논문 2010-47SD-4-7

다중 언폴딩 기법을 이용한 SHA-1 해쉬 알고리즘 고속 구현

(Implementation of High-Throughput SHA-1 Hash Algorithm using Multiple Unfolding Technique)

이 은 희*, 이 제 훈**, 장 영 조***, 조 경 록*

(Eun-Hee Lee, Je-Hoon Lee, Young-Jo Jang, and Kyoung-Rok Cho)

요 약

본 논문은 다중 언폴딩 기법을 이용한 고속 SHA-1 해쉬 알고리즘 구현 방법을 제시하였다. 제안된 SHA-1 해쉬 구조는 알고리즘의 반복적인 단계 연산을 언폴딩한 후 연산의 순서를 재 배열하고, 임계경로에 포함된 연산의 일부를 이전 단계에서 선행연산하여 임계경로의 길이를 줄였다. 제안된 SHA-1 해쉬 구조는 최대 118 MHz의 동작주파수에서 5.9 Gbps 처리량을 나타낸다. 이는 기존의 SHA-1 보다 전송량이 26% 증가하였고, 회로 크기가 32% 감소하는 결과를 얻었다. 또한 이 논문에서는 여러 개의 SHA-1 모듈을 시스템 레벨에서 병렬로 연결하여 여러 개의 SHA-1을 다중 처리하여 고속화를 할 수 있는 모델을 제안했다. 이 모델은 하나의 SHA-1을 사용하는 것보다 빠르게 데이터를 처리할 수 있고 입력되는 데이터의 최소한의 지연으로 처리 가능하다. 제안된 모델은 입력되는 데이터가 지연 없이 처리 되도록 하기 위해 필요로 하는 SHA-1의 FPGA 수를 구할 수도 있다. 고속화된 SHA-1은 압축된 메시지에 유용하게 사용될 수 있고 모바일 통신이나 인터넷 서비스 등의 강한 보안에 널리 이용가능하다.

Abstract

This paper proposes a new high speed SHA-1 architecture using multiple unfolding and pre-computation techniques. We unfolds iterative hash operations to 2 continuos hash stage and reschedules computation timing. Then, the part of critical path is computed at the previous hash operation round and the rest is performed in the present round. These techniques reduce 3 additions to 2 additions on the critical path. It makes the maximum clock frequency of 118 MHz which provides throughput rate of 5.9 Gbps. The proposed architecture shows 26% higher throughput with a 32% smaller hardware size compared to other counterparts. This paper also introduces a analytical model of multiple SHA-1 architecture at the system level that maps a large input data on SHA-1 block in parallel. The model gives us the required number of SHA-1 blocks for a large multimedia data processing that it helps to make decision hardware configuration. The hs fospeed SHA-1 is useful to generate a condensed message and may strengthen the security of mobile communication and internet service.

Keywords : cryptography, secure hash algorithm, hardware design, SHA-1

* 정회원, 충북대학교 정보통신공학과
(Dept. of Information and Communication Engineering, Chungbuk National University)
** 정회원, 강원대학교 전자공학과
(Dept. of Electronics Engineering, Kangwon National University)
*** 정회원, 한국기술교육대학교 정보기술공학부
(School of Information Technology Engineering, Korea University of Technology and Education)
※ 본 연구는 교육과학기술부와 한국산업기술재단의 지역혁신인력양성사업으로 수행된 연구결과임
접수일자: 2010년1월22일, 수정완료일: 2010년3월19일

I. 서 론

정보화 사회에서 자료를 보호하기 위한 암호화는 매우 중요하다. 현재 RSA, 3-DES 등이 안전한 암호화 방법으로 쓰이고 있으나 복호화하는데 시간이 많이 걸려 고속으로 동작시키기 어렵다는 단점을 갖는다. 또한 암호 처리 속도를 증가시키기 위해서는 소프트웨어 처리보다 고속 암호 처리용 하드웨어 구현이 필요하다.

해쉬 함수는 단방향으로 연산되는 함수로 한번 해쉬 함수에 의해 원본으로부터 해쉬 값을 생성했다면, 해쉬 값으로부터 원본 메시지를 알아내는 것이 불가능하다. 해쉬의 용도는 원본 메시지가 손상이 되었는지를 알아내는데 있으면 해쉬 값을 변경시키지 않고 원본을 조작하는 것이 극히 어렵다. 해쉬 함수는 패스워드 메커니즘을 비롯한 소프트웨어의 손상 유무를 가리는 데도 이용할 수 있으며 메시지의 손상을 방지하기 위해서도 널리 사용된다.

시스템이 고속화될수록 사용되는 해쉬 함수도 고속처리가 필요하다. 이와 같은 문제를 해결하기 위해 해쉬 함수 처리를 위한 고속 하드웨어 구현이 일반화되고 있다. 또한, 최근에 병렬처리를 위해 언폴딩(unfolding) 구조를 갖는 해쉬 함수가 사용되고 있다[2]. 이는 한 데이터 블록의 해쉬 함수 처리를 위해 총 80 사이클의 반복적인 단계연산을 수행하는 대신 한 클럭에 여러 개의 단계연산을 수행하도록 한다. 이를 통해 80개의 단계연산을 수행하기 위한 전체 클럭 수를 줄여 시간당 처리량을 늘려준다[1]. 그러나 언폴딩 하는 연산의 크기가 증가함에 따라 언폴딩된 단계연산들을 처리하기 위한 최대 지연 시간이 늘어나고 처리량은 낮아진다. 이 결과로 고속으로 들어오는 데이터를 처리할 수 없게 되고 병목 현상을 야기한다.

