Journal of The Korea Society of Computer and Information
Vol. 17 No. 4 , April 2012

2012-17-4-1-2

FPGA 기반 네트워크 침입탐지 시스템 하드웨어 설계 및 구현

김 택 훈 *, 윤 상 균 *

The Design and Implementation of Network Intrusion Detection System Hardware on FPGA

Taek Hun Kim *, SangKyun Yun *

요 약

침입 탐지에 가장 시간이 많이 소요되는 작업은 패킷 데이터에 침입 패턴이 있는지를 검사하는 심층 패킷검사이다. 고속 네트워크에서 이 작업을 실시간으로 처리하기 위해서는 하드웨어 기반 패턴매칭이 필요하다. 본 논문에서는 침입탐지 시스템 구현에 하드웨어 기반 패턴매칭을 사용할 수 있도록 네트워크의 패킷을 수집하여 Snort 패턴 규칙에 따라서 패턴매칭을 수행하고 결과를 소프트웨어에게 제공할 수 있도록 하는 하드웨어를 Virtex-6 FPGA를 사용하여 Microblaze 기반의 SoC 형태로 설계하여 구현하였다. 구현된 시스템은 인위적인 트래픽 생성과 실제 트래픽을 사용하여 동작을 검증하였고 패킷이 네트워크 인터페이스에서 메모리로 복사되는 동안 패턴매칭 동작을 정확하게 수행하여 소프트웨어에게 결과를 제공하였다. 본 연구 결과는 실시간 처리가 가능하도록 침입탐지 시스템을 고속화 하기위한 하드웨어로 사용될 수 있다.

▸ Keyword : 침입탐지, FPGA, 패턴매칭 하드웨어

Abstract

Deep packet inspection which perform pattern matching to search for malicious patterns in the packet is most computationally intensive task. Hardware-based pattern matching is required for real-time packet inspection in high-speed network. In this paper, we have designed and implemented network intrusion detection hardware as a Microblaze-based SoC using Virtex-6 FPGA, which capture the network input packet, perform hardware-based pattern matching for patterns in the Snort rule, and provide the matching result to the software. We verify the

• 제1저자 : 김택훈, 교신저자 : 윤상균
• 투고일 : 2012. 01. 16, 심사일 : 2012. 02. 06, 게재확정일 : 2012. 02. 08.
* 연세대학교 컴퓨터정보통신공학부(Dept. of Computer and Telecommunication Engineering, Yonsei University.)
※ 이 논문은 2010년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 것임(2010-0016600)

operation of the implemented system using traffic generator and real network traffic. The implemented hardware can be used in network intrusion detection system operated in wire-speed.

▸ Keyword : Instrusion Detection, FPGA, Pattern Matching Hardware

I. 서 론

최근에 네트워크를 통한 시스템 침입이 증가하고 있으며 이러한 위협에 대처하기 위하여 침입탐지 시스템을 사용한다. 침입탐지 시스템은 네트워크 트래픽을 검사하여 침입 시도를 미리 알려진 규칙에 근거하여 탐지한다. 침입탐지에 시간이 가장 많이 소요되는 작업은 패킷 데이터에 침입 패턴이 있는지를 검사하는 심층 패킷검사이다. 이러한 작업을 모든 패킷에 대해서 와이어 속도로 처리하기 위해서는 하드웨어 기반 패턴매칭이 필요하다.

침입 패턴은 초기에는 "abc"와 같은 형태의 고정 문자열 패턴으로 표현되었지만 최근에는 "ab*c"와 같은 형태의 정규표현식 패턴의 사용이 증가되는 추세이다. 대표적인 공개 소스 침입탐지 시스템인 Snort[1]와 Bro[2] 등에서도 침입패턴을 표시하는 데 고정 문자열 패턴 뿐 만 아니라 PCRE 형식[3]의 정규표현식을 함께 사용하고 있다.

하드웨어 기반 정규표현식 패턴매칭에 대한 연구가 여러 연구자들에 의해서 수행되었다[4-8]. 그렇지만 이들 중 많은 연구가 패턴매칭 하드웨어에 대한 구조를 제시하고, 하드웨어로 합성하여 검증하는 것에 그쳤다. 실제로 시스템으로 구현한 연구[8]에서는 네트워크 트래픽에 대한 패턴매칭을 수행하여 탐지 결과를 호스트 컴퓨터로 전달하는 시스템을 구현하였다. 그렇지만 이 연구에서는 고정 문자열 패턴만 처리하였으며, 자체의 프로세서를 갖지 못한 시스템이기 때문에 기능에 한계가 있다. 그리고 한 번에 8비트 단위로 처리되는 시스템이다.

이러한 기존 연구의 단점을 보완하기 위하여 본 논문에서는 한 번에 32비트 단위로 정규표현식 패턴매칭을 수행하고 자체의 프로세서를 가지고 있으며, 네트워크 트래픽에 대해서 Snort 패턴 규칙에 대한 패턴 매칭을 수행하여 탐지 결과를 자체의 프로세서가 처리하도록 하는 네트워크 침입탐지용 하드웨어를 설계하여 FPGA로 구현한다.

