JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 17, No. 8 : 1851~1856, August. 2013

적응적 병렬 검색을 지원하는 스트리밍 XML 파서

이규희[1] · 한상수[2*]

A Streaming XML Parser Supporting Adaptive Parallel Search

Kyu-Hee Lee[1] · Sang-soo Han[2*]

[1] Department of Computer and Telecommunications Engineering, Yonsei University, Wonju 220-710, Korea
[2] Department of Energy IT, Gachon University, Seongnam, 461-701, Korea

요 약

XML은 SOAP(Simple Object Access Protocol)와 REST(Representational State Transfer) 같은 웹서비스들을 위해 널리 사용되며 데이터를 표현하기 위한 사실상의 표준이다. DOM(Document Object Model)을 사용하는 XML 파서는 DOM 트리로 생성하여 메모리에 저장하는 전처리가 요구되기 때문에, 제한적 자원을 갖는 임베디드 시스템들은 일반적으로 전처리를 요구하지 않는 스트리밍 XML 파서를 채택하여 사용하고 있다. 본 논문에서는 FPGA(Field Programmable Gate Array)에서 적응적 병렬 검색을 사용하는 스트리밍 XML 파서를 위한 새로운 구조를 제안한다. 제안된 APSearch(Adaptive Parallel Search) 파서는 이전 연구들과 비교하여 소프트웨어의 오버헤드를 상당 수 감소시켰으며 XML 파싱을 위한 처리 시간이 약 2.55배와 2.96배 향상되었다. 따라서, 제안된 APSearch 파서는 XML 파싱을 가속화하기 위한 시스템들에 적합한 구조이다.

ABSTRACT

An XML is widely used for web services, such as SOAP(Simple Object Access Protocol) and REST (Representational State Transfer), and also de facto standard for representing data. Since the XML parser using DOM(Document Object Model) requires a preprocessing task creating a DOM-tree, and then storing it into memory, embedded systems with limited resources typically employ a streaming XML parser without preprocessing. In this paper, we propose a new architecture for the streaming XML parser using an APSearch(Adaptive Parallel Search) on FPGA(Field Programmable Gate Array). Compared to other approaches, the proposed APSearch parser dramatically reduces overhead on the software side and achieves about 2.55 and 2.96 times improvement in the time needed for an XML parsing. Therefore, our APSearch parser is suitable for systems to speed up XML parsing.

키워드 : FPGA, 스트리밍 XML파서, 하드웨어 파서, 병렬검색

Key word : FPGA, Streaming XMLParser, Hardware Parser, Parallel Search

접수일자 : 2013. 04. 02 심사완료일자 : 2013. 04. 29 게재확정일자 : 2013. 05. 15

*** Corresponding Author** Sang-Soo Han(E-mail: sshan@gachon.ac.kr, Tel:+82-31-750-5587)
Department of Energy IT, Gachon University, Seongnam, 461-701, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2013.17.8.1851 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

오늘날 대부분의 웹서비스들은 데이터를 표현하기 위해 구조화된 표준 문서인 XML(eXtensible Markup Language)을 사용하고 있다. 일반적으로 널리 사용되고 있는 SOAP(Simple Object Access Protocol)와 REST (Representational State Transfer)는 XML 기반의 메시지를 전달하는 웹서비스 방식이다[1][2].

XML의 파싱을 위해 현제 제공되는 API들은 이벤트 기반과 트리 기반으로 분류된다[3]. 대표적으로 SAX(Simple API for XML)[4]와 DOM(Document Object Model)이 여기에 속한다. 트리 기반 API는 문서 전체를 메모리에 적재하는 전처리 작업 후 파싱이 수행되어 데이터의 사용과 접근이 용이하지만 제한적 자원을 갖는 시스템들에는 적합하지 않다. 반면에 스트리밍 파서라고도 불리는 이벤트 기반 API는 입력되는 XML 문서를 작은 단위로 저장하여 입력과 동시에 처리하기 때문에 메모리에 적재하는 전처리 과정이 요구되지 않아 제한적 자원을 갖는 시스템들에 널리 사용되고 있다[5][6].