본 논문에서는 이와 같은 문제를 해결하기 위해 언폴딩을 통해 하나의 클럭에 여러 개의 단계연산을 수행할 수 있도록 하고, 최대지연시간을 결정하는 임계경로에 포함된 연산을 병렬처리하여 처리량을 높이는 방법을 제안한다. 특히 임계경로에 포함된 연산을 데이터 의존성이 없는 동일한 길이의 두 개의 데이터패스로 나눈 후, 그 중 하나는 이전 단계 연산에 선행처리를 수행한다. 따라서, 각 단계연산의 임계경로의 크기를 절반으로 줄여 단계연산의 최대 레이턴시를 기존 방법에 비해 반으로 줄여 처리량을 높이는 SHA-1 하드웨어를 제안한다. 또한 제안된 SHA-1 구조를 병렬로 구성하여 임의의 길이로 입력되는 전체 데이터워드를 지연 없이 병렬로 처리하는 시스템 모델을 제안한다.

본 논문은 다음과 같이 구성되어 있다. Ⅱ장에서는 SHA-1의 고속화를 위한 기존 방법들을 설명하고, Ⅲ장에서는 본 논문에서 제안하는 선행 연산 기법을 적용한 고속화 된 SHA-1 구조를 제안한다. Ⅳ장에서는 제안된 선행 연산 기법을 도입한 고속화된 SHA-1 연산기의 동작을 검증한다. 또한 시스템 레벨에서 SHA-1을 다중으로 연결하여 멀티 프로세싱하게 됨으로써 입력되는 데이터가 지연 없이 처리 될 수 있는 모델을 제안하고 이를 검증한다. 마지막으로 Ⅴ장에서 결론을 제시한다.

Ⅱ. SHA-1 해쉬 연산의 고속화 기법

1. 기존의 SHA-1 고속화 기법

SHA-1의 해쉬 연산 블록의 기본식은 식(1), (2), (3), (4), 그리고 식(5)로 나타낼 수 있다. SHA-1은 최종 해쉬 값을 얻기 위해 해쉬 연산 블록에서 기본 단계연산을 80번 반복 수행한다.

$$a_{t+1} = RotL^5(a_t) + f_{t+1}(b_t, c_t, d_t) + e_t + w_t + k_t \quad (1)$$

$$b_{i+1} = a_i \quad (2)$$

$$c_{t+1} = RotL^{30}(b_t) \quad (3)$$

$$d_{t+1} = c_t \quad (4)$$

$$e_{t+1} = d_t \quad (5)$$

여기에서 $RotL^x(y)$는 워드 y를 x 비트만큼 왼쪽으로 회전시키라는 표현이고 $f_t(x, y, z)$는 비선형함수이다. W_t 및 K_t 는 입력되는 데이터 스트림을 512 비트의 블록으로 나누고 패딩하여 만들어진 데이터 블록과 상수 값으로 해당 데이터 워드가 단계연산을 수행하기 전에 미리 알 수 있다. 나머지 변수 a_t, b_t, c_t, d_t, e_t의 경우 t 시점에서의 값으로 단계연산을 마친 후 레지스터에 저장하고 다음 단계연산을 위한 입력으로 들어간다. SHA-1은 이와 같은 해쉬 연산을 80번 반복하여 수행

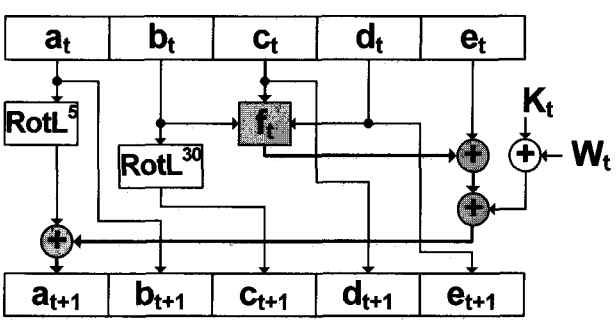


그림 1. SHA-1 해쉬 연산 블록
Fig. 1. Hash operation block of typical SHA-1.

함으로써, 임의의 길이의 데이터를 일정한 짧은 길이의 해쉬 값으로 만든다.

그림 1은 SHA-1의 기본적인 해쉬 단계연산 블록이다. 그림에서 굵은 선으로 표현된 데이터패스가 단계연산 블록의 임계경로이며, 최대 지연 시간은 하나의 비선형함수 f_t와 세 개의 가산기 지연 시간인 $3T_a$의 합으로 표현된다. 상수 W_t 및 K_t는 외부에서 미리 결정되어 입력되기 때문에 지연시간에 영향을 주지 않는다. 따라서, 이 단계연산의 최대지연시간 T_p는 f_t+$3T_a$이다.

언폴딩은 하나의 클럭 사이클 동안 여러 개의 단계연산을 수행함으로서, 필요한 전체 클럭 수를 줄일 수 있다. 그림 2는 언폴딩 계수가 2인 SHA-1의 해쉬 연산 블록으로 언폴딩을 하지 않은 방법에 비해 전체 해쉬 함수 처리를 위해서는 절반의 클럭이 필요하다. SHA-1의 연산을 하기 위하여 기본 80 클럭이 필요하지만 하나의 클럭 시간동안 두 단계연산을 처리하기 때문에 40 클럭이 소요된다.

해쉬함수의 언폴딩은 필요한 클럭 수의 단축 뿐만 아니라 병렬처리를 통한 임계경로를 단축시킬 수 있다는 장점을 갖는다. 그림 2에서 보여지듯이, 언폴딩 계수가 2일 때, 최대지연시간 T_p는 하나의 f_t 연산의 지연시간과 4개의 가산기를 통과하는 지연시간을 합한 f_t+$4T_a$이다. 따라서 언폴딩 계수가 2인 해쉬 연산은 기본적인 SHA-1의 단계 연산보다 최대지연경로는 25% 가량 길어지지만 연산에 필요한 클럭 수가 반으로 줄어들기 때문에 결과적으로 처리량이 35% 정도 증가한다.