시스템의 구현은 Xilinx사의 최신 Virtex-6 FPGA를 장착한 FPGA 보드에서 이루어졌다. 구현된 시스템에서 패턴매칭은 입력 패킷이 네트워크 인터페이스에서 DMA를 사용하여 메모리로 복사되는 동안에 이루어지도록 하여 와이어 속도의 패턴매칭이 가능하다.

이 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서 패턴매칭 하드웨어를 포함한 전체 시스템의 설계 및 구현에 대해서 다루고, 4장에서 구현한 시스템에 대한 평가를 수행하며, 5장에서 결론을 맺는다.

II. 관련 연구

1. 침입탐지 시스템과 정규표현식 패턴매칭

침입탐지 시스템은 모든 네트워크 트래픽을 분석하여 미리 수집된 침입 패턴 집합에 속한 패턴을 포함한 패킷을 찾는 작업을 수행한다. Snort는 대표적인 공개 소스 침입탐지 시스템으로서 침입 패턴을 나타내는데 고정 문자열 뿐 만 아니라 정규표현식도 사용한다.

와이어 속도로 침입탐지를 수행하기 위하여 하드웨어 기반의 패턴매칭 엔진의 설계가 필요하다. Sidhu 등이[4] 정규표현식에 대한 패턴매칭을 수행하는 NFA 기반 빌딩 블록을 구현하였다. Lin 등은[6] 접두패턴과 중간패턴에 대한 매칭 하드웨어를 공유하여 패턴매칭 엔진에 대한 최적화 설계를 하였다. Clark 등은[5] 고정 문자열 패턴에 대해서 한 번에 여러 바이트 단위로 패턴매칭을 수행하는 회로를 제시하였으며, Yun 등은 정규표현식 패턴에 대해서 다중 바이트 단위로 패턴매칭을 수행하는 회로[7]와 제한반복 정규표현식에 대한 패턴 매칭 회로[8]를 제시하였다.

패턴매칭 엔진을 사용하기 위해서는 컴퓨터 시스템에 결합되어야 한다. Katashita 등은[9] 네트워크 입력 패킷에 대해서 패킷분류 및 고정 문자열 패턴에 대해서 바이트 단위로 패턴매칭을 수행하여 결합한 결과를 호스트로 전달하는 침입탐지 하드웨어를 구현하였다. 네트워크 입력 패킷에 대한 와이어 속도 처리를 가능하도록 하였다. 그렇지만, 고정 문자열에 대한 처리만 수행하는 한계가 있으며, 이 하드웨어는 자체의 프로세서를 갖지 않고 소프트웨어적인 처리는 호스트에서 수행하도록 되어 있다.

2. FPGA를 사용한 SoC 구현

FPGA(Field Programmable Gate Array)는 메모리 기반의 정보를 사용하여 재구성이 가능한 논리블록, 상호연결

네트워크 및 입출력 블록들로 구성된 비메모리 반도체이다. 최근의 대용량 FPGA에서는 이러한 재구성 가능한 블록들로 구현한 수 있는 CPU가 제공되어 FPGA를 사용하여 SoC를 구현하는 것이 가능해졌다. 이러한 CPU를 소프트코어(SoftCore) 프로세서라고 하는 데 Xilinx의 MicroBlaze와 Altera의 Nios-II가 대표적이다. 그리고 많은 기본적인 주변 장치용 제어기에 대한 설계 IP도 제공 되어 이들을 CPU와 함께 사용하여 SoC를 구현할 수 있다.

그러므로 FPGA를 사용하여 SoC를 구현할 때에는 CPU를 포함한 기본적인 장치들은 설계가 제공되는 블록들을 사용하고, 패턴매칭 엔진과 같이 고유한 동작을 수행하는 회로는 Verilog 또는 VHDL과 같은 하드웨어 설계 언어로 직접 설계하고 두 종류의 설계를 통합하여 시스템을 구현하여 특정 목적의 하드웨어를 내장한 SoC 설계를 할 수 있다.

III. 침입탐지 시스템 하드웨어 설계

1. 침입탐지 시스템 하드웨어 구조

본 논문에서는 네트워크 침입탐지를 와이어 속도로 할 수 있도록 네트워크 인터페이스의 패킷 입력을 캡쳐하여 패턴매칭 엔진에 공급하고, 패턴매칭 엔진에서 Snort 규칙에서 정의된 패턴과 패킷 데이터와의 패턴매칭을 수행하여 매칭된 패턴에 대한 정보를 수집하고, 이 정보를 네트워크 인터페이스로부터 메모리로 패킷전송이 종료된 후에 CPU에 제공하도록 하는 침입탐지 시스템용 하드웨어를 FPGA로 설계, 구현한다.

이 시스템은 Xilinx사의 최신 FPGA인 Virtex-6를 장착한 ML605 Evaluation Kit[10]을 사용하여 설계, 구현하였다. 시스템의 전체적인 구조와 동작 흐름에 대해서 먼저 소개하고, 그 다음에 이러한 구조의 시스템을 ML605 Kit에서 어떻게 구현하였는지에 대해서 설명하고자 한다.