파싱이 순차 처리되는 소프트웨어 파서들은 성능의 한계성을 갖는데 이를 해결하기 위해 하드웨어 파서들이 연구되고 있다. 하드웨어 파서들은 높은 성능을 갖는 반면에 하드웨어 구조의 변경이 어렵기 때문에 재구성이 가능한 FPGA(Field Programmable Gate Arrays)의 사용이 일반적이다.

스트리밍 파서는 전체 XML 문서를 저장하지 않기 때문에 이전 데이터의 접근이 어렵고 스트리밍의 특성 때문에 파싱을 순차 처리하여 성능향상에 제약이 있다. 본 논문에서는 FPGA를 사용하여 스트리밍 XML을 처리하고 순차 처리 문제점을 해결할 수 있도록 적응적 병렬 검색이 가능한 하드웨어 기반 스트리밍 XML 파서를 제안한다.

본 논문의 2장에서는 XML의 기본 구성 요소들과 관련 연구들을 살펴본다. 3장에서는 제안된 구조의 하드웨어 및 동작을 서술하고, 4장에서는 제안된 파서의 평가와 합성결과를 제시하며, 5장에서 결론을 맺는다.

Ⅱ. 관련연구

XML은 구조화된 문서로서 그림 1과 같이 데이터를 엘리먼트의 쌍으로 표현한다. XML 데이터의 사용을 위해 파싱이 요구되는데, 제한적 메모리를 가지며 성능이 중요한 시스템에서는 스트리밍 XML 파서의 사용이 일반적이다[6]. XML의 입력과 동시에 파싱을 수행하는 스트리밍 파서는 전처리가 요구되지 않으며, 소프트웨어의 요청에 따라 결과를 응답하는 이벤트 기반 모델이다.

순차 처리되는 스트리밍 파서의 단점을 보완하기 위해, [5]의 저자들은 XML을 위한 하이브리드 병렬 처리 구조를 제시하였으나, 소프트웨어 파서의 한계성 때문에 높은 성능을 기대하기 어렵다. [6]의 저자들은 Schema 등에 대한 유효성 검사 등의 검증을 제외하고 파싱 집중적 하드웨어 파서를 제시하였으며 파싱된 엘리먼트에 재접근이 가능하도록 롤백 기능을 추가한 RBStreX(Roll-Back Streaming XML)를 제안하였다. RBStreX에서 제공된 명령들은 다음과 같다.

- ***getNext*** : XML을 엘리먼트 단위로 파싱하여 <엘리먼트 종류, 엘리먼트 값>을 반환.
- ***rollBack*** : 검색 성공 시, 파싱된 문서의 최상위 엘리먼트로 이동.

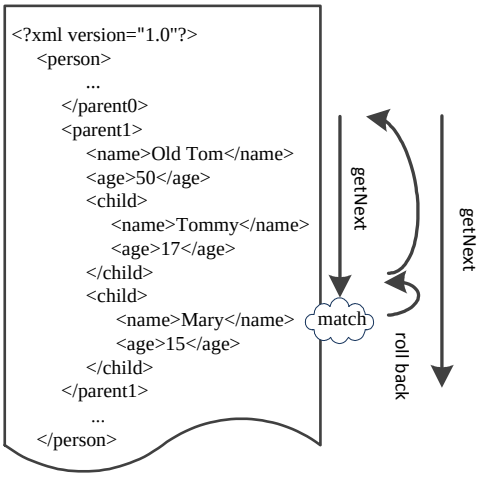

그림 1. RBStreX를 사용하는 시나리오
Fig. 1 Scenario of using RBStreX

RBStreX 구조는 검증 및 평가를 위해서 그림 1의 시나리오를 사용하였다. 본 논문에서는 그림 1에서 이름이 “Mary”인 엘리먼트를 찾는 과정을 **조건검색**이라 하고 “Mary” 부모에 대한 모든 자식들의 나이를 찾는 과정을 **결과검색**이라 표현한다. 조건검색 및 결과검색을 위해 RBStreX는 getNext 명령을 제공받아 처리하고 결과를 반환하여 엘리먼트의 종류와 값을 소프트웨어에서 판단하도록 하였는데, 이는 파서가 아닌 소프트웨어에서의 오버헤드가 된다.

