



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0147361
(43) 공개일자 2016년12월23일

(51) 국제특허분류(Int. Cl.)

H04L 9/14 (2006.01)

(52) CPC특허분류

H04L 9/14 (2013.01)

H04L 9/0618 (2013.01)

(21) 출원번호 10-2015-0084003

(22) 출원일자 2015년06월15일

심사청구일자 2015년06월15일

(71) 출원인

고려대학교 산학협력단

서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)

(72) 발명자

김현민

서울특별시 성북구 길음로 118, 414동 1801호 (길음동, 길음뉴타운)

홍석희

서울특별시 도봉구 마들로 859-19, 103동 107호 (도봉동, 도봉한신아파트)

김성곤

서울특별시 서초구 잠원로 150, 104동 103호 (잠원동, 잠원한신아파트)

(74) 대리인

특허법인충현

전체 청구항 수 : 총 12 항

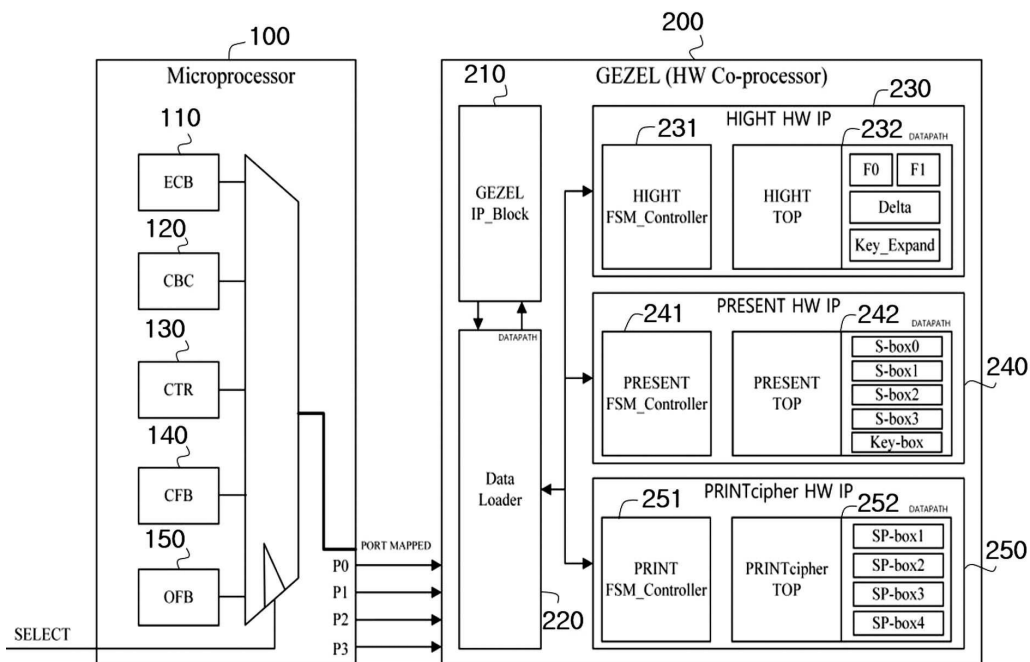
(54) 발명의 명칭 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치

(57) 요약

본 발명은 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 관한 것으로, 블록 암호 운용 방식에 따르는 마이크로프로세서 및 상기 마이크로프로세서와 연결되어 통신하는 GEZEL 하드웨어 코-프로세서(H/W co-processor)를 포함하며, 상기 GEZEL 하드웨어 코-프로세서는 GEZEL IP(intellectual property) 블록 및 데이터

(뒷면에 계속)

대표도 - 도1



패스(datapath)를 이용한 유한상태기계(Finite state machine with datapath, FSMD) 방식으로 동작시키는 하나 이상의 하드웨어 IP를 포함하여 구성된다.

상기와 같이 구성된 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 의하면, 데이터 패스를 이용한 유한상태기계(FSMD) 방식의 GEZEL을 이용하여 블록암호 및 경량암호 알고리즘의 하드웨어와 소프트웨어를 최적화하여 통합 설계함으로써, 경쟁효과(race condition)로부터 자유로워서 신호 타이밍 캐치가 쉽고, 교착 상태에 빠질 우려가 없으며, 하드웨어와 소프트웨어 상호간의 인터페이스를 최적화하여 전체적인 레이턴시(latency)를 감소시키고, 속도를 향상시킨다.

(52) CPC특허분류

H04L 9/0637 (2013.01)

H04L 2209/805 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	IITP-2015-H8501-15-1003
부처명	미래창조과학부
연구관리전문기관	정보통신산업진흥원
연구사업명	ITRC
연구과제명	제어시스템 보안 연구
기 여 율	1/1
주관기관	고려대학교 산학협력단
연구기간	2015.01.01 ~ 2015.12.31

명세서

청구범위

청구항 1

블록 암호 운용 방식에 따르는 마이크로프로세서; 및

상기 마이크로프로세서와 연결되어 통신하는 GEZEL 하드웨어 코-프로세서(H/W co-processor);를 포함하며,

상기 GEZEL 하드웨어 코-프로세서는,

GEZEL IP(intellectual property) 블록; 및

데이터패스(datapath)를 이용한 유한상태기계(Finite state machine with datapath, FSMD) 방식으로 동작시키는 하나 이상의 하드웨어 IP;를 포함하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량 암호 장치.

청구항 2

제 1항에 있어서,

상기 마이크로프로세서는,

전자 코드북(ECB), 암호 블록 체인(CBC), 카운터(CTR), 암호 피드백(CFB), 및 출력 피드백(OFB) 중 어느 하나의 방식을 선택하여 블록 암호를 운용하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량 암호 장치.

