情報処理学会論文誌　コンピューティングシステム　Vol. 2　No. 3　163–172 (Sep. 2009)

辞書圧縮の概念を用いたNIDS向けパターンマッチングアーキテクチャ

飯星貴裕[†1]　山口喜教[†1]　前田敦司[†1]

ネットワークのセキュリティシステムの１つにネットワーク侵入検知システム（NIDS）がある．この NIDS のスループットを向上させるため，ボトルネックとなっているパターンマッチング処理を専用ハードウェアで行う試みがなされている．しかしながら，一般的に専用ハードウェアには膨大なパターン集合とのマッチングを高速に行うためには，回路規模を大きくせざるをえないという問題がある．そこで，パターン集合中に現れる共通の部分文字列に着目した．パターンマッチング回路では，一般的にパターンそれぞれに対して検出回路が存在する．そのため，パターン集合中に同じ部分文字列が複数現れると，その部分文字列に対する回路が複数存在することになり，回路規模的に大きな無駄になる．本論文では，既存のパターンマッチングアーキテクチャをベースに，辞書圧縮の概念を用いることで同じ部分文字列に対する回路を共有し，効率的に回路を構成できるアーキテクチャを提案する．また，ここではそのアーキテクチャに従って回路を構成するためのパターン圧縮アルゴリズムを考案し，パターンマッチング回路を実装・評価した．その結果，ベースとしたアーキテクチャに比べ，スループットにほぼ影響を与えることなく，小さな回路規模で回路を構成できることを示した．

A New Pattern Matching Architecture for NIDS Based on the Concept of Dictionary Compression

Takahiro Iihoshi,[†1] Yoshinori Yamaguchi[†1] and Atusi Maeda[†1]

Network security systems are becoming more important in our society. The performance of a signature-based intrusion detection system (NIDS) is mostly dependent on a string matching processing. To accelerate this processing, several attempts of string matching circuits have been studied so far. To get the high throughput string matching circuit for the huge amount of pattern rules, generally we have to pay more hardware cost for it. To reduce this cost, we focused on the common substring that will appear in the several pattern rules. The similar concept of dictionary compression is applied to the generation of the pattern matching circuit. The matching circuit for the same substring in the different rules can be shared in the total circuit for NIDS. This can reduce the amount of the total hardware cost. In this paper, the algorithm of how to share the substring circuit and how to reduce the total hardware cost is presented. Finally, the performance evaluation based on the hybrid architecture is studied.

1. はじめに

近年，社会においてネットワーク上で展開されている種々のサービスは必要不可欠なものとなった．それにともない，それらのサービスを妨害する各種攻撃に対してセキュリティを保つことも必要不可欠となっている．

それらのセキュリティシステムの１つに，ネットワーク侵入検知システム（NIDS: Network Intrusion Detection System）がある．これは，サービスを提供するシステムとネットワークの接点付近に置くことで．そのシステムへの攻撃を検知し警告を発することができるものである．NIDS の攻撃検知の原理は，ルールセットに記述されている既知の攻撃パターンにマッチするものを，システムに対する通信から見つけ出すというものである．そのため，NIDS はパケットのヘッダだけでなく，ペイロード中のデータもパターンマッチングによって検査しなくてはならない．これが NIDS における処理の大半を占めている．

一般的な NIDS の代表例に Snort [1] があるが，ソフトウェアで実装されており，パターンマッチングの手法は進歩しているものの，高速なネットワークに対して十分な速度性能が得られていない．速度を満足できなければ，システムに対する通信のすべてを検査することができず，攻撃検知の精度が低下する．

そこで，十分な速度性能を得るため，専用ハードウェアとソフトウェアを協調させた NIDS を実装する試みがなされている[2)–10)]．これは，専用ハードウェアを用いてマッチングを行い，その結果をもとにソフトウェアがアクションを起こすというものである．しかしながら，パターン数が膨大になるにつれ，検知に必要な回路の規模も大きくなり，ハードウェアに実装できる容量を超えてしまうという問題がある．

一般的にパターンマッチング回路においては，パターン文字列[⋆1]それぞれに対して検出す

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba

⋆1 本論文における文字とは，一般的なアルファベットや記号のみでなく，制御コードなども含む 1 バイトのデータのことを指す．

る回路が必要である．そしてパターン集合中には複数現れる部分文字列が存在するため，その文字列を検出する回路がパターンマッチング回路全体に重複して現れる．この重複する回路を縮小できれば，パターンマッチング回路全体の回路規模を縮小できると考えられる．

本論文では，高いスケーラビリティを持つ文献 2) のアーキテクチャをベースに，辞書圧縮の概念を用いて同じ部分文字列を検出する回路を共有するアーキテクチャを提案する．まず，文献 2) のアーキテクチャについて簡単に述べた後，本論文における実装について述べる．次に，辞書圧縮の概念を用いたパターンマッチングアーキテクチャについて述べ，そのアーキテクチャに従って回路を構成するために考案したアルゴリズムについて述べる[⋆1]．最後に，FPGA 用の論理合成ツールによって評価を行い，その結果についての考察を述べる．

