社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－133（11）
2008／1／16

パターンを実行時に設定可能な正規表現ストリングマッチングマシンとFPGAによる実現

川中 洋祐† 若林 真一† 永山 忍†

† 広島市立大学 大学院 情報科学研究科
〒731-3194 広島市安佐南区大塚東3-4-1

あらまし ストリングマッチングとは，文字や文字に対する演算からなる文字列をパターンとし，与えられたパターンに一致する文字列を入力系列から検索する操作である．本論文では，正規表現のサブクラスをパターンとするストリングマッチングに対して専用ハードウェアを提案し，FPGA上に実現して評価を行う．提案ハードウェアは従来のハードウェアとは異なり，ストリングマッチングの実行時にパターンの設定が可能という大きな利点がある．
キーワード ストリングマッチング，正規表現，FPGA

A Regular Expression String Matching Machine Allowing Pattern Setting During Execution Time and Its FPGA Implementation

Yosuke KAWANAKA†, Shin'ichi WAKABAYASHI†, and Shinobu NAGAYAMA†

† Graduate School of Information Sciences, Hiroshima City University
3-4-1, Ozuka-higashi, Asaminami-ku, Hiroshima 731-3194, Japan

Abstract In this paper, a high-speed string matching circuit for searching a pattern in a given text is proposed. In the circuit, a pattern is specified by a class of restricted regular expressions. The main feature of the proposed hardware, which is distinct from previous string matching circuits, is that a pattern to be retrieved can be set during the execution time. The proposed hardware was implemented on an FPGA chip, and its effectiveness was evaluated.
Key words String Matching, Regular Expression, FPGA

1. はじめに

ストリングマッチングとは，文字や文字に対する演算からなる文字列をパターンとし，与えられたパターンに一致する文字列を入力系列から検索する操作である[1]．ストリングマッチングの主な応用としてはデータベースにおけるデータ検索などがあり，これまでに多くの研究が行われてきている．本研究では，正規表現のサブクラスがパターンとして与えられるストリングマッチングに注目する．

正規表現をパターンとするストリングマッチングは，近年，高速ネットワークのネットワーク侵入検知システム（NIDSs）のようなアプリケーションとして注目されている[3]．しかしながら，NIDSに対してソフトウェアでストリングマッチングを実現した場合，莫大な実行時間がかかってしまうため実用的ではない．このため，近年，ハードウェア上で効率よく実現するためのストリングマッチングアルゴリズムが研究されている．

ストリングマッチングをFPGAを用いてハードウェアで実現した場合，ソフトウェアで実現した場合に比べて実行時間の点で極めて有利だが，同時に一つの大きなの問題点が存在する．もしパターンが変更された場合，ハードウェアのHDL記述の生成，論理合成，配置配線，コンフィギュレーションファイルの生成といったFPGAにおける設計・実装工程をもう一度実行しなければならない．このFPGAの工程には少なくとも5分から通常は30分以上かかるため，ストリングマッチングのいくつかのアプリケーションに対しては致命的な欠点となる．例えば，ストリングマッチングを行うたびにパターンが変更されるデータベースシステムにおけるテキスト検索等においては，FPGAによるハードウェアでの実装は困難となる．

本論文では，任意のパターンを実行時に設定可能なストリングマッチング専用ハードウェアの提案と，そのFPGA上での実現について述べる[4]．提案ハードウェアは，正規表現のサブクラスをパターンとして設定可能であり，1文字単位の比較を行うセルと呼ばれる回路を1次元配列状に並べたシストリックアルゴリズム[5]で構成される．パターンは，ストリングマッチングを行う前に各セルに1文字ずつ記憶され，入力系列を1文字ずつセルに入力していくことでストリングマッチングを行う．