RBStreX의 오버헤드 단점을 극복하기 위해 [7]의 저자들은 조건검색을 위한 Search 명령을 제안하였다. Search 구조는 소프트웨어에서 한 번의 명령을 제공하여 파서에서 결과를 판단하도록 구현하였기 때문에 조건검색에서 RBStreX의 오버헤드를 두 배 감소시켰다. 그러나 결과검색에서 RBStreX와 동일하게 getNext를 사용하여 전체시스템의 성능향상은 기대하기 어렵다. 게다가, [6][7]의 파싱은 순차적으로 처리된다.

본 논문에서는 [6][7]의 구조에서 발생되는 오버헤드를 감소시키고 순차 처리 문제를 해결할 수 있도록 적응적 병렬 처리를 수행하는 하드웨어 파서를 제시한다.

Ⅲ. 하드웨어 기반 스트리밍 XML 파서

본 장에서는 순차 처리에 대한 성능저하를 막고 적응적 병렬처리가 가능한 새로운 명령어를 제시하고 적응적 병렬 검색을 수행하는 APSearch (Adaptive Parallel Search) 파서에 대한 동작 절차와 하드웨어 구조를 기술한다.

3.1. APSearch 파서의 동작

제안된 APSearch 파서는 소프트웨어로부터 입력된 명령에 따라 다른 작업을 수행하도록 설계되었다.

입력 명령에 따른 APSearch 파서의 흐름은 그림 2에 제시된 상태도에 기반하며, 여기에서 gNext와 Srch 그리고 pSrch는 각각 getNext, Search, pSearch(Parallel Search)를 의미한다. 그림 2의 상태도에 기반을 둔 APSearch 파서의 동작은 다음과 같은 일련의 절차를 따른다.

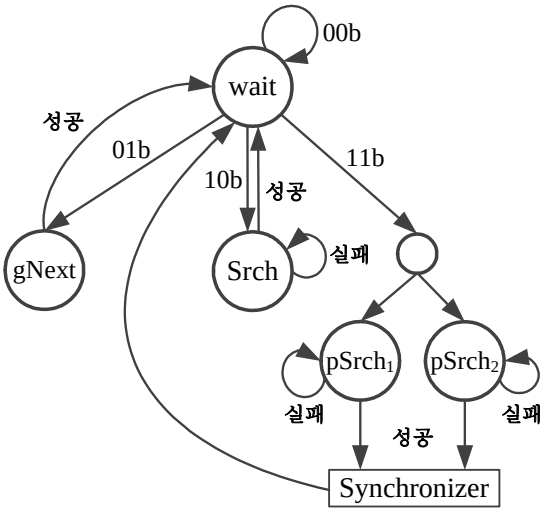

그림 2. 시스템 동작을 위한 상태도
Fig. 2 State diagram for system operation

Step 1. 조건검색(10b) 수행.
Step 2. 검색 성공 시, 병렬 검색(11b) 수행.
Step 3. 결과검색을 위해 Step2의 검색된 엘리먼트를 기준으로 pSrc1과 pSrc2가 병렬 수행.
Step 4. 하나의 검색이 성공한 경우, 다른 검색의 성공 또는 실패까지 동기화 블록에서 대기.
Step 5. 입력된 XML에서의 결과검색 완료 될 때까지 Step 2부터 반복 수행.

위와 같은 일련의 절차들은 그림 3과 같은 시나리오로 표현되며, 그림 3에서 병렬 검색이 가능하도록 이름이 “Shelly”인 하나의 자식 엘리먼트를 추가하였다.

3.2. APSearch 파서의 하드웨어 구조

제안된 APSearch 파서의 시스템 블록도는 그림 4와 같으며 음영으로 표시된 메모리들과 간단한 회로들로 구성된다. APSearch 파서는 XML 문서의 저장과 검색 및 출력을 위해 메모리를 사용하며, 그림 4에서 두 개의 pSearch 블록은 검색을 위한 메모리, 직렬화 블록은 검색결과를 저장하기 위한 직렬화 메모리를 의미한다.