청구항 3

제 1항에 있어서,

상기 하나 이상의 하드웨어 IP는,

경량암호 HIGHT 기법의 하드웨어 IP, 경량암호 PRESENT 기법의 하드웨어 IP, 및 경량암호 PRINTcipher 기법의 하드웨어 IP를 포함하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 4

제 3항에 있어서,

상기 경량암호 HIGHT 기법의 하드웨어 IP는,

상기 경량암호 HIGHT 기법의 각 데이터패스를 유한상태기계로 정의하여 제어하는 HIGHT 제어기;

상기 경량암호 HIGT 기법의 암호 알고리즘을 수행하는 HIGHT_round_dp 데이터패스;

키 스케줄(key schedule)을 수행하는 KEY expand 데이터패스;

룩-업 테이블(look-up table) 형태로 내부 함수의 출력 값을 갖는 F_0 및 F_1 데이터패스; 및

라운드의 서브키(subkey) 조합에 사용되는 델타(delta)값을 저장하는 delta 데이터패스;를 포함하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 5

제 4항에 있어서,

상기 HIGHT 제어기는,

서브키를 16개 사용하는 4라운드 수행 후 키스케줄을 수행하되,

4라운드 수행시 레지스터의 위치를 변경하는 리타이밍(retiming)을 적용하여 크리티컬 패스(critical path)의

지연(delay)을 감소시키는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 6

제 3항에 있어서,

상기 경량암호 PRESENT 기법의 하드웨어 IP는,

상기 경량암호 PRESENT 기법의 각 데이터패스를 유한상태기계로 정의하여 제어하는 PRESENT 제어기;

상기 경량암호 PRESENT 기법의 암호 알고리즘을 수행하는 RESENT_round_dp 데이터패스;

룩-업 테이블(look-up table) 형태로 입출력을 처리하는 복수 개의 S-box 데이터패스; 및

키 스케줄 시 사용하는 Key-box 데이터패스;를 포함하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 7

제 6항에 있어서,

상기 PRESENT 제어기는,

입력과 출력이 필요한 S-box를 상기 룩-업 테이블을 이용하여 하나의 데이터패스로 처리하되 S-box Layer의 입력값들을 레지스터로 선언함으로써 상기 PRESENT_round_dp 데이터패스와 상기 S-box 데이터패스 간에 발생할 수 있는 지연을 감소시켰으며,

하나의 사이클마다 S-box Layer, Permutation Layer, 및 다음 라운드의 AddRoundKey를 수행하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 8

제 3항에 있어서,

상기 경량암호 PRINTcipher 기법의 하드웨어 IP는,

상기 경량암호 PRINTcipher 기법의 각 데이터패스를 유한상태기계로 정의하여 제어하는 PRINTcipher 제어기;

상기 경량암호 PRINTcipher 기법의 암호 알고리즘을 수행하는 PRINTcipher_round_dp 데이터패스; 및

Permutation과 S-box Substitution을 수행하는 복수 개의 SP-box 데이터패스;를 포함하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 9

제 8항에 있어서,

상기 PRINTcipher 제어기는,

PRINTcipher 기법에 따른 Permutation과 S-box Substitution의 전처리 부분을 미리 하나의 테이블로 설정하여 연산량을 감소시키며,

반복 수행되는 라운드 연산의 루프(loop) 로직을 재구성하여 하나의 루프마다 데이터의 읽기 및 쓰기 횟수를 증가시켜 수행 사이클(cycle)을 감소시키는 언폴딩(unfolding)을 적용하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 10

제 1항에 있어서,

상기 마이크로프로세서는,

상기 GEZEL 하드웨어 코-프로세서 입출력 장치와 통신 시, 메모리 맵 입출력(memory mapped I/O)방식 또는 포트 맵 입출력(port mapped I/O)방식을 이용하여 연결되는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 11

제 1항에 있어서,

상기 마이크로프로세서는,

상기 GEZEL 하드웨어 코-프로세서와 통신 시, 암호화와 메시지 송수신이 동시에 병렬로 수행하는 핸드셰이크(handshake) 프로토콜을 이용하여 메시지를 송수신하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

청구항 12

제 11항에 있어서,

상기 핸드셰이크 프로토콜을 이용한 통신 방법은,

상기 마이크로프로세서가 상기 GEZEL 하드웨어 코-프로세서로 제1명령과 제2명령을 전송하고, 상기 GEZEL 하드웨어 코-프로세서는 상기 제1명령과 상기 제2명령을 수신하여 동작을 순차적 또는 한번에 병렬적으로 수행하는 단계;

상기 GEZEL 하드웨어 코-프로세서가 수신한 상기 제1명령과 상기 제2명령 중 먼저 완료된 동작의 제1완료 시그널을 상기 마이크로프로세서로 전송하는 단계;

상기 제1완료 시그널을 수신한 상기 마이크로프로세서가 제3명령과 제4명령을 생성하여 버퍼에 저장하는 단계;

상기 GEZEL 하드웨어 코-프로세서가 수신한 상기 제1명령과 상기 제2명령 중 나중에 완료된 동작의 제2완료 시그널을 상기 마이크로프로세서로 전송하는 단계; 및

상기 제2완료 시그널을 수신한 상기 마이크로프로세서가 상기 버퍼에 저장된 상기 제3명령과 상기 제4명령을 상기 GEZEL 하드웨어 코-프로세서로 전송하는 단계;를 반복 수행하는 것을 특징으로 하는 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치.

발명의 설명

기술 분야

[0001] 본 발명은 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 관한 것으로, 더욱 상세하게는 데이터패스를 이용한 유한상태기계(FSMD) 방식의 GEZEL을 이용하여 블록암호 및 경량암호 알고리즘의 하드웨어와 소프트웨어 통합 설계하는 경량암호 장치에 관한 것이다.

배경 기술

[0002] 암호 시스템은 크게 공개키 암호 시스템과 암/복호화에 동일한 키를 사용하는 대칭키 암호 시스템으로 분류된다. RFID나 USN 같은 저전력/경량화 환경에서는 안전성을 보장하면서도 효율적인 암호 알고리즘을 필요로 한다. 이러한 환경에서 평문을 블록단위로 암호화를 수행하는 대칭키 경량 암호가 많이 사용된다. 그리고, 경량 암호는 단독으로 사용되기보다는 다양한 길이를 가진 메시지를 암호화하기 위해 다양한 운용 모드와 함께 사용된다.

[0003] 암호 시스템을 설계할 때 하드웨어와 소프트웨어 각각의 모듈로 따로 설계하기 보다는 각 설계 방법의 이점들을 최대한 살릴 수 있는 하드웨어/소프트웨어 통합 설계 기법이 많이 적용되어 사용되고 있다. 다만, 하드웨어/소프트웨어의 언어적 특징이 상당히 다른 면이 있고, 최적화 방법도 상이하기 때문에 각각의 장점을 활용하여 효과적으로 설계하기 어려운 문제점이 있다.

선행기술문헌

비특허문헌

[0004] (비특허문헌 0001) Patrick R. Schaumont, "A Practical Introduction to Hardware/Software Codesign," Springer, Dec. 2012.

