Journal of The Institute of Electronics and Information Engineers Vol.52, NO.9, September 2015 http://dx.doi.org/10.5573/ieie.2015.52.9.135
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2015-52-9-16

다중바이트 기반 스트리밍 XML 하드웨어 파서의 설계

(Design of Multibyte-based Streaming XML Hardware Parser)

이 규 희*, 서 병 석**

(Kyu-Hee Lee and Byeong-Seok Seo©)

요 약

웹 서비스들은 데이터의 표현과 전송을 위해 구조화된 문서 XML을 채택하고 있다. SOAP나 REST는 XML을 이용하여 메시지를 송/수신하는 대표적 시스템들이다. XML 파서는 이벤트 기반과 DOM 기반 파서들로 나눌 수 있는데, 고속의 작업을 위해서 이벤트 기반인 스트리밍 파서가 널리 사용되지만, 순차 처리되는 특성을 갖기 때문에 성능향상에 제약이 있다. 본 논문에서는 스트리밍 파서에서의 순차 처리 특성을 해결하여 고성능 파서를 제안하기 위해 다중바이트 기반 하드웨어 파서를 제안한다. 제안된 구조는 다른 파서들과 비교하여 문자 비교에 사용되는 소비 클록의 수가 약 2.72배 감소하였고 약 7.8Gbps의 시스템 성능을 갖는다. 따라서, 제안된 MStreXHP 파서는 고성능 시스템들에서 스트리밍 XML 파서에 적합한 구조를 갖는다.

Abstract

Web-services employ XML that is the well-formed structure as a de-facto standard to represent data. SOAP or REST is one of the representative web-services using XML based massage passing systems. The XML parser can be divided into event driven and DOM tree. A streaming parser as an event driven is widely used for high-speed parsing. Since the streaming parser processes XML documents in sequence, they have any limitation to improve system performance. In order to improve speed of streaming XML parser, we present multibyte based streaming XML hardware parser using the element analyzer instead of the state machine. Compared to other parsers, the proposed MStreXHP can achieve about 2.72 times improvement in the number of clock cycles to be consumed in comparison of characters and sustain about 7.8Gbps throughput. Therefore, our MStreXHP is desirable for the streaming XML hardware parser on high-performance systems.

Keywords : XML parser, Streaming XML parser, FPGA

Ⅰ. 서 론

다양한 웹 서비스들은 구조화된 표준 문서인 XML(eXtensible Markup Language)을 이용하여 사용자에게 서비스를 제공한다. 대표적으로, SOAP(Simple Object Access Protocol)와 REST(Representional State Transfer)가 XML 기반의 메시지를 전달하는 웹 서비스 방식들이다[1]. XML의 파싱은 이벤트 기반 SAX (Simple API for XML)[2] 파서와 DOM(Document Object Model) 기반 파서가 널리 사용된다. DOM 파서는 문서 전체를 메모리에 적재한 후 파싱을 수행하기 때문에 접근이 용이하지만 전처리 작업이 요구되며 메모리 사용량이 증가한다. 반면에 스트리밍 파서인 이벤트 기반 파서들은 XML 문서가 입력되는 동시에 파싱을 수행하기 때문에 전처리 작업 및 추가적인 메모리 할당이 필요하지 않다. 따라서, 스트리밍 XML 파서들은 제한적 자원을 갖고 고속의 파싱이 요구되는 시스템

* 정회원, 상지영서대학교 국방정보통신과
(Department of National Defense Communication Engineering, Sangji Youngseo College)
** 평생회원, 상지영서대학교 국방정보통신과
(Department of National Defense Communication Engineering, Sangji Youngseo College)
© Corresponding Author(E-mail: seobs@sy.ac.kr)
Received ; July 1, 2015 Revised ; July 17, 2015
Accepted ; September 3, 2015

들에 적합한 구조이다.