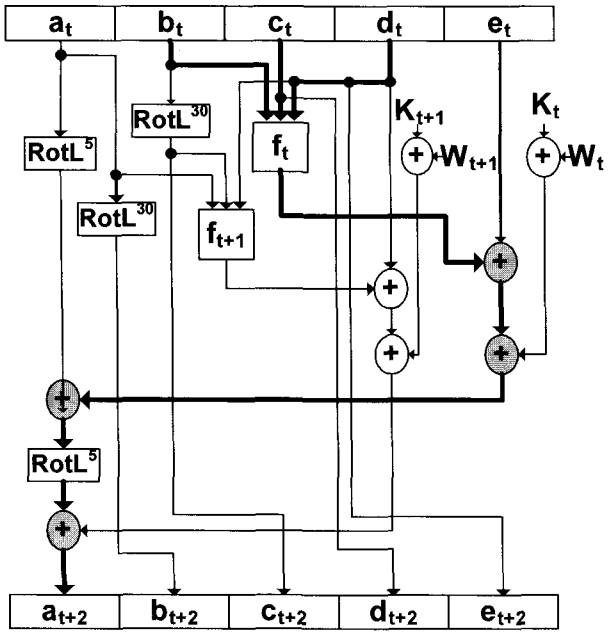


그림 2. 언폴딩 팩터가 2인 SHA-1 해쉬 연산 블럭
Fig. 2. 2-unfolded hash operation block of SHA-1.

언폴딩에 의해 SHA-1 해쉬 연산에 필요한 클럭 수는 감소하지만 최대지연시간은 증가하게 되어 처리량은 계속적으로 증가하지 않는다. 따라서 더 높은 처리량을 얻기 위해서 파이프라인 SHA-1 구조를 적용한다[3]. SHA-1 구조는 서로 다른 네 개의 비선형 함수를 가지고 20개의 클럭 사이클 동안 사용되며, 4개의 함수 블록의 병렬 처리가 가능하다. 파이프라인 구조는 SHA-1의 처리량을 4배로 증가시킬 수 있다[3].

2. 제안된 SHA-1 고속화 기법

앞 절에서 설명한 것처럼 기본 단계연산에서의 임계경로는 a_{t+1}을 구하기 위한 데이터패스이며, 이를 위한 최대 지연시간은 언폴딩을 하지 않은 기본 해쉬 단계연산의 경우 하나의 f_t 및 3개의 가산기 연산시간의 합으로 얻어진다. 로테이션의 경우 지연값이 거의 없어 임계경로에 포함되지 않는다. 따라서, 그림 1에서 나타낸 것처럼 a_{t+1}을 제외한 나머지 변수값 b_{t+1}, c_{t+1}, d_{t+1}, 그리고 e_{t+1}의 값은 이전 단계인 t번째 단계의 a_t, b_t, c_t, 그리고 d_t의 값을 이용하여 지연 없이 구할 수 있다. 이러한 특징을 이용하여 다음 단계의 연산에 필요한 변수 중 일부를 현재 단계의 연산과 병렬처리하여 임계경로를 줄일 수 있다[4].

본 논문에서는 임계경로를 줄이기 위해 새로운 변수를 정의한다. 임계경로인 a_{t+1}연산 데이터패스에 포함되는 변수중 k_t 및 w_t는 단계연산전에 결정되기 때문에 두 변수의 합 $k_t + w_t$값을 새로운 변수로 정의하고 전처리를 통해 해쉬 연산을 수행전에 미리 값을 구한 후 단계연산에 입력한다. 또한 해쉬 단계연산의 언폴딩 시 t+2 단계의 해쉬 연산 입력값중 t 단계에서 결정가능한 변수값을 이용한다. 식 (1)부터 식 (5)의 기본 단계연산중 e_{t+1}=d_t 이고 e_{t+2}=d_{t+1}=c_t 이기 때문에, $t+2$ 단계의 입력 e_{t+2}는 t단계의 입력 c_t로 결정된다. 그리고 $t+1$번째 단계에서 연산되는 $f_{t+1}(b_{t+1}, c_{t+1}, d_{t+1})$의 피연산자 b_{t+1}, c_{t+1} 그리고 d_{t+1}의 값이 a_t, $RotL^{30}(b_t)$, c_t와 동일하기 때문에 t 단계에서 연산할 수 있다. 마지막으로, 임계 경로에 포함된 $e_t + w_t + k_t$ 역시 두 단계 전에 미리 계산되고 f_t가 한 단계 전에 미리 계산되어 최대지연 시간을 줄이는 것이 가능하다.

본 논문에서는 식 (11), 식 (12), 식 (13)을 이용하여 l_t, m_t, 그리고 n_t을 새로운 변수로 정의하였다.

$$a_{t+1} = RotL^5(a_{t-1}) + l_t \quad (6)$$

$$b_{t+1} = a_t \quad (7)$$

$$c_{t+1} = RotL^{30}(b_t) \quad (8)$$

$$d_{t+1} = c_t \quad (9)$$

$$e_{t+1} = d_t \quad (10)$$

$$l_{t+1} = f_{t+1}(a_t, RotL^{30}(b_t), c_t) + m_t \quad (11)$$

$$m_{t+1} = c_t + n_t \quad (12)$$

$$n_t = k_{t+3} + w_{t+3} \quad (13)$$

위의 식을 적용하여 새로운 단계연산을 만들 수 있으며, 최대지연 시간 T_p는 f_t+T_a로 얻어진다. 이는 그림 1에서 나타낸 기본 SHA-1 단계연산의 최대지연시간 보다 두 개의 가산기 연산 시간만큼 줄어든다.

