ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
논문번호 18-01-13
j.inst.Korean.electr.electron.eng.Vol.22,No.1,87~92,March 2018
http://dx.doi.org/10.7471/ikeee.2018.22.1.87

정규표현식 프로세서에서의 효율적 문자 클래스 매칭을 위한 구조

Architecture for Efficient Character Class Matching in Regular Expression Processor

윤 상 균*★

SangKyun Yun*★

Abstract

Like CPUs, regular expression processors that perform regular expression pattern matching using instructions have been proposed recently. Of these, only REMPc provides features for character class matching. In this paper, we propose an architecture for efficient character class matching in a regular expression processor, which use character class bitmap format in a instruction operand field and implement the hard-wired character class comparator for several frequently used character classes. Using the proposed method, most of the character classes used in Snort rule can be represented by an operand or an instruction. Thus, character class matching can be performed more efficiently in the proposed archiecture than in REMPc.

요 약

보통의 CPU 처럼 명령어 기반으로 정규표현식 패턴 매칭을 수행하는 정규표현식 프로세서가 최근에 연구되었다. 이들 중 REMPc만이 문자 클래스 처리를 위한 기능을 제공한다. 본 논문에서는 정규표현식에서 사용 빈도가 높은 문자 클래스들에 대해서 명령어의 오퍼랜드 필드에 비트맵 방식으로 나타내고, 하드 배선 방식으로 이 문자 클래스에 대한 매칭을 수행하여 효율적인 문자클래스 매칭을 수행하는 구조를 제안한다. 제안한 방법을 사용하면 Snort 규칙의 문자 클래스에 대해서 대부분의 문자 클래스를 명령어의 한 오퍼런드 또는 한 명령어로 나타낼 수 있다. 이처럼 REMPc에 비해서 적은 수의 명령어를 사용하므로 효율적인 문자 클래스 매칭을 할 수 있다.

Key words : Regular expression matching, Character class matching, Regular expression processor, Custom processor architecture, FPGA, Intrusion detection

* Department of Computer and Telecommunication Engineering, Yonsei University, Wonju

★ Corresponding author

E-mail: skyun@yonsei.ac.kr Tel: 033-760-2267

※ Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No.2011-0025467).

Manuscript received Mar. 11, 2018; revised Mar. 18, 2018 ; accepted Mar. 20, 2018

Ⅰ. 서론

문자열 매칭은 네트워크 침입탐지 시스템, 프로토콜 파서 등의 응용분야에서 널리 사용된다. 문자열 패턴을 나타내는 데에 많은 문자열 패턴들을 간결하게 표현할 수 있는 정규표현식이 주로 사용되고 있다. 소프트웨어 기반의 정규표현식 매칭은 처리 속도에 제약이 있기 때문에 고성능 정규표현식 매칭을 위한 하드웨어 기반 구조들이 여러 연구자들에 의해서 제시되었다[1].

특히 정규표현식을 일련의 명령어로 나타내어 보통의 프로세서와 비슷한 방식으로 패턴 매칭을 수행하는 정규표현식 프로세서들이 제안되었으며

ReCPU [2-3]가 정규표현식 프로세서의 대표적인 예이다. 그리고 ReCPU의 단점을 개선한 정규표현식 프로세서인 SMPU(string matching processing unit)[4]와 REMP(regular expression matching processor)[5]도 제시되었다. 이 방법은 정규표현식 패턴이 갱신될 때에 FPGA를 재구성할 필요가 없이 보통의 프로그램처럼 명령어 메모리의 내용만 변경하여 사용할 수 있는 장점이 있다.

정규표현식 프로세서에서는 정규표현식에 대한 명령어를 명령어 메모리에 저장하고, 명령어를 순서대로 실행하면서 데이터 메모리에 저장된 데이터에 대한 패턴 매칭을 수행한다. ReCPU의 명령어는 패턴의 문자열을 명령어의 오퍼랜드로 나타내어 데이터와의 패턴 매칭에 사용하고, 단순 매칭, 반복, OR와 같은 연산을 연산코드로 나타내어 명령어의 실행 순서를 결정하는 데 사용한다.

