電気学会論文誌 D（産業応用部門誌）
IEEJ Transactions on Industry Applications
Vol.136 No.10 pp.692–697　DOI: 10.1541/ieejias.136.692

論　文

正規表現の先読みに対するマッチングハードウェアの改善

非会員　比嘉　秀斗*　正　員　山田　親稔*
非会員　宮城　　桂*　非会員　市川　周一**

Improvement in Look-ahead Matching Hardware for Regular Expression

Shuto Higa*, Non-member, Chikatoshi Yamada*, Member, Kei Miyagi*, Non-member, Shuichi Ichikawa**, Non-member

（2016年2月15日受付，2016年6月23日再受付）

IDS is used to detect virus patterns by packets in the network. The virus patterns are generally expressed as regular expression. The matching hardware for regular expressions has been extensively studied. However, the look-ahead and back referece approaches have not been studied very much. By using the matching method for the look-ahead pattern that has been already suggested, the matching result is added to memory. Therefore, the memory increases as look-ahead patterns increases. Moreover, the operating frequency becomes lower than that of conventional matching hardware. In this article, we propose an improved method of regular expression matching hardware for the look-ahead pattern. Using the proposed method, we show a decrese in memory and an improvement in the operating frequency.

キーワード：ネットワーク侵入検知システム，正規表現マッチング，先読み
Keywords: network intrusion detection system, regular expression matching, look-ahead

1. はじめに

急速なインターネットの普及に伴い，コンピュータウイルスや不正アクセス等によるハッカー攻撃が増大している。ハッカー攻撃の対策の一つとしてネットワーク侵入検知システム（Network Intrusion Detection System，NIDS）がある。NIDSとは，ネットワークを流れるパケットを取得して解析し，異常がないか監視する。そして，不正侵入や攻撃を検知したときには管理者に知らせる。NIDSでのウイルス検出はコンピュータウイルスを定義したパターンとネットワークパケットをストリングマッチングを行う。ストリングマッチングとは与えられたパターンを入力系列の文字列から検索する操作である。NIDSでは一般的にパターンを正規表現で記述している。

ネットワークの高速化に伴い，NIDSでは正規表現を高速にマッチングする必要があるが，ソフトウェアでは対応できなくなりつつある。そのためハードウェアによる正規表現マッチングの研究が盛んに行われている[(1)~(7)]。しかし，後方参照と先読みのための正規表現マッチングハードウェアに関する研究はあまり行われていない。

本稿では，既に提案されている正規表現の先読みに対するマッチングハードウェアの問題点を改善した手法を提案し評価する。提案手法では，先行研究で提案された先読みに対するハードウェアのメモリの使用量を減らす方法を提案する。章構成は，2章の原理でSnortとそのルールに用いられている正規表現について述べ，3章で先読みと先行研究について説明する。4章では先読みに対する提案手法の提案と評価について述べ，最後にまとめと今後の展開を示す。

2. 原　　理

〈2·1〉 Snort[(8)]　NIDSのソフトウェアによる代表的な実装としてSnortが知られている。Snortとは，IPネットワーク上でリアルタイムトラフィック分析及びパケットロギングを行うことが可能なオープンソースのNIDSである。これは，プロトコル解析，コンテンツ検索・マッチングを行うことができ，例えば，バッファオーバーフロー，ステルススキャン，CGIアタック等多くの攻撃を検出することができる。

Snortにおいてウイルスパターン（Snortルール）はPCRE（Perl Compatible Regular Expression）で記述されている。PCREとは，Perl互換の正規表現ライブラリのことである。

* 沖縄工業高等専門学校
〒905-2192　沖縄県名護市辺野古905
Okinawa National College of Technology
905, Henoko, Nago, Okinawa 905-2192, Japan
** 豊橋技術科学大学
〒441-8580　愛知県豊橋市天伯町雲雀ヶ丘1-1
Toyohashi University of Technology
1-1, Hibarigaoka, Tempaku-cho, Toyohashi, Aichi 441-8580, Japan

```
>ping 192.168.147.130

192.168.147.130 に ping を送信しています 32 バイトのデータ:
192.168.147.130 からの応答: バイト数 =32 時間 <1ms TTL=64
192.168.147.130 からの応答: バイト数 =32 時間 =1ms TTL=64
192.168.147.130 からの応答: バイト数 =32 時間 <1ms TTL=64
192.168.147.130 からの応答: バイト数 =32 時間 <1ms TTL=64

192.168.147.130 の ping 統計:
    パケット数: 送信 = 4、受信 = 4、損失 = 0 (0% の損失)、
ラウンド トリップの概算時間 (ミリ秒):
    最小 = 0ms、最大 = 1ms、平均 = 0ms
```