2. パターンマッチングアーキテクチャの基本構成

非決定性有限オートマトン（NFA）でパターンマッチを行うアーキテクチャの方式として，前のステートの発火信号と入力文字に対するマッチ信号を次のステートの発火条件としてステートの発火を伝えていくものがある[3)]．このようなステートマシンを NFA ステートマシンと呼ぶことにする．さらに，各文字と入力文字とのマッチ信号を，1 ビット信号にデコードした状態でパイプライン的に伝播させることでバッファし，必要なタイミングで複数のパイプラインから発火信号を得て文字列のマッチングを行う方式がある[5)]．文献 2) のアーキテクチャはこれらの考え方を融合させて用いており，文献中でハイブリッドアーキテクチャと呼ばれている．これは，**図 1** に示すように，文字ごとの発火信号をパイプライン化したものと，その信号を用いて遷移する NFA ステートマシンから構成されている．ハイブリッドアーキテクチャはステート遷移に用いる入力信号の幅を任意に設定できるという特長があり，高いスケーラビリティを得ることができる．これにより，**図 2** のようにパイプラインと NFA ステートマシンを入力バイト幅と同じ数だけ用意することで，複数バイト幅入力のときにも正規表現のパターンマッチングを可能にしている．

本論文におけるパターンマッチングアーキテクチャでは，このハイブリッドアーキテクチャをアーキテクチャのベースとしている．それは，文献 9) で示されているように，このアーキテクチャが実装方式などの工夫や効率化によって，高い性能とスケーラビリティを兼ね備えることが可能であるためである．また，NIDS におけるパターンマッチング回路の実

⋆1 パターン集合はかなり膨大であるため，人の手で回路を構成するのは現実的ではない．そのため，NIDS のルールセットからハードウェア記述言語のソースファイルを自動生成することで回路を構成する．

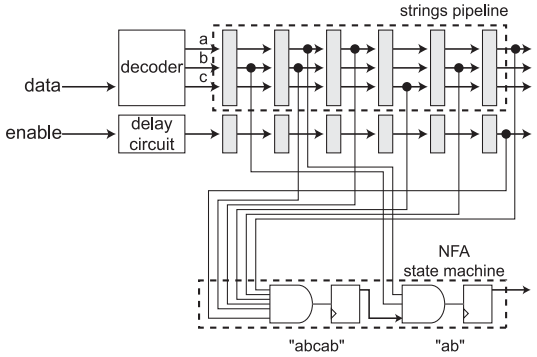

図 1　1 バイト入力ハイブリッドアーキテクチャに基づく回路の例
Fig. 1　The example of the circuit which is based on Hybrid Architecture on 1 byte width input.

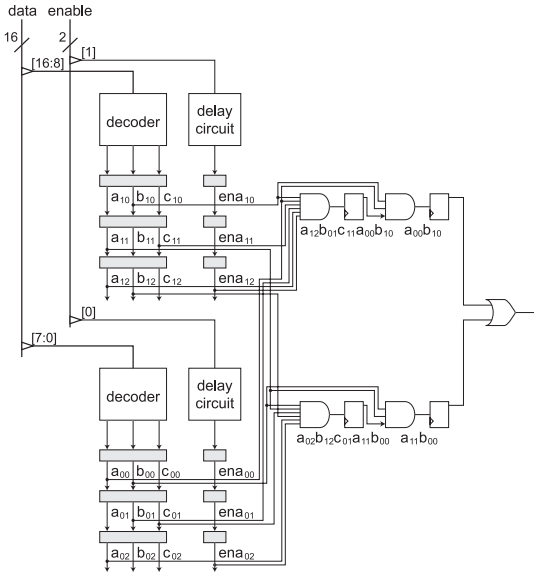

図 2　2 バイト入力ハイブリッドアーキテクチャに基づく回路の例
Fig. 2　The example of the circuit which is based on Hybrid Architecture on 2 byte width input.

現においては，パターンの更新に対応できるようにするために，回路の書き換えが可能な FPGA デバイスを用いることが一般的である．ハイブリッドアーキテクチャでは，ステート遷移に用いる入力信号の幅を任意に設定できるため，FPGA の LUT（Look Up Table）の入力線を効率良く使用することができるという利点もある．

2.1　ハイブリッドアーキテクチャの実装方式

ここでは，本研究で対象とするハイブリッドアーキテクチャに関する具体的な実装の考え方と工夫について述べる．

本論文において対象とするのは，Snort の検知ルールのうち，content・uricontent オプションに記述されている単純な文字列集合とのパターンマッチングである．したがって，正規表現のうち選択や閉包といった規則は対象外となり，各ステート遷移の論理は AND のみとなる．また，指定した content・uricontent オプションに記述した文字列に対して，アルファベットの大文字小文字を区別しないことを示す nocase オプションが存在する．これに対して本論文では，入力された文字が大文字でも小文字でも信号が発火するような専用の出力線をデコーダに用意することで対応する．また，入力のどの位置でマッチングを開始するかを指定する機能を設け，enable 信号を指定位置で入力することにより制御できるようにする．