그림 4의 SM(State Machine) 블록은 XML에서의 엘리먼트 종류와 콘텐트를 구분하는 상태머신이며 XML 파서의 기본 구성단위이다.

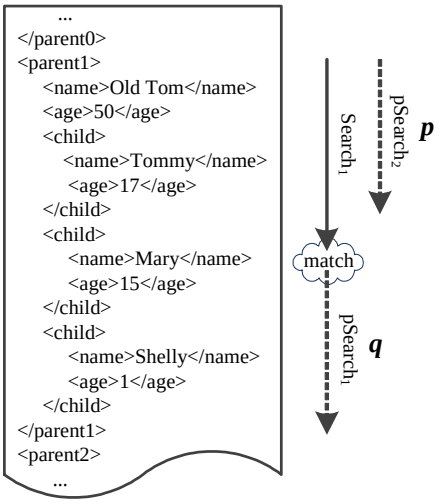

그림 3. APSearch를 사용하는 시나리오
Fig. 3 Scenario of using APSearch

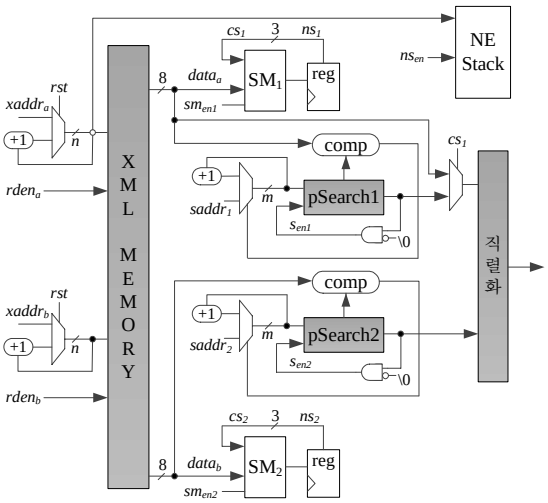

그림 4. APSearch 파서 블록도
Fig. 4 Block diagram of APSearch parser

제시된 SM은 3비트 상태레지스터를 이용하여 현재 상태에서 입력된 문자에 의해 다음 상태를 결정한다. 그림 5는 SM 블록의 상태도로서 원 안의 숫자는 현재 상태를 나타낸다. 상태 2는 시작 엘리먼트, 상태 6은 종료 엘리먼트, 상태 3에서 상태 1 의 전이는 콘텐트 파싱이 완료되었음을 의미한다.

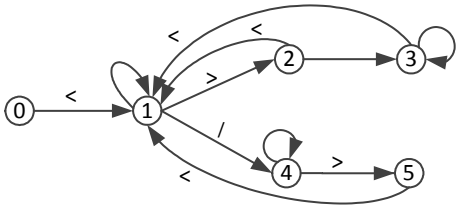

그림 5. 파싱을 위한 상태머신
Fig. 5 State machine for parsing

그림 4의 NE(Nested Element) Stack은 RBStreX의 방법과 유사하지만 본 논문에서는 결과검색에서 각 병렬 검색기의 시작과 종료 조건으로 사용한다. NEStack은 시작 엘리먼트(“<”)에서 시작주소를 push하고 종료 태그(“/”)에서 pop을 수행한다.

예를 들어, 그림 3에서 “Mary”가 검색되었을 때, NEStack에는 그림 6(a)와 같이 저장되어 있다. 이 때에 “Parent1”의 시작주소는 그림 3의 p에 대한 시작 위치가 되고 q에 대한 시작위치는 “Mary”+1이 되며, NEStack이 empty가 되면 종료된다.

