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ハードウェア

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)

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 上に実現することを目的とする.

2　ストリングマッチング

ストリングマッチングは文字列に関する基本操作の 1 つである [2]. ストリングマッチングには多くの定式化があるが, 基本的なストリングマッチングでは長さ N のテキストと長さ M(<N) のパターンが与えられたとき, テキスト中にパターンの現れる箇所があればその 1 つを示し, なければないと答える. ここで“パターン”とはテキストとアルファベットを共有する文字列である.

3　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 [3]. 本研究においても, 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.

3.1　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 [3]. 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.

1. ϕ を, 空集合を意味する Σ 上の正規表現とする.
2. ε を, $\{\varepsilon\}$(空語) を意味する Σ 上の正規表現とする.
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.
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,
   (a) $(R+S)$ を, R∪S を意味する Σ 上の正規表現とする.
   (b) (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.
   (c) (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.
5. 4 において各表現には () をつけて定義したが, 優先順位を
   ＊　(クリーネの閉包演算子)
   ・　(連接, 通常は表記しない)
   ＋　(和)
   の上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.
   例　$(ab)+(a(b)^*) \Rightarrow ab+ab^*$
6. 以上の 1 から 5 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.

パターンの例としては, $ab+ab^*a+(ab)^*ba$ や $(a+b)(bc^*a+abc^*)$ などがある. 入力系列 $bbcccaab$ に対してパターン $(a+b)(bc^*a+abc^*)$ が与えられたとすると, このときパターンとマッチする入力系列中の文字列は, 1 文字目から始まる $bbccca$ と, 6 文字目から始まる aab となる.

3.2　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 [1].

1. $? = a_0 + a_1 + a_2 +\cdots+ a_n$
2. $\text{-}a_i = a_0 + a_1 +\cdots+ a_{i-1} + a_{i+1} +\cdots+ a_n$
3. $a^{@} = a + \varepsilon,\qquad a\in\Sigma\cup\{?\}$
4. $\text{-}a_i^{@} = (\text{-}a_i)^{@}$
5. $\text{-}a_i{}^* = (\text{-}a_i)^*$
6. $?^* = (a_0 + a_1 + a_2 +\cdots\text{S}+ a_n)^*$
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.
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.
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.
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　$ab^*c+d+(\text{-}a^{@})?$
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.
    例　$a^*?(bc+d^{@})e$
    ここで C-展開可能とは, 例えば上記の例の場合, 連接演算子 (・) に関して $a^*?bce+a^*?d^{@}e$ と展開できることを意味している.

4　ストリングマッチングハードウェア

4.1　アーキテクチャ

提案アルゴリズムは 1 次元配列状にセルを接続したハードウェア上で実現される (図 1). セルの左端から 1 クロックごとにテキストが入力され, 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とマッチングを行う [1].

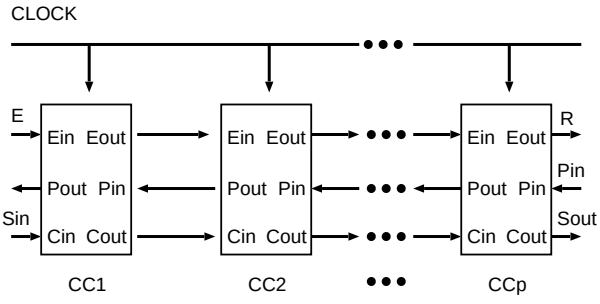

図 1　アルゴリズムのアーキテクチャ

ここで, セルとは 1 文字単位のマッチングを行う回路である. 各セルの構造を図 2 に示す.