고속의 파싱 성능을 내기 위해 하드웨어 기반 XML 파서들이 제안되고 있으며, TCAM (Ternary Content Addressable Memory) 또는 재구성이 가능한 FPGA (Field Programmable Gate Arrays) 등이 주로 사용되고 있다. TCAM은 “don't care” 비트를 이용하여 3가지 상태를 표현할 수 있기 때문에 파싱보다는 질의(query) 작업에 보다 효율적이며, 가격이 비싸고 높은 전력을 소비하는 단점을 갖는다. 반면에 FPGA는 gate 집합으로 구성되어 있어 파서의 설계 및 구조 변경이 용이하다. 그러나, TCAM과 비교하여 상대적으로 낮은 성능을 갖기 때문에, 본 논문에서는 ASIC (Application Specific Integrated Circuit)이나 TCAM에 상응하는 성능을 갖도록 다중바이트 단위로 파싱을 수행하는 고속의 XML 하드웨어 파서를 제안하며, XML의 DTD나 XMLSchema등은 고려하지 않는다.

본 논문의 Ⅱ장에서는 XML 파서들에 대한 관련 연구들을 살펴보고, Ⅲ장에서는 제안된 구조의 하드웨어 구조와 동작을 기술한다. Ⅳ장에서는 제안된 스트리밍 XML 하드웨어 파서의 성능을 평가하고 Ⅴ장에서 결론을 맺는다.

Ⅱ. 관 련 연 구

실시간으로 입력되는 XML 문서의 파싱 및 검색을 위해서 스트리밍 파서들이 널리 사용되고 있다. 스트리밍 파서는 입력과 동시에 파싱을 수행하기 때문에 고속으로 동작하지만, 이미 파싱된 XML 엘리먼트의 재접근이 어려운 단점을 갖는다.

RBStreX[3]는 스트리밍 파서로서 Roll-Back을 이용하여 이전에 파싱된 엘리먼트에 대한 재접근의 어려움을 해결하였다. 그림 1의 getNext 명령은 각 엘리먼트를 얻고자 할 때 사용되는 명령어로써, 엘리먼트를 파싱할 때 마다 소프트웨어에서 파서로 입력되기 때문에 오버헤드가 발생한다. 또한, XML의 파싱이 순차 처리되기 때문에 전체적인 성능 향상을 기대하기 어렵다.

순차 처리되는 스트리밍 파서의 단점을 보완하기 위해 DOM 트리 구조를 변형한 StrexTree[4] 파서가 제안되었다. StreXTree 파서는 DOM 트리에서 단말 노드가 되는 엘리먼트 세트(시작엘리먼트, 컨텐트, 종료엘리먼트)를 상위 엘리먼트의 자식 노드 대신에 형제 노드로

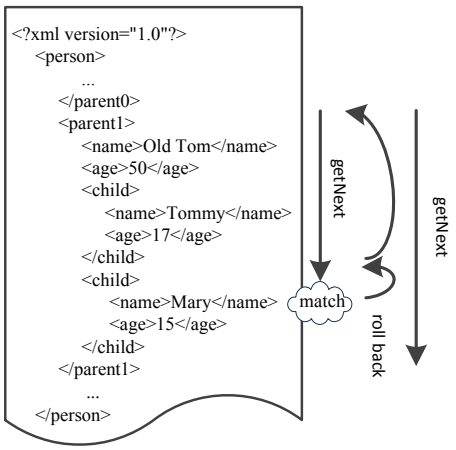

그림 1. 데이터 검색을 위한 XML 시나리오
Fig. 1. A scenario of an XML for data search.

```
QUERY1: for $x in doc("person.xml")/person
        return if ($x/name="Mary")
QUERY2: for $x in doc("person.xml")/person/parent/child
        return if ($x/age)
```

그림 2. 검색 시나리오를 위한 XQuery 문법
Fig. 2. XQuery syntax for the search scenario.

연결하여 검색 속도의 성능 향상을 이루었다. 그렇지만, 여전히 단일 바이트 단위로 XML 문서를 파싱하기 때문에 실시간 처리 속도를 기대하기 어렵다. 이전의 연구들은[3~4] 검색 속도의 성능을 비교하기 위해, 그림 1의 XML에서 이름이 “Mary”인 엘리먼트의 “parent”를 검색하여 모든 “child”에 대한 “age”들을 찾는 시나리오를 사용하였다.