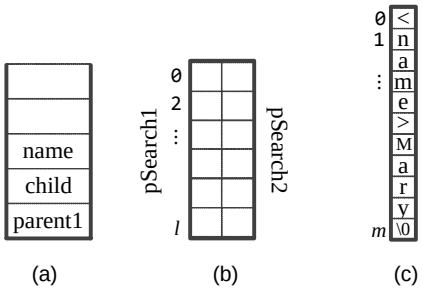

그림 6. APSearch 파서의 메모리 (a) NE 스택 (b) 직렬화 메모리 (c) 검색메모리
Fig. 6 Memroy of APSearch parser (a) NE Stack (b) Serial Memroy (c) Search Memroy

3.3. 적응적 병렬 검색

본 논문에서 제안된 APSearch 파서는 조건검색에서 pSearch1 블록만 활성화 되고 결과검색에서 병렬 검색을 수행하도록 pSearch2 블록이 함께 활성화된다. 검색 메모리는 조건검색과 결과검색에 필요한 문자들을 동적으로 저장하는데 그림 6(c)와 같이 검색 종료를 나타내는 null을 추가로 삽입한다. 입력되는 각 클록에서 XML 메모리의 문자와 검색메모리의 문자가 비교되며 두 문자의 내용이 일치하면 검색은 진행되고, 검색메모

리의 내용이 null이 되면 검색은 성공한다.

결과검색이 성공하였을 때 소프트웨어에 결과들을 반환하기 위해 파싱된 엘리먼트는 직렬화 블록에 저장된다. APSearch 파서의 직렬화 블록은 병렬 검색에서 발생된 두 개의 결과를 동시에 저장하도록 FPGA에 내장된 Block RAM(BRAM)을 이용하여 듀얼포트 RAM으로 구성된다. 이에 대한 구조는 그림 6(b)에 제시 하였으며 두 개의 결과를 동시에 저장해야 하는 경우에도 메모리 경합은 발생하지 않는다. APSearch 파서에서 각 블록들이 활성화되기 위한 조건들은 표 1과 같다.

표 1. 각 블록을 활성화하기 위한 조건
Table. 1 Conditions to enable each block

en신호	조건	en신호	조건
rdena	cmd[1]	sen1	cmd[1]
rdenb	cmd[0]	sen2	cmd[1]&cmd[0]
smen1	always	nsen	cmd[1]
smen2	cmd[1]&cmd[0]		

Ⅳ. 평 가

본 장에서는 제안된 APSearch 파서와 이전의 하드웨어 XML 파서들에 대한 성능 비교는 표 2와 같다. getNext는 파서에 제공되는 명령어 횟수 k와 결과를 소프트웨어에서 판단하는 시간 s가 소요된다. 파서에서 검색이 성공될 때까지 메모리에서 인출된 문자 수 n, 결과를 출력하기 위해 n 클록 사이클 소요된다.

Search는 검색의 성공까지 한 번의 명령만 제공되며 메모리로부터 문자 인출을 위해 n 클록 사이클 소요된다. 결과 반환은 성공을 의미하는 1비트로서 1 클록 사이클이 소요된다.

본 논문에서 제시한 pSearch 명령은 소프트웨어로부터 “Mary” 부모의 자식들의 수만큼 제공되고 이를 e라 정의한다. 두 개의 XML로 분리되어 pSearch1과 pSearch2에서 동시에 처리되므로 검색이 성공 할 때까지 메모리에서 인출된 문자열은 최대 p가 되며, 결과 검색에서 소프트웨어에 반환되는 문자열 수를 m이라 할 때에, n과 k에 대하여 식(1)의 관계가 성립된다. $\boldsymbol{p}$와 $\boldsymbol{q}$는 그림 3에서의 각각 분리된 XML을 나타낸다.

$$n = p + q,\ p \geq q \quad m \ll n \text{ and } e \ll k \tag{1}$$

표 3은 성능 비교를 위해 표 2에서 제시한 식에 기반을 두어 그림 3의 XML을 적용한 결과이다. Search 파서는 조건검색에서 Search 명령을 수행하여 RBStreX와 비교하여 2배 이상의 성능 향상이 있는 반면에 결과검색에서 getNext를 사용하여 동일한 성능을 갖는다. 제안된 APSearch 파서는 조건검색에서 Search 파서와 동일한 성능을 가지며, 결과검색에서 s를 제외하더라도 약 4.3배의 소비 클록이 감소되었다.