Fig. 1. Send ping.

```
[**] [1:1000000:0] test [**]
[Priority: 0]
01/10-00:16:19.820005 192.168.147.1 -> 192.168.147.130
ICMP TTL:128 TOS:0x0 ID:1825 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:61  ECHO

[**] [1:1000000:0] test [**]
[Priority: 0]
01/10-00:16:20.821914 192.168.147.1 -> 192.168.147.130
ICMP TTL:128 TOS:0x0 ID:1826 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:62  ECHO

[**] [1:1000000:0] test [**]
[Priority: 0]
01/10-00:16:21.826095 192.168.147.1 -> 192.168.147.130
ICMP TTL:128 TOS:0x0 ID:1827 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:63  ECHO

[**] [1:1000000:0] test [**]
[Priority: 0]
01/10-00:16:22.830006 192.168.147.1 -> 192.168.147.130
ICMP TTL:128 TOS:0x0 ID:1828 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:64  ECHO
```

Fig. 2. Snort alerts.

〈2･1･1〉 Snort の実行例　Snort の動作を確認するため Snort2.9.8.0 をインストールし検証した。侵入を検知する為に使用するルールは “alert icmp any any -> 192.168.147.130 any (msg:“test”; sid:1000000;)” とした。別の PC で Snort がインストールされている PC への ping 送信を Fig. 1 に，Snort のアラートを Fig. 2 に示す。実行した結果は，4 回の送信すべてに対してアラートを出力した。ルールの括弧 “( )” の中に正規表現を用いてパケットの中身を指定することができる。

〈2･2〉 正規表現　以下に Snort ルールで使用されている演算子を定義する [9][10]。

記号の有限集合を $\Sigma = \{a_1, a_2, \cdots, a_n\}$，$R_1$，$R_2$，$R_3$ をおのおの Σ 上の正規集合 R_1，R_2，R_3 を意味する正規表現とする。

〔定義 1〕 $R_1|R_2 = R_1 \cup R_2$

〔定義 2〕 $R_1* = \{\varepsilon\} \cup R_1^1 \cup R_1^2 \cup \cdots$

〔定義 3〕 $R_1? = R_1|\varepsilon$

〔定義 4〕 $R_1+ = R_1^1 \cup R_1^2 \cup \cdots$

〔定義 5〕 $R_1\{M\} = R_1^M$

〔定義 6〕 $R_1\{M, N\} = R_1^M \cup R_1^{M+1} \cup \cdots \cup R_1^N$

〔定義 7〕 $R_1\{M, \} = R_1^M \cup R_1^{M+1} \cup \cdots$

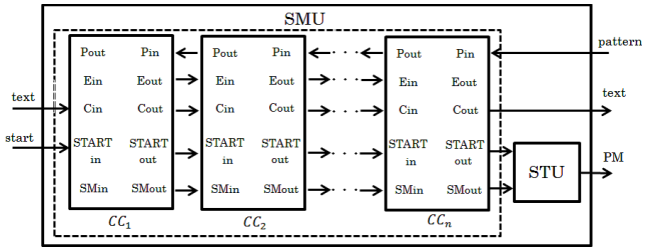


Fig. 3. Matching hardware for regular expression.

〔定義 8〕 $. = a_1|a_2|a_3|\cdots|a_n$

〔定義 9〕 $[a_1a_2a_3] = a_1|a_2|a_3$

〔定義 10〕 $[\hat{}a_i] = a_1|a_2|a_3|\cdots|a_{i-1}|a_{i+1}|\cdots|a_n$

〔定義 11〕 ‘^’ は文字列の先頭，‘$’ は文字列の末端もしくは改行の前にそれぞれマッチする。

〔定義 12〕 以下の演算子は次のように定義される。また，各演算子の大文字はその演算の否定を意味する。

- \n 改行文字
- \r リターン文字
- \w 英数字または ‘_’
- \s 空白文字
- \d 数字
- \b 単語境界（片方を \w，もう片方を \w 以外で挟まれた間の位置）