정규표현식에서는 패턴을 나타내는 데 단일문자 뿐 만 아니라 [ace]와 같이 문자들의 집합을 나타내는 문자 클래스도 사용된다. 그렇지만 ReCPU, SMPU와 REMP는 모두 문자 클래스에 대한 별도의 지원 기능이 없어서 문자 클래스를 문자들의 OR 연산으로 처리해야 하므로 비효율적이다.

REMPc[6]은 REMP의 명령어 형식을 수정하고 명령어를 추가하여 문자 클래스 매칭을 효과적으로 수행할 수 있는 기능을 제공하였다. REMPc에서는 명령어의 오퍼랜드를 문자 클래스에 속한 문자들로 나타내어, 한 명령어에서 문자 클래스에 속한 최대 4개까지의 문자와 데이터의 한 문자를 동시에 비교할 수 있게 하였다. 그리고 범위 비교 명령어를 사용하여 문자 클래스에 속한 한 개의 문자 범위와 한 개의 문자를 데이터의 한 문자와 동시에 비교할 수 있다. 그렇지만 REMPc에서 추가된 기능을 사용하더라도 흔히 사용되는 많은 문자 클래스들이 4개보다 많은 문자들로 구성되어 여전히 여러 개의 명령어를 사용해야 한다. 그리고 대소문자를 구분하지 않는 정규표현식 매칭도 많이 사용되는 데 이를 위한 기능이 없다.

본 논문에서는 이러한 REMPc의 단점을 보완하기 위하여 대소문자를 구분하지 않는 매칭 기능을 추가하고, 실제 응용에서 많이 사용되는 문자 클래스들에 대한 매칭 회로를 하드 배선 방식으로 구현하고, 이러한 문자 클래스들을 명령어의 한 오퍼랜드로 나타내도록 하여, 문자 클래스를 처리하는 데 사용되는 명령어 수를 감소시켜서 정규표현식 프로세서에서의 문자 클래스 처리 성능을 향상시킬 수 있게 하는 방법을 제안한다.

II. 관련 연구

ReCPU는 그림 1(a)와 같은 명령어 형식을 사용하며 다른 정규표현식 프로세서들도 비슷한 명령어 형식을 사용한다. 한 번에 많은 연속적인 데이터들에 대한 매칭을 동시에 수행할 수 있도록 패턴의 연속적인 문자열을 같은 명령어의 오퍼랜드로 사용하고, 그림 1(b)와 같이 오퍼랜드의 문자열 패턴을 한 문자씩 시프트된 위치의 데이터 문자열과 동시에 비교하는 비교기 클러스터가 구성되어, 임의의 위치에서 매칭을 시작할 수 있도록 하였다.

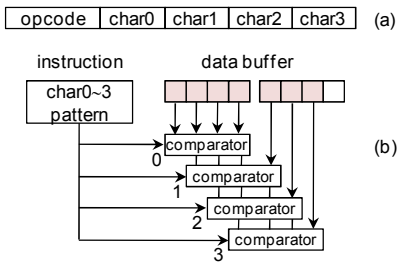

Fig. 1. ReCPU (a) instruction format (b) comparator cluster
그림 1. ReCPU (a) 명령어 형식 (b) 비교기 클러스터

정규표현식에서 문자 클래스는 다음과 같은 여러 가지 형태로 나타낼 수 있다.

- [ace] : 개별 문자들의 집합
- [a-d] : 문자 범위
- [a-fxz] : 문자 범위와 개별 문자들의 조합
- [^abcd] : 제외되는 문자들을 나타내는 부정(negated) 문자 클래스
- \s, \d : 미리 정의된 문자 클래스

기존의 정규표현식 프로세서의 오퍼랜드는 그림 1(a)와 같이 문자를 사용하므로 이러한 문자 클래스를 나타낼 수 없다.