표 2. 성능 비교
Table. 2 Performance comparison

	RBStreX	Search	APSearch
조건검색	2n+ks	n+2	n+2
roll_back	2	2	1
결과검색	$2n+\sum_{i=1}^{k} s_i$	$2n+\sum_{i=1}^{k} s_i$	$p+\sum_{i=1}^{e} m_i$

표 3. 그림 3의 예제를 이용한 파서들의 소요시간
Table. 3 Processing time of parsers using an example shown in Fig. 3

Approach	조건검색			결과검색			전체 소비 클록
	문자 수	명령어 수	소비 클록	문자 수	명령어 수	소비 클록	
RBStreX	105	18	228	187	32	406	595
Search	105	1	107	187	32	406	513
APSearch	105	1	107	88	3	94	201

표 4. APSearch 파서의 합성 결과
Table. 4 Synthesis result of APSearch parser

ALUTs	Memory	Fmax
43	2,043 bits	221 MHz

전체 소비 클록은 Search 파서보다 약 2.55배, RBStreX 파서보다 약 2.96배 감소되었다. 제안된 APSearch 파서는 Altera의 Stratix GX IV 230 FPGA를 사용하여 Quartus II 12.1로 합성되었으며 결과는 표 4와 같다.

Ⅴ. 결 론

본 논문에서는 순차처리 방식을 이용하는 스트리밍 XML 파서들의 성능을 향상시키기 위하여 적응적 병렬 검색을 수행하는 구조를 제안하였다. 제안된 APSearch 파서는 병렬 검색을 지원하는 구조로서 기존 스트리밍 XML 파서들보다 소프트웨어의 오버헤드를 상당히 감소시켰고, 파서의 전체 성능을 Search 파서 보다 약 2.55배 그리고 RBStreX 파서 보다 약 2.96배 향상 시켰다. 결론적으로, 제안된 APSearch 파서는 적응적으로 병렬 검색을 수행하여 고성능의 파싱을 요구하는 시스템들에 적용될 수 있는 적합한 구조이다.

감사의 글

본 연구는 2013년도 가천대학교(GCU-2013-R133) 지원본부의 지원에 의하여 이루어진 연구로서, 관계부처에 감사드립니다.

REFERENCES

[1] K. Hameseder, S. Fowler, and A. Peterson, “Performance Analysis of Ubiquitous Web Systems for SmartPhones,” *Int'l Symposium on PECATS*, pp. 84-89, 2011.

[2] F. Belqasmi, J. Singh, S. Y. B. Melhem, and R. H. Glitho, “SOAP-based vs. RESTful Web Services,” *Internet Computing, IEEE*, Vol. 16, pp. 54-63, 2012.

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

[4] SAX, http://www.saxproject.org.

[5] Y. Pan, Y. Zhang, and K. Chiu, “Hybrid Parallel for XML SAX Parsing,” *Web Services, ICWS '08. IEEE Int'l Conference*, pp.505-512, 2008.

[6] C. E. Chang, F. M. Asin and A. K. Mustapha, “RBStreX: Hardware XML Parser for Embedded System,” *Internet Technology and Secured Transactions, ICITST'09. International Conference*, pp.1-6, Nov. 2009

[7] Sae-Woon Kim, Kyu-Hee Lee, and Sang-Kyun Yun, “Implementation of a Hardware-based XML Parser Supporting Search”, *In Proceedings of 33th KIPS Conference*, pp. 41~ 44, 2012.

이규희(Kyu-Hee LEE)

2009년 연세대학교 전산학과 이학석사
현재: 연세대학교 전산학과 박사과정
※관심분야 : FPGA 설계, 임베디드시스템, NIDS

한상수(Sang-Soo Han)

1995년 홍익대학교 전자공학과 공학박사
현재: 가천대학교 교수
※관심분야 : FPGA 설계, 임베디드시스템, 지능 및 퍼지제어