FPGAによる高速かつ軽量なNFAパターンマッチング回路

片下敏宏[†] 前田敦司[†] 小野正人[†]
戸田賢二[††] 山口喜教[†]

近年のネットワークの高速化により，ネットワークサービスに対する攻撃や侵入を検知するNIDSでは，処理の中核であるパターンマッチング処理の高速化が必須となっている．本研究では，1クロックサイクルあたり4バイト同時に処理するNFAパターンマッチング回路を提案する．従来の1クロックサイクルあたり1バイトを処理するNFAパターンマッチング回路に対し，回路規模の増大を2倍程度に抑えつつ，処理性能を3倍以上向上させた．30,675文字のパターンマッチングを対象とした場合，Xilinx xc2vp100-6において回路規模は21,218 Slice，29,211 FF，40,960 LUTであり，処理性能は6.2 Gbps（193.78 MHz動作）である．さらに，同時に処理するバイト数を変化させて回路を評価した結果，本回路構成では8バイト同時に処理する場合に最も効率良く実装できることが分かった．

Fast and Compact NFA Pattern Matching Circuit Using FPGAs

Toshihiro Katashita,[†] Atsushi Maeda,[†] Masato Ono,[†]
Kenji Toda[††] and Yoshinori Yamaguchi[†]

Due to rising network traffic in recent years, improving processing throughput of the pattern matching is important in NIDS. In this paper, we propose a fast and compact NFA based pattern matching circuit which processes 4 bytes data at each clock cycle. This circuit is about triple faster than the former one which processes 1 byte data at each clock cycle, but it is about twice larger. On Xilinx xc2vp100-6, the throughput of this NFA circuit is 6.2 Gbps (192.78 MHz). And the area usage is 21,218 Slices, 29,211 FFs, and 40,960 LUTs for 30,675 patterns. We also evaluated our circuit in case of multiple bytes processing, and found processing eight bytes at each clock cycle with our circuit is most efficient.

1. はじめに

近年ではネットワークにおけるサービスに社会が依存する傾向にあり，ネットワーク上におけるセキュリティの重要性が高まっている．しかし，ネットワークを利用する人口が増加するに従い，ネットワークサービスに対する攻撃や侵入が増加し，サービスに障害をもたらす要因となっている．このネットワークサービスに対する攻撃や侵入を検知し防御を行う起点となる技術が，NIDS（Network Intrusion Detection System）である．NIDSでは常時ネットワークの状態を監視し侵入や攻撃の検知を行うため，その処理速度はネットワークのスループットと同じであることが望まれる．NIDSの1つに，攻撃・侵入ルールのパターンマッチングによる検知を行うSNORT[1]があるが，ソフトウェアによるパターンマッチング処理を行っているため，その処理速度は1 Gbpsに満たない．

NFA（Nondeterministic Finite Automaton）によりパターンマッチング回路を再構成可能なデバイスFPGA（Field Programmable Gate Array）へ構成し，高速な処理を行う研究がいくつかなされている[2)~4)]．NIDSでは，新たな攻撃や侵入に従ってマッチングパターンが変更されるためNFAによるパターンマッチングの場合ASICに実装するのは実用的でなく，FPGAのような再構成可能なハードウェアがパターンマッチングエンジンとして用いられる．

この従来のNFAによるパターンマッチング回路は，1クロックサイクルあたり1バイトを処理するため，その処理性能は1.96 Gbps（245.97 MHz，30,675文字，Xilinx xc2vp100-6時）程度であった．このNFA

† 筑波大学システム情報工学研究科
Systems and Information Engineering, University of Tsukuba
†† 産業技術総合研究所情報技術研究部門
Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology

回路を 1 クロックあたり複数バイト処理する構成に改変し，処理性能を向上させる研究がなされている[5]．しかし，改変による回路規模の増大は大きく，4 バイト処理する NFA 回路は 25,002 文字のマッチングパターンにおいて Xilinx xc2v-8000 の 100%を使用するものであった．

本研究では，回路規模の増大を抑えつつ，1 クロックサイクルあたり 4 バイト処理する NFA パターンマッチング回路を提案する．そして，SNORT ルールより抽出したマッチングパターンより NFA 回路を構成し，その回路規模，処理性能について検証を行った．その結果，SNORT ルールより抽出した 30,675 文字のマッチングパターンを対象とした回路でも Xilinx xc2v-8000 の 38%しか使用しないことが分かった．また，従来の 1 クロックあたり 1 バイト処理する NFA 回路に比べて，回路規模の増加を 2 倍程度に抑えることが可能であり，スループットは 6.2 Gbps（193.78 MHz, 30,675 文字，Xilinx xc2vp100-6 時）と 3 倍以上高めることができることが分かった．