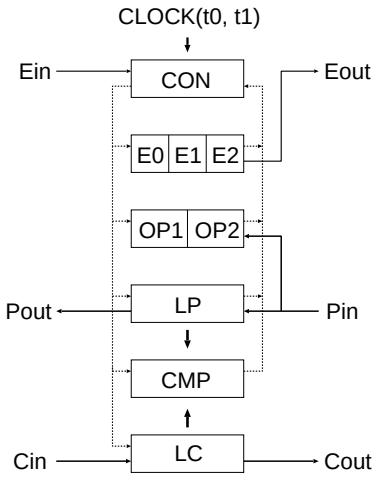

図 2　比較セル

4.2 アルゴリズム

セルの動作は 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．図 3 にセルのアルゴリズムを示す．各セルにはラッチ LP の他に，LC，OP1, OP2, E0, E1, E2 のラッチがある．LC には入力系列の 1 文字が記憶され，LP と LC を比較することでマッチングが行われる．また，LP にパターンがラッチされるとき，同時に OP1 には “?” と演算子 “-” が，OP2 には演算子 “*” と “@” がそれぞれラッチされる．マッチングを正しく行うためには，そのセルのマッチング結果を 2 クロック後に右隣のセルに出力する必要がある．そのため，最初にマッチング結果を E1 に記憶し，それを E2 に受け渡して右隣のセルに出力する．パターンにクリーネ演算子が含まれ，パターンを複数回繰り返す場合は，マッチングを繰り返すという情報を E0 に記憶し，1 度も繰り返さない (またはパターンが ε や@演算で ε となる) 場合は入力されたマッチング結果を E2 に代入することで，正しくマッチング動作を行うことができる．

```
t0: begin
      MOVE_INPUT_STRING;
      E:=Ein or E0;
      E2:=E1;
      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='*' then E2:=true;
        if OP1='@' then E2:=true;
        if OP1='ε' then E2:=true
      end
    end;
```
図 3　各セルのアルゴリズム

4.3 動作例

パターンを $a^{@}b^{*}?c$, 入力系列を S = $abbac$ とした場合の動作を図 4 に示す．図 4 の LP は各セルのラッチ LP である．パターンの入力はすでに行われているものとする．LP の下側は各セルのラッチ LC において，入力系列がクロックに同期して左から入力されつつ，各セルが図 3 のアルゴリズムを実行していることを表す．ここで，T_i は i 番目のクロック周期を表す．図中の各入力系列中の文字の左右の記号 (○, ◎, ×) は次の意味をもつ．○はそのクロック周期でマッチングアルゴリズム中の変数 R が true になることを示す．つまりマッチング動作をすることを示す．◎はマッチングの結果，マッチしたことを，×はマッチしなかったことを示す．例えば T_1 において，CC_1 は外部より入力系列の 1 番目の文字 a が入力される．それと同時に，E_{in} が true となるので，先ほどのアルゴリズムにより T_1 において変数 R が true となり，図中に○がつく．比較の結果，パターンと一致するので，◎となる．また，a には演算子@がついているので E2 も true となり，その結果，T_2 の CC_2 に○がつく．さらに，T_3 において CC_2 がマッチングの動作を実際に行うように CC_1 はフリップフロップ E1, E2 を用いて CC_2 に true を伝達するので，T_3 の CC_2 に○がつく．以後，クロックに同期してこのような動作を繰り返す．この例では，T_8 において，◎となり，この結果，T_9 で外部に出力されるので，入力系列 S はパターン P にマッチしたことになる．

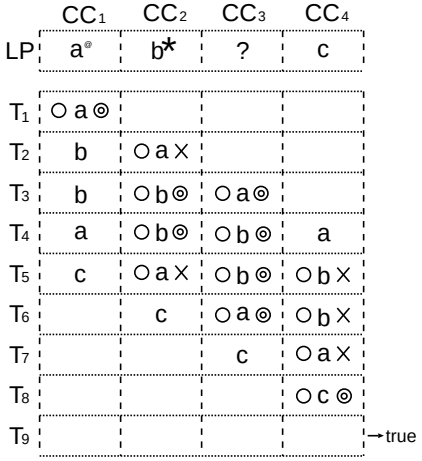

図 4　動作例

5 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　6.1e を使った．FPGA 評価ボードには 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 8140 行，論理合成の結果，回路で使用した LUT 数は 21017 である．回路の最大動作周波数は 316.1MHz であり，2000 文字の入力系列から 500 文字のパターンを検索するのに必要な検索時間は 8.06 μ s となるので，この回路は 1 秒あたり約 31 億文字の入力系列に対してマッチングを実行することが可能である．

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，提案ハードウェアの動作速度の向上等があげられる．

参考文献

[1] 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.
[2] 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.
[3] 富田悦次, 横森貴 :オートマトン・言語理論, 森北出版, 1992.