提案ハードウェアは，ハードウェア設計時にパターンを設定しなければならない従来のストリングマッチングハードウェアと異なり，ストリングマッチングの実行時にパターンの設定が可能であるという大きな利点がある．このため，提案ハードウェアはその場でパターンを変更しなければならないようなアプリケーションに適しており，そのようなアプリケーションにデータベースシステムにおけるテキスト検索などが挙げられる．さらに，NIDS に対してもこのハードウェアを使用することで，パターンの迅速な更新が可能になり，システムのセキュリティをより高めることが可能になる．

本論文の構成は以下の通りである．2 章ではパターンを定義するための言語クラスである正規表現とそのサブクラスの定義について述べ，3 章では提案ハードウェアアルゴリズムとその回路構成を，4 章では実験的評価を述べる．最後の 5 章は本論文のまとめとなっている．

2. 正規表現

ストリングマッチングにおいては，パターンとして単純な文字列だけでなく，形式言語に基づくパターンを用いることが一般的である．この場合，パターンを定義するための形式言語クラスとして，正規表現やそのサブクラスが使われることが多い．正規表現は受理アルゴリズムが簡潔で，パターンを定義するための言語としては実用上，十分な表現能力を持つ．正規表現が表す文字列の集合正規集合 (あるいは正規言語) と言う [8]．本論文においても，正規表現のサブクラスでパターンを定義するものとする．

2.1 定　　義

Σ を記号の有限集合 (アルファベットと呼ぶ) としたとき，Σ 上の正規表現は以下のように再帰的に定義される [8]．なお，正規表現特有の記号，ε, ϕ, (,), +, $*$ は Σ に含まれないものとする．

[定義 1]　ϕ を，空集合を意味する Σ 上の正規表現とする．

[定義 2]　ε を，$\{\varepsilon\}$(空語) を意味する Σ 上の正規表現とする．

[定義 3]　a を，$\{a\}$(ただし，$a\in\Sigma$) を意味する Σ 上の正規表現とする．

[定義 4]　R と S をおのおの Σ 上の正規集合 R, S を意味する正規表現としたとき，

（1）　$(R+S)$ を，R∪S を意味する Σ 上の正規表現とする．

（2）　(RS) を，R・S=$\{w=xy|x\in$R, $y\in$S$\}$ を意味する Σ 上の正規表現とする．

（3）　$(R)*$を，R*=$\{\varepsilon\}\cup R^1\cup R^2\cup\cdots$ を意味する Σ 上の正規表現とする．

[定義 5]　4 において各表現には ( ) をつけて定義したが，演算子間の優先順位を以下の数字の小さいのものほど強いと決めれば，普通の数式と同様に括弧は省略できる．

（1）　*　　(クリーネの閉包演算子)

（2）　・　　(連接，通常は表記しない)

（3）　+　　(和)

[定義 6]　以上の 1 から 3 の正規表現から出発して，4 における演算を有限回施して得られる表現のみが Σ 上の正規表現である．

なお，1 つの正規集合を表す正規表現は上記の定義に従った範囲内でも一般に幾通りもあり得る．例えば，正規集合 $\{0,1\}*$ を表す正規表現としては $(0+1)*$, $(0*1*)*$ などが存在する．

ここで，正規表現 R の表す正規集合を L(R) とする．また，2 つの正規表現 R, S において，L(R)=L(S) であるとき，R と S とは等価であるといい，R=S と表す．例えば，$L((0+1)*)=L((0*1*)*)=\{0,1\}*$ であるから，$(0+1)*=(0*1*)*$ である．

2.2 正規表現のサブクラス

入力文字系列を構成する文字の集合を $\Sigma=\{a_0,a_1,a_2,\cdots,a_n\}$ としたとき，Σ 上の正規表現において，$?\notin\Sigma$ を特殊文字，-, @を単項演算子とし，正規表現のサブクラスとして次のように定義する [7]．