さらに，1 クロックあたり処理するバイト数を変化させて評価を行った結果，スループットを 1 文字あたりに必要な回路規模で割った値で定義される performance 値は 8 バイトごとに処理する場合が最も高いことが分かった．本論文では，まず提案する NFA パターンマッチング回路構成について述べ，次に回路規模とスループットの検証について述べる．そして最後に，従来の研究におけるパターンマッチング回路との比較について述べる．

2. NFA パターンマッチング回路

GNU grep などのツールでは，マッチングパターンの正規表現から NFA を構成し，これを DFA（Deterministic Finite Automaton）に変換して処理を行っている．この正規表現から構成した NFA を OHE（One-Hot Encoding）方式により回路化する方法が Sidhu ら[2] により提案されている．NFA 回路では，同時刻に 1 つのステートを持つ DFA 回路[6)~8)] とは異なり，同時に複数のステートを持つ．

2.1 1 バイト処理の NFA 回路

従来の 1 クロックサイクルあたり 1 バイト処理する NFA 回路は**図 1** のような構成となる[3]．回路に入力された文字はコンパレータで比較され，比較結果は回路中の各ステートマシンに分配される．この比較結果によってステートマシンの状態が遷移し，特定のステートに遷移したときにマッチング結果が出力される．

図 1 の回路構成では，文字入力からマッチング結果

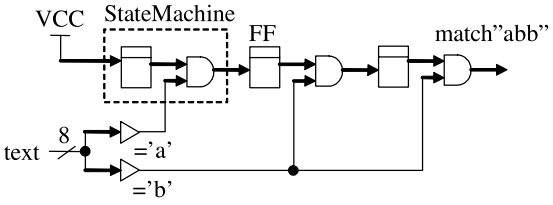


図 1 “abb” を検出する NFA 回路
Fig. 1 NFA circuit for “abb” matching.

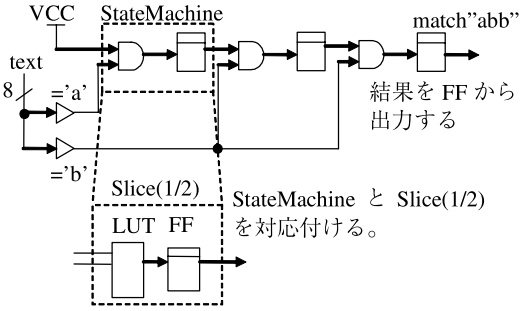


図 2 改良した “abb” を検出する NFA 回路
Fig. 2 Improved NFA circuit for “abb” matching.

出力までのパスに組合せ回路のみのパスが存在し，高速な回路構成に適さない．そこで，本研究では回路構成を**図 2** のように改良し，これを提案する回路の比較対象である 1 バイト処理 NFA 回路とした（以後，この回路を Single-NFA と呼ぶ）．

Single-NFA ではマッチング結果を FF（Flip-Flop）から出力する．また，ステートマシンを Xilinx virtex シリーズの論理回路ブロック Slice[9),10)] の形に対応させたことで，回路に必要な Slice 数を容易に推定できるようになった．図 1 のステートマシン構成では，組となっている FF と LUT が異なった Slice へマッピングされる場合があり，推定した Slice 数よりも実際の Slice 数が大きくなる場合がある．

NIDS では複数のマッチングルールに対して同時にパターンマッチングを行う．そのため，NFA 回路には複数のマッチングルールに対する NFA のステートマシンのツリーが含まれているが，このステートツリー中に含まれるステートには同じ状態を表しているものが存在する．**図 3** では，「abc」「abb」「acb」の 3 つのマッチングパターンに対する NFA のステートツリーを示している．この中で，「abc」と「abb」では最初の「ab」がマッチされたステートが同じ状態を表しており，このステートを共有化することができる．図 3 では 3 つのルールそれぞれで独立に NFA 回路を構成するとステート数は 9 となるが，同じ状態のステート

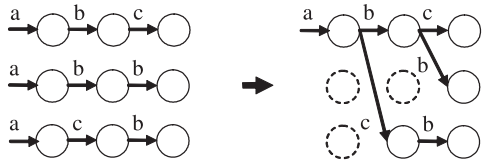


図 3 Single-NFA におけるステートの縮小
Fig. 3 State reduction on Single-NFA.

表 1 Single-NFA におけるステート数の削減
Table 1 State reduction on Single-NFA.