2.1.1　NFA ステートマシン

1 つのステート遷移に用いる入力文字の信号線が多ければ多いほど，ステートは少なくなり，回路規模は小さくなると考えられる．しかし，1 つの AND ゲートで実現できる AND 演算の入力数は一般的にある数に決まっており，その数を超える入力数で AND 演算を行う場合にはゲートを連結させなければならない．これにより，ゲート遅延が増大することが考えられる．そのため，LUT の連結が発生しない範囲内で可能な限り多くステート遷移に用いる入力文字の信号線を配線する．本論文では，LUT への入力が 6 本である最新 FPGA の Xilinx Virtex-5 [11] への実装を想定し，ステート遷移に入力文字の信号線を 5 本配線することとした．

2.1.2　デ コ ー ダ

本論文におけるデコーダには，文献 6)，7) で提案された比較器を用いる．この比較器は，8 ビット信号を上下 4 ビットずつ比較し，その結果の AND をとる方式であり，FPGA における実装では LUT の連結段数を低く抑えることができる．また，パイプライン化することで LUT の連結を防ぐことができる．nocase オプションに対応する出力については，大文字に対する上位 4 ビットの比較結果と小文字に対する上位 4 ビットの比較結果の OR をとり，

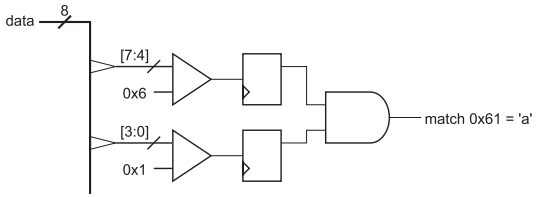

図 3　文献 6) で提案された比較器
Fig. 3　The comparator suggested in the bibliography 6).

その出力と下位 4 ビットの比較結果の AND をとる．FPGA は，入力ビットが LUT の入力線数以下で収まり出力が 1 ビットであるような組合せ回路を 1 つの LUT で構成することができるため，このように構成しても LUT の連結は発生しない．**図 3** に回路構成を示す．

2.1.3　文字列パイプライン

本論文では，パイプラインの実装に FlipFlop（FF）を連結したシフトレジスタを用いる．パイプラインレジスタは，各文字に対してそれぞれ必要な長さのみ実装する．

3.　辞書圧縮の概念を用いたアーキテクチャ

パターン集合中には，複数現れる部分文字列が存在する．**図 4** の例では文字列「ab」が重複して現れている．一般的にパターンマッチング回路においては，パターン文字列それぞれに対して検出する回路が必要である．そしてパターン集合中には複数現れる部分文字列が存在するため，その文字列を検出する回路がパターンマッチング回路全体に重複して現れる．これらの部分文字列を検出する回路の重複は大きな無駄であると考えられ，重複する回路を縮小することができれば回路規模を縮小することができると考えられる．

この重複する回路を縮小するアプローチは多数考えられるが，本論文では辞書圧縮の概念を用いる．辞書圧縮はデータ圧縮技術としてよく知られており，多く出現するデータの並びを特定のワードと 1 対 1 対応するように辞書に登録し，そのデータの並びを辞書のワードに置き換えることによって全体のデータサイズを縮小するというものである．この概念を用いパターン集合中の任意の位置に複数表れる部分文字列を辞書に登録して 1 つの記号に置き換えると，部分文字列それぞれに対する回路が 1 記号分になり，必要な論理ゲートを少なくすることができる．

本論文の辞書圧縮の概念を用いたアーキテクチャを Compress アーキテクチャと呼ぶことにする．

パターン 1: ”**abcabab**”
パターン 2: ”cb**ab**acbb**ab**b”
…

図 4　部分文字列が重複して現れるパターン集合の例
Fig. 4　The example of the set of patterns which has partial strings.

3.1　辞書圧縮の概念を用いたパターンマッチング回路に求められる要件

パターンマッチング回路では，パターン検出信号を他の回路やソフトウェアが入力として使用する場合が想定される．このとき，文字列が入力されてから検出信号を出力するまでの時間がパターンや入力によって違うと，使用できない場合や，それらに余計な動作を強いる必要がある場合が考えられ，汎用性に欠ける．このことから，パターンマッチング回路は，文字列が入力されてから検出信号を出力するまでの時間がすべてのパターンや入力で一定であることが望ましい．

パターンマッチング回路は，入力の幅を大きくして並列化を行うことによりスループットの向上を図る．このことから，入力の幅が大きくなるにつれて圧縮率が低くなるようなことがあってはならない．