[定義 1]　$?=a_0+a_1+a_2+\cdots+a_n$

[定義 2]　$-a_i=a_0+a_1+\cdots+a_{i-1}+a_{i+1}+\cdots+a_n$

[定義 3]　$a^{@}=a+\varepsilon$,　　$a\in\Sigma\cup\{?\}$

[定義 4]　$-a_i^{@}=(-a_i)^{@}$

[定義 5]　$-a_i*=(-a_i)*$

[定義 6]　$?*=(a_0+a_1+a_2+\cdots+a_n)*$

[定義 7]　正規表現において，各演算子 (*, +, ・, -, @) のオペランドとなっている部分正規表現を項と呼ぶ．項が-以外の演算子を含まないとき，特にその項を単一項と呼ぶ．

[定義 8]　単一項と連接演算子・のみからなる項を C-単純項，単一項とユニオン演算子+のみからなる項を U-単純項と呼ぶ．

[定義 9]　s を単一項とする．$s*$, $s^{@}$ をそれぞれ*-単純項，@-単純項と呼ぶ．単一項および C-単純項，U-単純項，*-単純項，@-単純項をあわせて単に単純項と呼ぶ．

[定義 10]　項が単純項，U-単純項以外の単純項の連接，又は，それらのユニオンである場合，その項を UC-項と呼ぶ．

[例 1]　$ab*c+d+(-a^{@})?$

[定義 11]　正規表現 P が UC-項，又は，それらの連接である場合，P は C-展開可能であると呼ぶ．

[定義 12]　正規表現 P が C-単純項，もしくは C-単純項をオペランドとするクリーネ閉包演算の項である場合，P を K-項と呼ぶ．

[定義 13]　正規表現 P が単一項，もしくは C-単純項のユニオンをオペランドとするクリーネ閉包演算の項である場合，P を KC-項と呼ぶ．

[例 2]　$(ab+c+def)*$

?はワイルド文字 (wild character)，もしくは長さが 1 の FLDC(Fixed-Length Don't Care) と呼ばれる．また，?*は VLDC(Variable-Length Don't Care) と呼ばれる [2]．

このサブクラスでは，各演算子 (*, +, @) のネスト構造を禁止している．この制約により，正規表現のサブクラスで表現されるパターンは通常の正規言語よりも制限されるが，既存のアプリケーションでストリングマッチングを行うには十分な表現能力を持つ．

3. ストリングマッチングハードウェア

ここでは，2.2 で定義した正規表現のサブクラスをパターンと

するストリングマッチング専用ハードウェアのアーキテクチャとハードウェアアルゴリズムについて説明する [7].

3.1 アーキテクチャ

提案アルゴリズムは 1 次元配列状に CC(Comparison Cell) と呼ばれる回路を接続することで実現される (図 1). セルの左端から 1 クロックごとに入力系列 S が入力され, あらかじめ左端のセルから順番に入力されて各セルに記憶されたパターン P とマッチングを行う.

提案ハードウェアのアーキテクチャは一様構造であり, グローバル信号がクロック以外に存在しないことから, このアーキテクチャはシストリックアーキテクチャ [5] とみなすことができる.

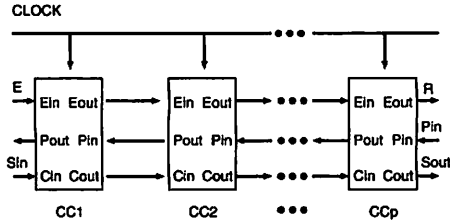

図 1 アルゴリズムのアーキテクチャ
Fig. 1 Architecture of algorithm.

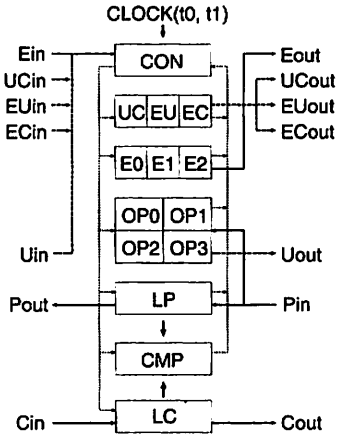

図 2 Comparison cell(CC)
Fig. 2 Comparison cell(CC).