ルール数	128	256	512	1,024	2,028
総文字数	1,427	2,630	6,223	14,953	30,675
ステート数（共有化前）	1,427	2,630	6,223	14,953	30,675
ステート数（共有化後）	1,140	2,161	4,335	10,206	20,487
削減率（%）	20.1	17.8	30.3	31.7	33.2

を共有化するとステート数は 6 へ縮小させることができることが分かる．

表 1 に，Snort2.2 のルール（全 2,028 ルール）からランダムに 128，256，512，1,024，2,028 個取り出したルールを解析し，ステートの共有化によってステート数がどの程度縮小するか検証した結果を示す．表 1 より，総文字数が増加するに従って，共有化により削減できるステート数が多くなっていることが分かる．また，削減率も，総文字数が 1,427 のときに 20.1%，30,675 のときに 33.2%と増加することが分かる．

2.2 4 バイト処理の NFA 回路

Single-NFA では 1 バイトのマッチングごとにステートマシンの状態が変化するが，これを 4 バイトのマッチングごとにステートを変化させるものが，1 クロックサイクルあたり 4 バイト処理する NFA である．**図 4** に Clark ら[5] により提案された 4 バイト処理の NFA 回路構成を示す（以後，Decoder-NFA と呼ぶ）．また，**図 5** に本論文で提案する NFA 回路を示す（以後，Quad-NFA と呼ぶ）．Decoder-NFA は，入力されたデータを 1 バイトごとにマッチングするコンパレータと，パターンマッチングルールごとの NFA ステートマシンよりなる．

Quad-NFA は，(1) 入力したデータを 1 バイトごとにマッチングするコンパレータ部，(2) コンパレータにより出力されたマッチング結果を論理積して 4 バイトマッチング結果を出力する 4-AND 部，(3) 4 バイトマッチング結果により状態を遷移させる NFA のステートマシン部，(4) NFA のステートマシンの出力を論理和しマッチング結果を出力する 4-OR 部の 4 つの回路よりなる．

Quad-NFA は，マッチングルールごとに NFA ステートマシンを持つ Decoder-NFA に対し，マッチングルールすべてに対して 1 つの NFA ステートマシンのみを持つ点が異なる．また，4 文字マッチング結果の 4-AND がステートマシンに組み込まれていない点も異なっている．

Quad-NFA では，1 クロックサイクルあたり 4 バイトごとに入力されたデータは 1 バイトごとに分割され，1 バイトのコンパレータへ入力される．比較結果を 4-AND 部で論理積された 4 バイトマッチング結果は NFA ステートマシン部の各ステートマシンへ分配される（**図 6**）．この比較結果によってステートマシ

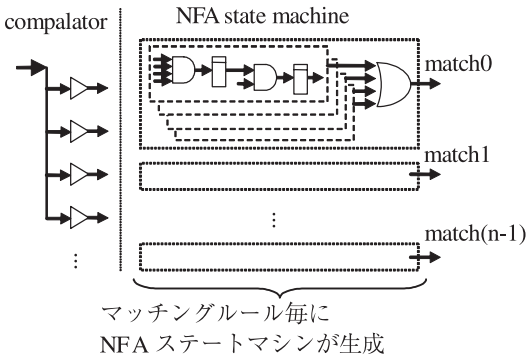


図 4 Decoder-NFA [5] の回路構成
Fig. 4 Circuit composition of Decoder-NFA [5].

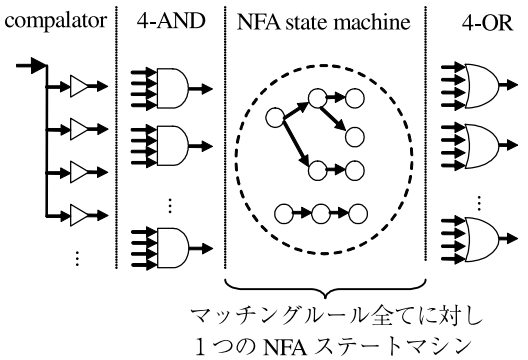


図 5 提案する Quad-NFA の回路構成
Fig. 5 Circuit composition of Quad-NFA.

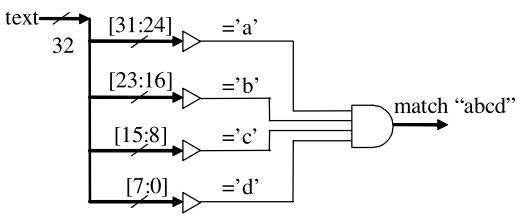


図 6 4 バイトマッチングを得る回路の構成
Fig. 6 Circuit for 4 bytes matching.

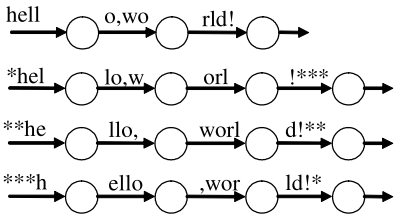


図 7 4 バイト処理の NFA のステートマシン
Fig. 7 NFA state machine for 4 bytes processing at each cycles.

表 2 Quad-NFA におけるステート数の削減
Table 2 State reduction on Quad-NFA.