제안된 선행 연산하는 방법으로 SHA-1의 단계연산 블록으로 고속화를 구현하였다. 언폴딩을 적용한 SHA-1에서 면적당 처리량을 비교해 봤을 때 언폴딩 계수 2일 때 가장 높다. 본 논문에서는 이러한 비용도 고려하여 언폴딩 계수 2를 갖는 SHA-1 구조를 최적화 하였다. 또한 처리량을 높이기 위해 해쉬 연산에 4단 파이프라인 방법을 적용하여 회로를 구현하였다.

그림 3은 고속화를 위해 제안된 SHA-1의 해쉬 연산 블록이다. SHA-1 구조는 선행 연산 블록과 SHA-1 단계연산 블록에 언폴딩 계수 2를 적용하여 4단 파이프라인 구조로 설계하였다. 언폴딩 계수 2로 연산에 사용하는 클럭 수가 80에서 40으로 줄어든다[4]. 블록에 4단의 파이프라인으로 매 10 클럭마다 해쉬 값을 얻을 수 있고 처리량이 4배로 증가하였다[5].

선행 연산 기법에서 새로 정의된 변수를 초기화하기 위해 한주기의 클럭이 필요하고, SHA-1의 해쉬 연산을 위해서 언폴딩으로 인해 반으로 줄어든 40 클럭이 소요된다. 따라서 해쉬 연산을 완료하기 위해서는 총 41 클럭이 필요하다.

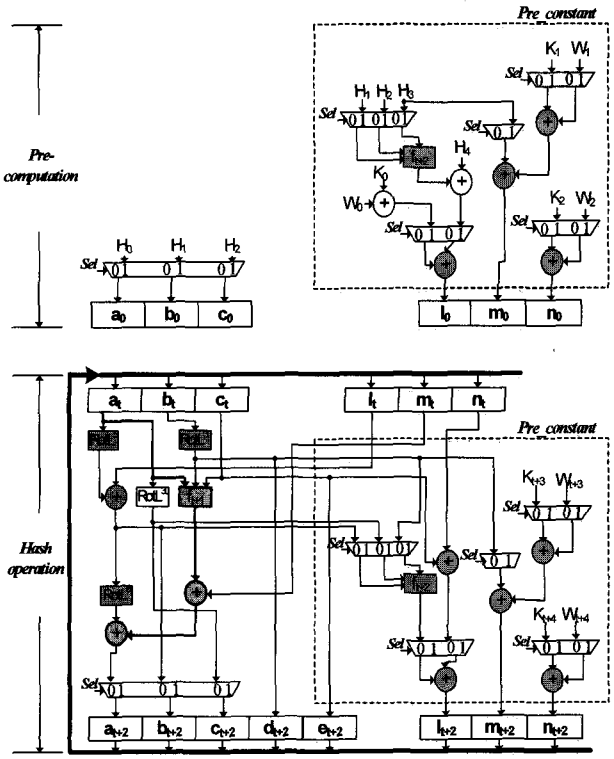


그림 3. 제안된 SHA-1 해쉬 연산 블럭
Fig. 3. Proposed hash operation block of SHA-1.

그림 3에서 보이듯이, 선행처리를 위한 첫 번째 클럭과 이후 40회의 반복연산을 구분하기 위해서 선행처리기 내의 *sel* 입력을 이용하여 제어한다. 총 5개의 멀티플렉서가 있고 *sel* 입력이 1인 경우 선행처리기로 사용되며, 미리 정의한 세 개의 변수 l_t, m_t, 그리고 n_t의 초기값을 계산한다. 두 번째 클럭부터 40회 동안 *sel* 입력은 0으로 유지되어 SHA-1의 해쉬 연산을 하게 된다. 따라서 총 41 클럭 후에는 160비트의 해쉬값을 얻는다.

Ⅲ. 시스템 레벨의 SHA-1 병렬 고속화 모델링

SHA-1은 2^{64} 보다 작은 임의의 길이의 데이터를 부호화하는 것이 가능하다. 그렇기 때문에 어떤 길이의 데이터를 어떤 순서로 처리하느냐에 따라서 연산을 완료하는데 걸리는 시간이 달라진다. 하나의 SHA-1로 연속적으로 들어오는 데이터를 처리하는 구조에서는 데이터를 처리하고 있는 도중에 입력된 데이터는 현재 데이터의 처리 시간만큼 지연이 된 후에 처리된다. 따라서, 하나의 SHA-1로 데이터를 처리하는 것 보다 여러 개를 병렬로 처리를 하면 연산을 기다리는 시간의 낭비를 줄여준다.

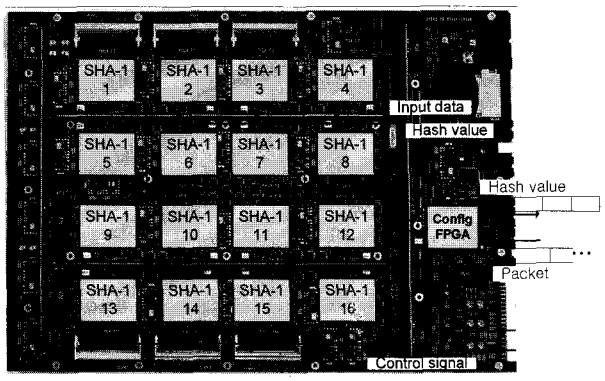


그림 4. 병렬로 연결된 SHA-1 시스템 모델
Fig. 4. Parallel SHA-1 system model.