발명의 내용

해결하려는 과제

- [0005] 본 발명이 해결하고자 하는 기술적 과제는 블록암호 및 경량암호 알고리즘의 하드웨어와 소프트웨어를 통합적으로 설계한 최적화 기법이 존재하지 않았으며, 그로 인해 통합 설계 과정에서 나타날 수 있는 데드락(deadlock), 레이턴시(latency) 및 경쟁 효과(race condition) 발생의 문제점을 극복하기 위한 것으로서, 데이터패스를 이용한 유한상태기계(FSMD) 방식의 GEZEL을 이용하여 블록암호 및 경량암호 알고리즘의 하드웨어와 소프트웨어를 통합 설계하는 경량암호 장치를 제공하는 데에 그 목적이 있다.

과제의 해결 수단

- [0006] 상기 목적을 달성하기 위해 본 발명은 블록 암호 운용 방식에 따르는 마이크로프로세서 및 상기 마이크로프로세서와 연결되어 통신하는 GEZEL 하드웨어 코-프로세서(H/W co-processor)를 포함하며, 상기 GEZEL 하드웨어 코-프로세서는 GEZEL IP(intellectual property) 블록 및 데이터패스(datapath)를 이용한 유한상태기계(Finite state machine with datapath, FSMD) 방식으로 동작시키는 하나 이상의 하드웨어 IP를 포함하여 구성된다.
- [0007] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 마이크로프로세서는 전자 코드북(ECB), 암호 블록 체인(CBC), 카운터(CTR), 암호 피드백(CFB), 및 출력 피드백(OFB) 중 어느 하나의 방식을 선택하여 블록 암호를 운용하는 것을 특징으로 한다.
- [0008] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 하나 이상의 하드웨어 IP는 경량암호 HIGHT 기법의 하드웨어 IP, 경량암호 PRESENT 기법의 하드웨어 IP, 및 경량암호 PRINTcipher 기법의 하드웨어 IP를 포함하는 것을 특징으로 한다.
- [0009] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 경량암호 HIGHT 기법의 하드웨어 IP는 상기 경량암호 HIGHT 기법의 각 데이터패스를 유한상태기계로 정의하여 제어하는 HIGHT 제어기와, 상기 경량암호 HIGHT 기법의 암호 알고리즘을 수행하는 HIGHT_round_dp 데이터패스와, 키 스케줄(key schedule)을 수행하는 KEY expand 데이터패스와, 룩-업 테이블(look-up table) 형태로 내부 함수의 출력 값을 갖는 F_0 및 F_1 데이터패스 및 라운드의 서브키(subkey) 조합에 사용되는 델타(delta)값을 저장하는 delta 데이터패스를 포함하는 것을 특징으로 한다.
- [0010] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 HIGHT 제어기는 서브키를 16개 사용하는 4라운드 수행 후 키스케줄을 수행하되 4라운드 수행시 레지스터의 위치를 변경하는 리타이밍(retiming)을 적용하여 크리티컬 패스(critical path)의 지연(delay)을 감소시키는 것을 특징으로 한다.
- [0011] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 경량암호 PRESENT 기법의 하드웨어 IP는 상기 경량암호 PRESENT 기법의 각 데이터패스를 유한상태기계로 정의하여 제어하는 PRESENT 제어기와, 상기 경량암호 PRESENT 기법의 암호 알고리즘을 수행하는 PRESENT_round_dp 데이터패스와, 룩-업 테이블(look-up table) 형태로 입출력을 처리하는 복수 개의 S-box 데이터패스 및 키 스케줄 시 사용하는 Key-box 데이터패스를 포함하는 것을 특징으로 한다.
- [0012] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 PRESENT 제어기는 입력과 출력이 필요한 S-box를 상기 룩-업 테이블을 이용하여 하나의 데이터패스로 처리하되 S-box Layer의 입력 값들을 레지스터로 선언함으로써 상기 PRESENT_round_dp 데이터패스와 상기 S-box 데이터패스 간에 발생할 수 있는 지연을 감소시켰으며, 하나의 사이클마다 S-box Layer, Permutation Layer, 및 다음 라운드의 AddRoundKey를 수행하는 것을 특징으로 한다.
- [0013] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 경량암호 PRINTcipher 기법의 하드웨어 IP는 상기 경량암호 PRINTcipher 기법의 각 데이터패스를 유한상태기계로 정의하여 제어하는 PRINTcipher 제어기와, 상기 경량암호 PRINTcipher 기법의 암호 알고리즘을 수행하는 PRINTcipher_round_dp 데이터패스 및 Permutation과 S-box Substitution을 수행하는 복수 개의 SP-box 데이터패스를 포함하는 것을 특징으로 한다.
- [0014] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 PRINTcipher 제어기

는 PRINTcipher 기법에 따른 Permutation과 S-box Substitution의 전처리 부분을 미리 하나의 테이블로 설정하여 연산량을 감소시키며, 반복 수행되는 라운드 연산의 루프(loop) 로직을 재구성하여 하나의 루프마다 데이터의 읽기 및 쓰기 횟수를 증가시켜 수행 사이클(cycle)을 감소시키는 언폴딩(unfolding)을 적용하는 것을 특징으로 한다.

[0015] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 마이크로프로세서는 상기 GEZEL 하드웨어 코-프로세서 입출력 장치와 통신 시, 메모리 맵 입출력(memory mapped I/O)방식 또는 포트 맵 입출력(port mapped I/O)방식을 이용하여 연결되는 것을 특징으로 한다.

[0016] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 마이크로프로세서는 상기 GEZEL 하드웨어 코-프로세서와 통신 시, 암호화와 메시지 송수신이 동시에 병렬로 수행하는 핸드셰이크(handshake) 프로토콜을 이용하여 메시지를 송수신하는 것을 특징으로 한다.

[0017] 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 있어서, 상기 핸드셰이크 프로토콜을 이용한 통신 방법은 상기 마이크로프로세서가 상기 GEZEL 하드웨어 코-프로세서로 제1명령과 제2명령을 전송하고, 상기 GEZEL 하드웨어 코-프로세서는 상기 제1명령과 상기 제2명령을 수신하여 동작을 순차적 또는 한번에 병렬적으로 수행하는 단계와, 상기 GEZEL 하드웨어 코-프로세서가 수신한 상기 제1명령과 상기 제2명령 중 먼저 완료된 동작의 제1완료 시그널을 상기 마이크로프로세서로 전송하는 단계와, 상기 제1완료 시그널을 수신한 상기 마이크로프로세서가 제3명령과 제4명령을 생성하여 버퍼에 저장하는 단계와, 상기 GEZEL 하드웨어 코-프로세서가 수신한 상기 제1명령과 상기 제2명령 중 나중에 완료된 동작의 제2완료 시그널을 상기 마이크로프로세서로 전송하는 단계 및 상기 제2완료 시그널을 수신한 상기 마이크로프로세서가 상기 버퍼에 저장된 상기 제3명령과 상기 제4명령을 상기 GEZEL 하드웨어 코-프로세서로 전송하는 단계를 반복 수행하는 것을 특징으로 한다.