ルール数	128	256	512	1,024	2,028
総文字数	1,427	2,630	6,223	14,953	30,675
ステート数（共有化前）	1,811	3,398	7,759	18,025	36,759
ステート数（共有化後）	1,524	2,929	5,871	13,278	26,571
削減率（%）	15.8	13.8	24.3	26.3	27.7

ンの状態が遷移し，特定のステートに遷移した結果の論理和がとられマッチング結果が出力される．

4 バイト処理の NFA ではマッチングパターンの先頭が 4 バイト中の先頭になるとは限らず，パターン先頭のズレごとの NFA が必要となる．よって，NFA のステートツリーの個数は Single-NFA の 4 倍となる．一方，ステートツリーの深さは短くなるため，実質の総ステートマシン数は 4 倍とはならない．たとえば，「hello,world!」の場合，Single-NFA では 12 ステートであるが，Quad-NFA や Decoder-NFA では 15 ステートとなる（**図 7**）．図 7 中の “*” は空白を表している．

Quad-NFA のステートマシンは Single-NFA と同様にステートの共有化を行うことができる．一方，Decoder-NFA では共有化を行っていない．**表 2** に，Quad-NFA におけるステート共有化によるステート数の削減を Single-NFA と同様に検証した結果を示す．

表 2 より Quad-NFA の場合も Single-NFA と同様に，総文字数に従って共有化できるステート数が増加していることが分かる．ステート数の削減率も同様に総文字数 1,427 のとき 15.8%，30,675 のとき 27.7%と増加している．しかし，ステートの遷移条件が複雑であるため，Single-NFA に比べて削減率は低くなっている．

4 文字マッチング結果を得る 4-AND も，ステートマシンと同様に共有化できる．4-AND は “abc” と “bbc” の場合のように，ステートマシンでは共有化できないマッチングパターンでも，**図 8** の点線で示すような共有化できる部分が存在する．

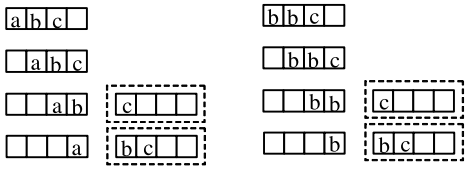


図 8 4-AND が共有化できる例
Fig. 8 Example of 4 input AND gate reduction.

表 3 共有化による 4-AND 数の削減
Table 3 4 input AND gate reduction on Quad-NFA.

ルール数	128	256	512	1,024	2,028
総文字数	1,427	2,630	6,223	14,953	30,675
4-AND 数（共有化前）	1,524	2,929	5,871	13,278	26,571
4-AND 数（共有化後）	1,340	2,463	4,598	8,811	15,189
削減率（%）	12.1	15.9	21.7	33.6	42.8

表 3 に共有化による 4-AND 数の削減をステートマシンの場合と同様に検証した結果を示す．ステートマシンの遷移と 4 文字マッチングは対であり，ステート数と 4-AND 数は共有化前では同数となる．よって共有化前の 4-AND 数は，ステート共有化後のステート数とした．

表 3 より，総文字数の増加に従って共有化できる 4-AND 数が大幅に増加していることが分かる．また，削減率も総文字数の増加に従って 12.1%から 42.8%と大幅に増加している．

3. 実験と検証

提案する Quad-NFA 回路を論理合成し，Single-NFA に対する回路規模の増加とスループットの向上について検証を行った．さらに，Quad-NFA を 8，16，32，64 バイト同時に処理するように変化させた場合の検証も行った．

3.1 自動化ツールによる NFA 回路の生成

実験では，Snort のルールセットから NFA 回路の生成を自動化するツールを作成した．本ツールは，Snort ルールからのマッチングパターンの取り出しを行い，Single-NFA，Quad-NFA のステートマシンをそれぞれ生成する．そして，生成したステートマシンのステート共有化を行った後，回路を生成する．また，Quad-NFA では回路生成前に 4-AND の共有化も行う．

このツールを用いて，Snort 2.2 のルールからランダムに 128，256，512，1,024，2,028 個取り出したルールより Single-NFA，Quad-NFA 回路を生成し，その

表 4 論理合成結果（回路規模）
Table 4 Resource usage.

ルール数	文字数	Single-NFA			Quad-NFA（4）				Quad-NFA（8）			
		Slice	FF	LUT	Slice	FF	LUT	増加率（%）	Slice	FF	LUT	増加率（%）
128	1,427	736	1,317	1,400	1,673	1,771	3,216	227	3,060	2,467	5,904	416
256	2,630	1,369	2,477	2,457	3,088	3,400	5,935	226	5,665	4,706	940	414
512	6,223	2,659	4,924	4,794	5,702	6,687	10,959	214	9,596	8,901	18,238	361
1,024	14,953	6,103	11,381	11,271	11,460	14,713	22,057	188	19,542	19,139	37,159	320
2,028	30,675	12,250	22,791	22,606	21,218	29,211	40,960	173	37,910	38,101	72,661	309

表 5 論理合成結果（回路規模）
Table 5 Resource usage.