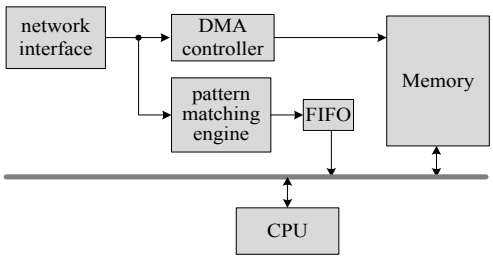

그림 1. 시스템 기본 구조
Fig. 1. System Basic Architecture

시스템의 기본 구조는 그림 1과 같다. 네트워크 인터페이스에서 입력되는 패킷은 DMA를 사용하여 메모리로 전송된 후에 CPU가 사용하게 되는 데, DMA 전송 동안에 네트워크 인터페이스의 패킷 입력을 패턴매칭 엔진에서 동시에 입력받아서 침입탐지 패턴들과 패턴매칭을 수행하며, 매칭된 결과는 FIFO (First In First Out) 버퍼에 저장된다. DMA 전송이 종료되면 DMA 제어기는 CPU에 인터럽트 요청을 하여 입력 패킷에 대한 처리를 하도록 하는 데, 침입탐지 소프트웨어가 시간이 많이 소요되는 패턴매칭을 직접 수행하는 것 대신에 패턴매칭 엔진에서 제공하는 매칭 결과를 이용함으로써 와이어 속도로 동작하는 침입탐지용 패턴매칭 시스템을 구현할 수 있도록 한다.

이러한 구조의 시스템을 Virtex-6 FPGA를 장착한 ML605 보드에 다음과 같이 구현하였다. CPU로 Xilinx FPGA에 구현가능한 32비트 RISC 기반의 소프트코어 프로세서인 MicroBlaze를 사용하였다. 네트워크 인터페이스는 10/100/ 1000 Mbps의 전송속도를 지원하는 이더넷 제어기인 TEMAC (Tri-mode Ethernet MAC) IP를 사용하였다. 네트워크 인터페이스와 메모리 간의 패킷 전송을 위한 DMA 제어를 위하여 MPMC(Multi-Port Memory Controller) IP를 사용하였다. 이 제어기는 64비트 DDR3 SDRAM과 MicroBlaze CPU를 XCL (Xilinx Cache Link)를 통하여 연결하며, 고속 전송을 위하여 데이터용 XCL (DXCL)과 명령어용 XCL (IXCL)을 분리하여 데이터 캐쉬와 명령어 캐쉬와의 전용 채널을 제공한다. 그리고 DMA 제어기를 포함하여 32비트 외부장치와 64비트 메모리 간의 DMA를 수행할 수 있게 한다.

TEMAC은 패킷을 32비트 단위로 전송하므로 패턴매칭 엔진도 한 번에 32비트 씩 처리를 해야 한다. 이를 위하여 패턴매칭엔진은 다중 바이트 단위로 패턴매칭을 수행하도록 설계된 구조[7]를 기반으로 설계하였으며 시스템의 나머지 부분과 연결이 될 수 있도록 하는 인터페이스를 추가로 설계하였다. 패턴매칭 엔진은 패킷헤더의 주요 필드를 추출하여 레지스터에 저장하고 패턴매칭 결과들을 FIFO에 저장한다. 그리고 CPU가 이 레지스터와 FIFO의 값을 읽을 수 있도록 PLB (Peripheral Local Bus) 버스를 통한 CPU 인터페이스를 구성하였다. 이 이외에 시스템 구성에 필요한 인터럽트 제어기, 플래시 메모리 제어기, UART와 GPIO 등을 제공된 IP를 사용하여 FPGA에 내장하였다. 그림 2는 Virtex-6 FPGA를 사용하여 구현한 시스템의 구성도이다. 점선 안이 FPGA 내에 구성된 부분을 나타낸다.

FPGA로 구현되는 시스템은 Xilinx사의 임베디드 개발 도구인 EDK(Embedded Development Kit)의 XPS(Xilinx

Platform studio)와 BSB(Base System Builder)를 이용하여 설계하였으며 패턴매칭 엔진은 Verilog 언어로 별도로 설계하여 시스템에 추가하였다. 그리고 FPGA 합성 도구인 Xilinx ISE Design Suite 12.2를 사용하여 Virtex-6 FPGA를 타겟 디바이스로 하여 합성하였다.

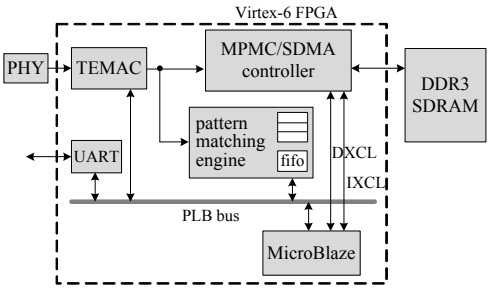

그림 2. Virtex-6 FPGA기반 시스템 구성도
Fig. 2. Virtex-6 FPGA-based System Architecture

2. 패턴매칭 엔진 설계

본 시스템의 설계에서 가장 핵심이 되는 부분이 패턴매칭 엔진이다. 패턴매칭 엔진은 크게 세 부분으로 나뉜다.