REMPc[6]는 문자 클래스 매칭을 지원하도록 REMP를 수정한 것이다. 이전의 정규표현식 프로세서들은 여러 개의 문자 오퍼랜드들을 같은 수의 연속적인 데이터 문자들과 동시에 비교(4대4 비교)하지만 REMPc는 [abcd]와 같은 문자 클래스 매칭을 지원하기 위해서 여러 개의 문자 오퍼

랜드들을 하나의 데이터 문자와 비교(4대1 비교)하는 기능을 선택하여 사용할 수 있도록 하였다. 이 기능을 사용하면 보통의 문자 클래스 표현에 필요한 명령어의 개수가 약 1/4로 감소한다. 그리고 [^abc]와 같은 부정 문자 클래스 매칭 기능도 제공한다. 그림 2는 REMPc의 명령어 형식으로서 연산코드(opcode)에 플래그 비트 C와 N이 추가되었다. C 또는 N이 1일 때에 4대1 비교기를 사용하여 문자 클래스 매칭을 하며, 특히 N이 1일 때에는 매칭 결과를 반대로 사용하는 부정 문자 클래스 매칭을 지원한다. 그리고 문자 범위 매칭을 지원하기 위하여 CMPR(compare with range)와 ORR(simple OR with range) 명령어를 추가했다.

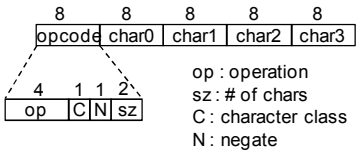

Fig. 2. Instruction Format of REMPc
그림 2. REMPc의 명령어 형식

문자 개수가 4개 이하인 짧은 문자 클래스는 한 개의 REMPc 명령어로 나타낼 수 있다. 다음은 문자 클래스 [abcd]에 대한 명령어 예이다. 명령어에서 [abcd] 표기는 4대1 비교(C=1, N=0)를 사용하는 것을 의미한다.

```
CMP [abcd] ; 플래그 C=1, N=0, 오퍼랜드 a b c d
```

문자 개수가 4를 초과하는 긴 문자 클래스는 하나의 명령어로 나타낼 수 없고, 짧은 문자클래스들의 OR 연산으로 나타낼 수 있으며 여러 개의 OR 명령어를 사용해야 한다. 다음은 문자클래스 [abcdepqrsxy]에 대한 매칭을 REMPc 명령어들을 사용하여 나타낸 예로서 4개의 명령어를 사용한다.

```
ORS +4          ; OR 다음 위치 지정
OR [abcd]       ; OR 연산. 매칭되면 OR 다음 위치로
                ; 실패하면 다음 명령어로
OR [pqrs]       ; OR 연산. 위와 같음
E_OR [xy]       ; OR 마지막. 성공하면 OR 다음위치로
                ; 실패하면 문자클래스 매칭 실패
...             ; OR 다음 위치(ORS +4에서 지정)
```

III 제안하는 문자클래스 매칭 구조

1. 기본 아이디어

정규표현식 프로세서에서 정규표현식 매칭을 효율적으로 수행하기 위해서는 실행되는 명령어 수를 줄일 필요가 있다. REMPc에서 제공되는 문자 클래스 처리 기능을 사용하더라도 긴 문자 클래스는 여전히 여러 개의 명령어를 사용하여 표현되며, 짧은 문자 클래스는 한 명령어로 처리할 수 있을지라도 다른 패턴을 함께 처리하지 못하므로 문자 클래스를 처리하는 명령어는 한 개의 데이터 문자만 처리할 수 있다. 그리고 대소문자를 구분하지 않는 문자 매칭도 필요한 데 기존의 정규표현식 프로세서에서는 대소문자 2개로 구성된 문자클래스로 다루어야 하므로 비효율적이다.

사용 빈도가 높은 문자 클래스를 명령어에 한 개의 오퍼랜드로 나타낼 수 있도록 이들에 대한 매칭 회로를 하드 배선 방식으로 구현한다면 다음과 같은 두 가지 이점을 가지게 된다. 다음 예에서 [0-9]는 매칭회로가 구현되어 한 오퍼랜드로 나타낼 수 있는 문자 클래스라고 가정한다.