〔定義 13〕 ‘(’ と ‘)’ で囲まれた部分正規表現をグループと呼ぶ。パターンの左から q 番目のグループにマッチした文字列を ‘$q’ とした時，‘\q’ は文字列 ‘$q’ を意味する演算子である。

〔定義 14〕 “$R_1(?=R_2)R_3$” は R_1 と一致したすぐ後に R_2 と R_3 の両方と一致するような文字列の集合を意味する。ここで，R_2 と一致する文字列は R_3 と一致する文字列の部分文字列，またはその逆であることに注意されたい。

〔定義 15〕 “$R_1(?!R_2)R_3$” は R_1 と一致したすぐ後に R_2 には一致せず R_3 と一致するような文字列の集合を意味する。

〔定義 16〕 定義 2 から 7 の各演算子の後ろに ‘?’ を続けることによって，最低限の繰り返しでマッチングを実行する。

〔定義 17〕 各修飾子は以下のマッチングを行う。

- i 大文字小文字の違いを無視
- s ‘.’ が改行文字ともマッチ
- m ‘^’ と ‘$’ が各行の行頭・行末とマッチ
- U 定義 16 における繰り返し最大と最小の意味の入れ替え

ここで，定義 13 を後方参照，定義 14 を肯定先読み，定義 15 を否定先読みと呼ぶ。次の章で先読みについて詳しく説明する。

〈2･3〉 正規表現マッチングハードウェア [7]　本節では，既存の先読みに対応していない既存の正規表現マッチングハードウェアについて説明する。Fig. 3 に既存の正規表現マッチングハードウェアのアーキテクチャを示す。アー

キテクチャはシストリックアルゴリズムに基づくストリングマッチングユニット（SMU）と NFA に基づく状態遷移ユニット（STU）からなっている。SMU は n 個の比較セル（CC）を 1 次元配列上に接続した構造で，与えられた正規表現 R の遷移文字列（極大単純部分表現）の比較を行い，比較結果を STU に伝える。STU はその比較結果に基づき状態遷移を管理する。動作周波数は，186 [MHz] となっている。

3. 先読みに対する先行研究

〈3･1〉 先読みについて 先読みには肯定先読み “$R_1(?=R_2)R_3$” と否定先読み “$R_1(?!R_2)R_3$ の二種類ある。R_1 と R_2，R_3 は任意の正規表現である。Fig. 4 にテキスト “collarblue” による肯定先読みと否定先読みのマッチング例を示す。(a) は先読みを用いない正規表現で，(b) と (c) は肯定先読みと否定先読みである。先読みは，等価な基本演算子（‘.’ と ‘|’，‘∗’）のみを使った正規表現にも書き換えることができる。しかし，一般的に正規表現の変換は長いパターンを作ってしまう。そのため，先読みは，等価な正規表現に変換することなく短いパターンのままでハードウェアで実現できるのが望ましい。

〈3･2〉 先行研究[1] 先行研究では，パターンが “$R_1(?=R_2)R_3$” としたとき，まず先読みパターン R_2 に対するマッチングを行い，次にそのマッチング結果を用いて，先読みパターンを除いた残りのパターン “R_1R_3” に対するマッチング行う手法とリングバッファによるテキストの入れ替えを行うバッファ機構について提案している。先読み演算はパターン全体 “$R_1(?=R_2)R_3$” または “$R_1(?!R_2)R_3$” を指し，先読みパターンは R_2 だけを指す用語とする。

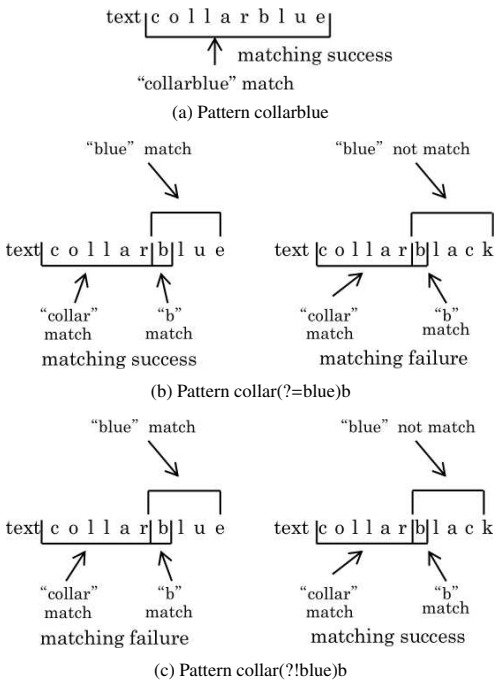


Fig. 4. Examples of matching for look-ahead.