첫째, 네트워크 인터페이스에서 메모리로 DMA 전송되는 입력 패킷을 캡쳐하고, 헤더의 필요한 필드를 추출하여 레지스터에 저장하거나 패킷 데이터를 패턴매칭 모듈로 전달하는 입력 인터페이스 모듈.

둘째, 패킷 데이터에 대해서 4바이트 단위로 Snort 규칙에 대한 패턴매칭을 수행하고 매칭 결과를 FIFO에 저장하는 정규표현식 패턴매칭 모듈.

셋째, 헤더 필드 추출 결과를 저장하는 레지스터와 매칭 결과를 저장하는 FIFO 값을 CPU에서 읽도록 하기 위한 PLB 버스 인터페이스 모듈.

그림 3은 이러한 패턴매칭 엔진에 대한 구성도이다. 이 그림에서 앞서 언급한 세 부분은 점선으로 구분하였다. 각 부분에 대해서 순서대로 소개하도록 한다.

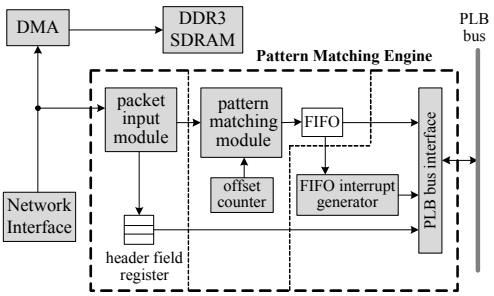

그림 3 패턴매칭 엔진 구성도
Fig. 3. Pattern Matching Engine Architecture

2.1 입력 인터페이스 모듈

DMA로 전송되는 네트워크 입력 패킷을 캡쳐하는 패턴매칭 엔진의 입력 인터페이스를 설계하기 위해서는 DMA 전송과 관련한 제어신호를 이용하여야 한다. 이더넷 제어기인 TEMAC 모듈에서 수신패킷 프레임은 LocalLink header, payload(패킷), footer로 구성되며 LocalLink 인터페이스의 32비트 데이터 채널을 통하여 MPMC/SDMA로 전송된다. 이 프레임의 payload가 네트워크 입력패킷이다.

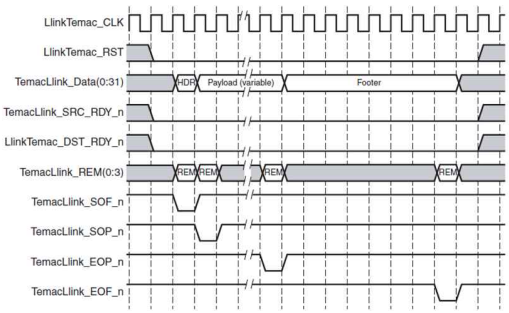

그림 4 LocalLink 수신 타이밍도
Fig. 4. LocalLink Receiving Timing Diagram

그림 4는 LocalLink를 통한 이더넷 프레임 수신과 관련된 제어신호의 타이밍도이다. frame과 payload의 시작과 끝을 각각 알려주기 위해서 SOF(start of frame), EOF(end of frame), SOP(start of payload), EOP(end of payload) 신호를 제공한다. TEMAC과 MPMC/SDMA 사이의 흐름 제어를 위하여 DST_RDY와 SRC_RDY 신호를 사용한다. 이러한 제어신호를 사용하여 패킷 전송 신호의 변화에 따라서 상태를 변화하여 패턴매칭 모듈로의 입력을 제어하는 유한상태 기계 회로를 설계하였다. 이 유한상태 기계는 그림 5와 같은 상태도를 갖는다. 각 상태는 수신대기, header 수신, payload(네트워크 패킷) 수신, footer 수신, 수신 종료의 동작과 관련된다.

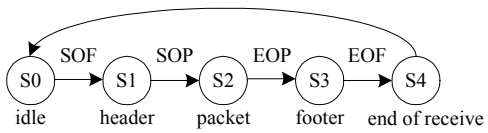

그림 5 입력 인터페이스 상태도
Fig. 5. State Diagram of Input Interface

payload 수신 상태에서 SRC_RDY와 DST_RDY가 동시에 활성화되는 경우에 패킷 입력이 유효하며 이 경우의 패킷 입력을 검사한다. payerload 수신 상태가 되어 패킷이 입력

되기 시작하면 입력 인터페이스 모듈에서는 먼저 입력 패킷의 헤더를 검사하여 소스 MAC 주소, 목적지 MAC 주소, 소스 IP 주소, 목적지 IP 주소, 소스 port 번호, 목적지 port 번호 및 프로토콜 필드를 추출하여 관련된 헤더 필드 레지스터에 저장한다. 그리고 입력 패킷의 데이터 부분이 입력되기 시작하면 패턴매칭 모듈로 입력 패킷을 전달한다.

2.2 패턴매칭 모듈

패턴매칭 모듈은 NFA 기반으로 구성된 4바이트 단위로 처리하는 기존 연구에서 제시된 정규표현식 패턴매칭 회로[7]를 포함하고 있다. 이 회로는 미리 주어진 패턴 집합에 따라서 구성되며 입력 인터페이스 모듈에서 전달되는 패킷 데이터와의 패턴매칭을 수행한다.