3.2 Comparison Cell

CC とは 1 文字単位のマッチングを行う回路である. CC の動作は複雑なので, まず CC の構造とパターンに K-項と KC-項を許可しない場合の CC の動作について説明する. パターンに K-項と KC-項を許可する場合の動作については後述する.

図 2 に CC の構造を示す. CC は, 制御回路 CON, フラグ UC, EU, EC, E0, E1, E2, レジスタ OP0, OP1, OP2, OP3, LP, LC, そして比較器 CMP で構成される. CON は CC の全体の動作の制御を行う回路で, 各 CC はグローバルクロック信号に同期して動作する. UC, EU, EC は UC-項がパターンとして与えられた場合のストリングマッチングを実現するために使用されるフラグである. また, E0, E1, E2 はストリングマッチングの基本機能を実現するために使用されるフラグである. OP0, OP1, OP2, OP3 は演算子と括弧を記憶するレジスタである. OP0 は'(' と')' を, OP1 は'-' と'ε' を, OP2 は'*' と'@' を, そして OP3 は'+' をそれぞれ記憶する. また LP はパターン中の 1 文字を, LC は入力系列中の 1 文字をそれぞれ記憶するレジスタである. CMP はレジスタ LP と LC に記憶された文字を比較する比較器である.

3.3 アルゴリズム

図 3 に CC のアルゴリズムを示す. CC の動作は 2 つのフェーズで構成される. 1 つはクロック t_0 で動作する入力フェーズで, 各セルのラッチ LP に記憶されたパターンと, そのセルでのマッチング結果を右隣のセルに出力する. もう 1 つはクロック t_1 で動作するマッチングフェーズである.

各 CC は与えられた入力系列が右端の CC から出力されるまでこの動作を繰り返す. ここではパターンの入力動作については省略する. また'MOVE_INPUT_STRING' はクロックに同期して右隣の CC に入力系列をシフトすることを意味している.

```
t0: begin
      MOVE_INPUT_STRING;
      if (OP0='(') or (UCin and (OP0 ≠')'))
         then UC:=true;
      if Uin then E:=EUin or E0
         else if (UC and (OP0 ≠'(') or (OP0=')')
              then E:=ECin or E0
              else E:=Ein or E0;
      if UC or (OP0=')') then begin
         if (OP3='+') or (OP0=')')
              then E2:=Ein or E1
              else begin
                   EC:=E1;
                   if OP0 ≠'(' then E2:=Ein end end
         else E2:=E1;
      if EUin and UC then EU:=true else EU:=false;
          E0:=false;
          E1:=false
  end;
t1: begin
      if E then begin
         if OP1='-' then R:=notCMP(LP,LC)
              else R:=CMP(LP,LC)
         if R or (OP1='?') then begin
              E1:=true;
              if OP2='*' then E0:=true
         end;
         if (OP2='*') or (OP2='@') or (OP1='ε') then
              if (not UC) or (OP3='+') then E2:=true
                   else EC:=true;
         if OP0='(' then EU:=true
      end
  end;
```

図 3 各セルのアルゴリズム
Fig. 3 Algorithm of a comparison cell.

3.3.1 基本動作

CC の基本動作について簡単に説明する. いま, クロック周期 T_j において i 番目の CC(CC_i) が入力系列の k 番目の文字 (s_k) をレジスタ LC に記憶していてマッチング動作を実際に行っているものとし, その動作を図 4 に示す. CC_i におけるマッチングの結果に応じて, CC_i は他の CC にその結果を知らせ, 次のマッチングの動作をそれらの CC に行わせる必要がある. CC_i の結果に応じて他の CC が行うべきことを図 5 に示す. この図

において，△は CC_i におけるマッチングの結果，マッチした場合に限りそのクロック周期でマッチングを行うことを表している．この動作は，クロック周期 T_j でフラグ E1 を 1 にし，次のクロック周期 T_{j+1} で E1 の値を E2 へ代入することで実現される．□は CC_i のマッチングの結果に関わらず，そのクロック周期でマッチングを行うことを表している．この動作はフラグ E2 を 1 にすることで実現される．また◇は CC_i において，p_i と s_k がマッチした場合，T_{j+1} 以降の各クロック周期において，マッチングが失敗するまでマッチング動作を継続することを表している．この動作はフラグ E0 を 1 にすることで実現される．

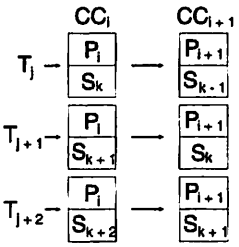

図 4　基本的なストリングマッチング
Fig. 4　Basic string matching.