첫째, 이러한 문자 클래스는 다른 문자 패턴들과 함께 한 명령어에서 사용될 수 있으므로, 한 번에 여러 개의 연속된 데이터 문자를 처리할 수 있게 된다. 예를 들어 패턴 [0-9]abc는 다음과 같은 명령어로 표현되어 한 번에 4개의 연속된 데이터 문자와 매칭을 수행할 수 있다.

```
CMP [0-9]abc    ; C=0, N=0, 오퍼랜드 [0-9], a, b, c
                ; 4대4 비교 (4개의 연속된 문자와 비교)
```

둘째, 한 오퍼랜드로 나타낼 수 있는 문자 클래스를 포함하는 긴 문자 클래스인 경우에 추가되는 문자가 3개 이내이면 한 명령어로 나타낼 수 있다. 예를 들어 문자 클래스 [0-9]를 포함하는 문자 클래스 [0-9apz]는 다음과 같이 문자 클래스 구성 문자를 4개의 오퍼랜드로 사용하는 한 명령어로 나타낼 수 있다.

```
CMP [0-9apz]    ; C=1, N=0, 오퍼랜드 [0-9], a, p, z
                ; 4대1 비교 (한 문자와 동시 비교)
```

이와 같이 사용 빈도가 높은 문자 클래스에 대한 매칭 하드웨어를 미리 구현하고 한 오퍼랜드로 나타내어 사용할 수 있도록 하면 사용되는

명령어 개수를 감소시켜서 문자 클래스 처리를 효율적으로 할 수 있다.

그리고 비교기에서 대소문자 구분이 없는 문자 매칭을 지원하고 오퍼랜드에서 이에 대한 매칭을 나타낼 수 있다면 대소문자 구분없는 문자 매칭을 효율적으로 수행할 수 있을 것이다. 이러한 방법을 지원하도록 제안되는 정규표현식 프로세서를 REMPcm이라고 부를 것이다.

2. 오퍼랜드 필드 형식

명령어의 한 오퍼랜드에 문자 뿐 만 아니라 미리 구현되는 문자 클래스도 나타낼 수 있도록 하기 위해서 REMPcm에서는 그림 2의 명령어 형식을 그림 3와 같이 오퍼랜드 필드를 1비트를 추가하여 9비트로 확장한다. 추가된 비트가 0이면 나머지 8비트는 문자를 나타내고, 1이면 나머지 8비트는 문자 클래스들을 나타낸다.

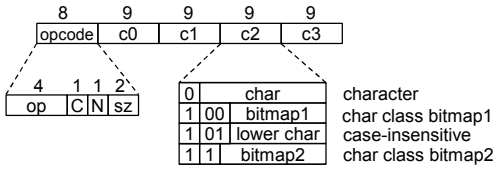

Fig. 3 instruction and operand format
그림 3 명령어와 오퍼랜드 형식

그림 3의 문자 클래스 오퍼랜드 형식에서 대소문자 구분이 없는 문자 매칭은 8비트 필드를 알파벳 문자(상위 두 비트가 01)로 나타내어 표현하며, 6비트 비트맵(상위 두 비트 00)과 7비트 비트맵(상위 한 비트가 1)을 사용하여 최대 13개의 미리 구현된 문자 클래스를 나타낼 수 있다.

같은 비트맵에 속하는 문자 클래스들은 한 오퍼랜드에 함께 나타낼 수 있으므로 함께 사용되는 빈도가 높은 문자클래스들을 같은 비트맵에 속하게 지정한다. 예를 들어서 문자클래스 [0-9]와 [A-F]가 같은 비트맵의 필드들로 표현되게 한다면 [0-9A-F]도 한 오퍼랜드로 표현될 수 있다.