본 논문에서는 이전 연구들에서 사용한 시나리오에 대한 가독성을 높이기 위해, XML에서 널리 사용되고 있는 XPath 및 XQuery (XMark Query)[5]를 이용하였으며 질의들을 재 정의하여 그림 2에 나타내었다. 그림 2의 Q_1 ($QUERY_1$)은 “person.xml” 파일에서 “person” 엘리먼트의 하위 엘리먼트들을 검색하여 이름이 “Mary”인 엘리먼트를 반환하는 질의 표현이다. Q_2($QUERY_2$)는 “person.xml” 파일에서 “person/parent/child” 하위의 모든 “age” 엘리먼트들의 값을 반환하는 질의 표현이다.

본 논문에서는 순차 처리의 한계성을 해결하고 전처리 및 병합과정이 요구되지 않는 다중바이트 기반 스트리밍 하드웨어 파서(MStreXHP: Multibyte based Streaming XML Hardware Parser)를 제안한다. MStreXHP 파서는 상태머신을 사용하지 않는 새로운

인코딩 기법을 적용하여 엘리먼트를 검색할 수 있는 하드웨어 구조이다.

Ⅲ. 다중바이트 기반 스트리밍 XML 파서

1. 제안된 하드웨어 파서의 시스템 구조

XML 문서의 순차적 파싱은 단일 문자 단위로 수행되기 때문에 파서의 속도 향상에 한계성을 갖는다. 따라서, XML 파서의 고속 동작을 위해 다중바이트 기반 파싱이 요구된다. 대다수의 XML 파서들은 상태머신을 이용하여 입력되는 문자에 따른 전이를 통해 파싱을 수행한다. 만일, 다중바이트 단위로 파싱을 한다면 입력 문자열에 대한 상태 수가 급격히 증가하여 회로 복잡도가 높아지고 병목현상이 발생할 수 있다.

제안된 MStreXHP 파서는 상태머신 대신에 인코딩된 엘리먼트 정보를 이용하여 다중바이트 단위로 파싱한다. 그림 3은 본 논문에서 제안하는 다중바이트 기반 하드웨어 파서의 전체 시스템 블록도이다. 일반적으로 이더넷 인터페이스는 32비트 단위로 처리하기 때문에 본 논문에서는 4바이트 단위의 검색 및 파싱 하드웨어를 제시한다. 제안된 파서는 4개 문자단위로 그림 3의 input 레지스터로 입력되고, 다음 클록에서 새로운 4개 문자가 입력되면 이전 문자들이 buf 레지스터로 이동된다. 두 개의 레지스터들로부터 엘리먼트 분석기들과 바이트정렬 레지스터로 문자들이 제공되면 파싱은 시작된다. 입력된 문자들을 이용하여 시작엘리먼트 분석기와 종료엘리먼트 분석기에서는 미리 정의된 인코딩정보와 비교하여 결과를 출력한다. 만일 시작엘리먼트 분석기에서 시작엘리먼트가 검색되면 해당 위치의 벡터에 1이 출력된다. 벡터의 값들 중 하나 이상이 1인 경우, 해당

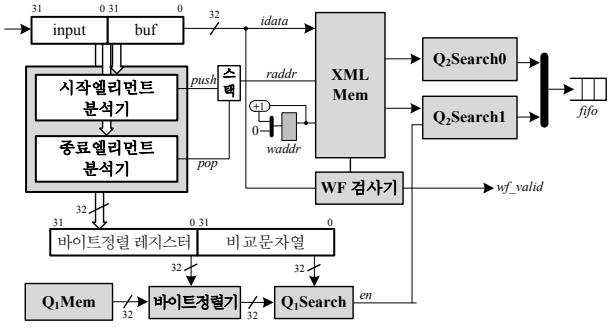

그림 3. MStreXHP의 하드웨어 구조
Fig. 3. Hardware architecture of MStreXHP.