발명의 효과

[0018] 상기와 같이 구성된 본 발명에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 의하면, 데이터패스를 이용한 유한상태기계(FSMD) 방식의 GEZEL을 이용하여 블록암호 및 경량암호 알고리즘의 하드웨어와 소프트웨어를 최적화하여 통합 설계함으로써, 경쟁효과(race condition)로부터 자유로워서 신호 타이밍 캐치가 쉽고, 교착 상태에 빠질 우려가 없다.

[0019] 또한, 하드웨어와 소프트웨어 상호간의 인터페이스를 최적화하여 전체적인 레이턴시(latency)를 감소시키고, 속도를 향상시키는 효과가 있다.

도면의 간단한 설명

[0020] 도 1은 본 발명의 일 실시 예에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치의 구조를 나타내는 블록도이다.

도 2는 본 발명의 일 실시예에 따른 HIGHT 유한상태기계(FSM)의 전체 흐름을 도시한 도면이다.

도 3은 본 발명의 일 실시예에 따른 PRESENT 유한상태기계(FSM)의 전체 흐름을 도시한 도면이다.

도 4는 본 발명의 일 실시예에 따른 PRINTcipher 유한상태기계(FSM)의 전체 흐름을 도시한 도면이다.

도 5는 본 발명의 일 실시예에 따른 핸드셰이크 프로토콜을 이용한 통신 방법을 나타내는 순서도이다.

발명을 실시하기 위한 구체적인 내용

[0021] 본 발명은 다양한 변형 및 여러 가지 실시 예를 가질 수 있는바, 특정 실시 예들을 도면에 예시하고 상세한 설명에 보다 상세하게 설명하고자 한다. 본 발명을 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 본 발명의 요지를 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다.

[0022] 먼저 본 발명에서 하드웨어와 소프트웨어 통합 설계를 위한 GEZEL에 대해서 설명한다. GEZEL은 소프트웨어와 하드웨어 통합설계가 용이하도록 개발된 언어이자 설계 플랫폼이다. 또한 GEZEL의 문법은 직관적이라서 소프트웨어 설계에 익숙한 설계자들이 하드웨어를 쉽게 설계할 수 있다.

[0023] 사이클 기반의 하드웨어 기술 언어인 GEZEL은 데이터패스를 이용한 유한상태기계(Finite State Machine with Datapath, FSMD) 방식으로 되어있다. 다른 하드웨어 설계언어는 유한상태기계(Finite State Machine, FSM)기반인데 반해 데이터패스가 결합된 유한상태기계를 구성하기 때문에 하드웨어 구현 최적화 이론을 적용하기 쉽다.

- [0024] 특히, GEZEL은 하드웨어 단독 설계뿐만 아니라 하드웨어와 소프트웨어의 통합 설계에도 적합하도록 개발되었다. 통합 설계에 있어서 GEZEL은 ARM, 8051 마이크로프로세서의 IP 코어 블록을 지원하기 때문에 별도의 컴파일 없이도 GEZEL에서 지원하는 IP 블록 코드를 삽입하는 것만으로도 ARM, 8051 마이크로프로세서들과 통신이 가능하다.
- [0025] 이하, 본 발명의 일 실시예를 첨부된 도면을 참조하여 GEZEL을 이용한 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치에 대해서 상세히 설명하기로 한다.
- [0026] 도 1은 본 발명의 일 실시예에 따른 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치의 구조를 나타내는 블록도이다. 도 1을 참조하면 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치는 마이크로프로세서(100)와 GEZEL 하드웨어 코-프로세서(200)를 포함하여 구성된다.
- [0027] 마이크로프로세서(100)는 전자 코드북(Electronic Code Book, ECB) 운용모드(110), 암호 블록 체인(Cipher Block Chaining, CBC) 운용모드(120), 카운터(Counter, CTR) 운용모드(130), 암호 피드백(Cipher FeedBack, CFB) 운용모드(140) 및 출력 피드백(Output FeedBack, OFB) 운용모드(150)의 5개의 운용모드를 통해 하드웨어/소프트웨어의 통합설계를 구축한다.
- [0028] 전자 코드북 운용모드(110)는 운용 방식 중 가장 간단한 구조를 가지며, 암호화하려는 메시지를 여러 블록으로 나누어 각각 암호화하는 방식으로 되어있다.
- [0029] 암호 블록 체인 운용모드(120)는 각 블록이 암호화되기 전에 이전 블록의 암호화 결과와 엑스클루시브 오아(exclusive or, XOR) 로직으로 결합되며, 첫 블록의 경우에는 초기화 벡터가 사용된다. 상기 초기화 벡터가 같은 경우 출력 결과가 항상 같기 때문에, 매 암호화마다 다른 초기화 벡터를 사용해야 한다.
- [0030] 카운터 운용모드(130)는 블록 암호를 스트림 암호로 바꾸는 구조를 가진다. 카운터 운용모드(130)는 각 블록마다 현재 블록이 몇 번째인지 값을 얻어, 그 숫자와 난스(nonce)를 결합하여 블록 암호의 입력으로 사용한다. 각 블록 암호에서 연속적인 난수를 얻은 다음 암호화하려는 문자열과 XOR 로직으로 결합한다. 카운터 운용모드(130)는 각 블록의 암호화 및 복호화가 이전 블록에 의존하지 않으므로 병렬적으로 동작하는 것이 가능하다. 또한, 암호화된 문자열에서 원하는 부분만 복호화하는 것도 가능하다.
- [0031] 암호 피드백 운용모드(140)는 블록 암호를 자기 동기 스트림 암호로 변환한다. 암호 피드백 운용모드(140)의 동작 방식은 전자 코드북 운용모드(110)와 유사하며, 특히 암호 해제 방식은 전자 코드북 운용모드(110) 암호화의 역순과 매우 유사하다.
- [0032] 출력 피드백 운용모드(150)는 블록 암호를 동기식 스트림 암호로 변환한다. XOR 명령의 대칭 때문에 암호화와 암호 해제 방식은 완전히 동일하다.
- [0033] 마이크로프로세서(100)는 상기의 5가지 운용모드를 이용함으로써 사용 플랫폼과 상황에 맞추어 운용모드와 암호를 선택할 수 있는 유연성을 제공한다. 또한, 마스터키가 하드웨어에 저장된 것이 아닌 소프트웨어로부터 입력받는 것임으로 추후 키가 유출되었을 경우 쉽게 수정 및 업데이트가 가능하다는 안전성 측면에서의 유연성을 갖는다.
- [0034] 설계 측면에서도 하드웨어와 소프트웨어의 통합 설계 시 각 대칭키 암호의 알고리즘 중 키 스케줄 부분을 소프트웨어에서 구현하여 미리 각 라운드키 값을 사전 계산하고, 계산된 라운드키를 하드웨어로 보내는 방식으로도 구현할 수 있다. 그에 따라 하드웨어에서는 키 스케줄 연산 처리과정이 삭제됨으로 면적이 감소하고 속도 증대 효과를 기대할 수 있다.
- [0035] 다음으로 GEZEL 하드웨어 코-프로세서(200)는 GEZEL IP 블록(210), 데이터 로더(220), HIGHT 하드웨어 IP(230), PRESENT 하드웨어 IP(240), PRINTcipher 하드웨어 IP(250)를 포함한다.
- [0036] GEZEL IP 블록(210)은 GEZEL을 이용하여 설계한 하드웨어와 소프트웨어 간의 통신이 쉽게 이루어질 수 있도록 GEZEL에서 제공하는 통신을 위한 IP 블록으로, 본 발명에서는 마이크로프로세서(100)와 GEZEL 하드웨어 코-프로세서(200) 간의 통신이 가능하도록 연결해주는 IP 블록을 의미한다.
- [0037] 데이터 로더(220)는 GEZEL 하드웨어 코-프로세서(200) 내부의 메모리로 구성되어 있으며, 마이크로프로세서(100)의 명령어 또는 데이터를 받아 데이터 로더(200)에 미리 저장하여 GEZEL 하드웨어 코-프로세서(200)에서 빠른 연산이 이루어지도록 한다. 여기서 하드웨어와 소프트웨어 통합 설계시 레이턴시(latency)에 가장 많은 영향을 미치는 부분이 소프트웨어와 하드웨어 간의 데이터를 읽고 쓰는데 있다는 것을 감안한다면, 데이터 로더(220)는 하드웨어에서 중요한 역할을 한다.