3.2　Compress アーキテクチャ

本論文では，前述したように，高い汎用性と複数バイト幅入力時の高圧縮率を達成するため，ハイブリッドアーキテクチャをベースにする．ハイブリッドアーキテクチャでパターン集合がすべて単純な文字列であるとき，検出すべき文字列が入力されてから検出を示す信号を出力するまでのクロック数はすべてのパターンで同一である．Compress アーキテクチャでもこの性質を損なわないように設計することで，高い汎用性を確保する．

Compress アーキテクチャでは，入力文字列中の辞書に登録されている部分文字列を検出する回路と，その信号を流すパイプラインを構成する．このパイプラインに流れる信号を NFA ステートマシンに入力すると，信号線 1 本で部分文字列の検出を行うことができる．これにより NFA ステートマシンに入力する信号線の数が減り，NFA ステートマシンのステート数を削減することができる．5 文字以上の部分文字列が辞書に登録された場合，回路構成によっては検出回路において LUT の連結が発生する可能性がある．そこで本論文では，ハイブリッドアーキテクチャの NFA ステートマシンの回路構成を検出回路に応用することで，LUT の連結が発生しない状態で回路を構成可能にする．

ここで，単純にハイブリッドアーキテクチャに部分文字列検出回路を追加して信号をパイプラインに流すと，検出信号が対応する部分文字列の末尾より 1 クロック遅れてしまう．このクロック遅れが発生すると，検出すべき文字列が入力されてから一定クロック後に検出されなくなったり，圧縮を行えない部分が存在したりする弊害が生じる．この問題に対応するため，文字列パイプラインにフォワーディングステージを設けることにより，部分文字列の検出信号のクロック遅れを隠蔽する．部分文字列検出回路をカスケード接続する場合には，その段数+1 だけのフォワーディングステージが必要である．本論文では，部分文字列検出回路のカスケード接続が起きないように回路を構成し，フォワーディングステージの段数を 1 段とする．

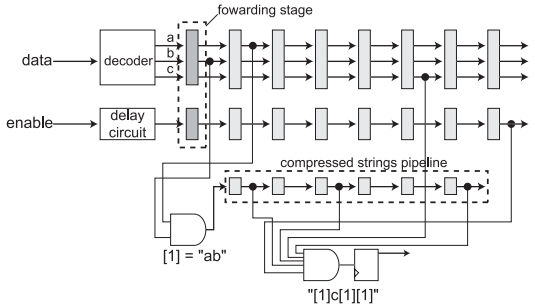

図 5　1 バイト入力 Compress アーキテクチャに基づく回路の例
Fig. 5　The example of the circuit which is based on Compress Architecture on 1 byte width input.

図 5 に「abcabab」を検出する 1 バイト入力の回路構成の例を示す．部分文字列「ab」に対応するパイプラインから NFA ステートマシンに入力することにより，2 文字分の検出を 1 本の信号線を用いるのみで行うことができる．これにより，パターン全体の検出を 5 本の信号線で行うことができ，結果として NFA ステートマシンのステート数が図 1 より 1 つ少なくなる．

複数バイト幅入力の場合にはハイブリッドアーキテクチャと同様に回路を構成することによって対応することができる．複数バイト幅入力のときには，文字の種類が入力バイト幅倍になり，部分文字列の種類も入力バイト幅倍になるが，NFA ステートマシンも入力バイト幅と同じだけ用意される．そのため部分文字列の出現数が変わることはなく，圧縮率が落ちることはないと考えられる．

3.3　回路圧縮アルゴリズム

Compress アーキテクチャに基づいて回路を構成するために，本論文で実装する回路圧縮アルゴリズムについて述べる．

```
while true do
  文字または辞書の組の出現数をカウント;
  降順ソート;
  if 閾値以上の出現数の組が無い then
    break;
  end if
  for all 閾値以上の出現数の組 do
    対応する辞書を生成;
  end for
  for all 生成した辞書 do
    for all パターン集合 do
      対応する文字または辞書の組を辞書に置換;
    end for
  end for
end while
```

図 6　replace フェーズの擬似コード
Fig. 6　The pseudo code of replace phase.

最適な圧縮を行う問題は NP 困難であるため，本論文ではこの問題に対するヒューリスティクスを考案した．本論文の回路圧縮アルゴリズムは 2 つのフェーズに分かれている．まず 1 つ目のフェーズは，パターン集合中において出現頻度の高い部分文字列を辞書に登録し，パターン集合中の該当部分を辞書記号に置き換えるものである．これを「replace フェーズ」と呼ぶ．次に 2 つ目のフェーズは，辞書それぞれにおいて，その辞書による回路規模縮小効果を最適化するものである．これを「reduce フェーズ」と呼ぶ．