엘리먼트가 저장되어야 하는 XML 메모리의 주소를 스택에 push한다. 종료엘리먼트 분석기의 동작은 시작엘리먼트 분석기와 동일하며 벡터 값에 1이 있는 경우, 스택의 최상위에 저장된 XML 메모리 주소가 pop된다. 또한, 스택을 사용하여 엘리먼트들의 중첩구조 및 계층구조를 나타낼 수 있는데 이는 RBStreX[3]의 구조와 유사하다.

Q_1 질의 검색은 Q_1Mem에 저장된 검색 조건들과 입력되는 4개의 문자가 동시에 비교되도록 한다. Q_1Mem은 소량의 메모리로서 소프트웨어에서 검색 명령어와 함께 질의 내용이 제공되어 시작엘리먼트의 시작태그(“<”)부터 순차적으로 Q_1Mem에 저장된다. 4개의 문자를 비교하기 위해서는 4개의 4바이트 비교기가 요구된다. 이때에 Q_1Mem에는 엘리먼트 시작태그부터 저장되어 있지만, 입력 문자열은 임의의 위치에서 시작태그가 올 수 있기 때문에 바이트 단위의 정렬이 필요하다. 바이트 정렬이 완료된 후에는 Q_1Search에서 매 클록마다 4바이트 단위로 Q_1 질의를 비교할 수 있다. Q_1질의에서 매칭이 성공하면 두 개의 병렬 검색기에서 Q_2 질의 검색이 수행된다. Q_2의 검색은 Q_2Search0 회로에서 루트 엘리먼트 부터 Q_1의 매칭이 성공한 엘리먼트까지 비교되며, Q_2Search1 회로는 그 다음 엘리먼트부터 XML 문서의 마지막 엘리먼트까지 비교한다. 두 개의 검색기에 대한 결과는 *fifo*에 의해 순차적으로 출력된다.

2. 엘리먼트분석기

다중바이트 기반 파싱에서 상태머신의 사용은 파서 회로의 복잡도를 높이고 성능 열화를 발생시킨다. 본 논문에서는 낮은 회로 복잡도를 가지며 파싱 성능을 높이기 위해서 XML의 태그들과 대응되는 2비트의 인코딩 값들을 사용한다. 모든 엘리먼트는 2개 문자 단위로 인코딩된 2비트와 비교되는데, 이는 XML의 시작엘리먼트는 “<”로 시작하지만, 종료엘리먼트가 “</”로 구분되기 때문이다. 입력 문자에 대한 인코딩 값은 표 1과 같다.

표 1에서 정의된 인코딩 비트들은 시작엘리먼트 분석기와 종료엘리먼트 분석기에 제공되어 그림 4와 같이

표 1. 문자에 대한 인코딩 값
Table 1. Encoding value for a character.

시작태그(“<”)	닫힘태그(“/”)	종료태그(“>”)	나머지
01b	10b	11b	00b

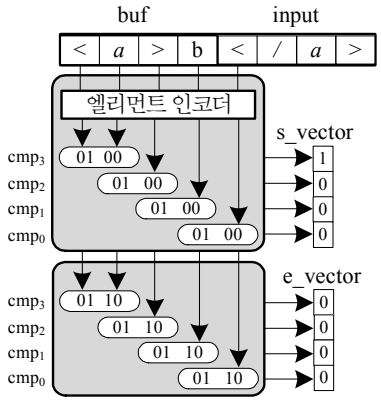

그림 4. 엘리먼트분석기
Fig. 4. Element analyzer.

해당 입력문자열에 대하여 시작엘리먼트와 종료엘리먼트를 판별한다. 비교된 4개의 결과들은 각각 시작엘리먼트 벡터(s_vector)와 종료엘리먼트 벡터(e_vector)에 저장된다. s_vector와 e_vector는 그림 2의 스택을 위한 push와 pop 신호로 사용되며, 제안된 구조에서 스택은 Well-Formed (WF) 유효성 검증을 위하여 사용된다.