- [0038] HIGHT 하드웨어 IP(230)는 HIGHT 알고리즘을 이용한 하드웨어 IP로 HIGHT 제어기(231) 및 HIGHT 데이터 패스(232)를 포함하여 구성된다.
- [0039] HIGHT 제어기(231)는 각 데이터패스를 유한상태기계로 정의하여 상기 각 데이터패스들을 이용한 연산 기능을 수행한다.
- [0040] HIGHT 데이터 패스(232)는 HIGHT 암호 알고리즘 전체를 수행하는 HIGHT_round-dp 데이터패스와, 키 스케줄을 수행하는 KEY expand 데이터패스, 룩-업 테이블(look-up table) 형태로 내부 함수와의 출력 값을 갖고 F_0 , F_1 데이터패스 및 라운드의 서브키(subkey) 조합에서 사용되는 델타(delta)값을 저장하는 delta 데이터패스의 총 5개의 데이터패스로 구성된다.
- [0041] 상기와 같은 구성으로 HIGHT 제어기(231)는 서브키를 16개 사용하고 난 이후에 입력된 비밀키의 상위 64비트와 하위 64비트가 각각 8비트씩 순환 왼쪽 시프트를 하는 것에 착안하여 최적화를 수행하였다. 즉, 서브키를 16개 사용하는 4라운드 수행 후 키 스케줄을 수행할 수 있게 구현하고, 4라운드 수행 시 발생하는 크리티컬 패스(critical path)의 증가문제를 해결하기 위하여 리타이밍(retiming) 기법을 적용하여 크리티컬 패스의 지연을 감소시킨다.
- [0042] 여기서 HIGHT 제어기(231)의 최적화 방법은 도 2를 참조하여 설명한다. 도 2는 본 발명의 일 실시예에 따른 HIGHT 유한상태기계(FSM)의 전체 흐름을 도시한 도면으로 총 8개의 상태로 구성된다.
- [0043] 먼저 Check start(310) 상태에서 통신 모듈과 소프트웨어의 통신이 연결되었는지 확인하여, 연결이 되지 않았다면 암호화를 시작하지 않는다. 정상적으로 연결이 되고 상기 통신 모듈로부터 평문과 키를 입력받았다면 다음 상태로 넘어간다.
- [0044] 다음으로 INIT(320) 상태에서는 외부 통신 모듈로부터 평문과 키를 입력받은 후 HIGHT 내부 레지스터에 저장한다. 그리고 HIGHT 암호화를 수행하는 동안 사용될 레지스터를 초기화하는 상태를 의미한다.
- [0045] 다음으로 Pre round(330) 상태에서는 라운드 시작 전에 입력 받은 평문에 대해서 화이트닝키를 삽입하는 과정을 수행한다.
- [0046] 다음으로 Sub_Key(340) 상태에서는 HIGHT의 라운드키로부터 2라운드에 사용될 8개의 서브키를 생성한다. delta 데이터패스의 출력값과 현재 키를 조합하여 서브키를 생성한다.
- [0047] 다음으로 Round(350) 수행 상태에서는 앞서 생성된 8개의 서브키를 이용하여 2라운드를 수행하고 그 값을 레지스터에 저장한다. 이때 HIGHT는 마지막 라운드가 기존 라운드와 다르기 때문에 마지막 라운드는 32라운드 알고리즘을 수행하고 최종 화이트닝키를 삽입한 값을 출력한다. 이 상태에서는 키 스케줄을 시작하기 전에 3라운드 중 처음 2라운드만 수행했을 때는 다시 서브키를 생성하는 Sub_Key(340) 상태로 회귀되고, 4라운드가 모두 수행되었으면 키 스케줄을 수행하는 상태로 이행된다.
- [0048] 다음으로 Delta Change(360) 상태에서는 라운드의 서브키 조합에 사용되는 델타 데이터패스의 입력값인 delta_constant를 갱신한 후 레지스터에 저장한다.
- [0049] 다음으로 Key schedule(370) 상태에서는 키를 갱신하는 키 스케줄을 수행하는 상태로, 현재 키의 상위 64비트와 하위 64비트 각각 8비트씩 좌측 순환 이동(left shift rotation)시킨 값을 키 레지스터에 저장한다.
- [0050] 마지막으로 FINISH(380) 상태에서는 최종 라운드까지 수행된 암호문을 외부 통신 모듈로 전송한다. 그리고 HIGHT의 시작 상태를 알려주는 레지스터를 다시 초기화시킨다.
- [0051] 이상과 같이 HIGHT 하드웨어 IP(230)의 구성을 확인하였으며, 다음으로 PRESENT 하드웨어 IP(240)에 대해서 설명한다.
- [0052] PRESENT 하드웨어 IP(240)는 PRESENT 알고리즘을 이용한 하드웨어 IP로 PRESENT 제어기(241) 및 PRESENT 데이터 패스(242)를 포함하여 구성된다.
- [0053] PRESENT 제어기(241)는 각 데이터패스를 유한상태기계로 정의하여 상기 각 데이터패스들을 이용한 연산을 수행한다.
- [0054] PRESENT 데이터 패스(242)는 PRESENT 암호 알고리즘 전체를 수행하는 PRESENT_round_dp 데이터패스와, 룩-업 테이블의 형태로 S-box 0~3 입/출력을 처리하는 4개의 S-box 데이터패스 및 키 스케줄시 사용하는 Key-box 데이터패스로 총 6개의 데이터패스로 구성되며, PRESENT 암호 알고리즘의 라운드 함수는 AddRoundKey, S-box Layer

