스토리지 시스템을 위한 해쉬 알고리즘 회로 개발

김재운○ , 이후웅 , 원유집

한양대학교

ragingwind@hanyang.ac.kr , oihtoto@ece.hanyang.ac.kr , yjwon@ece.hanyang.ac.kr

Implementing SHA-1 hash function

Jae-woon Kim○ , Hu-ung Lee , Youjip Won

Hanyang University

요 약

본 논문에서는 스토리지 시스템의 데이터 중복 확인에 사용되는 SHA-1 해쉬 알고리즘을 하드웨어로 구현하여 시스템에 적용하고, 그 성능을 소프트웨어적 처리 방법과 비교 하였다.

1. 서 론

근래에 네트워크 시스템의 발달은 디지털 데이터의 폭증을 가져왔다. 최근에는 가상화 기술, 클라우드 컴퓨팅과 같은 새로운 기술들이 등장하며 데이터의 양은 더욱 더 증가하고 있는 추세다. 이러한 데이터들을 저장, 관리하기 위한 스토리지 시스템 또한 그 중요성과 더불어 발전에 발전을 거듭하고 있다[1][2].

비대해진 데이터들은 시스템 내에 많은 부분 동일한 내용을 갖게 되고 이러한 중복 데이터를 관리하는 방법에는 일반적으로 파일을 청크(chunk) 혹은 블록(block)이라는 단위 조각으로 나누어서 그 조각을 비교하는 방법이 사용되고 있다. 각각의 청크들은 고유의 핑거프린트를 가지게 되며, 중복성 증명을 위한 하나의 방법으로 해쉬 알고리즘을 사용하게 된다.

중복성 증명을 위한 해쉬 알고리즘으로는 MD5와 SHA-1을 주로 사용한다. MD5는 128비트, SHA-1은 160비트 값을 생성하며, 사실 이 기법들은 암호화된 서명을 생성하는 방법으로 널리 사용되는 기술이다. 각 기법들은 각 청크가 유일한 값이라고 인식되도록 하여 통계적으로 유의한 값을 생성한다. 따라서 만약 두 개의 청크가 하나의 해쉬 값을 가지고 있을 경우에는 그 청크들은 같은 정보를 담고 있는 것이다.

고용량 스토리지 내에서의 해쉬 값의 생성은 적지 않은 CPU 자원의 사용을 요구하게 된다. 본 논문에서는 하드웨어로 구현된 SHA-1 알고리즘을 위한 프로세서를 시스템에 도입하여, 소프트웨어로 해쉬 데이터를 처리, 비교 했을 때와의 시간 차이를 확인하고자 한다.

2. SHA-1 알고리즘

일반적인 SHA-1 해쉬 연산의 구조는 그림 1과 같다. 연산의 단위는 512비트를 한 블록으로 하여 이루어지며, 각 블록 당 기본 단계 연산을 80회 반복 수행한다.

그림 1에서 a, b, c, d, e는 160비트 해쉬 블록을 다섯 개로 나눈 32비트 크기의 블록이며, w는 k는 512비트의 데이터 스트림을 32비트 블록으로 나눈 데이터 블록과 각 연산의 단계별 사용 상수 값을 말한다. ROTLx(y)는 y를 x비트 만큼 왼쪽으로 회전시키라는 표현이고, f는 비선형 함수이다.