기존 연구[7]의 NFA 기반 회로는 매 클럭마다 패킷 데이터가 입력되는 것을 가정하여 회로를 설계하였으나, 패턴매칭 모듈 구현에서 SRC_RDY 또는 DST_RDY가 비활성화될 때에는 새로운 데이터가 입력되지 않으므로 NFA의 상태 진행이 정지해야 한다. 따라서 SRC_RDY와 DST_RDY가 동시에 활성화할 때에 Payload_Valid 신호를 생성하여 패턴매칭 활성화 신호로 사용하여, 이 신호가 1일 때에만 패턴매칭이 진행되도록 하였다. 그리고 payload의 길이가 4의 배수가 아닌 경우에는 마지막 데이터는 일부 바이트들이 유효하지 않을 수 있다. 각 바이트의 유효성을 나타내는 REM[0:3] 신호를 사용하여 유효하지 않은 바이트에 대해서는 매칭을 수행하지 않도록 하였다. 패턴매칭 모듈에서 패턴매칭이 성공하면 각 패턴들에 대한 인코딩된 번호가 출력된다. 이 패턴 번호는 패턴매칭이 성공하였을 때의 위치를 나타내는 32비트 단위의 오프셋과 함께 FIFO에 저장된다.

패턴매칭 모듈 설계에 대한 Verilog 코드는 Snort 규칙을 입력으로 사용하여 회로 생성기 프로그램에 의해서 자동적으로 생성된다.

2.3 CPU 인터페이스 설계

입력 인터페이스 모듈에서 추출된 패킷 헤더 필드들을 저장한 레지스터들과 패턴매칭 모듈에서 매칭된 패턴의 번호들을 저장한 FIFO를 CPU에서 읽을 수 있도록 하기 위해서 레지스터들과 FIFO에 대한 PLB 버스 인터페이스를 구성하였다. 시스템의 다른 구성 요소는 Xilinx의 EDK를 사용하여 설계를 하지만 패턴매칭 엔진은 직접 Verilog로 설계가 되므로 PLB 버스의 신호와 프로토콜을 이해한 후에 설계해야 한다. XPS의 “Create and Import Peripheral Wizard” 기능을 이용하면 PLB 버스 프로토콜에 맞춘 기본적인 인터페이스 하드웨어 템플릿 코드를 생성해준다. 이 코드를 기반으로 본 시스템에 알맞도록 코드를 수정하여 레지스터와 FIFO에 대한 PLB 버스 인터페이스를 설계하였다. PLBv4.6 호환 주변장치는 PLB IPIF(IP Interface) 모듈[11]을 사용하여 설계하는 데, 이 모듈은 PLB 버스와 사용자 회로 사이의 인터페이스를 빠르게 구현하고 높은 적응력을 가질 수 있도록 하는 IPIC(IP Interconnect)라고 하는 인터페이스를 제공한다. 사용자 회로는 IPIC 신호를 이용하여 PLB 버스 인터페이스를 설계한다. CPU 인터페이스로 접근할 수 있는 레지스터들과 FIFO에 대해 표 1과 같이 오프셋 주소를 부여하여 CPU에서 각 레지스터들을 접근할 수 있게 하였다.

표 1 패턴매칭 엔진의 레지스터의 오프셋 주소
Table 1. Offsets of Registers in Pattern Matching Engine

Offset	Bit	Name
0x0	0-31	Destination MAC (Reg0)
0x4	0-31	Source MAC (Reg1)
0x8	0-31	Source IP (Reg2)
0xc	0-31	Destination IP (Reg3)
0x10	0-31	Source, Destination Port (Reg4)
0x14	0-31	Protocol Information (Reg5)
0x18	0-31	FIFO Status (Reg6)
0x1c	0-31	FIFO Read (Reg7)
0x20	0-31	Reserved (Reg8)
0x24	0-31	Reserved (Reg9)

입력 패킷이 메모리로 DMA 전송이 완료되면 CPU로 인터럽트 요청을 하게 되며, 이때에 CPU가 레지스터 및 FIFO를 읽어서 침입탐지 프로그램을 수행하게 된다. 그런데 이 동안에 다음 패킷이 DMA 전송되어 FIFO에 저장될 수 있다. 이때에 FIFO가 완전히 채워져서 빈 공간이 없을 수 있다. 이러한 문제를 해결하기 위해서 FIFO가 미리 정해놓은 한계점 이상 채워질 때에 인터럽트를 발생시켜서 DMA 전송이 완료되지 않았더라도 CPU가 FIFO에 저장된 정보를 메모리로 이동시켜서 FIFO를 비우게 한다. 이렇게 하면 FIFO에 빈 공간이 없어서 오버플로우가 되는 경우는 발생하지 않는다. 그리고 패킷에 대한 패턴매칭이 완료되면 다음 패킷과 구분하기 위하여 빈 데이터를 FIFO에 추가적으로 넣는다. 이를 통하여 두 패킷에 대한 패턴 번호들이 FIFO에 연속적으로 저장되더라도 둘 사이를 구분할 수 있다.