본 장에서는 SHA-1을 병렬로 연결하여 멀티 프로세싱하는 시스템을 모델링하고 최적화구조를 얻는 결과를 기술한다. 그림 4는 연속적으로 입력되는 데이터를 처리하기 위한 FPGA 어레이 구조 platform 이다. 입력되는 임의의 길이의 데이터를 순차적으로 SHA-1에 분배하여 병렬로 처리한다. 따라서 첫 번째 데이터와 두 번째 데이터를 동시에 병렬로 처리하게 됨으로써 처리 시간은 둘 중 더 긴 처리시간 만큼 소요된다.

그림 4를 지연을 도입한 블록도 로 나타내면 그림 5와 같다. 입력되는 데이터의 길이가 일정할 때 처리시간은 식 (14)로 얻어진다. 데이터 패킷의 길이가 l 인 N개의 데이터가 병렬 처리 SHA-1 시스템으로 입력될 때의 처리 시간이다. 여기서 P_N은 네트워크의 성능, P_H는 해쉬 연산기의 성능이고 $n=\left\lceil \frac{l}{512} \right\rceil + 1$이다.

$$T_N = \frac{(N-1)\times l}{P_N} + \frac{512}{P_N} + n \times \frac{512}{P_H} \qquad (14)$$

첫 번째 SHA-1이 동작하는 중간에 두 번째 데이터의 512 비트만큼 입력되면 두 번째 SHA-1은 동작을 시작할 수 있다. 또한 입력되는 네트워크 속도가 SHA-1의 처리 속도보다 훨씬 빠르고 SHA-1도 병렬로 연산이 되므로 마지막 데이터의 바로 전 데이터까지 들어오는 시간 동안 그 이전 데이터에 대한 SHA-1 연산은 진행 중이고 마지막 데이터의 512 비트만큼 입력되면 마지막 SHA-1이 동작한다.

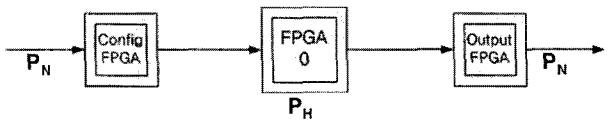


그림 5. FPGA array에 SHA-1 매핑
Fig. 5. SHA-1 mapping to FPGA array.

$N-1$번째까지의 데이터가 네트워크로 입력되는 시간은 $\frac{(N-1)\times l}{P_N}$ 이다. $\frac{512}{P_N}$는 네트워크를 통해 첫 번째 데이터의 512 비트가 입력될 때 걸리는 시간이다. SHA-1이 동작하기 위한 기본 데이터 길이는 512 비트이기 때문이다. SHA-1은 512 비트 블록 단위로 연산을 하는데 $\frac{512}{P_H}$는 SHA-1이 데이터 한 블록을 처리하는데 걸리는 시간이고 거기에 데이터를 512 비트 블록으로 나눈 수인 n을 곱하면 길이가 l인 데이터를 처리하는데 걸리는 시간이다. 즉, 패킷의 길이가 l인 N개의 데이터를 처리하는데 걸리는 시간은 마지막 데이터의 바로 전 데이터까지 입력되는 시간과 마지막까지 동작하는 SHA-1의 연산 시간을 합한 것이다.

그러나 실제 네트워크상에서 전송되는 데이터의 길이는 항상 같을 수는 없다. 데이터의 길이가 같지 않으면 각각의 데이터에 대한 SHA-1 연산기의 연산시간 또한 달라진다. 따라서 실제 네트워크상에서 임의의 길이로 입력되는 데이터를 처리하는 데 걸리는 시간과 그 데이터를 지연 없이 처리하는데 필요한 SHA-1의 FPGA 개수를 구하기 위해서는 각 데이터의 처리시간과 데이터의 입력 시간을 비교할 필요가 있다.

첫 번째 데이터가 입력되는데 걸리는 시간을 구한다. 길이가 l_1인 첫 번째 데이터가 네트워크를 통해 입력되고 첫 번째 SHA-1에서 연산하는데 걸리는 시간은 식 15이다. 여기서 $n_1=\left\lceil \frac{l_1}{512} \right\rceil + 1$이다. 이는 길이가 l_1인 데이터를 512 비트 블록으로 나눈 수이다. 첫 번째 데이터는 SHA-1이 연산을 시작할 수 있는 최소 데이터 길이인 512 비트가 입력되고 연산이 완료되면 식 (15)와 같이 출력 네트워크로 해쉬 값을 출력한다.

$$T_1 = \frac{512}{P_N} + n_1 \times \frac{512}{P_H} \qquad (15)$$

두 번째 데이터가 입력되는데 걸리는 시간을 구한다. 길이가 l_2인 두 번째 데이터가 네트워크를 통해 입력되고 SHA-1에서 처리되는 시간은 식 (16)이다. 여기서 $\frac{l_1}{P_N}$은 첫 번째 데이터가 입력되는데 걸리는 시간이다. 두 번째 데이터가 입력이 되려면 첫 번째 데이터가 모두 입력되어야 하기 때문에 그 시간을 더하고 두

번째 데이터도 마찬가지고 512 비트만큼 입력이 되고서야 SHA-1이 동작할 수 있다. 첫 번째 SHA-1이 연산을 끝내는데 걸리는 시간, T_1 과 식 (15)에서 두 번째 데이터의 512비트까지 입력되는데 걸리는 시간, $\frac{l_1}{P_N}+\frac{512}{P_N}$ 을 비교하여 T_1 이 크면 SHA-1의 FPGA 수를 늘려준다. 입력되는 데이터가 두 개뿐이라면 이 데이터를 처리하는데 걸리는 총 시간은 T_2 이다.

$$T_2=\frac{l_1}{P_N}+\frac{512}{P_N}+n_2\times\frac{512}{P_H} \quad (16)$$

이렇게 이어서 들어오는 데이터의 입력 시간의 비교를 통한 방법으로 N개의 입력 데이터를 모두 비교하여 입력되는 데이터의 지연 없이 연산이 가능한 SHA-1의 FPGA 개수를 구할 수 있다.