〈3･2･1〉 先行研究のマッチング手法 パターン “$R_1(?=R_2)R_3$” に対するマッチング手法の動作例を Fig. 5 に示す。先読みマッチングにおいて，まず先読みパターン R_2 と一致する部分文字列の先頭を見つける。そして同じテキストに対し R_2 を除くパターンでのマッチングを行い，R_3 のマッチング開始位置が R_2 と同じなら R_3 のマッチングを行う。それ以外は，マッチングは失敗しているので行わない。上記は肯定先読みの場合で否定先読みの場合は逆の動作となる。

〈3･2･2〉 パターンと一致する部分文字列の先頭文字の検索 マッチングエンジンは，まず先読みパターンと一致する部分文字列を見つける必要がある。しかしながら，正規表現マッチングハードウェアはテキストの先頭からマッチングを行うとパタンと一致する部分文字列の末尾文字は検出できるが，先頭文字は検出できない。

そこで，元のパターン P を逆順にした逆順パターン P^R をハードウェアに入力し，その後，テキストも逆順で入力していく。この時も，P^R と一致する部分文字列の末尾文字を検出するが，Fig. 6 に示すように，逆順の末尾文字は元のパターン P の先頭文字となっている。これにより，P と一致する部分文字列の先頭文字を漏れなく見つけることが

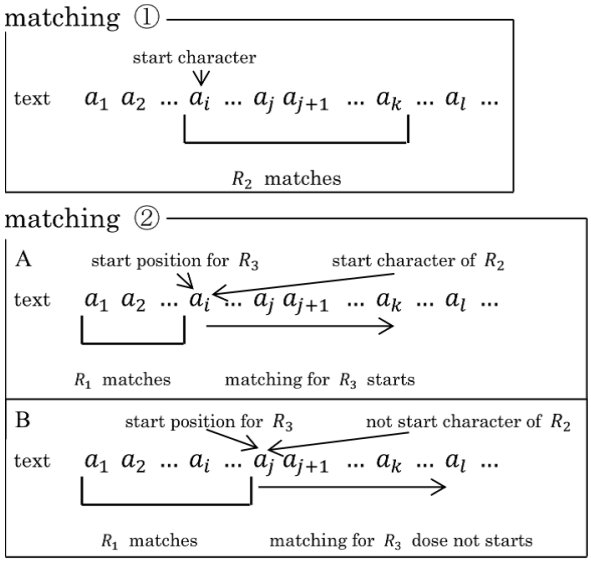


Fig. 5. Matching method of previous research.

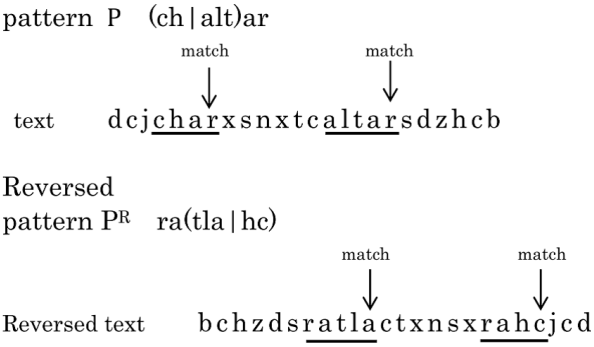


Fig. 6. Matching for a reversed pattern.

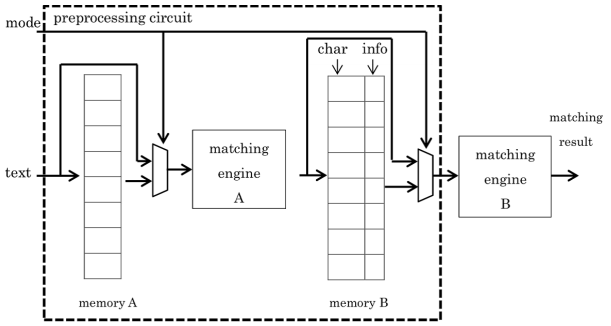


Fig. 7. Architecture of previous research.