및 Permutation Layer의 세가지 과정을 포함한다.

- [0055] 상기와 같은 구성으로 PRESENT 제어기(241)는 S-box Layer의 입력과 출력이 필요한 모듈인 룩-업 테이블을 하나의 데이터패스로 구현하였기 때문에 S-box Layer의 입력값들을 레지스터로 선언하여 PRESENT_round_dp 데이터패스와 S-box 데이터패스 간에 발생할 수 있는 지연을 최소화하였다. 따라서 한 사이클마다 S-box Layer, Permutation Layer 및 다음 라운드의 AddRoundKey 를 수행한다. 그 이유는 레지스터에 할당된 값은 다음 라운드에 사용할 수 있으므로 S-box 입력을 중심으로 사이클을 나누는 것이 효율적이기 때문이다. 그리고 레지스터를 8비트 단위로 분할한 이후에 비밀키와 XOR를 수행함으로써 64비트 단위 XOR를 수행할 때보다 효율적이다. 또한 소프트웨어인 마이크로프로세서(100)상에서 구현된 키 스케줄링에 의해 전처리된 라운드 키값들을 전송받아서 사용하므로 하드웨어의 효율을 증가시킬 수 있다.
- [0056] 여기서 PRESENT 제어기(241)의 최적화 방법은 도 3을 참조하여 설명한다. 도 3은 본 발명의 일 실시예에 따른 PRESENT 유한상태기계(FSM)의 전체 흐름을 도시한 도면으로 총 6개의 상태로 구성된다.
- [0057] 먼저 Check start(410) 상태에서 통신 모듈과 소프트웨어의 통신이 연결되었는지 확인하여, 연결이 되지 않았다면 암호화를 시작하지 않는다. 정상적으로 연결이 되고 통신 모듈로부터 평문과 키값을 입력받았다면 다음 상태로 넘어간다.
- [0058] 다음으로 INIT(420) 상태에서는 외부 통신 모듈로부터 평문과 키값을 입력 받은 후 PRESENT_round_dp 내부 레지스터에 저장한다. 그리고 암호화를 수행하는 동안 사용될 레지스터를 초기화하는 상태이다. 본 발명의 실시예들은 라운드 수행 알고리즘이 첫 라운드 AddRoundKey 과정을 수행하지 않도록 구성되어 있으므로 초기 레지스터에 미리 AddRoundKey를 수행하여 저장하는 것이 바람직하다.
- [0059] 다음으로 Round(430) 상태에서는 PRESENT의 라운드 함수를 수행하는 상태이다. 라운드 수행 알고리즘은 S-box Layer, Permutation Layer 및 다음 라운드의 AddRoundKey를 수행한 이후에 8비트 레지스터 8개에 나누어 저장한다. 이 상태를 최종라운드까지 수행한다.
- [0060] 다음으로 Key schedule(440) 상태에서는 소프트웨어로부터 받은 각 라운드키값을 갱신하고 라운드 상수인 ctr을 증가시킨다. 이때 키는 키값을 저장하는 레지스터에 저장된다.
- [0061] 다음으로 S-box(450) 상태에는 Present 암호 알고리즘의 S-box 구성에 맞는 대치(substitution) 연산을 수행한다.
- [0062] 마지막으로 Finish(460) 상태에서는 최종 라운드까지 수행 시 암호문을 외부 통신 모듈로 전송한다. 그리고 PRESENT의 시작상태를 알려주는 레지스터를 초기화 시킨다.
- [0063] 이상과 같이 PRESENT 하드웨어 IP(240)의 구성을 확인하였으며, 다음으로 PRINTcipher 하드웨어 IP(250)에 대해서 설명한다.
- [0064] PRINTcipher 하드웨어 IP(250)는 PRINTcipher 알고리즘을 이용한 하드웨어 IP로 PRINTcipher 제어기(251) 및 PRINTcipher 데이터 패스(252)를 포함하여 구성된다.
- [0065] PRINTcipher 제어기(251)는 각 데이터패스를 유한상태기계로 정의하여 상기 각 데이터패스들을 이용한 연산을 수행한다.
- [0066] PRINTcipher 데이터 패스(252)는 PRINTcipher 암호 알고리즘을 전체를 수행하는 PRINTcipher_round_dp 데이터 패스와, 룩-업 테이블의 형태로 3비트 단위로 Permutation과 S-box Substitution을 수행하는 4개의 SP-box 데이터패스로 구성된다.
- [0067] 상기와 같은 구성으로 PRINTcipher 제어기(251)는 라운드 함수가 한 사이클 당 2라운드를 수행하게 언폴딩(unfolding) 기법을 적용하여 총 사이클을 감소시켰다. 여기서 언폴딩은 프로그램의 루프 로직을 풀어서 한 번에 여러 연산을 수행하는 방법으로, 이때 공간의 소비가 있으므로 속도적 측면과 공간의 효율을 고려해야 한다.
- [0068] 또한 PRINTcipher 제어기(251)는 암호화 과정 중 Permutation과 S-box Substitution을 따로 처리하지 않고 하나의 테이블로 미리 설정하여 연산량을 감소시킨다. 따라서 PRINTcipher의 기존 구조인 Key XOR - Permutation - S-box Substitution을 Key XOR - SP-box 구조로 구성하였고 라운드 연산을 수행 시 SP-box 이전에 레지스터를 두어 현재 라운드의 SP-box, 다음 라운드의 Key XOR를 하나의 프로세스로 구성한다. 그리고 상기 프로세스를 한 사이클 당 두 번 수행하게 구성한다. 이로 인해 총 수행 사이클이 감소되는 효과를 얻을 수 있다.