3.3.1　replace フェーズ

replace フェーズでは，部分文字列を辞書に登録し，該当部分を辞書記号に置き換える．本論文における replace フェーズでは貪欲法のモデルを採用し，辞書記号も文字として扱いパターン集合全体において文字または辞書の組の出現数を数え上げ，閾値以上の回数現れなくなるまで段階的に辞書記号に置き換えていく．文字の組のどちらかに辞書記号が含まれている場合には，辞書部分をその辞書記号が示す文字列に還元してから登録することで，部分文字列検出回路のカスケード接続を防ぐ．**図 6** に擬似コードを示す．

本論文の NFA ステートマシンでは，1 つのステートで最大 5 つの文字または辞書記号のマッチを検出する．よって，1 つの辞書記号がパターン集合中に 5 回現れたとき，その辞書は 1 つ分のステートを削減すると考えることができる．本手法では，辞書それぞれにパイプラインを用意するため，必ず 1 つ以上の FF を使用する．以上のことから，辞書記号がそれぞれ 5 回以上現れなければ回路を削減する可能性はないといえるため，本論文では閾値を 5 に設定した．

```
辞書を対応する文字列の長い順にソート;
最適状態に現在状態をコピー;
for all 辞書 do
  ステージ番号降順ソート;
  for all ステージ do
    if 辞書の使用数が閾値未満 then
      if 最適回路削減効果 <= 0 then
        現在の辞書記号をすべて還元;
        現在の辞書を削除;
      end if
      break;
    end if
    if 辞書の回路削減効果更新 then
      現在状態を最適状態に保存;
    end if
    現在のステージに対応する辞書を還元;
  end for
end for
```

図 7　reduce フェーズの擬似コード
Fig. 7　The pseudo code of reduce phase.

3.3.2　reduce フェーズ

reduce フェーズでは，辞書によるステート削減数と回路コストのトレードオフを計算し，辞書が最大の回路削減効果を得られる状態を探索して最適化する．**図 7** に擬似コードを示す．

パイプラインには，その出力線が NFA ステートマシンに入力されるレジスタと，次のステージのレジスタに入力されるのみで NFA ステートマシンに入力されないレジスタがある．仮に前者のレジスタをホワイトレジスタ，後者のレジスタをブラックレジスタとすると，ブラックレジスタは回路規模の観点から大きな無駄であるといえる．しかし，ブラックレジスタより後方のステージに 1 つでもホワイトレジスタがある場合，ホワイトレジスタに信号を伝送しなければならないため，ブラックレジスタを削除することができない．replace フェーズによって圧縮されたパターン集合では，圧縮文字列パイプラインに連続したブラックレジスタを大量に持つため，辞書の回路削減効果が低下している．また，ブラックレジスタを少ししか持たない辞書であっても，その辞書記号の出現数が少なく，回路コストに回路

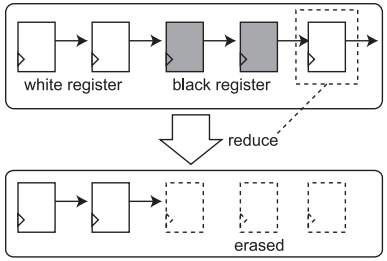

図 8　ホワイトレジスタの削除によるブラックレジスタの削除例
Fig. 8　The example of erasing the black registers by erasing the white register.

削減効果が見合っていない場合がある．このような辞書は無駄であるため，削除する必要がある．

ブラックレジスタが存在するのは後方にホワイトレジスタが存在するためであり，後方のホワイトレジスタを削除すれば前方のブラックレジスタを削除できる．このような例を**図 8** に示す．圧縮文字列パイプラインにおけるホワイトレジスタを削除するためには，圧縮パターン集合の対応する辞書記号を元の部分文字列に相当する別の形で表す必要がある．そこで，ホワイトレジスタに対応する辞書記号を，後方から順に元の文字の組に還元する．辞書記号を文字の組に還元すると辞書記号の出現数が減り，パターン集合の圧縮率が低下するデメリットはあるが，ブラックレジスタを削除することができるメリットが生じる．これはトレードオフの関係であることから，最も辞書の回路削減効果が高くなる状態を線形探索することで最適化を行う．また，最適化の段階で，辞書が回路を削減する見込みがないような場合には，その辞書記号をすべて還元し，辞書を削除することも行われている．

ある辞書が表す部分文字列の中に他の辞書が表す部分文字列が含まれていることがある．もし後者の辞書を先に最適化した場合，前者の辞書を最適化した際に後者の辞書記号が現れてしまい，後者の辞書の最適性を損なう．そのため，この探索は，対応する文字列の長い辞書から行わなくてはならない．

4. 評　　価

ハイブリッドアーキテクチャと Compress アーキテクチャに基づく回路をそれぞれ構成し，比較することで Compress アーキテクチャの性能を評価する．本論文においては，入力バイト幅 1〜3 のときについてそれぞれ回路を構成する．