그림 4에서 입력된 문자열들은 두 개의 문자 단위로 정의된 인코딩 비트들과 비교하여 엘리먼트의 종류를 판별 할 수 있다. 예를 들어, cmp_0 비교기는 현재 buf 레지스터의 마지막 문자와 input 레지스터의 첫 문자를 비교하여 s_vector에 대한 출력을 결정할 수 있다. 만일 s_vector의 bitwise-OR 연산이 1이면 시작엘리먼트가 저장되어야 하는 Xmem(XML Memory)의 주소를 스택에 저장한다. 마찬가지로, e_vector의 값은 스택에서 pop 신호로 사용된다.

3. 바이트정렬기 및 4바이트 비교기

그림 3에서 바이트정렬 레지스터에 저장된 입력문자열은 Q_1Mem에 저장된 Q_1 질의 조건과 처음 클록에서 비교되고, 다음 클록부터 4바이트 비교기에서 매칭 작업이 수행된다. 다중바이트 기반 매칭에서 바이트정렬기의 필요성은 질의 조건이 항상 엘리먼트의 시작태그부터 저장되어 있고, 입력문자열이 XML의 임의의 위치에서 시작될 수 있기 때문이다. 예를 들어, Q_1Mem에 저장된 Q_1 질의 조건이 (“<name>Mary”)이라 할 때, 그림 5와 같이 입력문자열이 구성된다면 처음 클록에서 cmp_0을 통해 매칭이 성공하며 다음 클록부터는 4개의 문자단위로 비교 된다. 여기에서는 “e>ma”가 다음 클록에 비교된다. 만일 매칭이 실패하면 7개 문자가 쉬프트되고, 매칭이 성공하면 4개의 비교기에 대한 출력 벡터에 의해 쉬프트가 결정된다.

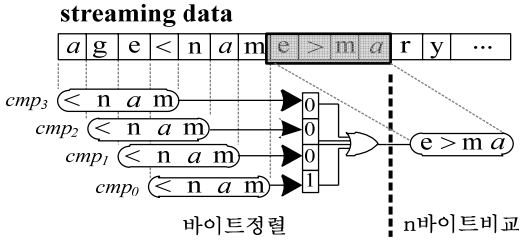

그림 5. 바이트 정렬 및 4바이트 비교기
Fig. 5. Byte alignment and 4 bytes comparator.

Ⅳ. 평 가

스트리밍 파서들에 대한 평가는 소프트웨어에서 제공되는 명령어 수 및 파싱과 검색이 완료 될 때까지의 클록 수를 비교한다. 제안된 MStreXHP 파서의 성능 평가 및 비교를 위해, 그림 1의 XML 문서를 이용한다. 그림 1의 XML 문서에서 전체 문자 수를 N, 파서에서 실제 비교되는 문자수를 m, 그리고 소프트웨어에서 제공되는 명령어 수를 k라 정의한다.

표 2에는 XML 파서들의 질의 검색을 위한 소요 시간을 분석하여 일반화하였다. RBStreX 파서는 Q_1과 Q_2 질의 검색을 위해 모든 엘리먼트 마다 getNext 명령어가 제공되고 엘리먼트의 문자 비교 및 결과 판별이 XML 파서와 소프트웨어에서 각각 발생하기 때문에 Q_1 질의에서 2m, Q_2 질의에서 2N의 비교가 이루어진다. StreXTree 파서는 Q_1 질의에서 한 번의 명령어가 제공되고, 결과 판별을 위한 한 번의 클록이 소요된다. 표 2를 보면, StreXTree가 Q_1 질의에서 RBStrex 보다 많은 문자 비교를 수행하지만, 이는 XML을 StreXTree로 변환하기 위하여 마지막 엘리먼트까지 파싱하기 때문이다. Q_2 질의에서는 문서의 모든 문자를 비교하지 않아 RBStrex 파서 보다 우수한 성능을 갖는다.

제안된 MStreXHP 파서는 명령어 제공 및 결과 출력

표 2. 질의들을 위한 검색 소요시간 분석
Table 2. Analysis of search time for the Queries.