T_j	T_{j+1}		T_{j+2}
CC_i	CC_i	CC_{i+1}	CC_{i+1}
P			△
P*	◇	□	△
P$^{\oplus}$		□	△
ε		□	

図 5　p^*，$p^{\oplus}$，ε のストリングマッチング
Fig. 5　String matching of p^*，$p^{\oplus}$ and ε.

フラグ E2 の値がその CC でのマッチングの結果として回路の出力端子 R より出力される．右隣の CC ではその値を enable 信号 Ein として受け取り，Ein が 1 ならばその CC でマッチングを開始する．

アルゴリズムの動作例を図 6 に示す．ここでパターンは $a^{\oplus}b^*?c$，入力系列は *abbac* とし，パターンはストリングマッチングが開始される前に記憶されているものとする．図中の記号で，○はその CC で LP と LC に記憶されている文字のマッチングを実行することを，◎はその結果マッチしたことを，×はマッチしなかったことを示す．LP と LC の文字がマッチした場合，そのときフラグ E1 は 1 となり，その値は次のクロック周期でフラグ E2 へ代入される．各 CC の E2 の出力は右隣の CC と接続されており，もし左隣の CC の E2 が 1 になった場合，その CC はマッチングを開始する．

3.3.2　K-項のマッチング

K-項のマッチングは上述の他の項に対するストリングマッチングよりもさらに複雑である．紙面の都合上，ここではアルゴリズムの基本的なアイデアのみ説明し，アルゴリズムの形式的記述は省略する．

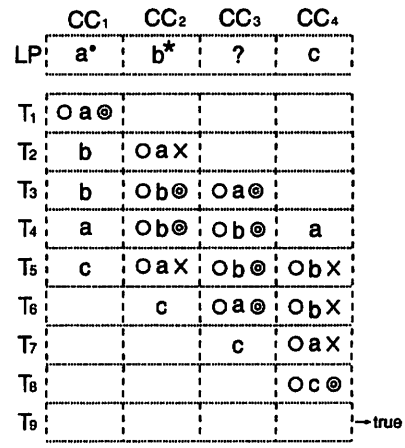

図 6　UC-項のストリングマッチング
Fig. 6　String matching of a UC-term.

以下に例を用いて K-項のストリングマッチングのアルゴリズムについて説明する．パターン (*abcdef*)*と入力系列 *abcdefabcdef* がどのようにマッチするかを図 7 に示す．

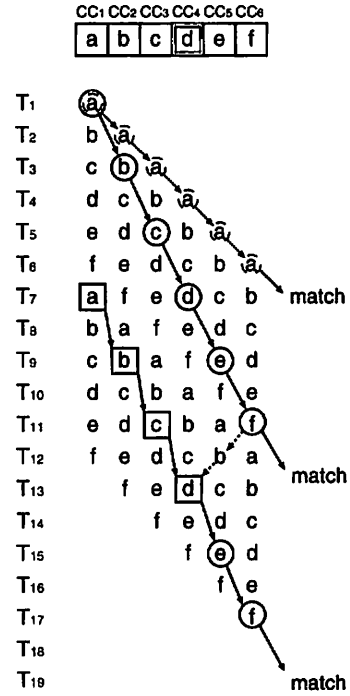

図 7　K-項のストリングマッチング
Fig. 7　String matching of a K-term.