3. 문자클래스 매칭을 위한 비교기

오퍼랜드의 비트맵으로 표현되는 13개의 문자 클래스에 대해서는 하드배선 방식으로 입력 문자 데이터와의 문자 클래스 매칭을 수행하는 비교기 회로를 직접 설계한다. 그림 3의 오퍼랜드 형식를 처리할 수 있는 비교기 구조는 그림 4와 같으며 이 비교기는 9비트 오퍼랜드와 입력 문자 데이터를 비교하는 단위 비교기로서 사용된다.

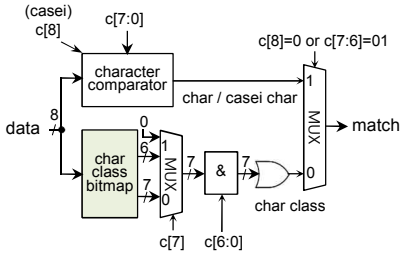

Fig. 4 Comparator supporting a new operand format
그림 4 새 오퍼런드 형식을 지원하는 비교기 구조

그림 4의 아래부분의 “char class bitmap” 회로는 비트맵으로 지정된 문자 클래스에 대한 비교회로를 만들어서 13개의 출력을 제공하며, 이 출력은 어느 비트맵에 속하느냐에 따라서 두 개의 그룹으로 구분된다. 비트맵 종류(c[7])에 따라서 두 개의 그룹의 출력 중 하나를 제공받아서 명령어 오퍼랜드의 7비트 비트맵과 AND 연산을 하여 연산 결과가 1인 비트가 존재하면 비트맵으로 지정된 문자 클래스 매칭이 이루어진 것이다. bitmap1은 실제로 6비트를 사용하므로 상위 1비트는 0으로 고정된 비트맵 비교기 출력을 제공한다.

그림 4의 윗부분은 문자 비교기로서 대소문자 구분이 없는 문자 매칭을 지원하기 위해서 문자 클래스 오퍼랜드를 나타내는 c[8]이 1일 때에는 문자 비교기에 입력되는 데이터의 bit 5를 1로 입력되게 하여 알파벳 문자를 소문자로 만들어서 비교를 하도록 한다. 그리고 c[8]이 0일 때 뿐 만 아니라 c[7:6]이 01인 경우에도 문자 비교기의 출력이 매칭 출력으로 사용되도록 하여 대소문자 구분이 없는 문자 매칭을 구현하였다.

4. Snort 규칙을 위한 문자클래스 비트맵

그림 3과 같은 오퍼랜드 형식에서 문자 클래스 비트맵의 지정은 응용에 따라서 문자 클래스들의 사용 빈도가 차이가 있으므로 달라질 수 있다. 대표적인 침입탐지 시스템인 Snort[7]에서 침입 패턴을 기술하는 Snort 규칙에 포함된 정규표현식을 대상으로 하여 효율적인 처리를 위한 문자 클래스 비트맵을 지정해 보았다. 이를 위하여 먼저

Snort v2.975의 규칙에서 사용하는 문자 클래스들을 분석하였다.

이 규칙에는 1759개의 PCRE(Perl Compatible Regular Expression)형식의 정규표현식 패턴이 포함되어 있으며 이 패턴들은 7629개의 문자 클래스를 사용한다. 이 문자 클래스들 중에서 중복되지 않고 서로 구분이 되는 문자 클래스들은 192개이다. [0-9a-z]와 [a-z0-9] 같이 형태는 다르지만 실제로는 같은 문자 클래스는 한 개로 세었다. 이 중에서 71개가 부정 문자 클래스이고, 121개가 부정이 아닌 보통 문자 클래스이다.

Snort 규칙에서 사용 빈도가 높은 보통 문자 클래스들은 표 1과 같으며, 사용 빈도가 높은 부정 문자 클래스들은 표 2와 같다.