파서	$Query_1$	$Query_2$
RBstreX[3]	$2m + k$	$2N + k$
STreXTree[4]	$N + 2$	$m + k$
Ours	$\lceil m/4 \rceil + 1 + 2$	$\lceil N/4 \rceil + 1 + 2$

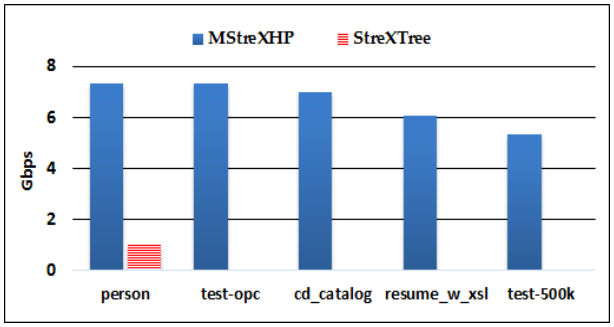

그림 6. 스트리밍 XML 파서의 성능 비교
Fig. 6. Performance Comparison of streaming XML parsers.

표 3. 다른 연구들과의 성능 비교
Table 3. Comparison with other works.

파서	QUERY1			QUERY2			WF 검사	Tot. (clk)
	ch	cmd	Q_{1CLK}	ch	cmd	Q_{2CLK}		
RBStreX[3]	153	26	334	234	40	510	×	844
StreXTree[4]	234	2	236	40	2	42	○	278
Ours	153	2	41	153	2	61	○	102

에 각 1클록이 소요되며, Q_1 질의에서 m 개의 문자를 4 바이트 단위로 비교하기 때문에 총 $\lceil m/4 \rceil + 1$의 클록을 소비한다. 또한 Q_2 질의는 두 개의 검색기가 병렬로 처리되기 때문에, Q_2 질의의 소비클록을 Q_{2CLK}이라 하면 식 (1)과 같이 소비 클록을 추정 할 수 있다. 여기에서, p는 Q_1 질의가 성공한 엘리먼트까지 비교한 문자 수이며, q는 그 엘리먼트 이후부터 문서 마지막까지 존재하는 문자 수를 의미한다.

$$Q_{2CLK} = \lceil p/4 \rceil + 1, \quad \text{if } N = p + q, (p \geq q) \tag{1}$$

식 (1)에서 최악의 경우 Q_1 질의의 대상 엘리먼트가 문서의 마지막에 위치한다면 p=N-r이 될 수 있다. 여기에서 r은 마지막 종료엘리먼트의 문자 길이를 나타낸다. 그러므로, p를 N이라 가정하여 제안된 MStreXHP 파서에서의 전체 검색 시간을 추정하여 표 2에 나타내었다.

표 3은 표 2의 검색 소요 시간을 이용하여 그림 1의 XML에 대한 실제 소비되는 클록 수를 이전 연구들과 비교한 것이다. 표 3에서 *ch*는 비교되는 문자수를 나타내며, *cmd*는 소프트웨어에서 제공되는 명령어 수, 그리고 Q_{1CLK}과 Q_{2CLK}는 검색이 성공될 때까지의 소비 클록 수를 의미한다. 제안된 MStreXHP 파서는 이전 연구들과 비교하여 약 2.72배에서 최대 8.27배까지 소비 클록이 감소하여 향상된 성능을 갖는다.

표 4. MStreXHP의 합성결과
Table 4. Synthesis report of MStreXHP.

BenchMark	Size(KB)	ALUT	Memory(bits)	Fmax(MHz)
test-opc.xml	1.79	384	17,480	228.57
cd_catalog	5.66	387	33,872	218.87
resume_w_xsl	50.5	391	525,424	188.89
test-500k	500	480	4,195,456	166.97

제안된 MStreXHP 파서는 Altera의 Stratix IV FPGA 디바이스를 사용하였고 합성 도구는 QuartusII 13.0을 이용하였다. 제안된 구조의 하드웨어 합성을 위하여 그림 1의 XML 예제 대신에 일반적으로 XML 파서의 평가를 위해 사용되는 벤치마크 파일들을 이용하였으며, 합성결과는 표 4에 나타내었다. 표 4에서 ALUT는 Altera의 Stratix IV FPGA를 이루는 로직 배열의 기본 단위를 의미한다.