4.1 評 価 環 境

本論文では，論理合成ツールを用いて評価を行う．論理合成ツールは Xilinx ISE 9.2.04i –xst J.40，ターゲットデバイスは Xilinx Virtex-5（xc5vlx220t-1-ff1738）を用いた．論理合成ツールのオプションはデフォルトである．

使用するルールセットは Snort2.8 のシグネチャ（更新日：2008 年 5 月 1 日）であり，content・uricontent オプションに記述されているパターンの総数は，重複を取り除いて 6,229 個（100,666 文字）である．

4.2 評 価 結 果

ハイブリッドアーキテクチャと Compress アーキテクチャそれぞれにおいて回路を構成する．用いるパターン集合は，全パターンから無作為に 1,024 個，2,048 個，4,096 個抽出したものと，全パターンのものを用いる．スループットには，論理合成によって求められた最大動作周波数[⋆1]にそれぞれの回路構成の入力ビット幅を掛けたものを用いる．**表 1** に評価結果を示す．

5. 考　　察

4.2 節の結果について，回路規模とスループットの観点から考察する．

5.1 回 路 規 模

表 1 から，全パターンに対して回路を構成したとき，Compress アーキテクチャはハイブリッドアーキテクチャに比べ，1 バイト入力のとき 8.12%，2 バイト入力のとき 12.67%，3 バイト入力のとき，16.57%の回路規模縮小に成功していることが分かる．このことから，Compress アーキテクチャは，入力バイト幅によらず高い回路規模縮小効果が得られることが実証された．さらに高い圧縮率を実現するアルゴリズムで圧縮を行えば，さらなる回路規模縮小効果を得られると予想される．また，パターン集合が小さいときにハイブリッドアーキテクチャより Compress アーキテクチャの回路規模が大きくなっているのは，フォワーディングステージのコストが圧縮による回路規模縮小効果よりも大きいためであると考えられる．

5.2 デバイス使用率

本論文における回路構成では，使用したデバイスの 50% 程度を使用した．本論文で使用

⋆1 論理合成では，配線長の影響を仮想的に反映した回路遅延が出力され，これは対象デバイスにおけるほぼ最大値であると考えられる．回路の規模によっては，論理合成で求められた配線遅延よりも非常に大きな配線遅延がかり，動作周波数が大きく落ちることがある．

表 1　評価結果
Table 1　The evaluation result.

			ハイブリッドアーキテクチャ			Compress アーキテクチャ		
input width	#pattern	#char	#Bit Slice	Frequency	Throughput	Bit Slice	Frequency	Throughput
1 Byte	1,024	9,557	7,605	580.720	4,646	7,798	577.034	4,616
	2,048	26,435	13,778	577.034	4,616	13,713	577.034	4,616
	4,096	49,947	20,710	575.374	4,603	20,168	575.374	4,603
	6,229	100,666	32,172	568.182	4,545	29,560	574.053	4,592
2 Byte	1,024	9,557	9,627	577.034	9,233	10,080	577.034	9,233
	2,048	26,435	18,315	574.053	9,185	18,506	577.034	9,233
	4,096	49,947	30,923	553.499	8,856	29,313	575.374	9,206
	6,229	100,666	53,802	547.104	8,754	46,985	566.572	9,065
3 Byte	1,024	9,557	11,041	575.374	13,809	11,808	577.034	13,849
	2,048	26,435	21,758	575.374	13,809	21,927	575.374	13,809
	4,096	49,947	38,896	552.084	13,250	36,913	577.034	13,849
	6,229	100,666	72,659	550.339	13,208	60,619	551.915	13,246

Frequency: MHz　Throughput: Mbps

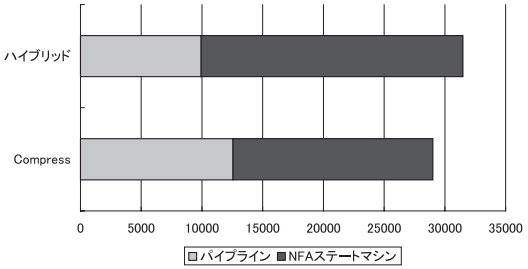

図 9　1 バイト入力のときの論理記述上のパイプラインと NFA ステートマシンの FF 数
Fig. 9　The number of FF in Pipeline and NFA State Machine on the logical description on 1 byte width input.

したデバイスは中位から上位クラスのものである．

パターン集合の大きさが最上位クラスのデバイスに収まりきれないほどである場合，パターン集合を複数のデバイスに分割して実装することが想定される．この際，部分文字列の出現数によってクラスタリングし，本手法の効果が大きくなるような分割を行う必要があると考えられる．

5.2.1　各部の回路規模

ハイブリッドアーキテクチャと Compress アーキテクチャでは各部の回路規模がどのように違うのかを検証する．論理合成ツールは全体の回路規模しか求めることができないため，ここでは論理記述上の FF の数を計数することによって調べた．これは，FPGA の内部構造が LUT と FF の組から構成されているため，使用数の多い方が回路規模を決定するからである．**図 9** に，パイプラインと NFA ステートマシンに用いられている論理記述上の FF の数のグラフを示す．なお，パターン集合は 6,229 個すべてであり，入力バイト幅は 1 バイトである．グラフから，圧縮によりパイプラインに要する回路規模は増大しているが，NFA ステートマシンの回路規模がそれ以上に縮小されることにより，全体の回路規模が縮小されていることが分かる．