まず，K-項は空語とマッチするため，入力系列が出力される前に "match" 信号を出力しなければならない．提案ハードウェアでは，各 CC は特別なフラグ EK を持っており，K-項内の最初の CC から最後の CC までこのフラグを使用して "match" 信号が伝達される．K-項の左端の CC の EK は常に 1 となって

おり，任意のK-項のCCでは，左隣のCCのEKが1のとき自身のEKも1となる．K-項の右端のCCは，自身のEKが1となったとき“match”信号として1を出力する．図7では，点線の円がEKが1であることを示している．

次に，K-項はクリーネ演算子のオペランドとマッチする．この例では，$(abcdef)$*は$abcdef$とマッチする．このマッチングはこれまでのパターンマッチと同じように実行される．この図では，実線の円がどのようにして“match”信号がCC間を伝播されたかを示している．

最後に，K-項はパターンの任意の回数の繰り返しとマッチしなければならない．このマッチングを実現させるために，決定セルと仮マッチングという概念を導入する．仮マッチングとは，K-項の左端のCCが現在の入力系列とマッチしたときに，あたかも与えられたクリーネ閉包のパターンと入力系列がマッチしたかのように開始するマッチングのことであり，この図では仮マッチングの成功は四角で示されている．実際に入力系列がパターンとマッチするように，仮マッチングの結果は確認されなければならない．この確認は，決定セルと呼ばれる特別なCCで行われる．K-項の右端のCCは“match”信号を決定セルに向けて後方に伝播させる．図ではこの後方への信号の伝播は点線で示される．もし決定セルが仮マッチングの信号と右端のCCからの後方への信号を受け取った場合，仮マッチングは通常のマッチングへと変更される．もし信号を受け取らなかった場合マッチングは失敗となり，仮マッチングはこの時点で終了となる．この図ではCC_4が決定セルとなる．

決定セルの位置は以下のように決定される．K-項の長さをpとし，K-項のCCを$CC_1, CC_2, \cdots, CC_p$とする．そのとき決定セルCC_dの位置は$d = \lfloor \frac{2(p+1)}{3} \rfloor$となる．

pの値によっては，仮マッチングの結果の信号と右端から後方に伝播される信号は同じクロックサイクルで到着しない場合がある．pの値によって，全部で以下の3つ場合が存在する．

（1） ($p \bmod 3 = 0$): 2つの信号は同じクロックサイクルで決定セルに到着

（2） ($p \bmod 3 = 1$): 仮マッチングの結果の信号が1クロックサイクル遅れて決定セルに到着

（3） ($p \bmod 3 = 2$): 仮マッチングの結果の信号が2クロックサイクル遅れて決定セルに到着

この提案ハードウェアの設計では，どんなCCも決定セルとして指定されることができる．パターンがハードウェアに記憶されるとき，パターン中の各K-項に対して決定セルが指定される．

3.3.3 KC-項のマッチング

KC-項のマッチングは仮マッチングと決定セルの概念を拡張することで実現している．KC-項の右端のCCが決定セルとして指定され，KC-項内のすべての単一項とC-単純項で仮マッチングが行われる．このとき，第i項で仮マッチングが開始されるとKS_i信号を，仮マッチングに成功するとKT_i信号を，失敗するとKF_i信号を決定セルに出力する．そのため，決定セルを含む項ではKT，KFは出力されず，そのマッチング結果は決定セルに記憶される．また，これらの信号が決定セル以外のCCに入力された場合はそのまま右隣のCCに出力する．

決定セルは，各項から送られたKTの中から“match”信号として出力されるKTを選択する．このとき，各項から出力されるKSが重要となる．KC-項のマッチングでは，KC-項の繰り返しが1回目の場合は入力系列の先頭から，繰り返しがn回の場合，n-1回目の繰り返しでKC-項の任意の項とマッチングに成功した入力系列の文字をs_{k-1}とすると，s_kからそれぞれマッチングに成功しなければならない．そのため決定セルでは，“match”信号が出力されるときに入力されたKSを選択候補とし，その項からKTが出力された場合のみそれを“match”信号として出力する．また，決定セルでは信号が出力された項と同時に何回目の信号かも記憶するので，同じ項からKS，KS，KF，KTといった順番で信号が出力された場合にも正しくマッチングを行える．