ルール数	文字数	Quad-NFA（16）				Quad-NFA（32）				Quad-NFA（64）			
		Slice	FF	LUT	増加率（%）	Slice	FF	LUT	増加率（%）	Slice	FF	LUT	増加率（%）
128	1,427	6,007	3,816	11,458	816	12,052	6,551	22,923	1,638	23,960	12,058	45,663	3,255
256	2,630	11,111	7,177	21,112	812	21,767	12,159	41,316	1,590	45,284	22,039	85,709	3,308
512	6,223	19,707	13,716	37,401	741	39,013	23,133	74,055	1,467				
1,024	14,953	39,761	28,344	75,493	651								

回路規模とスループットを検証した．用いた乱数はステート数・4-AND 数の削減を検証した際と同じものである．

Quad-NFA の同時に処理するバイト数は次の手順で変化させた．まず，NFA ステートマシンをバイト数に合わせて遷移するように変更し，4 バイトの場合と同様にステートの共有化を行う．これにともない，結果出力の OR はバイト数と同じ入力数となる．(8 バイトのときは，8-OR となる）次に，ステート遷移条件であるマッチング結果を 4 バイト境界で区切り 4-AND 部の生成と共有化を行う．最後に 4-AND 部の結果の論理積をとってステートマシンに入力する回路を追加する．

回路の検証を行う際，Single-NFA，Quad-NFA それぞれの入出力ポートに FF を配置した．これは，Xilinx ISE の論理合成ツール xst 上で正確な動作周波数測定を行うためである．xst では最大動作周波数を算出する際，FPGA の入力から FF へのパスと FF から出力のパスが I/O のセットアップ/ホールドパスとして認識され，正しい最大動作周波数が推定されない．FF を配置することで，NFA 回路の入出力のパスも含まれた正確な最大動作周波数の算出を行っている．

3.2 回路規模の検証

ツールで生成した NFA 回路を ISE 6.3sp3 を用いて論理合成し，回路規模の検証を行った．ツールでは Single-NFA 回路と，Quad-NFA の回路構成で 4，8，16，32，64 バイト同時に処理する回路を生成している．(以後，4 文字同時に処理する Quad-NFA を Quad-NFA（4）と同時処理文字数を括弧中に示す）実装対象 FPGA は XAUI（10 Gigabit Attachment Unit Interface）が接続可能な Xilinx xc2vp100-ff1704-6 とした．論理合成時の設定は xst の初期設定と同じとした．

表 4，**表 5** に論理合成した回路規模結果を示す．表 4，表 5 中の Slice 増加率は，Quad-NFA の Slice 数を Single-NFA と比較した数値である．デバイスの容量を大幅に超える結果は記載していない．なお，Quad-NFA（64）のルール数 256 の場合は回路規模がデバイスの規模を 1,188 Slice 超えているが，参考数値として記載している．

また，**図 9** にそれぞれの回路規模と，Quad-NFA（4），Quad-NFA（8）の回路規模を Single-NFA と比較した増加率を示す．

表 4，表 5 より，Single-NFA と比較して Quad-NFA（4）の回路規模の増大が 1.73〜2.28 倍となっていることが分かる．さらに，総文字数の増加に反して規模の増加率は減少していることが分かる．これは，次の 2 点の要因によるものと考えられる．

(1) 回路全体に対する比較器部の回路規模
Quad-NFA（4）の比較器の数は Single-NFA の 4 倍となるが，最大でも Single-NFA で 256 個，Quad-NFA（4）で 1,024 個である．1 つの比較器は LUT を 3 個使用するため比較器部の差は最大で 2,304 LUT であるが，NFA の回路規模が増加するに従って影響は小さくなると考えられる．

(2) 4-AND 部の共有化
2 章で述べたとおり，総文字数の増加に従って 4-AND は共有化により大幅に削減できる．これにより，Quad-NFA の総文字数の増加に対する回路規模の増大が抑えられたと考えられる．