- [0069] 나아가, SP-box 데이터패스 입력값을 3비트 레지스터 16개로 선언하고, Key XOR 과정을 SP-box 데이터패스 내부에서 수행하도록 구성한다. 그에 따라 48비트의 Key XOR 과정에서 3비트 단위로 나누어져 Key XOR 과정을 수행하여 속도 측면에서의 효율성이 있다.
- [0070] 여기서 PRINTcipher 제어기(251)의 최적화 방법은 도 4를 참조하여 설명한다. 도 4는 본 발명의 일 실시예에 따른 PRINTcipher 유한상태기계(FSM)의 전체 흐름을 도시한 도면으로 총 5개의 상태로 구성된다.
- [0071] 먼저, Check start(510) 상태에서 다른 암호화 동일하게 통신 모듈과 소프트웨어의 통신이 연결되었는지 확인하여 연결이 되지 않았다면 암호화를 시작하지 않는다. 정상적으로 연결이 되고 통신 모듈로부터 평문과 키값을 입력 받았다면 다음 상태로 넘어간다.
- [0072] 다음으로 INIT(520) 상태에서는 외부 통신 모듈로부터 평문과 키값을 입력 받은 후 PRINTcipher_round_dp 내부 레지스터에 저장한다. 그리고 본 실시예에서는 PRINTcipher의 라운드 수행 알고리즘이 첫 라운드 Key XOR 과정을 수행하지 않도록 구성되어 있으므로 초기 레지스터에 저장할 때 미리 Key XOR를 수행하여 저장한다.
- [0073] 다음으로 2R-Round(530) 상태에서는 PRINTcipher의 2라운드 함수를 수행한다. Permutation - S-box Substitution 과정을 미리 테이블로 설정한 3비트 SP-box 데이터패스를 설계하고 SP-box 데이터패스 내부에서 Key XOR를 수행한다.
- [0074] PRINTcipher_round_dp에서는 SP-box 데이터패스의 출력값을 입력받고 상기 과정을 한 사이클 당 두 번 수행하여 한 사이클에 두 개의 라운드가 수행된다.
- [0075] 다음으로 SP-box(540) 상태에서는 PRINTcipher의 특성을 나타내는 대치 및 치환(Substitution and Permutation) 연산을 수행한다.
- [0076] 마지막으로 Finish(550) 상태에서는 암호문을 외부 통신 모듈로 전송한다. 그리고 PRINTcipher의 시작 상태를 알려주는 레지스터를 초기화시킨다.
- [0077] 이상으로 하드웨어와 소프트웨어 통합 설계가 가능한 경량암호 장치의 마이크로프로세서(100)와 GEZEL 하드웨어 코-프로세서(200)의 구성을 확인하였다. 다음으로 마이크로프로세서(100)와 GEZEL 하드웨어 코-프로세서(200) 간의 통신방법에 대해서 설명한다.
- [0078] GEZEL을 이용한 통합설계에서 마이크로프로세서(100)가 하드웨어 입출력 장치와 통신 시 사용하는 방식은 메모리 맵 입출력(memory mapped I/O) 방식과 포트 맵 입출력(port mapped I/O) 방식이 있다. 메모리 맵 입출력 방식은 통신 시 입출력과 메모리의 주소 공간을 분리하지 않는 방식이고, 포트 맵 입출력 방식은 입출력 회로와 메모리 주소 공간을 분리하는 방식이다. 입출력 회로가 메모리에 비해 속도가 느린 경우 이 두 공간을 분리하면 속도와 공간의 효율성이 있다. 본 발명의 실시예에서는 IP 블록을 연결 시 포트 맵 입출력 방식을 사용하여 속도와 공간의 효율성을 향상시켰으며, 이는 본 발명의 통신 방법을 포트 맵 입출력 방식으로 한정하는 것은 아니다.
- [0079] 하드웨어와 소프트웨어 간의 통신 시 신뢰성을 높이기 위해 마이크로프로세서(100)와 GEZEL 하드웨어 코-프로세서(200) 간에 개선된 핸드셰이크(handshake) 프로토콜을 사용한다. 서로 통신이 확립되면 완료가 되었다는 완료 시그널(done signal)을 보내어 데이터를 주고 받을 수 있는 상태가 되었는지 확인한다. 도 5를 참조하여 개선된 핸드셰이크 프로토콜을 이용한 통신 방법에 대해서 구체적으로 설명한다.
- [0080] 도 5는 본 발명의 일 실시예에 따른 핸드셰이크 프로토콜을 이용한 통신 방법을 나타내는 순서도이다.
- [0081] 먼저 마이크로프로세서(100)가 GEZEL 하드웨어 코-프로세서(200)로 제1명령과 제2명령을 전송하고, GEZEL 하드웨어 코-프로세서(200)는 제1명령과 제2명령을 수신하여 동작을 순차적 또는 한번에 병렬적으로 수행한다(S10).
- [0082] 다음으로 GEZEL 하드웨어 코-프로세서(200)가 수신한 제1명령과 제2명령 중 먼저 완료된 동작의 제1완료 시그널을 마이크로프로세서(100)로 전송한다(S20). 제1완료 시그널을 수신한 마이크로프로세서(100)는 제3명령과 제4명령을 생성하여 버퍼에 저장한다(S30).
- [0083] 다음으로 GEZEL 하드웨어 코-프로세서(200)가 수신한 제1명령과 제2명령 중 나중에 완료된 동작의 제2완료 시그널을 마이크로프로세서(100)로 다시 전송한다(S40). 제2완료 시그널을 수신한 마이크로프로세서(100)는 버퍼에 저장된 제3명령과 제4명령을 GEZEL 하드웨어 코-프로세서(200)로 전송한다(S50).
- [0084] 이상으로 핸드셰이크 프로토콜을 이용한 통신 방법을 확인하였다. 상기 단계를 반복적으로 수행하여 통신을 수

행하며, 이를 통해 GEZEL 하드웨어 코-프로세서(200)에서 암호화 수행 중에도 마이크로프로세서(100)와 메시지를 주고 받을 수 있다.