3. 시스템 동작을 위한 소프트웨어 구현

앞에서 구현한 시스템을 동작시키기 위해서는 시스템을 기동하고 주변장치들을 초기화하는 프로그램과 패턴매칭 엔진의 정보를 이용하여 추가적인 작업을 수행하는 프로그램들이 필요하다.

초기화 프로그램은 FPGA의 하드웨어 구성 정보와 합쳐져서 하나의 비트스트림(bit stream) 파일로 생성되어 FPGA 디바이스에 함께 프로그래밍 된다. 응용 프로그램은 크기가 크므로 하드웨어 구성정보와 합쳐질 수 없으며, 플래시와 같은 외부 메모리에 저장된다.

FPGA 디바이스에 대한 비트스트림 정보는 Compact Flash(CF) 메모리에 저장되며 전원이 인가될 때에 FPGA에 적재되어 하드웨어와 초기화 프로그램을 구성하게 된다. 초기화 프로그램은 FPGA 내부의 BRAM에 적재된다. BRAM의 크기에 제약이 있기 때문에, 실제의 시스템 프로그램은 CF 메모리 카드에 저장하고, BRAM에 적재되는 초기화 프로그램은 CF 메모리에서 시스템 프로그램을 읽어서 외부 DRAM에 적재하여 실행시키는 역할을 수행한다. 초기화 프로그램이 적재하는 시스템 프로그램 모듈은 ELF 바이너리 형식으로 만들어서 플래시 메모리에 저장된다. 그림 6에 전원 인가시의 동작 수행에 대한 흐름을 나타내었다.

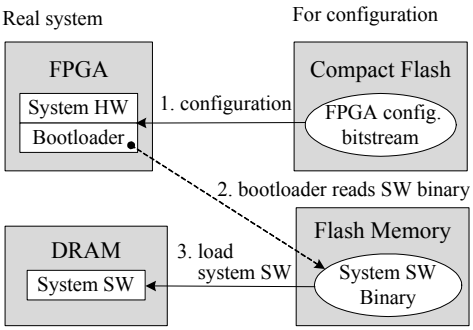

그림 6 전원 인가 시의 수행 흐름
Fig. 6 Execution Flow during Power-on

구현된 시스템 프로그램은 설계한 하드웨어 시스템의 동작을 확인하기 위해서 작성되었다. 시스템 프로그램은 기본적으로 시스템의 각종 주변장치의 초기화 및 시스템 설정을 먼저 수행한다. 설계한 침입탐지 하드웨어의 동작을 확인하기 위하여 DMA 동작이 원활하게 수행되도록 관리해야 하며, DMA 완료에 의한 인터럽트 시에 패킷 헤더가 저장된 레지스터와 패턴매칭 엔진의 검사 결과가 저장된 FIFO의 데이터들을 읽어서 침입탐지 하드웨어가 정상적으로 패킷을 처리하였는지 여부를 확인할 수 있게 하는 소프트웨어가 포함되어야 한다. 구현한 시스템 프로그램에는 이러한 프로그램이 포함되어 있다.

시스템 프로그램은 TEMAC과 메모리 간의 DMA 동작을 관리하기 위해서 MPMC/SDMA 제어기를 제어하고, DMA가 사용하는 메모리 버퍼를 관리하는 동작을 수행한다. SDMA 제어기는 분산-집합(Scatter- Gather) DMA를 지원하여 긴 패킷에 대해서 비연속적인 메모리 공간으로 DMA 전송을 가능하게 하며, 프로그램은 이와 관련된 관리 기능도 수행한다. 버퍼의 크기는 이더넷 프레임을 저장하기에 충분하고, 관리하기 쉽도록 2KB로 정하였다. 연속적인 패킷 전송에 대처할 수 있도록 8개의 버퍼를 링 체인으로 구성하여 관리하였다.

그리고 시스템에 대한 사용자 인터페이스를 제공하기 위하여 명령어 처리기도 함께 구현하였다. 명령어 처리기는 시스템과의 상호 작용을 위해 필요한 기능들을 명령어로 제공하고 명령어 처리기를 통하여 명령어를 입력받아서 해당되는 기능을 수행한다. 제공되는 명령어들은 메모리 읽기/쓰기, LED, text LCD 및 푸쉬버튼과 같은 기본 입출력 장치에 대한 인터페이스, 그리고 침입탐지 하드웨어에 의한 입력 패킷 처리 결과 출력 등의 기능을 수행한다. 그림 7은 명령어 처리기 화면을 보여준다.

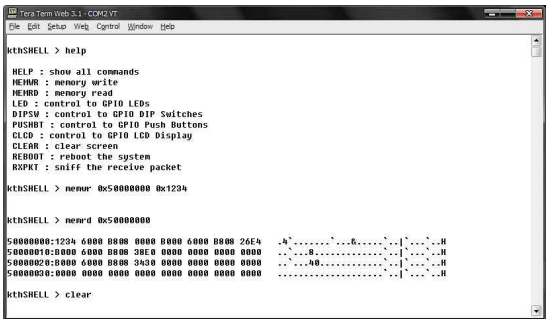

그림 7 명령어 처리기
Fig 7. Command Interpreter

IV. 동작 검증 및 평가

설계한 시스템은 Virtex-6 FPGA를 장착한 Xilinx사의 ML605 보드를 사용하여 구현하였다. 설계한 시스템에는 포함된 패턴매칭 엔진은 정규표현식 패턴과 고정문자열 패턴을 포함하여 총 1,212개의 Snort 패턴을 처리한다. 그림 8은 동작을 테스트 중인 ML605 보드의 사진이다.