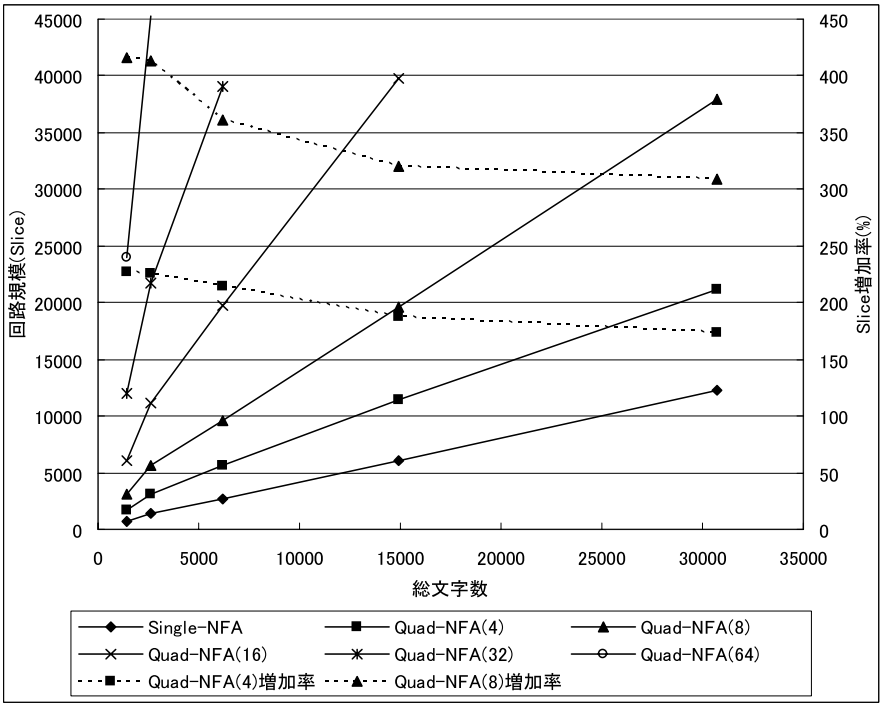


図 9 Single-NFA と Quad-NFA の回路規模と，Single-NFA に対する Slice 増加率
Fig. 9 Circuit size of Single-NFA and Quad-NFA.

処理するバイト数を 1 から 4 とした場合は回路規模の増大を 2.28 倍と文字数の倍率以下であるが，8，16，32，64 と変化させた場合，4.16，8.16，16.38，33.08 倍と，4 の場合と比べて 2 倍程度ずつ増加することが分かった．

3.3 スループットの検証

次に，NFA 回路の最大動作周波数を算出しスループットの検証を行った．xst の設定や対象 FPGA は回路規模の検証と同じである．

表 6，**表 7** に論理合成により得られたスループットの結果を示す．表 6，表 7 中の向上率は，Quad-NFA のスループットを Single-NFA と比較した数値である．

表 6，表 7 より，Quad-NFA（4）のスループットは Single-NFA に比べて 3.10〜3.15 倍と向上することが分かる．また，同時処理バイト数を 8 とすることで 10 Gbps のスループットが達成可能であることが分かる．

総文字数の増加に従って，Single-NFA，Quad-NFA ともスループットが低下しているが，これは回路中で共有している信号が増加しており，この信号のファンアウトが増大によりディレイが増加したためであると考えられる．

また，ファンアウトが大きい信号がツール上で複製されずロジックレベルが高いパスにファンアウトが大きい信号が集中すると，Quad-NFA（8）のルール数 1,024 の場合に見られるように，動作周波数が低下してしまう場合があることが分かった．このファンアウトの集中を避ける対策として，回路生成ツールであらかじめ多く共有されている信号を特定し，HDL 上で信号を複製する方法が考えられる．

3.4 予備的実験

Quad-NFA によるマッチング回路を FPGA ボード上へ実装し，Gigabit Ethernet PHY と接続して予備的な実験を行った．実装は，FPGA ボード REX [11] に Gigabit Ethernet PHY チップを接続した環境上に行った．FPGA は Xilinx xcv2000e-6 である．この実験によりスループット 1 Gbps 時の Quad-NFA の動作確認ができたため，今後は検証において対象としたデバイスによる実装を行い，スループット等を検証する予定である．

4. 関 連 研 究

従来の NFA を複数文字同時に処理させるように改変して高速化した Decoder-NFA [5] は，4 バイト処理の場合スループットを 5.34〜8.49 Gbps に向上させている．しかし，Decoder-NFA は回路規模が大きく，

表 6 論理合成結果（スループット）
Table 6 Throughput.

ルール数	文字数	Single-NFA		Quad-NFA（4）			Quad-NFA（8）		
		frequency（MHz）	throughput（Mbps）	frequency（MHz）	throughput（Mbps）	向上率（%）	frequency（MHz）	throughput（Mbps）	向上率（%）
128	1,427	261.5	2092.1	205.7	6582.3	315	204.7	13102.7	626
256	2,630	259.2	2073.4	204.7	6551.3	316	196.4	12570.0	606
512	6,223	250.2	2001.8	196.0	6270.8	313	182.8	11700.2	584
1,024	14,953	248.8	1990.3	193.8	6201.0	312	166.9	10682.7	537
2,028	30,675	246.0	1967.8	193.8	6201.0	315	192.5	12319.6	626

表 7 論理合成結果（スループット）
Table 7 Throughput.

ルール数	文字数	Quad-NFA（16）			Quad-NFA（32）			Quad-NFA（64）		
		frequency（MHz）	throughput（Mbps）	向上率（%）	frequency（MHz）	throughput（Mbps）	向上率（%）	frequency（MHz）	throughput（Mbps）	向上率（%）
128	1,427	180.4	23094.3	1,104	175.3	44884.7	2,145	159.8	81815.0	3,911
256	2,630	177.1	22671.0	1,093	167.3	42820.1	2,065	156.9	80338.9	3,875
512	6,223	160.2	20499.7	1,024	159.6	40848.9	2,041			
1,024	14,953	159.6	20424.4	1,026						

xc2v-8000 において総文字数 25,002 までしか対応できない．一方，本研究の Quad-NFA（4）を同じデバイスに実装した場合では，Snort2.2 の全ルール 2,028 時の 30,675 文字で Slice 使用率が 40%と回路規模が小さい．これは，Quad-NFA ではマッチングルールすべてに対して 1 つの NFA を持ち，その中でステートの共有化を行い，4-AND も共有化を行っているためである．

