International Journal of Smart Home
Vol.8, No.1 (2014), pp.45-52
http://dx.doi.org/10.14257/ijsh.2014.8.1.06

An Innovative Approach for Regular Expression Matching Based on NoC Architecture

Cui Linhai

Harbin University of Science and Technology, Harbin, China
cuilinhai@hrbust.edu.cn

Abstract

An new regular expression (regex) matching method based on Network-on-Chip(NoC) architecture is proposed in this paper. The idea is to combine a new kind of regex matching engine implemented in hardware with NoC architecture to get a high matching rate. The Regex matching was performed by partitioning the regex into several parts to make the finite state machine (FSM) simpler. Each part of regex can be matched by an engine cell core, and each core communicates with other cores by routers on a NOC topology. This method is suitable for different rule libraries in Deep packet inspection (DPI) and is easy to change the rule library of regex stored in memory. The engine was designed and implemented based on field programming gate array (FPGA) as a prototype and a model to implement the architecture in Application-Specific Integrated Circuits (ASIC) is also discussed in this paper. The experimental results show that this method can make regex matching much faster than traditional methods.

Keywords: *Network-on-chip; Regular expression matching; Finite state machine; Deep packet inspection; Reconfigurable design; Throughput*

1. Introduction

Network security is becoming more important than ever before. With the continuous development of network technology in recent years, how to protect integrity, confidentiality, availability and controllability of network information becomes an important research issue of network security.

Intrusion detection system (IDS) is a commonly used way for detecting network attack attempts, behavior and results of the attack to ensure the integrity, confidentiality and availability by monitoring the operation of a network system and the tightly coupled data transmitted over the entire network according to a set of preset values. It extracts statistical characteristics of packets flow, and makes intelligent analysis and comparison to the packets it concerned to detect which packet flow is illegal according to the preset rules. And these preset rules are usually described in reguex.

Regex refers to a single string used to describe a series of strings in computer science. In many text editors or other tools, regex is often used to retrieve and / or replace text which matches a particular pattern. Many programming languages support the use of regex for string manipulation.

With the development of DPI, traditional pattern sets used to filter packets have been gradually replaced by rule sets of regex. A regex is often represented by a finite state machine called deterministic finite automata (DFA) or non-deterministic finite automata (NFA) for most string matching. Although the matching performance of regex is better than that of traditional string matching, the memory space required for converting regex into finite automata is huge. In general-purpose processor systems, the algorithm's efficiency will be seriously affected because the access time of memory is much longer than matching time. So to perform regex matching in hardware has become a new hotspot.

Network on-chip is a structure which can provide data path for several different cores or processors to make the entire system efficient in parallel. And the backtracking problem of NFA generated according to regex can be also solved by using the parallel operation features of multi-cores.

An FPGA is an integrated circuit or chip which can be configured or programmed by customers or designers after manufacturing. It can be used to implement any logical function that ASIC could perform. The configuration of FPGA is specified by using hardware description language (HDL). It contains programmable logic components called "logic blocks" and a hierarchy of reconfigurable interconnects that allow the blocks to be wired together. Logic blocks can be configured to perform complex combinational functions. In most FPGAs, the logic blocks also include memory elements which may be simple flip-flops or blocks of memory.

For saving area, the regex to be matched was stored in cache or memory, so does the op-code that controls the engines. Experiments show that each engine cell only cost 48 logic cells in our verification design on FPGA.

2. Background

Regex gains widespread application in DPI together with other network applications. Since it is difficult for a system of traditional Von Neumann architecture to match a regex at a desired speed, contemporary computer network transmission rate, the way to match regex by using integrated circuit or hardware was employed.

2.1. Matching Algorithm based on DFA

Implementing regex matching in hardware has been widely studied before. DFA is one of the main methods for implementing regex. Two-dimensional table storage structure was used to implement regex for majority of matching methods based on DFA. In DFA, the next state to which the current state should jump can be easily found by accessing memory only once. However, the storage cost of DFA depends on the number of status and the number of edges.

The number of status of DFA is related closely with the structure of regex. In DPI, the rules of regex are more complex. When converting regex into DFA, the number of status is over tens of thousands.

In the paper of Fang Yu [1], rewriting rules was presented for two common types of regex which may lead to the number of status of DFA excessive. At the same time, he also proposed a way which partitioning regex into groups and compiled them into several matching engines, which can significantly reduce the DFA storage space required.

Kumar S [2] proposed an algorithm called D^2FA which replace a group of common edge of some status of the DFA with a single default edge. And their experimental results show that the memory space reduced by more than 95% in such a manner.