III. 검증 및 결과

1. 제안된 SHA-1 고속화 검증

본 논문에서 제안된 SHA-1 구조의 성능 평가를 수행하기 위해 Xilinx사의 Vertex-2 FPGA에서 합성하였다. 제안된 SHA-1 구조는 SHA 표준에 제시된 테스트 샘플을 통하여 기존에 제안된 SHA-1 구조와 성능 평가를 수행하였다[1].

표 1은 본 논문에서 제안된 SHA-1 구조와 기존 제시된 SHA-1 구조와 최대동작주파수, 성능, 그리고 회로 크기 비교 결과를 보여준다. 공정한 비교를 위해 모두 동일한 FPGA에서 합성한 결과를 비교하였다. 첫 번째와 두 번째 SHA-1 구조는 언폴딩을 사용하지 않고 80회의 반복적인 단계연산을 사용한 구조이며, 세 번째는 언폴딩을 사용하지 않았지만, 4단 파이프라인을 채택하여 성능을 크게 향상시켰다[5~7]. 네 번째와 다섯 번째 구조는 4단 파이프라인 구조를 채택하였을 뿐만 아니라 해쉬 단계 연산을 언폴딩하였다. Y. K. Lee가 제안한 구조는 언폴딩 계수 4를 채택하였고, H. Michail이 발표한 구조는 언폴딩 계수 2를 채택하였다[8~9]. 특히, H. Michail이 제안한 SHA-1 구조는 가장 최근에 발표된 SHA-1 구조로 91 MHz의 최대 동작 주파수에서 4.7 Gbps의 처리량을 갖는다[9]

SHA-1 해쉬 알고리즘의 고속 구현을 위해서는 표 1에 나타낸 것처럼 언폴딩 기법과 파이프라인 구조를 동시에 채택하는 방식이 필수적이다. 본 논문에서 제안한 SHA-1 구조는 언폴딩 계수 2를 갖는 해쉬 단계 연산기로 구현되었고, 4단 파이프라인 구조를 갖는다. 본 논문에서 제안된 SHA-1 단계연산의 최대지연시간은 하나의 f_t를 연산하는데 걸리는 시간과 3개의 가산기를 연산하는 데 걸리는 시간을 합한 f_t+2T_a이다. 해쉬 단계연산기와 변수의 일부를 미리 연산하는 선행처리기를 분리하여 이를 병렬로 처리하여 임계경로의 길이를 기존 SHA-1 구조에 비해 절반으로 감소시켰다.

본 논문에서 제안된 SHA-1 구조는 해쉬 함수 결과를 출력하기 위해 반복적인 해쉬 단계 연산을 위한 40 클럭과 이에 앞서 전처리를 수행하기 위한 하나의 클럭을 포함하여 총 41 클럭 사이클이 요구된다. 118 MHz의 최대동작주파수에서 5.9 Gbps의 처리 속도를 나타낸다. 제안된 SHA-1은 최대동작 주파수나 처리량에서 모두 이점을 가진다. 표 1에 보여지듯이 기존 SHA-1 구조중 가장 동작 속도가 빠른 것과 비교하여 26% 가량의 처리량 증가를 가져왔다.

또한 제안된 SHA-1 구조의 회로 크기는 2,894 slice로 구현되며, 언폴딩 계수가 2인 기존 SHA-1 회로에 비해 32% 가량 감소시켰다. 특히 선행연산을 위한 선행처리기와 해쉬 단계 함수 연산에 사용되는 연산 블록을 공유하고 멀티플렉서로 제어하도록 하여 회로의 크기를 크게 감소시켰다. 그림 3에서 Pre_constant 라고 하는 박스 안에 있는 부분이 초기값을 구하고 실제 해쉬 연산에서 미리 연산을 해 두는 부분을 나타낸다. 선행처리기, Pre_constant 블록에 5개의 멀티플렉서 추가로 다섯 개의 가산기와 하나의 f_t 함수만큼의 크기를 줄일 수 있다.

표 1. 제안된 SHA-1의 성능 결과의 비교
Table 1. Comparison results with the other SHA-1 design.

Implemen-tation	Freq. (Mhz)	Throughput (Mbps)	Area (Slices)	Number of Cycle
[5]	162.0	1024.0	854	80
[6]	38.6	900.0	1550	20
[7]	55	1300	8980	80
[8]	41.5	3541.0	4258	24
[9]	91.0	4715.0	4848	40
Proposed	118	5919.6	2894	41

2. SHA-1 병렬 고속화 모델링 검증

병렬로 연결된 SHA-1의 고속화 시스템 모델을 시뮬레이션을 통해 해석한다. 그림 6는 데이터의 크기가 0.1 Kbyte부터 10 Mbyte인 1024개의 데이터를 임의의 순서로 입력할 때 필요한 SHA-1의 FPGA 수와 운용도이다. 1024개의 데이터가 입력되는 동안 현재 동작하고 있는 SHA-1의 FPGA의 개수를 볼 수 있고 또한 시뮬레이션이 끝나면 지연 없이 처리하는데 필요한 SHA-1의 FPGA 개수를 알 수 있다.

이 모델링에서 지연 없이 모든 데이터를 처리하는데 필요한 FPGA 개수는 11개이다. 그림에서 보면 200번째 데이터가 들어올 즈음에 9개의 SHA-1이 동작하고 있는데 이미 그 이전에 10개의 SHA-1이 필요하기 때문에 그때는 하나의 SHA-1은 동작하지 않고 있는 것을 알 수 있다. 입력되는 데이터를 지연없이 처리 가능하지만 처리 되는 중간에 동작하지 않는 SHA-1이 늘어나 하드웨어 측면에서 효율이 떨어지는 단점이 있다.