NFA でない方式のパターンマッチング回路は，DFA [6)~8)]，CAM [12)]，KMP [13)] による方式による回路などが提案されている．それぞれの回路のスループットと回路規模の比較を**表 8** に示す．表中の LE（Logic Element）数は Slice 数を 2 倍して算出している．Xilinx 社 FPGA の Slice は 2 個の LC（Logic Cell，LE と同構成[5)]）より構成されているためである[9),10)]．

また，論文 5) で評価基準に用いられている Performance 値を算出し表に示している．

$$\begin{aligned} Performance &= Throughput \times Density \\ &= Throughput \times (\#char) \div LE \\ &= Throughput \times (\#char) \div Slice \div 2 \end{aligned}$$

なお，Sugawara ら[8)] により提案されている DFA は FPGA 中の RAM を使用する実装であるため表 8 に記載していない．

表 8 より，1 文字あたりの Slice 数は，すべての同時処理するバイト数において，Single-NFA と Quad-NFA が最も低回路規模であることが分かる．また，実装対象とするデバイスは異なるためスループット値は単純に比較できないものの，Quad-NFA（8）の Performance 値が最も高い値であることから，Quad-NFA（8）が最も効率の良い回路実装方式であることが分かる．

5. ま と め

1 クロックサイクルあたり 4 バイト処理する NFA パターンマッチング回路を提案し，その回路規模，処理性能について検証を行った．その結果，1 バイト処理の NFA 回路に比べて回路規模の増大を 2 倍程度に抑えつつスループットを 3 倍以上向上させることが可能であることが分かった．また，同時処理させるバイト数を変化させて検証を行った結果，Quad-NFA の構成で 8 バイト同時に処理させる回路が最も効率が良いことが分かった．

さらに，これまでなされている研究で提案されている 4 バイト処理のパターンマッチング回路と比較しても，提案する回路は 1 文字あたりの Slice 使用数が半数以下と回路規模が非常に小さい．

しかし，同時に処理させるバイト数を増やすと，ステートツリーの長さが短くなり共有できるステートは減少するほか，コンパレータに必要な LUT も増大するため FPGA 中の Flip-Flop 使用数に比べて LUT の使用数が非常に大きくなってしまう．このことから Quad-NFA 回路の同時処理文字数を単純に増大させた場合の，回路規模の増大を抑えることが難しくなる．そのため，同時に処理するバイト数を増やしても回路規模増大を抑える新たな方法を考案することが課題である．

また，ファンアウトが大きいコンパレータ部の回路

表 8 パターンマッチング回路のスループットと回路規模の比較
Table 8 Comparison of pattern matching circuits.

	device	input bit	frequency (MHz)	throughput (Mbps)	#char	LE	LEchar	performance (MB/ (s・LE))
Single-NFA	xc2vp-100	8	245	1,960	30,675	24,500	0.80	2,454
Quad-NFA	xc2vp-100	32	193	6,176	30,675	42,436	1.38	4,464
	xc2vp-100	64	192	12,288	30,675	75,820	2.47	4,971
	xc2vp-100	128	159	20,352	14,953	79,522	5.32	3,827
	xc2vp-100	256	159	40,704	6,223	78,026	12.54	3,246
	xc2vp-100	512	159	81,408	1,427	47,920	33.58	2,424
Decoder NFA [5]	xcv-1000e	8	100	800	17,537	19,698	1.12	712
	xc2v-8000	8	253	2,024	17,537	29,281	1.67	1,212
	xc2v-8000	32	218	6,976	17,537	54,890	3.13	2,229
	xc2v-8000	64	114	7,296	17,537	93,180	5.31	1,373
	xc2vp-125	128	129	16,512	7,996	93,180	11.65	1,417
	xc2vp-125	256	141	36,096	2,001	63,012	31.49	1,146
	xc2vp-125	512	195	99,840	250	47,748	190.99	523
DFA [7]	xcv-2000e	32	37	1,184	420	8,134	19.37	61
CAM [12]	xc2v-6000	32	303	9,696	18,032	64,268	3.56	2,720
	xc3s-5000	32	154	4,928	18,032	66,556	3.69	1,335
	xc2v-3000	8	335	2,680	18,032	17,538	0.97	2,755
KMP [13]	xc2vp-4	8	221	1,768	32	102	3.19	555
	xc2vp-4	8	285	2,280	32	65	2.03	1,122

構成を見直し，回路生成ツール上で回路中のファンアウト数の推定による適切な信号の複製を行うことで，回路規模の増大を抑えつつ効果的に最大動作周波数の向上を図る予定である．