이전 연구들은 그림 1의 작은 XML 예제만 사용하였기 때문에 합성 결과의 정량적 비교가 어렵지만 StreXTree에서 제시한 합성결과를 이용하여 그림 6에서 성능을 비교하였다. 또한, 본 논문에서 사용된 벤치마크용 파일들을 이용하여 합성한 결과를 토대로 시스템 성능을 기술하였다. 그림 6에 따르면, 제안된 구조는 StreXTree와 비교하여 하드웨어 처리 성능이 약 2.55배 향상되었다. 비록 XML 문서의 크기가 500KB로 증가하였을 때 하드웨어 성능이 저하되지만 이는 XML 문서가 수 백개의 FPGA 내의 메모리에 분산되어 저장되어 처리 주파수가 감소하기 때문이다. 그러나, 대부분의 FPGA 보드에서 제공하는 이더넷 처리속도가 125MHz임을 감안한다면, 제안된 MStreXHP 파서는 실시간으로 입력되는 XML을 충분히 처리할 수 있다.

V. 결 론

XML 파서들에서 문서의 파싱은 계산 집중적 작업으로써 파서의 성능을 결정하는 가장 중요한 요소이다. 일반적으로 스트리밍파서와 DOM 기반 파서들이 널리 사용되는데 스트리밍 파서는 전처리 작업이 요구되지 않고 적은 메모리를 사용할 수 있는 구조를 갖는다.

본 논문에서는 파싱 이전에 전처리가 요구되지 않고 메모리 사용을 줄일 수 있는 스트리밍 파서에 대한 하드웨어 구조를 제안하였다. 제안된 MStreXHP 구조는

순차 처리 파싱을 이용하는 스트리밍 파서의 단점을 개선한 다중바이트 기반 고성능 스트리밍 XML 하드웨어 파서이다.

제안된 MStreXHP 파서는 이전 스트리밍 하드웨어 파서들과 비교하여 약 2.72~8.27배 감소된 클록 수를 소비하였고, 하드웨어의 전체적인 성능에서도 약 2.55배 향상되었다. 또한 1.79KB~500KB의 크기를 갖는 XML 문서를 처리할 수 있음을 하드웨어 합성 결과를 제시하여 검증하였다. 그러므로, 제안된 다중바이트 기반 스트리밍 XML 파서 구조는 시스템의 전체 성능과 파서 안에서의 소비 클록이 감소된 효율적인 스트리밍 XML 하드웨어 파서이다.

REFERENCES

[1] Dai, GuiPing, and Yong Wang. “XML-Based Structural Representing Method for Information of Things in Internet of Things.” Advances in Computer Science and Information Engineering. Springer Berlin Heidelberg, pp. 9-15. 2012.

[2] E. R. Harold, “An Introduction to StAX,” [Internet].Available: http://www.xml.com/pub/a/2003/01/17 /stax.html.

[3] C. E. Chang, M. Faisal, and K. M. Azhar, “RBStreX: Hardware XML parser for embedded system.” Internet Technology and Secured Transactions, ICITST, 2009.

[4] Kyu-Hee Lee and Sang-Soo. Han, “A Streaming XML Hardware Parser using a Tree with Failure Transition”, Journal of Korea Inst. Inf. Commun. Eng., Vol. 17, No. 10, pp. 2323~2329, Oct. 2013.

[5] XQuery Tutorial, http://www.w3schools.com/XQuery.

저 자 소 개

이 규 희(정회원)

2007년 연세대학교 컴퓨터정보통신공학부 학사 졸업.
2009년 연세대학교 전산학과 석사 졸업.
2015년 연세대학교 전산학과 박사 졸업.
2015년~현재 상지영서대학교 국방정보통신과 조교수

<주관심분야 : 네트워크보안, 임베디드시스템, FPGA>

서 병 석(평생회원)

2001년 연세대학교 전산학과 학사 졸업.
2008년 연세대학교 전산학과 석사 졸업.
2011년 연세대학교 전산학과 박사 수료.
2012년~현재 상지영서대학교 국방정보통신과 조교수

<주관심분야 : 병렬처리, 임베디드시스템, GPU, 클라우드컴퓨팅>