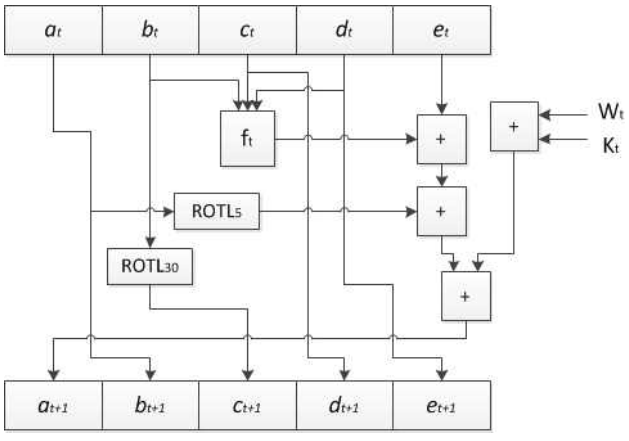

그림 1. SHA-1 해쉬 연산 블록

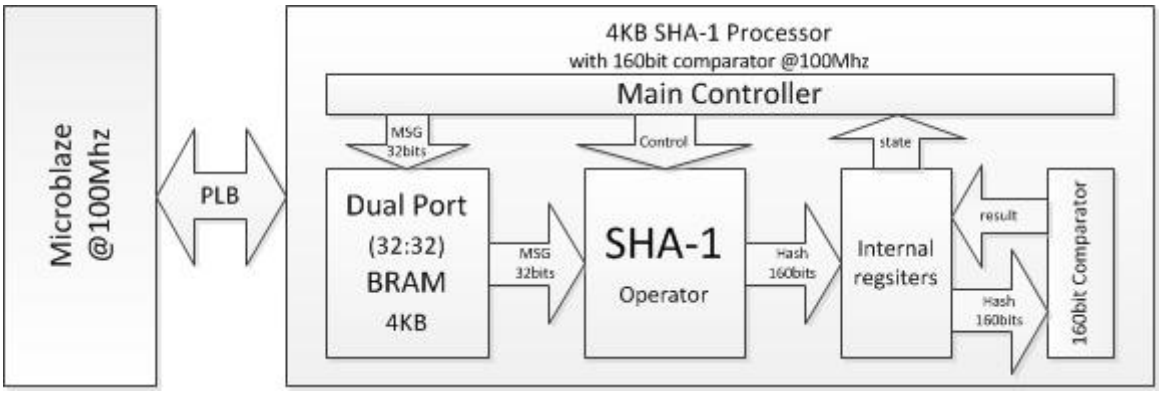

그림 2. 시스템 블록 다이어그램

3. 시스템 구현 및 성능 측정

본 논문에서는 SHA-1 해쉬 알고리즘의 하드웨어 구현과 성능 평가를 수행하기 위해 Xilinx사의 Virtex-6 FPGA에서 합성하였다. 또한 시스템의 메인 컨트롤러는 Xilinx사에서 제공되는 microblaze core를 IP로 사용하였으며, 내부 SHA-1 구조는 "FIPS PUB 180-1" SHA-1 표준 문서[3]를 기반으로 작성되었고, 비교를 위한 소트프웨어는 "RFC-3174" SHA-1 표준 문서의 C 소스를 사용하였다.

3.1. SHA-1 프로세서

그림 2를 통해 본 논문에서 구현된 SHA-1 프로세서의 전체적인 시스템 블록 다이어그램을 볼 수 있다. 시스템에서는 4Kbyte 단위의 데이터 청크 입력을 기본으로 가정하였으며, 이를 위한 버퍼로 내부에 4Kbyte BRAM이 사용되었다. SHA-1 프로세서는 4Kbyte의 메시지 데이터와 비교를 위한 160비트 해쉬 데이터를 microblaze로부터 전달 받아 SHA-1 알고리즘 연산을 처리한 뒤 각 값을 비교, 그 결과를 내부 레지스터에 기록한다.

3.2 데이터 처리 시간 비교

데이터 처리 시간의 측정 기준은 microblaze로부터 4Kbyte 메시지를 입력하는 순간부터 160bit의 올바른 해쉬 값이 출력되는 순간까지의 시간을 측정하였고, 이를 microblaze의 내부 타이머를 이용하여 cycle 수를 측정 후 us로 환산 하였다. 구현된 시스템에서 SHA-1 표준 문서의 C 소스를 이용해 4Kbyte 데이터를 처리한 결과 약 96,619us 시간이 걸렸으며, 이를 하드웨어로 처리했을 때는 약 839us의 시간이 소요되었다.

3.3 시스템의 성능 개선

구현 된 시스템에서 현재 성능에 영향을 가장 많이 미치는 부분은 단연 SHA-1 알고리즘의 연산 부분이다. 정해진 80회의 반복 연산을 줄일 수 있다면 전체적인 시스템의 성능 또한 증가될 것이다. SHA-1 알고리즘의 고속화를 위한 방안은 이미 다양한 방법[4-7]들이 제안되어 있다. 추후 진행 될 연구에서는 이 중 주된 방법인 unfolding과 pipelining을 이용한 개선 된 SHA-1 알고리즘 회로를 사용하여 좀 더 향상된 성능의 시스템을 구현하고자 한다.

4. 결론

본 논문에서는 스토리지 시스템에서 중복 데이터를 확인하기 위한 SHA-1 해쉬 알고리즘을 하드웨어로 구현, 적용하여 이를 통해 CPU 자원을 이용한 소프트웨어적 처리와의 성능 차이를 확인하였다.

확인 결과, 하드웨어에 의한 처리 시간은 소프트웨어에 의한 처리 시간의 0.8%에 불과하였으며, 이는 스토리지 시스템에서의 해쉬 알고리즘을 위한 보조 프로세서의 필요성을 야기한다.

5. 참고문헌

[1] J. McKnight, T. Asaro, and B. Babineau, "Digital Archiving:End-User Servey and Market Forecast 2006-2010," The Enterprise Strategy Group Jan. 2006.
[2] Gartner, "Gartner Identifies the Top 10 Strategic Technologies for 2009," http://www.gartner.com, 2008.
[3] FIPS PUB 180-2, Secure Hash Standard(SHA-1), National Institute of Standards and Technology (NIST), 1996.
[4] A. Kakarountas, G. Theodoridis, T. Laopoulos, and C. Goutis, "High speed FPGA impliementation of the SHA-1 hash function," IDAACS, pp. 211-215, 2005.
[5] Y. K. Lee, H. Chan and I. Verbauwhede, "Throughput optimized SHA-1 architecture using unfolding transformation," ASAP, pp. 354-359, 2006.
[6] H. Michail, and C. Goutis, "Holistic methodology for designing ultra high-speed SHA-1 hashing cryptographic module in hardware," EDSSC. pp. 1-4, 2008.
[7] Liehui Jiang, Yuliang Wang, Qiuxia Zhao, Yi Shao, Xiaoli Zhao, "Ultra High Throughput Architectures for SHA-1 Hash Algorithm on FPGA", CiSE. pp.1-4, 2009.