Brodie B [3] proposed a way processing multiple characters within each state rather than handling one character to reduce the expansion of the state transition table. The ASIC chip was taped out successfully on support of a fund, and the throughput of which can achieve 4Gb \ s. However, the use of the memory cell reached a very high degree.

2.2. Matching Algorithm based on NFA

The traditional NFA is not very efficient in matching compared with DFA because of the unconditional transfer and backtracking. For DFA, only one certain state transition can be produced for each input. But for NFA, each input may generate many state transitions and empty transitions or unconditional transfer. But NFA still attract some people's attention. In fact, Sidhu R proposed five kinds of structures for implementing regex on FPGA in 2001. Almost all kinds of regex can be expressed in these 5 structures via a certain algorithm. However, the disadvantage of this method is that the state itself stored by a logic circuit, rather than the memory on the circuit. So, the amount of hardware description language code is huge.

Floyd and Ullman [4] discussed the implementation of NFAs using hardware. Since regex can be represented in FSM. Sidhu and Prasanna [5] proposed an algorithm to implement regex matching in regular expression NFA (RE-NFA) architecture on FPGA. Yang and Prasanna [6] improved this architecture by processing 2 characters per clock cycle, and they got a result which has a concurrent throughput of 14.4Gbps for 760 regex matching. It is almost the fastest one in our survey result by RE-NFA. For the L7 with 70 protocol filters, the system throughput is less than 10Mbps, and more than 90% of the CPU time is spent in regular expression matching. These are all explained by Fang Yu, et al[7] , Brodie[8] , which transmit the regular expression into a FSM by a state table have 256 columns for ASCII character set. They analyzed and implemented their design using a plausible ASIC, and achieved a 16Gbps throughput.

2.3. Network on Chip

Research on NoC began in 1999 [9]. It was originally part of the design methodology exploring communication in system-on-a-chip (SoC) on system-level. NoC is a very broad concept, from hardware communication structure, middleware, operating systems, communication services to design methods and tools [10]. From the perspective of system architecture, NoC study focus on its topology, communication protocols, signal integrity and low power consumption.

With the development of deep submicron, nanometer CMOS technology and SoC technology, chip multiprocessors (CMP) technology began to occur. Bus architecture widely used in traditional von Neumann architecture brings many problems such as bandwidth limitations, signal integrity, signal delay and global synchronization in SoC design, which affects the performance of CMP.

Compared with the single bus structure, NoC technology has the characteristics of reconfiguration in addition to connecting more IP components. SoC is a reconfigurable design which can save costs, improve design reliability and shorten time to market.

For the NoC architecture, there have been various topologies include mesh, torus, ring, butterfly and irregular interconnection networks. [11, 12] Compared to other topologies, some researchers have suggested that 2-D mesh architecture for NoC will be efficient, ease of implementation in terms of latency, power consumption[13]. However, as NoC architectures are based on packet-switched networks, so that a suitable and efficient principle for design of routers for NoC is important [14].

3. Design of the Matching Engine

As a regex matching engine, the traditional and its expansive implementation is nothing more than a NFA based on character and logic element or a lookup table based on memory. However, the former consume large number of logical elements in same structure because each character requires a particular matching circuit and there is a long chain structure circuit for each regex. And for the latter, although fewer logical elements were used, extensive memory space was used. Although the lookup table model is more suitable for FPGA, due to the demand for storage performance and space, it is difficult to use on FPGA.

3.1. Structure of the Matching Engine

A specific matching engine was designed to handle regex matching. The structure based on memory cell / lookup table is an innovation of this paper. The engine is a specific matching engine dedicated to matching the regex. And it is also a general-purpose engine used for replaceable regex library. As a basic unit of regex matching structure, the engine includes two parts, a controller and a memory. Engine unit's main role is to extract characters which need to be matched from memory every cycle. And then the extracted characters will be handed over to the controller to determine its next state. As an engine cell, a basic circuit for character matching is designed by using FPGA. Figure 1 shows the top level structure of the engine.