できる。

〈3･2･3〉 先行研究のアーキテクチャ 以下では，先行研究のアーキテクチャについて説明する。アーキテクチャを Fig. 7 に示す。提案アーキテクチャは先読みパターンの逆中に対するマッチングを行う前処理回路と先読みパターンを除いたパターンに対する正規表現マッチングを行うマッチングハードウェア B からなる。また，前処理回路は 2 つのスタックメモリと正規表現からマッチングハードウェア A からなる。先読みを含まないパターンに対しては，2 つのエンジン A と B を直接つなぐことで，1 つのエンジンとして正規表現マッチングを行える。以下に動作を示す。

（1） マッチングが開始される前に，逆順先読みパターンをマッチングハードウェア A に設定し，先読みを除くパターンをマッチングハードウェア B に設定する。
（2） スタックメモリを用い，テキストを逆順にする。
（3） マッチングハードウェア A は先読みパターンと一致する部分文字列の末尾文字を検出し，末尾文字と分かるように情報を追加する。
（4） スタックメモリを用い，テキストの順を元に戻す。
（5） マッチングハードウェア B は前処理回路で得られた情報を用いて先読みパターンを除くパターンに対してマッチングを行う。

このアーキテクチャでは，上記 (2) と (4) の動作はテキストが無限の場合は，動作を終了することが無く正しく動作しない。しかし，NIDS であるため与えられるテキストはパケットであり，長さは有限であるため NIDS においては有効である。ここで用いられているスタックメモリはリングバッファと同じ構造でパケットを逆順にする仕組みになっている。詳しくは文献 (1) を参照されたい。

〈3･3〉 先行研究の問題点 先行研究では，逆順先読みパターンのマッチング結果をスタックメモリ B に保存しているので，先読みパターンが増えるとスタックメモリ B に追加する情報が増えてしまう。また，既存の正規表現マッチングハードウェアより動作周波数が低く，既存の正規表現マッチングハードウェアの性能を落とさなければならないという問題点がある。

4. 提案手法

本章では，先行研究の節で述べた問題点を改善した，先読みに対するハードウェアマッチング手法を提案する。提案ハードウェアは，先行研究の問題点を解決するためにスタックメモリ B とマッチングエンジン B を入れ替え，メモリに追加する情報を減らした。また，先読みを含まない場合も逆順でマッチングを行うことで，モード選択を省いた。

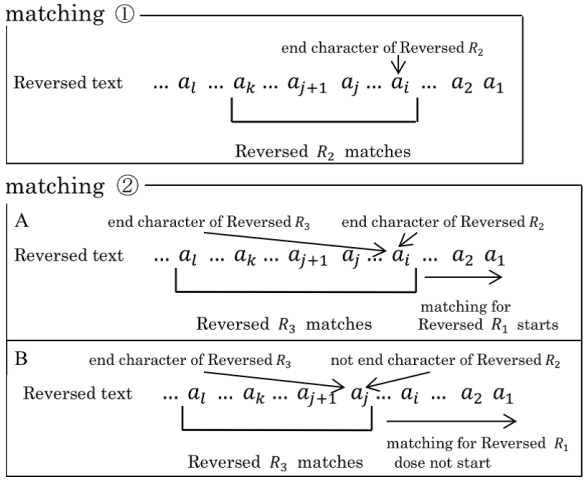


Fig. 8. The proposed matching method.

〈4･1〉 提案マッチング手法 提案手法では，まず逆順先読みパターン R_2 に対するマッチングを行い，次にそのマッチング結果を用いて，先読みパターンを除いた残りのパターンを逆順にして逆順テキストに対するマッチングを行う。パターン “$R_1(?=R_2)R_3$” に対するマッチング手法の動作例を Fig. 8 に示す。先読み演算において，まずテキストを逆順にして逆順先読みパターン R_2 と一致する部分文字列の末尾文字を見つける。そして同じテキストで先読みパターン R_2 を除く逆順パターンでマッチングを行い，逆順パターン R_3 と一致する部分文字列の末尾文字位置が逆順先読みパターン R_2 と一致した部分文字列の末尾文字と同じなら，逆順パターン R_1 のマッチングをおこなう。そうでなければ，マッチングは既に失敗しているので，逆順パターン R_1 のマッチングは行わない。上記は肯定先読みの場合であり否定先読みの場合は，逆順パターン R_3 と一致する部分文字列の末尾文字位置が逆順先読みパターン R_2 と一致した部分文字列の末尾文字が違う場合にマッチングを行う。