하나의 SHA-1을 이용하여 연산할 때와 병렬로 연결하는 구조를 이용하여 연산 할 때 입력되는 데이터를 처리하는 데 걸리는 시간을 비교하여 병렬로 연결한 SHA-1의 멀티 프로세싱 연산이 얼마나 빠르게 연산하는지 검증한다. 그림 7은 병렬로 연결된 FPGA의 개수에 따른 데이터 처리 시간이다. 각각의 병렬로 연결된 SHA-1 시스템에서 입력되는 데이터를 동작 중인 FPGA 중 빨리 끝나는 곳에 입력하여 데이터의 지연을 최소한으로 한다. 하나의 SHA-1을 이용하는 구조에서 1024개의 데이터를 입력하여 연산을 마칠 때까지 모든

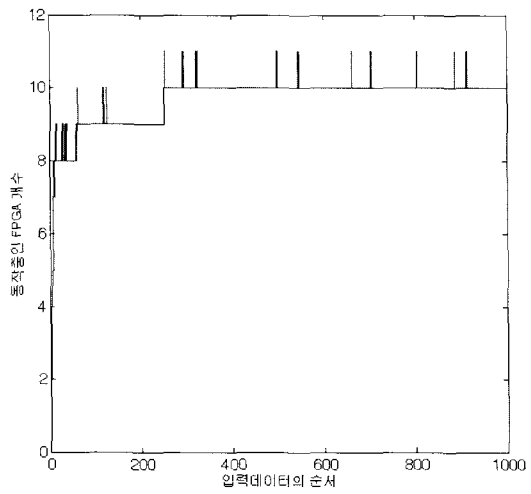


그림 6. 필요한 SHA-1의 FPGA 수
Fig. 6. The number of the necessary FPGA of SHA-1.

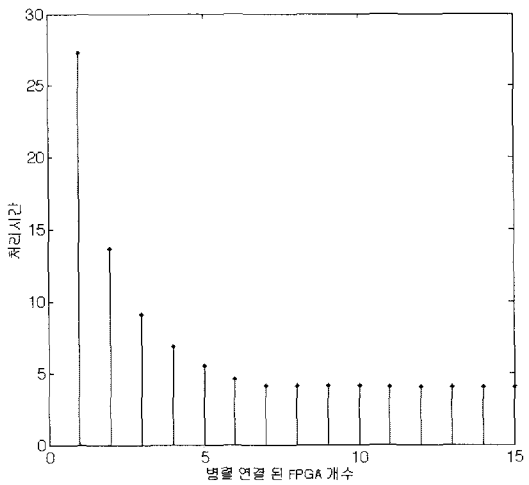


그림 7. 병렬로 연결된 FPGA의 개수에 따른 데이터 처리 시간
Fig. 7 Processing time according to the number of parallel FPGA of SHA-1 system.

데이터를 처리하는데 걸리는 시간은 27.3 초이고 두 개의 FPGA를 병렬로 연결하여 데이터를 처리하는데 걸리는 시간은 13.7 초이다. 이 두 결과에서 두 개의 FPGA를 병렬로 연결하여 데이터를 처리할 때 하나를 사용하는 것 보다 절반의 시간밖에 걸리지 않는다.

병렬로 연결된 SHA-1의 FPGA에 데이터가 입력이 되면 데이터의 입력시간과 각 FPGA의 이전에 입력된 데이터를 처리하는데 걸리는 시간을 비교하여 처리가 끝난 FPGA에 그 데이터를 입력한다. 병렬로 연결된 FPGA 수가 7 보다 많아지면 거의 4.05 초로 수렴한다. 이는 하나의 SHA_1 FPGA를 이용하여 입력되는 데이터를 처리하는 것 보다 대략 7배 정도 빠르다는 것을 알 수 있다. 그리고 데이터의 지연을 최소화 하는 방향에서 면적을 줄이는 SHA-1의 FPGA 개수는 7 개 이다. 이 결과는 데이터가 입력되는 네트워크의 속도와 사용하는 SHA-1의 성능 그리고 입력되는 데이터의 크기에 따라 차이는 있지만 멀티프로세싱 모델링을 통해 필요한 FPGA 수를 구할 수 있고 하나의 SHA-1에 비해 몇 배 고속화되는지 알 수 있다.

Ⅳ. 결 론

정보와 자료의 기밀성과 무결성을 확인하는 방법으로 나타난 것이 해쉬 알고리즘이다. SHA-1은 가장 대중화된 해쉬 알고리즘 중에 하나이고 고속화의 연구에 가장 적합한 알고리즘이다. 고속 SHA-1을 개발하기 위

하여 기존의 알고리즘들의 성능을 분석하였고 선행 연산 기법을 도입하여 임계경로를 줄이는 방법으로 고속화하는 하드웨어 구조를 제안하였다.

이 논문에서는 선행 연산 기법을 도입하여 임계경로를 줄여줌으로써 동작주파수를 높이고 처리량을 키워주는 SHA-1의 하드웨어 구조를 제안했다. 제안된 선행 연산 기법의 SHA-1 고속화 기법을 적용하고 여기에 팩터가 2인 언폴딩 기법과 네 개의 파이프라인 구조를 적용한 하드웨어 구조를 제안한다. 이 고속화 SHA-1의 최대동작 주파수는 118 MHz이고 처리량은 5.9 Gbps이다. 이는 기존의 고속화 SHA-1 보다 26% 정도 처리량이 향상되었다. 또한 선행 연산 기법을 도입하기 위해 새로 정의한 변수의 초기화를 위한 하드웨어 부분의 추가 없이 해쉬 연산부분을 함께 사용하기 때문에 32% 정도 면적이 감소하였다.