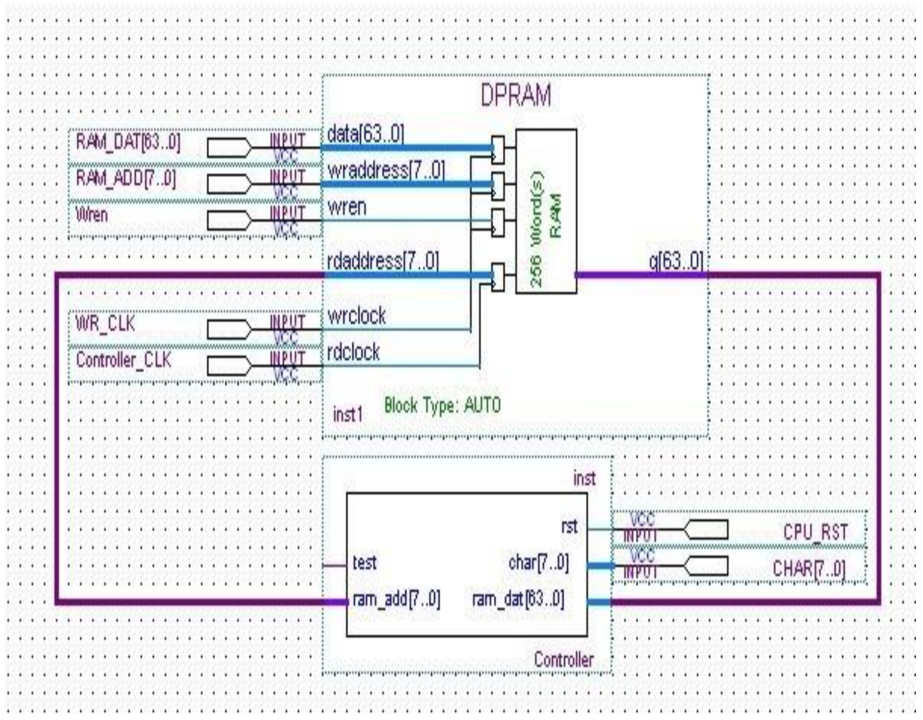


Figure 1. Top Level Structure of the Engine

The engine has a stable timing performance and has the ability to handle almost all meta-characters. Combined with the NoC architecture, this engine cell could process almost all regex Meta-character.

The controller has several modes of dealing with different Meta-characters. It depends on the 32 bit instruction set that described in Table 1. The solution for general meta-characters was described in Table 1. A32 bit instruction model was defined for such a general data engine, of which the first four bits scripts the op-code, 5-12 is the first character, 13 to 20 is the second character, and 21 to 32 bits indicates the suffix.

Table 1. Controller Modes and Instruction Set

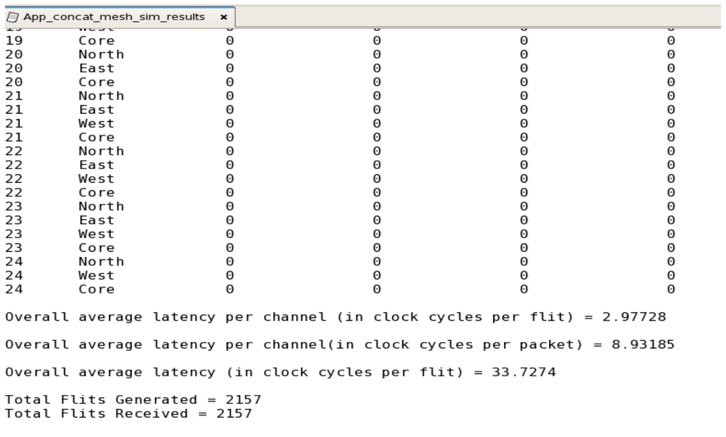
Meta-character example and description	Op-code (i0~i3)	Mode	Postfix (i20~i31)
ab: only ASCII-characters	0001	Controller match ‘a’ with data stream character input at the this clock, and match ‘b’ at the next clock	If ’ab’ was 2 character or 1 of the pattern (may be only one character here), here store the place in which pattern
[a-z]:a class range of character.	0010	Controller judge if the ASCII value of data stream character input between ‘a’ and ‘z’	Here store the place in which pattern, and if it was a apart-NFA status over.
[^ab]:any character but ‘ab’	0100	Controller match ‘a’ with data stream character input at the this clock, and match ‘b’ at the next clock	Data that let controller make the result negate and the place in which pattern, and if it was an apart-NFA status over.
(ab)+: match ‘ab’ once or more and data (a)+ is acceptable.	1000	Controller match ‘a’ with data stream character input at the this clock, and match ‘b’ at the next clock. pointer=pointer-n	Data that let controller make the next part of NOC that store the rest of regular expression available.
a*:match 0 times or more.	1001	Controller match ‘a’ with data stream character	Data that let controller make the next part of NOC that store the rest of regular expression available.