〈4･2〉 提案アーキテクチャ 提案手法を実現するためのアーキテクチャについて説明する。提案アーキテクチャを Fig. 9 に示す。提案アーキテクチャは逆順先読みパターンに対するマッチングを行うマッチングハードウェア A と先読みパターンを除いた逆順パターンに対するマッチングを行うマッチングハードウェア A とテキストを逆順にしてそれを元の順に戻す 2 つのスタックメモリからなる。スタックメモリは，先行研究のリングバッファ構造を用いる。パターンと一致する部分文字列の末尾文字を検出できる正規表現マッチングハードウェアであれば，パターン依存型とパターン非依存型のどちらでも，提案アーキテクチャの

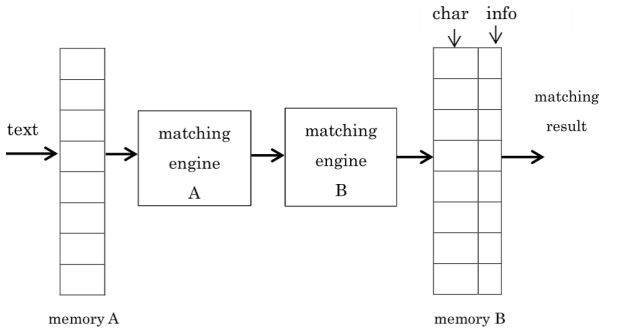


Fig. 9. The proposed architecture.

Table 1. 1,500 bytes packet size.

	Used LUTs	Used BRAMs	Maximum clock frequency
Previous research	163	3	251 [MHz]
Proposed method	148	2	270 [MHz]

Table 2. 65,536 bytes packet size.

	Used LUTs	Used BRAMs	Maximum clock frequency
Previous research	439	92	145 [MHz]
Proposed method	222	69	200 [MHz]

正規表現マッチングハードウェアとすることができる。先読みを含まないパターンに対しては，1 つのエンジンとして逆順パターンの正規表現マッチングを行う。以下に提案アーキテクチャの動作を示す。

（1） マッチングが開始される前に，逆順先読みパターンをマッチングハードウェア A に設定し，先読みを除く逆順パターンをマッチングハードウェア B に設定する。

（2） スタックメモリを用いて，テキストを逆順にする。

（3） マッチングハードウェア A は逆順先読みパターンと一致する部分文字列の末尾文字を検出し，末尾文字と分かるように情報を追加する。

（4） マッチングハードウェア B は前処理回路で得られた情報を用いて先読みパターンを除く逆順パターンに対してマッチングを行いその結果の情報を追加する。

（5） スタックメモリを用いてマッチング結果とテキストを元の順に戻す。

この提案手法により，先行研究での先読みに対するマッチングハードウェアと比べ，スタックメモリ B に追加する情報がマッチング結果のみとなったので，先読みパターンの数が増えても FPGA の使用するブロック RAM 数とブロック RAM 制御の Slice 数を増やす必要がなく，動作周波数の向上に繋がると考えられる。

〈4･3〉 実験的評価 提案手法を評価するため，ツールとして ISE Design Suite 14.4 を用い，先行研究を比較するため対象デバイスを Xilinx Virtex6（XC6VLX240T-1FFG1156）とした。パケットサイズ 1,500 bytes の場合とパケットサイズ 65,536 bytes の使用 LUT 数，使用ブロック RAM 数，最高動作周波数の比較をそれぞれ Table 1 と Table 2 に示す。

パケットサイズ 1,500 bytes の場合は，使用 LUT 数，使用 BRAM 数，最高動作周波数に大きな差が無かったが，それぞれの向上を確認できる。

パケットサイズ 65,536 bytes の場合は，使用 LUT 数，使用 BRAM 数，最高動作周波数に大きな向上を確認でき，最高動作周波数が，従来の正規表現マッチングハードウェアの動作周波数の 186 [MHz] より高くなり，従来の正規表現マッチングハードウェアの性能を低下させることなく，マッチングを行うことができる。

5. まとめと今後の課題

本稿では，既に提案されている正規表現の先読みに対するマッチングハードウェアを改善したマッチング手法を提案し評価した。この提案により，先行研究よりメモリの使用量が減り，動作周波数が従来の正規表現マッチングハードウェアより高くなることを示した。

今後の展開として，第一に今回提案した回路を実際に FPGA に実装して，実際にネットワーク上に流れるパケットを処理する実験を行い，その評価を行う必要がある。第二に，正規表現の後方参照については完全ハードウェアのみで実装する方法は示されていないため，後方参照に対するマッチングハードウェアを検討する必要がある。