특히, 본 논문에서는 입력되는 데이터를 고속으로 처리하기 위해 SHA-1을 시스템 레벨에서 병렬 고속화를 할 수 있는 모델을 제안했다. 이 모델은 하나의 SHA-1을 이용하는 것보다 빠르게 데이터를 처리할 수 있고 필요로 하는 SHA-1 연산기를 매핑하는 FPGA 수를 구할 수 있다. 연속적인 데이터 패킷이 입력될 때 데이터의 지연없이 처리하는데 필요한 SHA-1의 FPGA 개수는 11개이고 처리시간은 4.04초이다. 그리고 7개의 FPGA를 병렬로 연결된 시스템에서 최소한의 지연으로 입력되는 데이터를 처리하는데 걸리는 시간은 4.05초이다. 처리속도는 비슷하지만 최소한의 지연으로 면적은 작게 증가하는 시스템은 FPGA가 7개인 시스템이다. 하나의 FPGA를 이용하여 데이터를 처리하는 것보다 7개가 병렬로 연결된 SHA-1 시스템을 이용하여 데이터를 처리하는 것은 시스템 크기는 7배 증가하나, 단일 SHA-1 보다 크게 7배 빠르게 병렬 처리가 가능하다.

제안된 고속 SHA-1을 IP화하여 암호 프로세서 등에 활용이 가능하다. 정보 보안, 이동통신 및 인터넷 서비스의 기본 구성요소인 해쉬 암호 알고리즘의 고속화를 통한 다양한 분야의 활용성 및 응용성을 향상 시킬 수 있고 또한 고속 해쉬 함수의 하드웨어 구현을 통하여 안정성과 암호화의 효율을 향상 시킬 수 있다. 또한 정보가 많고 긴 데이터를 압축하여 색인 할 때 간편할 수 있도록 사용되기도 한다.

참 고 문 헌

[1] FIPS PUB 180-2, Secure Hash Standard (SHA-1), National Institute of Standards and Technology (NIST), 1996.

[2] H. Michail, A. Kakarountas, G. Selimis, and C. Goutis, "Optimizing SHA-1 hash funtion for high throughput with a partial unrolling study," PATMOS, pp. 591-600, 2005.

[3] H. Michail, A. Kakarountas, O. Koufopavlou and C. Goutis, "A low power and high throughput implementation of the SHA-1 hash function," ISCAS, pp. 23-26, 2005.

[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] I. Yiakoumis, E. Papadonikolakis, E. Michail, P. kakarountas, and E. Goutis, "Maximizing the Hash function of authentication code," IEEE, pp.9-12, 2006

[6] J. Diez, S. Bojanic, Lj. Stanimirovic, C. Carreras, and O. Nieto-Taladriz, "Hash algorithms for cryptographic protocols: FPGA implementations," TELFOR, pp. 26-28, 2002.

[7] N. Skavos, et al., "An ultra high speed architecture for VLSI implementation of Hash function," ICECS, pp. 900-993, 2003.

[8] Y. K. Lee, H. Chan and I. Verbauwhede, "Throughput optimized SHA-1 architecture using unfolding transformation," ASAP, pp. 354-359, 2006.

[9] H. Michail, and C. Goutis, "Holistic methodology for designing ultra high-speed SHA-1 hashing cryptographic module in hardware," EDSSC. pp. 1-4, 2008.

저 자 소 개

이 은 희(학생회원)

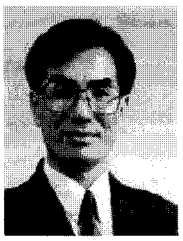
2003년 충북대학교 정보통신 공학과 학사 졸업.
2010년 충북대학교 정보통신 공학과 석사 졸업.
<주관심분야 : 통신회로 설계, 암호알고리즘, 신호처리>

이 제 훈(정회원)

1999년 충북대학교 정보통신 공학과 공학사
2001년 충북대학교 정보통신 공학과 공학석사
2005년 충북대학교 정보통신 공학과 공학박사
2005년~2006년 미국 Univ. of Southern California 박사후 과정
2006년~2009년 충북대학교 BK21 충북정보기술 사업단 초빙조교수
2009년~현재 강원대학교 삼척캠퍼스 전자공학과 조교수
<주관심분야> 저전력 고속 회로 설계, 마이크로프로세서 설계, Platform 기반 SoC 설계

장 영 조(정회원)

1979년 경북대학교 전자공학과 공학사
1982년 경북대학교 대학원 공학석사
1992년 한국과학기술원 전기 및 전자공학과 공학박사
1981년~1993년 한국전자통신연구원 자동설계 기술개발부 선임연구원
1993년~현재 한국기술교육대학교 정보기술공학부 교수
1987년~1988년 UIUC CSL 방문연구원
1999년~2000년 MDT in San Jose, 초빙연구원
<주관심분야 : 영상처리 SoC 설계, 임베디드 SoC 설계, 생체신호 SoC 설계 >



조 경 록(정회원)-교신저자

1977년 경북대학교 전자공학과 학사 졸업.
1989년 일본 동경대학교 전자공학과 석사 졸업
1992년 일본 동경대학교 전자공학과 박사 졸업.
1979년~1986년 (주)금성사 TV연구소 선임 연구원
1999년~2000년 Oregon State University 객원 교수
1992년~현재 충북대학교 전자정보대학 교수
<주관심분야 : 통신시스템 LSI 설계, 저전력 고속 회로 설계, Platform 기반의 SoC 설계>