping regular expressions to FPGA together with the sharing of character classes for the evaluation. The pattern matching unit is designed to accept one character per clock cycle. The measured reduction results are shown in the table I. The table contains the utilization of FPGA resources for the implementation of pattern matching unit without NFA reduction by similarity, the utilization of FPGA resources if NFA reduction by similarity is used and finally how many look-up tables (LUT) and flip-flops (FF) was removed by the NFA reduction by similarity. The average reduction of LUTs is 15.8% and the average reduction of FFs is 34.8% for the Snort rule sets.

IV. CONCLUSION AND FUTURE WORK

In this paper we propose to use NFA reduction by similarity to achieve efficient utilization of FPGA resources. This reduction is independent on concrete NFA mapping to the FPGA. The NFA reduction by similarity is able to reduce significant amount of FPGA resources.

In future work, we want to modify the algorithm to preserve information about the matched rule. Another reduction algorithms and architectures can be used together with simulation reduction. For example special blocks for PCRE constructions introduced in [4] reduce constructions that are irreducible by reduction by similarity. We want to evaluate the reduction by combined approaches in future work.

ACKNOWLEDGMENT

This research has been partially supported by the Research Plan No. MSM, 0021630528 – Security-Oriented Research in Information Technology and the grant BUT FIT-S-10-1.

REFERENCES

[1] R. Sidhu and V. K. Prasanna, "Fast Regular Expression Matching using FPGAs," in *Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2001)*, April 2001, pp. 227–238.

[2] C. Clark and D. Schimmel, "Efficient Reconfigurable Logic Circuits for Matching Complex Network Intrusion Detection Patterns," in *Field Programmable Logic and Application, 13th International Conference*, Lisbon, Portugal, 2003, pp. 956–959.

[3] C.-H. Lin, C.-T. Huang, C.-P. Jiang, and S.-C. Chang, "Optimization of regular expression pattern matching circuits on FPGA," in *Proc. of Conference on Design, Automation and Test in Europe (DATE 06)*, Munich, Germany, 2006, pp. 12–17.

[4] I. Sourdis, J. Bispo, J. Cardoso, and S. Vassiliadis, "Regular expression matching in reconfigurable hardware," *Journal of Signal Processing Systems*, vol. 51, pp. 99–121, 2008.

[5] R. Milner, "An algebraic de nition of simulation between programs," in *2nd IJCAI*. British Computer Society, 1971, pp. 481–489.

[6] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke, "Computing simulations on finite and infinite graphs," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, ser. FOCS '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 453–.

[7] L. Iliea and S. Yu, "Algorithms for computing small nfas," in *Mathematical Foundations of Computer Science 2002*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2002, vol. 2420, pp. 328–340.

[8] K. Thompson, "Programming techniques: Regular expression search algorithm," *Commun. ACM*, vol. 11, pp. 419–422, June 1968.

[9] T. Jiang and B. Ravikumar, "Minimal nfa problems are hard," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1991, vol. 510, pp. 629–640.

[10] Netbench, "Project WWW Page," `http://merlin.fit.vutbr.cz/ant/netbench/index.html/`, 2011.