Table 1. Frequently used character classes
표 1. 사용빈도가 높은 문자클래스

char class	count	char class	count
\s	3896	\x2f\x5c	23
\x22\x27	556	\x2e\x5b\x5c	22
0-9	514	\x2b\x3d\x7e 0-9A-Za-z	20
\w	390	0-9A-Za-z	20
\x2f\x3f\x5c	195	A-Za-z	19
0-9a-f	100	\x26\x3f	16
a-z	64	\s\x22\x27	15
0-9a-z	62	dlnpsx	13
0-9A-F	36	a-z\x2b[\x2f-\x39] \x3d	12

Table 2. Frequently used negated character classes
표 2. 사용빈도가 높은 부정 문자클래스

char class	count	char class	count
0-9	514	0-9a-f	100
\w	390	\s\x22\x27\x3e	16
\x2f\x3f\x5c	195	\s\x29\x2c\x5c	11

이러한 사용 빈도를 바탕으로 많이 사용하는 보통 문자 클래스들을 1개의 오퍼랜드로 나타내고, 거의 모든 문자 클래스들을 4개 이하의 오퍼랜드를 사용하여 1개의 명령어로 나타낼 수 있도록 표 3과 같이 비트맵을 지정하였다.

Table 3. Character class bitmap for Snort rule
표 3. Snort 규칙을 위한 문자클래스 비트맵

bit	6	5	4	3	2	1	0
char class1(00)	not used	\x22 \x27	\x2f \x5c	\x2e \x5b \x5c	dlnpsx	\x2f-\x39	\x3f
char class2(1)	\s	0-9	A-Z	a-z	A-F	a-f	_

표 3과 같이 비트맵으로 지정된 문자 클래스들 뿐 만 아니라 이들을 결합한 문자 클래스들도 한 오퍼랜드로 나타낼 수 있다. 표 4는 한 오퍼랜드로 표현되는 문자 클래스의 예를 보여준다.

Table 4. Examples of character classes encoded in an operand
표 4. 한 오퍼랜드로 표현되는 문자 클래스의 예

char class	operand encoding		char class	operand encoding	
[0-9]	11	0100000	[\x2f\x5c]	100	010000
[0-9a-f]	11	0100010	[\x2f\x5c\x3f]	100	010001
[0-9a-fA-F]	11	0100110	a	0	01100001
[0-9A-Za-z_] \w	11	0111001	[Aa] (case insensitive)	1	01100001

Ⅳ 평가

표 3의 문자클래스 비트맵을 사용하면 Snort 규칙에서 사용되는 192개의 문자 클래스들은 표5와 같은 개수의 오퍼랜드 또는 정규표현식 프로세서 명령어를 사용하여 나타낼 수 있다. 이 표에서 '/' 다음에 있는 숫자는 부정 문자 클래스에 대한 것이다. 대부분의 문자 클래스가 1개 명령어를 사용하여 나타낼 수 있으며, 전체 문자 클래스의 79% 정도를 1개의 오퍼랜드로 나타낼 수 있다.

이전에 제안되었던 문자 클래스 처리를 지원하는 정규표현식 프로세서인 REMPc에서도 많은 문자 클래스들이 여러 개의 명령어를 사용해야 하는 데 비해서, 제안된 구조를 갖는 REMPcm은 표 5와 같이 대부분의 문자 클래스들을 1 오퍼랜드 또는 1 명령어를 사용하여 매칭을 처리할 수 있어서 문자 클래스에 대한 효율적인 매칭이 가능하다.

Table 5. The number of REMPcm instructions for representing character class
표 5. 문자클래스 표현에 사용되는 REMPcm 명령어 개수

	# of unique char class	total # of char class
1 operand	22	6018
1 instruction (2~4 operands or 1 range)	85 / 69	198 / 1396
2 instructions (5~8 operand, or 1 range and 1~4 operands)	13 / 2	13 / 3
3 instructions	1	1