謝 辞

本研究は，豊橋技術科学大学高専連携教育研究プロジェクト及び日本学術振興会の科学研究費補助金（No.40412902）の支援を受けて行われた。

文 献

（1） 若葉陽一・永山 忍・稲木雅人・若林真一：「パターン非依存正規表現マッチングエンジンに対する先読み演算の実現」, 信学技報, VLD2011-136 (2012)

（2） 川中洋祐・若林真一・永山 忍：「NIDS 専用正規表現マッチングマシンの構成とその FPGA 実装」, 信学技報, VLD2008-123, CPSY2008-85, RECONF2008-87 (2009)

（3） 空閑洋平・松谷健史・櫨山寛章・中村 修：「FPGA による拡張性の高いネットワーク試験環境の実現」, 信学技報, IA2012-85 (2013)

（4） 川中洋祐・若林真一・永山 忍：「パターン非依存型正規表現ストリングマッチとその FPGA 実装」, 信学論 (D), Vol.J92-D, No.12, pp.2159–2167 (2009)

（5） 関山 守・戸田健二・神徳徹雄：「大規模 FPGA を用いた規模拡張可能な高速ネットワーク侵入検知システムの開発」, 信学技報, CPSY2012-96, DC2012-102 (2013)

（6） 中原啓貴・笹尾 勤・松浦宗寛：「オートマトンの分解に基づく正規表現マッチング回路について」, 信学技報, VLD2010-98, CPSY2010-53, RECONF2010-67 (2009)

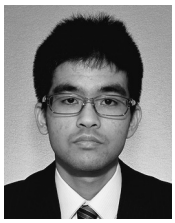
（7） 若葉陽一・若林真一・稲木雅人・永山 忍：「シストリックアルゴリズムと NFA に基づくパターン非依存正規表現マッチングハードウェア」, 信学論 (D), Vol.J96-D, No.10, pp.2139–2149 (2013)

（8） Cisco and/or its affiliates:Snort, https://www.snort.org/

（9） Jon Allen (JJ):Perl programming documentation, http://perldoc.perl.org/

（10） Jeffrey E.F Fridel：「詳説正規表現」, 田和勝 訳, オライリー・ジャパン (2003)

比 嘉 秀 斗（非会員） 2016 年沖縄工業高等専門学校情報通信システム工学科卒業。現在，エスアイエス・テクノサービス（株）技術開発部に勤務。



宮 城 桂（非会員） 2008 年高知工科大学情報システム工学科卒業。2010 年同大学大学院修士課程修了。2014 年同大学大学院博士課程修了。同年沖縄工業高等専門学校情報通信システム工学科助教。現在に至る。博士（工学）。自己同期型パイプラインを用いた低消費電力 VLSI の研究に従事。電子情報通信学会会員。

山 田 親 稔（正員） 2000 年琉球大学大学院理工学研究科博士前期課程修了。2004 年同大学大学院博士後期課程単位取得満期修了。同年拓殖大学北海道短期大学専任講師。2007 年沖縄工業高等専門学校情報通信システム工学科助教。2009 年同高等専門学校准教授。2014 年ビクトリア大学（カナダ）客員研究員。2015 年沖縄工業高等専門学校情報通信システム工学科准教授。博士（工学）。形式的設計検証，リコンフィギャラブルシステムの研究・教育に従事。電子情報通信学会会員。

市 川 周 一（非会員） 1985 年東京大学理学部卒業。1987 年東京大学大学院理学系研究科修士課程修了。1987 年新技術事業団 創造科学推進事業（ERATO）後藤磁束量子情報プロジェクト研究員。1991 年三菱電機（株）LSI 研究所，システム LSI 開発研究所勤務。1994 年名古屋大学工学部助手。1997 年豊橋技術科学大学工学部知識情報工学系講師。2001 年豊橋技術科学大学工学部知識情報工学系助教授。2007 年豊橋技術科学大学工学部知識情報工学系准教授。2010 年豊橋技術科学大学大学院工学系研究科准教授。2011 年沼津工業高等専門学校制御情報工学科教授。2012 年より，豊橋技術科学大学大学院工学系研究科教授。現在に至る。理学博士。並列計算機，並列処理，および専用計算システムアーキテクチャの研究に従事。IEEE（senior member），電子情報通信学会（シニア会員），ACM，情報処理学会，各会員。