[0085] 상기된 본 발명의 실시예들에 따르면, 본 발명은 경량암호가 사용될 수 있는 IoT 제품이나 RFID/USN 및 스마트 카드 등에 탑재되어 효율적인 암호화 연산에 사용될 수 있으며, 기존에 탑재되어 있는 PRINTcipher 경량암호 기반의 전자 제품에 대체하여 탑재할 수 있고, 다양한 운용모드에 바로 적용할 수 있어 제품 및 플랫폼에 상관없이 적용할 수 있다는 장점을 갖는다.

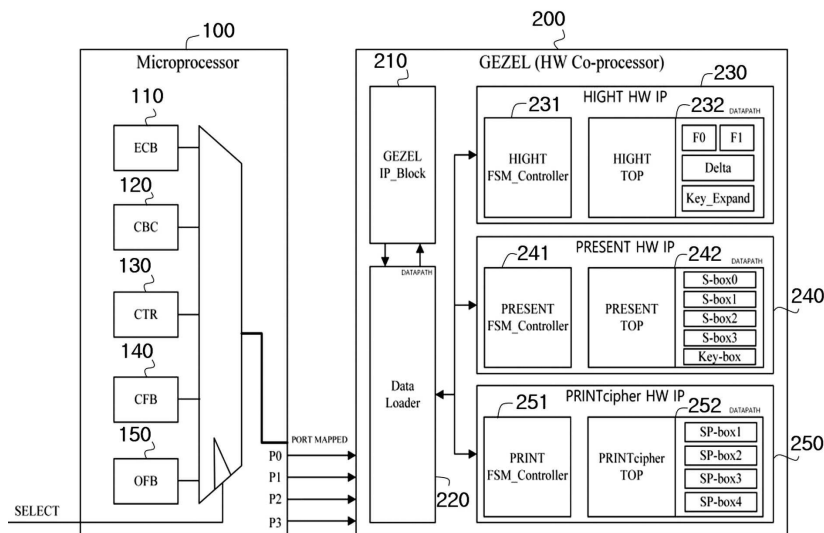
[0086] 본 명세서에 기재된 본 발명의 실시 예와 도면에 도시된 구성은 본 발명의 가장 바람직한 실시 예에 관한 것이고, 발명의 기술적 사상을 모두 포괄하는 것은 아니므로, 출원시점에 있어서 이들을 대체할 수 있는 다양한 균등물과 변형 예들이 있을 수 있음을 이해하여야 한다. 따라서 본 발명은 상술한 실시 예에 한정되지 아니하며, 청구범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 기술분야에서 통상의 지식을 가진 자라면 누구든지 다양한 변형실시가 가능한 것은 물론이고, 그와 같은 변경은 청구범위 기재의 권리범위 내에 있게 된다.

부호의 설명

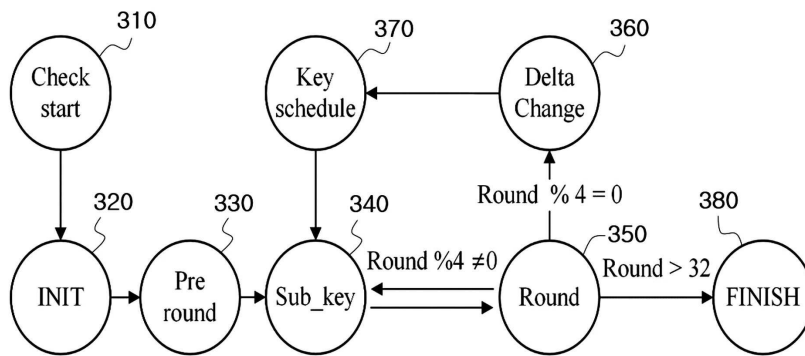
[0087] 100: 마이크로프로세서 110: 전자 코드북(ECB) 운용모드
120: 암호 블록 체인(CBC) 운용모드 130: 카운터(CTR) 운용모드
140: 암호 피드백(CFB) 운용모드 150: 출력 피드백(OFB) 운용모드
200: GEZEL 하드웨어 코-프로세서 210: GEZEL IP 블록
220: 데이터 로더 230: HIGHT 하드웨어 IP
240: PRESENT 하드웨어 IP
250: PRINTcipher 하드웨어 IP

도면

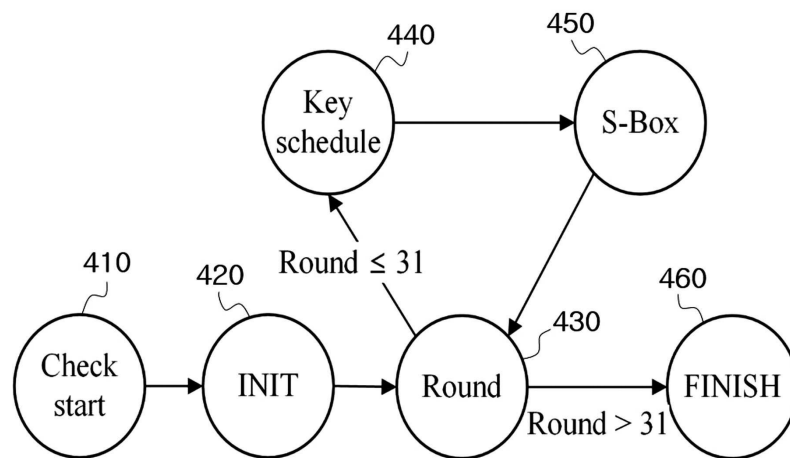
도면1



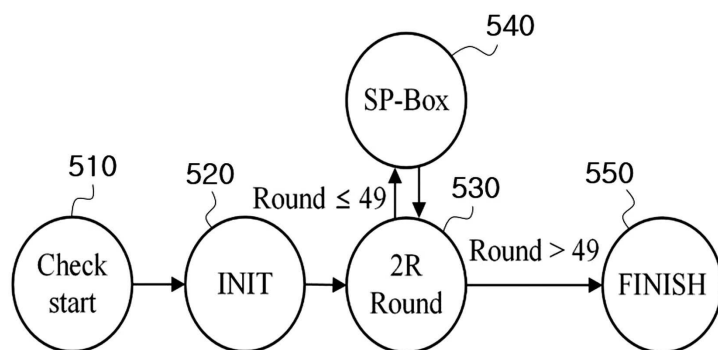
도면2



도면3



도면4



도면5