제안된 구조의 REMPcm 프로세서는 Verilog HDL로 설계되어 Altera Stratix IV GX FPGA (device 이름: EP4SGX230 KF40C2)를 타겟 디바이스로 하여 Quartus Prime v16.1 프로그램을 사용하여 합성하였다. 프로그램과 데이터는 FPGA 내부의 블록 메모리에 초기값으로 지정하여 구현하였으며, 어려 종류의 문자클래스를 포함한 정규표현식에 대하여 동작을 확인하였다. 표 6은 설계된 정규표현식 프로세서의 FPGA 합성결과이다. ALUT를 REMPc보다 조금 더 많이 사용하며, 가능한 최대 클록 속도는 하드웨어 복잡도 때문에 149.66 MHz로서 REMPc의 166.42MHz [6]보다는 약간 낮다. 그렇지만 문자 클래스 사용빈도가 높은 경우에는 제안된 구조가 효과적이다.

Table 6. FPGA Synthesis result for REMPcm
표 6. REMPcm에 대한 FPGA 합성 결과

Device	ALUTs	Registers	Freq	Speed
REMPcm	394	158	149.66 MHz	~4.79 Gbps
	394 ALU/Reg pairs			
REMPc [6]	341	158	166.42 MHz	~5.32 Gbps
	341 ALU/Reg pairs			

V 결론

정규표현식 패턴을 CPU와 유사한 방식으로 명령어로 나타내어 패턴 매칭을 처리하는 정규표현식 프로세서에서 문자 클래스 매칭을 효율적으로 수행하기 위하여 REMPc의 문자 클래스 매칭 기능에 추가하여 많이 사용되는 문자 클래스에 대해서 오퍼랜드 필드에 비트맵 방식으로 나타내고 하드 배선 방식으로 매칭을 수행하여 효율적인 문자클래스 매칭을 수행하는 구조를 제안하였다. 이 방법을 사용하면 대부분의 문자 클래스를 한 오퍼랜드 또는 한 명령어로 나타낼 수 있어서 이전의 정규표현식 프로세서에 비해서 효율적인 문자 클래스 매칭을 할 수 있다.

References

[1] J. C. Bispo, I. Sourdis, J. M. Cardoso, and S. Vassiliadis, "Regular expression matching for reconfigurable packet inspection," in *IEEE Int. Conf. Field Programmable Technology (FPT'06)*, 2006. DOI:10.1109/FPT.2006.270302

[2] M. Paolieri, I. Bonesana, M. Santambrogio, "ReCPU: a parallel and pipelined architecture for regular expression matching," in *Proc. IFIP Int. Conf. VLSI-SoC*, 2007. DOI:10.1109/VLSISOC.2007.4402466

[3] I. Bonesana, M. Paolieri, and M. Santambrogio, "An adaptable FPGA-based system for regular expression matching," in *Proc. Conf. Design, Automation and Test in Europe (DATE'08)*, 2008. DOI:10.1109/DATE.2008.4484852

[4] Q. Li, J. Li, J.Wang, B. Zhao, and Y. Qu, "A pipelined processor architecture for regular expression string matching," *Microprocess. Microsy.*, vol. 36, no. 6, pp. 520-526, 2012. DOI:10.1016/j.micpro.2012.04.004

[5] B. Ahn, K.H. Lee, and S.K. Yun, "Regular expression matching processor supporting efficient repetitive operations," *Journal of KIISE:Computing Practices and Letters,* Vol. 19, No. 11, pp. 553-558, 2013. DOI:10.4204/EPTCS.62.3

[6] S.K. Yun, "Regular expression matching processor supporting character class matching," *Journal of KIISE*, vol 42, no. 10. pp. 1280-1285, 2015. DOI:10.5626/JOK.2015.42.10.1280

[7] "Snort - Network Intrusion Detection," https://www.snort.org

BIOGRAPHY

SangKyun Yun (Member)

1984 : BS degree in Electronics Eng.ineering, Seoul National University.
1986 : MS degree in Electrical Engineering, KAIST
1995 : PhD degree in Electrical Engineering, KAIST
1992~2001 : Professor, Department of Computer Science, Seowon University
2001~ : Professor, Department of Comptuer and Telecom. Engineering, Yonsei Univ. Wonju