As can be seen from Table 1, we have adopted a split treatment for some of the meta-characters which may cause backtracking and interlock in the regex, *i.e.*, breaking regex from the point where these meta-characters occur. And then identification and communication are performed through the suffix. While for the character group, it is entirely carried out as a separate character matches. Since the structure of the engine may easily lead to interlock for backtracking in NFA, so we want to break the regex to solve this problem. In fact, due to multiple engine are interconnected with a single bus, timing problems are often caused due to bus control, and the data stream and the engine cannot be matched synchronously. So the network on-chip architecture was adopted to develop the matching engines.

3.2. Performance of Matching Engine

It is precisely because a structure was needed for implementing the engines we have designed and for algorithm to breaking regex, and network on-chip architecture is just in the position, so, simulations were performed for each of the proposed traditional network on-chip architecture on the NIRGAM platform, and also for the possible future expansion of the architecture to see which is better for regex matching. The resources of data flows are from DARPA2000 download at MIT Lincoln Laboratory. Some of which are suitable for aggressive behavior identified by Snort.

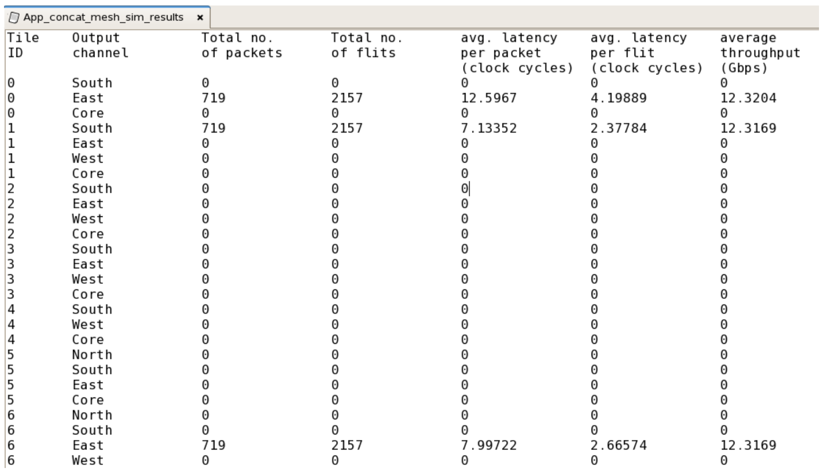
For the mesh structure, simulations were figured out by using Nirgam, System C and HDL model, as shown in Figure 2.



```
19      Core        0         0         0         0
20      North       0         0         0         0
20      East        0         0         0         0
20      Core        0         0         0         0
21      North       0         0         0         0
21      East        0         0         0         0
21      West        0         0         0         0
21      Core        0         0         0         0
22      North       0         0         0         0
22      East        0         0         0         0
22      West        0         0         0         0
22      Core        0         0         0         0
23      North       0         0         0         0
23      East        0         0         0         0
23      West        0         0         0         0
23      Core        0         0         0         0
24      North       0         0         0         0
24      West        0         0         0         0
24      Core        0         0         0         0

Overall average latency per channel (in clock cycles per flit) = 2.97728

Overall average latency per channel(in clock cycles per packet) = 8.93185

Overall average latency (in clock cycles per flit) = 33.7274

Total Flits Generated = 2157
Total Flits Received = 2157
```

Figure 2. Mesh Structure of the Simulation Result

As can be seen from Figure 2 , for the 4 × 6 mesh structure, the overall signal delay is approximately an average of 3 cycles for each stream, and is about 9 cycles for each package. While as can be seen in Figure 3, the simulation results for 2157 streams and 719 packets, the average system throughput is 12.32Gbps. and can see that in addition to the engine unit number 0 matches the additional feedback information , the required a long time, other data is stable is less than 3 cycles, each core is to ensure that almost all of the matching can be completed one cycle.



```
Tile    Output      Total no.   Total no.   avg. latency    avg. latency    average
ID      channel     of packets  of flits    per packet      per flit        throughput
                                            (clock cycles)  (clock cycles)  (Gbps)
0       South       0           0           0               0               0
0       East        719         2157        12.5967         4.19889         12.3204
0       Core        0           0           0               0               0
1       South       719         2157        7.13352         2.37784         12.3169
1       East        0           0           0               0               0
1       West        0           0           0               0               0
1       Core        0           0           0               0               0
2       South       0           0           0               0               0
2       East        0           0           0               0               0
2       West        0           0           0               0               0
2       Core        0           0           0               0               0
3       South       0           0           0               0               0
3       East        0           0           0               0               0
3       West        0           0           0               0               0
3       Core        0           0           0               0               0
4       South       0           0           0               0               0
4       West        0           0           0               0               0
4       Core        0           0           0               0               0
5       North       0           0           0               0               0
5       South       0           0           0               0               0
5       East        0           0           0               0               0
5       Core        0           0           0               0               0
6       North       0           0           0               0               0
6       South       0           0           0               0               0
6       East        719         2157        7.99722         2.66574         12.3169
6       West        0           0           0               0               0
```