5.2.2　回路規模増大傾向

回路規模についてグラフにプロットし，パターン集合の総文字数に対する回路規模の増大傾向について調べた．**図 10** にグラフを示す．グラフ中の凡例のカッコ内は入力バイト幅を示している．グラフから，Compress アーキテクチャは，回路規模の増大傾向を抑えられていることにより，回路規模を削減していることが分かる．このことから，さらにパターン集合が大きくなれば，さらなる回路規模削減効果を期待できると考えられる．

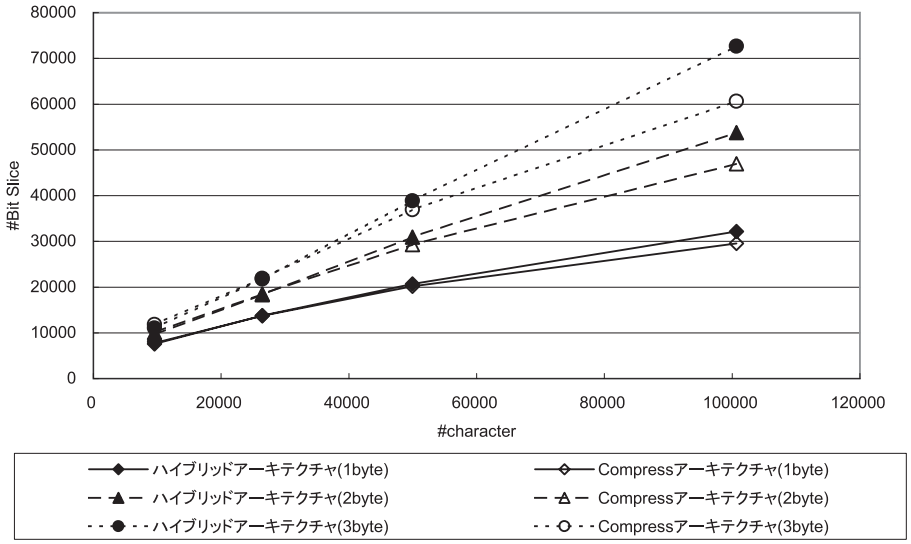

図 10　回路規模のグラフ
Fig. 10　The graph of the size of circuits.

5.2.3　パターン集合中の部分文字列の出現数分布

本論文で用いたパターン集合それぞれにおいて，部分文字列の出現数分布を調べる．部分文字列の一部が別の部分文字列の一部に重なることを許し，パターン集合において文字列長が 2 の部分文字列が出現する回数を計数した．**図 11** に度数分布を示す．

図 11 から，パターン集合が大きくなるにつれて，部分文字列の出現数の偏りが大きくなっていることが分かる．このことから，パターン集合中における部分文字列の出現数の偏りが，本手法の回路規模縮小効果に影響しているといえる．

5.3　スループット

表 1 から，ほとんど差はないことが分かる．これは，Compress アーキテクチャの設計がハイブリッドアーキテクチャと同様に最小限の LUT 連結段数で構成されているためである．

このことから，Compress アーキテクチャはスループットのペナルティなしに回路規模の削減を行うことができているということができる．

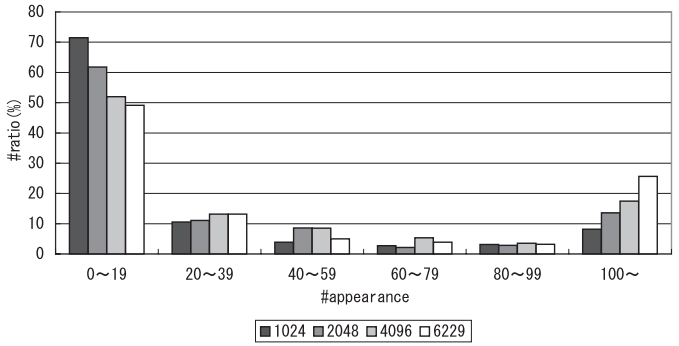

図 11　部分文字列の度数分布
Fig. 11　The frequency distribution of substrings.