구현된 시스템의 동작을 다음과 같이 검증하였다. 먼저 설계 시스템이 탐지 가능한 특정 패턴들을 포함한 패킷을 호스트에서 입력하여 네트워크에 제공한 후에 이를 정상적으로 캡쳐하여 패턴을 탐지하는 지 확인하였다. 이를 위하여 트래픽 생성기 프로그램을 작성하였다. 그림 9와 그림 10은 호스트에서 패킷 데이터를 입력하는 화면과 구현한 시스템에서 패턴

을 탐지하여 offset과 함께 패턴 번호를 출력한 화면을 보여준다. 고정문자열 패턴과 정규표현식 패턴을 포함한 여러 패턴에 대해서 테스트한 결과 구현한 시스템은 패킷 헤더를 정확하게 추출하고 패턴매칭을 정확하게 수행하여 결과를 제공함을 확인하였다.

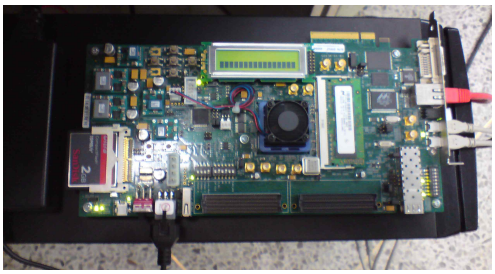

그림 8 ML605 기반 구현 시스템
Fig. 8 System Implemented on ML605 board

그리고 실제 네트워크 트래픽에 대해서도 동작을 검증하였다. 실제 네트워크 트래픽의 내용을 확인하기 위하여 패킷분석기 프로그램인 wireshark를 사용하여 스니핑한 결과를 함께 확인하여 구현한 시스템의 동작을 검증하였다. 그림 11과 그림 12는 구현한 시스템에서 매칭한 결과와 wireshark에서 스니핑한 결과를 보여준다. 구현한 패턴매칭 엔진에서 PID 102번에 해당하는 패턴은 "public"으로서 wireshark의 패킷 캡쳐 결과와 일치함을 확인하였다. 그리고 패턴의 오프셋 위치도 일치하였다.

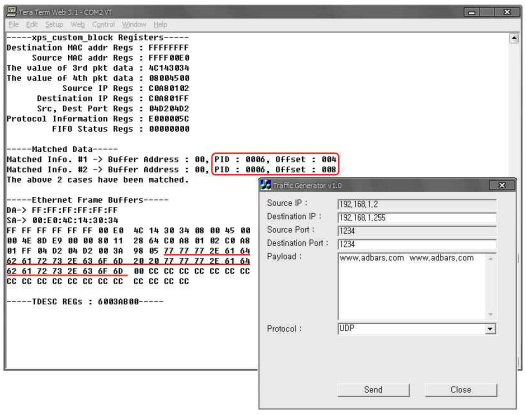

그림 9 고정문자열 패턴에 대한 수행 결과
Fig. 9 Pattern Matching Result for Fixed Patterns

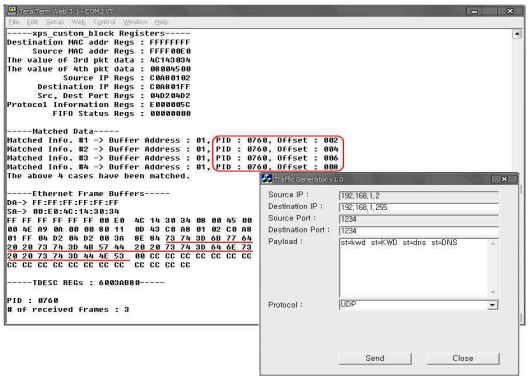

그림 10 정규표현식 패턴에 대한 수행 결과
Fig. 10 Pattern Matching Result for Regular Expressions

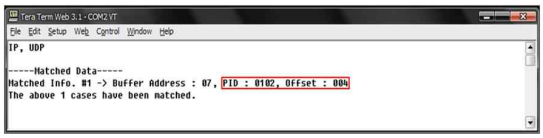

그림 11 실제 트래픽에 대한 구현 시스템의 매칭 결과
Fig. 11 Result in the Implemented System for Real Traffic

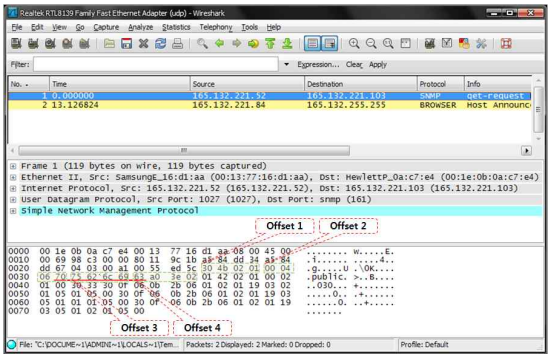

그림 12 실제 트래픽에 대한 Wireshark의 스니핑 결과
Fig. 12 Sniffing Result in Wireshark for Real Traffic

이와 같이 본 연구에서 설계하여 구현한 FPGA 기반 침입탐지 시스템 하드웨어는 스니핑한 네트워크 입력 패킷에 대해서 필요한 헤더 정보를 정확하게 추출하고, 패턴매칭을 정확하게 수행함을 확인하였다.