パターン$(abc+bcd+de)$*と入力系列$abcde$がどのようにマッチするかを図8に示す．

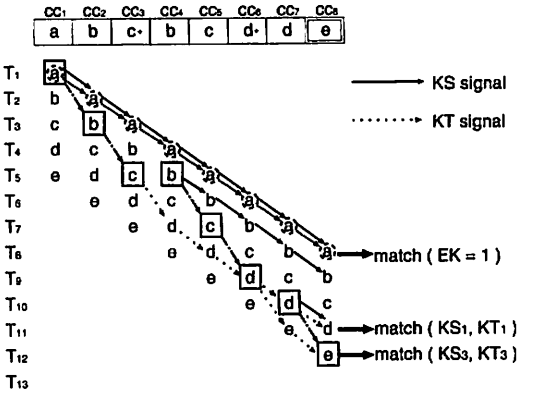

図8 KC-項のストリングマッチング
Fig. 8 String matching of a KC-term.

まず，KC-項も空語とマッチするため，この場合はK-項と同様にして“match”信号を出力する．

次に，入力系列の先頭文字aから仮マッチングに成功したものを決定セルは選択する．ここで決定セルはEKが1になったときに入力されたKS_1を選択候補とする．マッチング結果KT_1が2クロック後に出力されているため，決定セルは“match”信号としてKT_1を出力する．これはKC-項の最初の項abcがマッチングに成功したことを意味する．

1回目の繰り返しで入力系列のcまでマッチングに成功したため，2回目の繰り返しはdからとなる．ここでは1回目の繰り返しで“match”信号が出力されたと同時に入力されたKS_3が選択候補となる．第3項での仮マッチングに成功したため，次のクロックで決定セルはこの結果を記憶し，“match”信号として出力する．3回目以降の繰り返しについても，同様の方法マッチングを行う．

3.3.4 KC-項のマッチングアルゴリズムの正当性の証明

入力系列のs_0からs_kまでのマッチングにおいて，常に正しくKC-項のマッチングを行うためには，以下の条件を必ず満た

さなければならない．ここで繰り返しn回目のマッチングとは，n-1回目で s_{m-1} までのマッチングは終了している場合，n回目で s_m から開始するマッチングである．

（1） 最初の繰り返しは入力系列の先頭文字 s_0 からマッチングを開始．

（2） 繰り返しk回目のマッチングを開始する場合，k-1回目のマッチングまでマッチングに成功．

（3） 繰り返しl回目のマッチングに失敗した場合，たとえl+1回以降の全てのマッチングに成功したとしてもKC-項とはマッチしない．

最初の条件は，1回目の繰り返しにおいて決定セルのEKが1になったときに入力された KS_i を選択候補とすることで満たされる．これは図7や8からわかるように，その CC に先頭文字 s_0 が入力されたときのみEKが1になるためである．

2つめの条件は，繰り返しk回目のマッチングの場合，k-1回目のマッチングで“match”を出力するときに入力された KS_i を選択候補とすることで満たされる．この条件では，例えば2回目で s_a からマッチングを開始する場合に1回目で s_{a-1} までのマッチングに成功していなければならないが，最初の条件が常に満たされることと補題1と2より，s_{a-1} までのマッチングに成功した場合，必ず s_a が決定セルに入力されるときに“match”信号が出力される．選択候補の条件から，このときに KS_i は決定セルに入力されなければならず，このとき一緒に入力される文字が s_a であるため，2回目は s_a から開始されることになる．3回目以降も同様のことが言えるので条件2は満たされる．

もしl回目のマッチングに失敗した場合，“match”信号は出力されない．そのためl+1回目でマッチングに成功したとしても，KS_i が選択候補となる条件を満たさないため，KT_i は“match”信号として出力されず，以降の繰り返しについても同様のことがいえるため，最後の条件も満たされる．