6. 関 連 研 究

論文 10) では，部分文字列を検出する回路の重複を縮小するために，パターン中で同じ位置に現れる共通部分文字列に対する検出回路のみを共有化している．また，論文 8) で提案されている手法では，本論文と同様に辞書圧縮の概念を用いている．しかしながら，3.1 節で定義する要件のうち，2 番目を満たさないうえ，LUT の連結が発生するため，スループットが低くなる．

7. 今後の課題

7.1　配置配線による評価

本論文では，評価に配置配線ではなく，論理合成の結果を用いている．FPGA では，配置配線を行うことで，実際にチップに実装する際の動作周波数を求めることができる．しかしながら，予備実験を行った結果，配置配線には非常に大きな時間がかかるうえ，入出力ピン設定やタイミング制約などによって結果が大きくばらつき，回路規模が大きい場合においては配線遅延を抑制する別の工夫が必要であることが分かった．そのため，今回は配置配線による評価は行わなかった．今後，結果のばらつきと，回路規模が大きい場合の配線遅延を抑える手段を確立したうえで，配置配線による評価を行う予定である．

7.2　シフトレジスタ LUT の活用

FPGA には，シフトレジスタを LUT を用いて構成するシフトレジスタ LUT と呼ばれる

機能がある．かつては動作が非常に低速であったが，近年では非常に高速に動作するものがある．本手法において使用すれば，さらなる回路規模縮小効果が得られると期待されるが，圧縮アルゴリズムが複雑化するため，今回は使用を見送った．今後使用可能な圧縮アルゴリズムを考案し，実装評価する予定である．

8. おわりに

NIDS 向けのパターンマッチング回路の回路規模を縮小するため，辞書圧縮の概念を用いたパターンマッチングアーキテクチャを提案した．また，これに基づいて回路を構成するための回路圧縮アルゴリズムを考案し，パターンマッチング回路を実装した．

本論文で実装したパターンマッチング回路はベースとしたパターンマッチング回路から，入力バイト幅 1〜3 のときにスループットを保ったまま 8.12%〜16.57%の回路規模縮小に成功した．さらに大きなパターン集合や圧縮率の高い圧縮アルゴリズムで回路を構成すれば，さらなる高い回路規模縮小効果を得られると考えられる．

今後の課題として，配置配線による評価，シフトレジスタ LUT の活用があげられる．

謝辞　多くの貴重な知見をいただいた査読者の皆様に，深く感謝いたします．

参 考 文 献

1) Roesch, M.: Snort — Lightweight Intrusion Detection for Networks, *Proc. 13th Large Installation System Administration Conference* (*LISA'99*), Seattle, Washington, USENIX, pp.229–238 (1999).
2) Moscola, J., Cho, Y.H. and Lockwood, J.W.: A Scalable Hybrid Regular Expression Pattern Matcher, *Proc. 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (*FCCM'06*), pp.337–338, IEEE (2006).
3) Sidhu, R. and Prasanna, V.K.: Fast Regular Expression Matching Using FPGAs, *Proc. 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (*FCCM'01*), pp.227–238, IEEE (2001).
4) Clark, C.R. and Schimmel, D.E.: Scalable Pattern Matching for High Speed Networks, *Proc. 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (*FCCM'04*), pp.249–257, IEEE (2004).
5) Baker, Z.K. and Prasanna, V.K.: A Methodology for Synthesis of Efficient Intrusion Detection Systems on FPGAs, *Proc. 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (*FCCM'04*), pp.135–144, IEEE (2004).
6) Katashita, T., Maeda, A., Toda, K. and Yamaguchi, Y.: Highly Efficient String Matching Circuit for IDS with FPGA, *Proc. 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (*FCCM'06*), pp.285–286, IEEE (2006).
7) Katashita, T., Maeda, A., Toda, K. and Yamaguchi, Y.: A Method of Generating Highly Efficient String Matching Circuit for Intrusion Detection, *Proc. International Conference on Field Programmable Logic and Applications* (*FPL*), pp.799–802, IEEE (2006).
8) 小野正人，片下敏広，前田敦司，山口喜教：データ圧縮技術による NFA パターンマッチング回路の効率的実現手法，信学技報，Vol.105, No.226, pp.37–42 (2005).
9) 飯星貴裕，山口喜教，前田敦司：NIDS 向け NFA ハイブリッドパターンマッチング回路の効率化，信学技報，Vol.108, No.180, pp.1–6 (2008).
10) Sourdis, I., Bispo, J., Cardoso, J.M.P. and Vassiliadis, S.: Regular Expression Matching in Reconfigurable Hardware, *Int. Journal of VLSI Signal Processing Systems*, pp.99–121, Springer (2007).
11) Xilinx Inc.: Virtex-5 ファミリ概要 (2008).

(平成 21 年 1 月 27 日受付)
(平成 21 年 4 月 23 日採録)

飯星　貴裕（学生会員）

2008 年筑波大学第三学群情報学類卒業．学士（情報工学）．同年筑波大学大学院システム情報工学研究科入学．現在筑波大学大学院システム情報工学研究科在学中．主として論理回路設計，アルゴリズム設計，ハードウェア・ソフトウェア協調処理に関する研究に従事．

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）．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．

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）．電気通信大学大学院助手，筑波大学講師．筑波大学大学院助教授を経て現在筑波大学大学院システム情報工学研究科准教授．システムプログラム，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．