본 연구에서 설계한 하드웨어는 입력 패킷이 네트워크 인터페이스에서 메모리로 DMA 전송되는 동안에 패턴매칭이 동시에 이루어진다. 네트워크 침입탐지 시스템 소프트웨어가 이 하드웨어를 사용하면 패턴매칭 작업의 부담을 줄일 수 있으므로 실시간으로 패턴 탐지를 수행할 수 있게 된다.

Ⅵ. 결 론

본 연구에서 구현한 시스템은 네트워크 패킷입력을 하드웨어로 캡쳐하여 패턴매칭을 수행하여 소프트웨어로 패턴매칭을 수행할 필요를 없도록 하여 와이어 속도의 침입탐지가 가능하게 한다. 그리고 4바이트 기반의 하드웨어기반 패턴매칭을 구현하여 고속 동작을 수행하도록 하였다. 그리고 FPGA에 내장되는 MicroBlaze 소프트코어 CPU를 사용하여 SoC 형태로 시스템을 구현하였다.

본 연구에서는 전체적인 시스템 구조에서부터 기본적인 소프트웨어까지의 침입탐지시스템 구현에 대한 가이드라인을 제시하였고 구현한 시스템이 패턴매칭 동작을 정상적으로 수행함을 확인하였다.

구현한 시스템의 하드웨어는 패킷 헤더 추출과 패턴매칭만 수행하며, 완전한 침입탐지 시스템을 구현하기 위해서는 침입탐지 소프트웨어에서 추출된 헤더를 분석하여 패킷분류를 한 후에 FIFO에 저장된 패턴매칭 결과 중에서 패킷분류 결과와 관련이 없는 것을 폐기하고 관련된 결과만 분석하여 침입탐지를 수행해야 한다. 이러한 소프트웨어의 구현은 향후 과제이다. 또한 패킷분류를 하드웨어로 구현하여 패턴매칭 결과와 결합하면 침입탐지 소프트웨어는 더욱 간단하게 구현될 수 있을 것이다.

참고문헌

[1] Snort web site, http://www.snort.org

[2] Bro web site, http://bro-ids.org

[3] PCRE - Perl Compatible Regular Expressions, http://www.pcre.org

[4] R. Sidhu and V.K. Prasanna, "Fast Regular Expression Matching using FPGAs," IEEE Symp. Field- Programmable Custom Computing Machines, (FCCM'01), pp. 227-238, 2001.

[5] C. R. Clark and D. E. Schimmel, "Scalable Parallel Pattern Matching on High Speed Networks," IEEE Symp. Field-Programmable Custom Computing Machines (FCCM'04), pp. 249-257, 2004.

[6] C.-H. Lin, C.-T. Huang, C.-P. Jiang, and S.-C. Chang, "Optimization of Pattern Matching Circuits for Regular Expression on FPGA," IEEE Trans. VLSI Systems, Vol.15, No.12, pp.1303-1310, Dec. 2007.

[7] S.K. Yun and K.H. Lee, "A Hardware Architecture of Multibyte-based Regular Expression Pattern Matching for NIDS," Journal of Korea Information and Communicaitons Society(KICS), Vol.34, No.1B, pp.47-55, Jan. 2009.

[8] S.K. Yun and K.H. Lee, "A Hardware Architecture of Regular Expression Pattern Matching for Deep Packet Inspection," Journal of Korea Society of Computer and Information(KSCI), Vol.16, No.5, pp.13-22, May 2011.

[9] T. Katashita, Y. Yamaguchi, A. Maeda and K. Toda, "FPGA-Based Intrusion Detection System for 10 Gigabit Ethernet," IEICE Trans. Info. & Sys, Vol. E90-D, No.12, pp.1923-1931, Dec. 2007.

[10] Xilinx ML605 Evaluation Kit, http://www.xilinx.com/products/devkits/EK-V6-ML605-G.htm

[11] Xilinx PLB IPIF IP CORE Datasheet, http://www.xilinx.com/support/documentation/ip_documentation/ plb_ipif.pdf

저 자 소 개

김 택 훈

2009 : 연세대학교 컴퓨터공학 학사.
2011 : 연세대학교 전산학과 석사.
현 재 : (주)피앤피시큐어 정보보안기술연구소
관심분야 : 정보보안, 임베디드시스템,
Email : thkim@pnpsecure.com

윤 상 균

1984 : 서울대학교 전자공학과 학사.
1986 : KAIST전기및전자공학과 석사.
1995 : KAIST전기및전자공학과 박사.
현 재 : 연세대학교 컴퓨터정보통신공학부 교수
관심분야 : 임베디드시스템, NIDS, 컴퓨터하드웨어
Email : skyun@yonsei.ac.kr