以下のいくつかの補題は，上記の条件を提案アルゴリズムが常に満たすことを証明するために用いる．紙面の都合上，証明は省略する．

[補題1] KS_i と一緒に s_p が，KT_i と一緒に s_q が決定セルに入力された場合，第i項では s_p から s_{q-1} までの仮マッチングに成功する．

[補題2] KS_i が選択候補とされた場合，対応した KT_i が必ず“match”信号として出力される．

[補題3] 項 t_i と決定セルの動作はKC-項に含まれる項数に依存しない．

補題3より，KC-項が任意の単一項とC-単純項のユニオンからなる場合でもこれら3つの条件は満たされるので，提案アルゴリズムは常に正しいマッチングを行うことが証明できる．

[定理1] 提案アルゴリズムはKC-項に対するパターンマッチングを正しく実行する．

4. 実験的評価

提案ストリングマッチングハードウェアをFPGA上に実装し，実際にテキストといくつかのパターンを与えてストリングマッチングを行った．ハードウェア記述はVerilog-HDLを用い，FPGA設計ツールにはXilinx社のISE Version9.2iとSynplify Premier8.5.1，シミュレーションツールにはModelSim XE II 6.1eを使った．FPGA評価ボードには東京エレクトロンデバイス製FPGA評価ボードTD-BD-PCI-Express2DVI(ボード上のFPGAはXilinx社のXC4VLX100-11F1513)を用いた．

提案ハードウェアを実現するために記述したVerilog-HDLの行数は8223行，論理合成の結果，回路で使用したLUT数は21017である．回路の最大クロック周波数は322.0MHzであった．これは1秒あたり約3億2000万文字の入力系列に対してストリングマッチングを実行することが可能である．しかしFPGA評価ボードの制約より，実際に実験で使用されたクロック周波数は125MHzである．シミュレーションおよび実機を用いた実験により，提案ハードウェアが正しく動作することを確認した．

5. おわりに

本論文では，パターンを実行時に設定可能な正規表現ストリングマッチングマシンの提案と，そのFPGA上での実現を行った．評価実験の結果，提案手法は高速にストリングマッチングを実行可能であることを示した．また，提案手法ではストリングマッチングの実行時に任意のパターンの設定を行えることから，NIDSやデータベースにおけるテキスト検索のような広範囲のアプリケーションに対して適用可能である．今後の課題としては，パターンとして正規表現のより広いサブクラスを許した場合のアルゴリズムへの拡張，より便利な正規表現の記述形式の導入，よりエリア効率のよい提案ハードウェアの設計，そして提案ハードウェアの動作速度向上等があげられる．

本研究の一部は平成19年度科学研究費補助金基盤研究(C)(課題番号18500042)，および平成19年度広島市立大学特定研究費により行った．

文　献

[1] J. Aoe: *Computer Algorithms: String Pattern Matching Strategies*, IEEE Computer Society Press, 1994.

[2] L. A. Hollaar: Text retrieval computers, *IEEE Computer*, Vol.12, No.3, pp.40-50, 1979.

[3] B. L. Hutchings, R. Franklin, D. Cover: Assisting network intrusion detection with reconfigurable hardware, *Proc. 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp.111-120, 2002.

[4] Yosuke Kawanaka, Shin'ichi Wakabayashi, Shinobu Nagayama: Design and FPGA Implementation of a High-Speed String Matching Engine, *Proc. 14th Workshop on Synthesis And System Integration of Mixed Information technologies*, R2-2, pp.122–129, 2007.

[5] H. T. Kung: “Why systolic architectures?,” *IEEE Computer*, Vol.15, No.1, pp.37–45, 1982.

[6] R. セジウィック: アルゴリズム 第2巻=検索・文字列・計算幾何, 近代科学社, 1992.

[7] 若林真一: 制限された正規集合を受理するVLSI向きパターン・マッチング・マシン, 信学技報, AL85-33, 1985.

[8] 富田悦次, 横森貴: オートマトン・言語理論, 森北出版, 1992.