謝辞 提案する回路構成によって 8 バイト同時処理する際に回路実装の効率が最も高くなるという知見は，査読者の示唆による実験により得られた．ここに感謝の意を表する．

本研究の一部は，日本学術振興会平成 15 年科学研究費基盤研究（B）(2) 15300013「書き換え可能デバイスによる高速パケット処理の研究」による．

参 考 文 献

1) Roesch, M.: Snort — lightweight intrusion detection for networks, *13th Systems Administration Conference, LISA '99*, Seattle, WA (1999).
2) Sidhu, R. and Prasanna, V.K.: Fast Regular Expression Matching using FPGAs, *Proc.IEEE FCCM 2001* (2001).
3) Hutchings, B.L., Franklin, R. and D.C.: Assisting Network Intrusion Detection with Reconfigurable Hardware, *FCCM2002* (2002).
4) Clark, C.R. and Schimmel, D.E.: Efficient Reconfigurable Logic Circuits for Matching Complex Network Intrusion Detection Patterns, *FPL2003*, pp.956–959 (2003).
5) Clark, C.R. and Schimmel, D.E.: Scalable Pattern Matching for High Speed Networks, *FCCM2004*, Napa, California, pp.249–257 (2004).
6) 栗原 純，丹羽雄平，前田敦司，山口喜教：FPGA/ソフトウェア協調処理による侵入検知システムの提案，信学技報，Vol.102, No.276, pp.11–16 (2002).
7) Moscola, J., Lockwood, J., Loui, R.P. and Pachos, M.: Implementation of a Content-Scanning Module for an Internet Firewall, *FCCM2003*, pp.31–38 (2003).
8) Sugawara, Y., Inaba, M. and Hiraki, K.: Over 10 Gbps String Matching Mechanism for Multi-Stream Packet Scanning Systems, *FPL2004* (2004).
9) Xilinx: Virtex-E 1.8 V Field Programmable Gate Arrays.
10) Xilinx: Virtex-II Platform FPGAs: Complete Data Sheet.
11) Kodama, Y., Katashita, T. and Sayano, K.: REX: A Reconfigurable Experimental System for Evaluating Parallel Computer Systems, *IEICE Trans. Inf. Syst.*, Vol.E86-D, No.10, pp.2016–2024 (2003).
12) Sourdis, I. and Pnevmatikatos, D.: Pre-decoded CAMs for Efficient and High-Speed NIDS Pattern Matching, *FCCM2004*, Napa, California, pp.258–267 (2004).
13) Baker, Z.K. and Prasanna, V.K.: Time and Area Efficient Pattern Matching on FPGAs, *FPGA2004*, Monterey, California (2004).

(平成 17 年 1 月 24 日受付)
(平成 17 年 5 月 6 日採録)

片下 敏宏

1999 年筑波大学大学院工学研究科退学（修士）．現在，同大学院システム情報工学研究科博士課程在学中．主として FPGA，回路設計，ネットワークセキュリティに関する研究に従事．

前田 敦司（正会員）

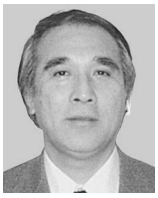
1994 年慶應義塾大学大学院理工学研究科数理科学専攻単位取得退学．博士（工学）（慶應義塾大学 1997 年）．1997 年電気通信大学大学院情報システム学研究科助手．2000 年筑波大学電子・情報工学系講師．2004 年筑波大学大学院システム情報工学研究科助教授（現職）．並列/分散処理，コンピュータアーキテクチャ，プログラミング言語の実装，ガーベッジコレクション等に興味を持つ．日本ソフトウェア科学会，ACM 各会員．

小野 正人

2004 年筑波大学第三学群情報学類卒業．現在，同大学大学院システム情報工学研究科博士課程在学中．主として FPGA の利用法，回路設計，ネットワークセキュリティに関する研究に従事．

戸田 賢二（正会員）

1982 年慶應義塾大学大学院工学研究科修士課程修了．同年電子技術総合研究所入所．以来，並列コンピュータのアーキテクチャの研究に従事し，記号処理用データ駆動計算機や実時間処理用並列計算機の開発を行った．近年は，組み込み応用をターゲットとし，開発環境の整備とともに，実時間処理用ハードウェアやネットワークの実用化研究を推進中．



山口 喜教（正会員）

1972 年東京大学工学部電子工学科卒業．同年通商産業省工業技術院電子技術総合研究所入所，計算機方式研究室長等を経て，1999 年筑波大学電子・情報工学系教授．博士（工学）（東京大学 1993 年）．現在，筑波大学システム情報工学科教授．高級言語計算機，並列計算機アーキテクチャ，並列実時間システム，ネットワーク侵入検知システム等の研究に従事．1991 年情報処理学会論文賞，1995 年市村学術賞受賞．著書『データ駆動型並列計算機』（共著）．IEEE Computer Society，ACM，電子情報通信学会各会員．