Figure 3. Simulation Results for 2157 Streams and 719 Packets

4. Experimental Results

We test the functions of engine cell on DE2 (EP2C35F672C6) board which with Altera’s FPGA chip and low performance RAM on it. The engine cell worked for the test case with all kinds of meta-character in Table 1. When test data stream input, the result of device return is correct. In a NOC simulator (NIRGAM), we tried 2D-mesh, and it has a 12.88 Gbps throughput 13.3311 average cycles latency with 64 cores (only 38 cores in use). For our mesh-pro topology diagonal only one-way down to the next level have a 17.9447Gbps throughput but some test cases have the cycle latency from 7 to 38.7004 with 38 cores.We change the router algorithm with mesh-pro topology 2-way diagonal and grouping method of group level regular expression, for 754 rules in snort, throughput over 19.98Gbps throughput and 7.6-8 cycles latency with 48 cores in use.

5. Conclusions

This paper proposed a new NOC architecture for regular expression matching with a new improved topology mesh-pro and a general reusable design. Our method makes full use of the advantage of REM by NOC architecture. This method solves the communication problem between multiple cores for regular expression and improve the throughout of the system. The experiment result shows that this model can process 6 character or meta-character per clock, and the throughput is over 19.98Gbps for 754 REMs.

Acknowledgements

This work is supported by Science and Technology Research Funds of Educational Department of Heilongjiang Province under Grant No. 12531132.

References

[1] F. Yu, Z. F. Chen, Y. L. Diao, T. V. Lakshman and R. H. Katz, "Fast and memory efficient regular expression matching for deep packet inspection", Bhuyan LN, Dubois M, Eatherton W, eds. Proc. of the 2006 ACM/IEEE Symp. On Architecture for Networking and Communications Systems. New York: ACM, **(2006)**, pp. 93-102.

[2] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection", Rizzo L, Anderson TE, McKeown N, eds. Proc. of the ACM SIGCOMM 2006 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM, **(2006)**, pp. 339-350.

[3] B. Brodie, R. Cytron and D. Talor, "A scalable architecture for high-throughput regular-expression pattern matching", //ISCA, **(2006)**.

[4] R. W. Floyd and J. D. Ullman, "The Compilation of Regular Expressions into Integrated Circuits", vol. 29, no. 3, **(1982)** July, pp. 603-622.

[5] R. Sidhu and V. K. Prasanna, "Fast Regular Expression Matching Using FPGAs", IEEE Symposium on Field Programmable Custom Computing Machines, **(2001)** April.

[6] Y.-H. E. Yang, W. Jiang and V. K. Prasanna, "Compact architecture for high-throughput regular expression matching on FPGA", Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, **(2008)**, pp. 227-238.

[7] F. Yu, "Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection", UCB tech. report, EECS-2005-8.

[8] B. C. Brodie, D. E. Taylor and R. K. Cytron, "A Scalable Architecture For High Throughput Regular Expression Pattern Matching", 33rd International Symposium on Computer Architecture (ISCA'06), **(2006)**.

[9] K. Goossens, "A Ethereal Network on Chip: Concepts, Architectures, and Implementations", IEEE Design & Test of Computers, vol. 22, no. 5, **(2005)**.

[10] P. P. Pande, "Performance Evaluation and Design Trade-offs for Network-on chip Interconnect Architectures", IEEE Transctions on Computers, vol. 54, no. 8, **(2005)**, pp. 1025-1040.

[11] J. Dally and B. Towles, "Principles and Practices of Interconnection Networks Morgan Kaufmann", **(2004)**.

[12] P. Pratim Pande, C. Grecu, M. Jones, A. Ivanov and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", IEEE Transactions on Computers, vol. 54, no. 8, **(2005)**, pp. 1025-1040.

[13] A. Agarwal, C. Iskander, H. Multisystems and R. Shankar, "Survey of network on chip (NoC) architectures & contributions", Journal of Engineering, Computing and Architecture, **(2009)**.

[14] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage and E. Waterlander, "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip", IEEE Proc. on Computers and Digital Techniques, vol. 150, no. 5, **(2003)** September, pp. 294-302.