

RISC-V SoC의 커스텀 IP용 인터페이스를 이용한 캐시 부채널 공격 (Cache Side-Channel Attacks Exploiting the RISC-V Coproprocessor Interface on an SoC Platform)

황 예 원 [†] 서 태 원 ^{**} 구 건 재 ^{***}
(Yewon Hwang) (Taeweon Suh) (Gunjae Koo)

요약 현재 System-on-Chip(SoC)은 다양한 서드파티 IP를 활용하여 설계된다. 외부 공급자로부터 제공된 커스텀 IP는 내부적으로 검증하기 어렵기 때문에 악의적인 공급자가 삽입한 하드웨어 트로이목마에 의해 심각한 보안 위협이 될 수 있다. 본 논문은 최근 주목받고 있는 RISC-V 오픈소스 SoC 플랫폼에서 보조프로세서를 위한 명령어 전달에 사용되는 RoCC 인터페이스를 이용한 캐시 부채널 공격 시나리오를 제시한다. 본 논문에서는 공격자가 RoCC 명령어를 통해 커스텀 IP 내부의 악의적인 메모리 접근 로직을 제어함으로써 Flush+Reload 형태의 캐시 부채널 공격을 효과적으로 수행할 수 있음을 입증했다. 이 방법은 기존 캐시 부채널 공격 방법보다 9.4배 빠르게 플러시 단계가 가능하여 공격자가 좀 더 수월하게 기밀 정보를 유출할 수 있다. 이 논문은 이러한 공격 시나리오를 통해 오픈소스 프로세서를 사용하는 SoC 설계에서 하드웨어 보안 공격에 대비한 방어 전략의 필요성을 제안한다.

키워드: 캐시 부채널 공격, RISC-V SoC, 하드웨어 트로이목마, 서드파티 IP 보안

Abstract A modern System-on-Chip (SoC) incorporates multiple third-party intellectual properties (IPs) provided by external vendors. Such third-party IPs can be vulnerable to security attacks exploiting hardware Trojans. Namely, attackers may include malicious hardware logic that can perform unauthorized operations within a third-party coprocessor. In this paper, we present a cache side-channel attack scenario that exploits the coprocessor interface, called RoCC, in a RISC-V open-source SoC platform. We demonstrate that attackers can effectively execute a Flush+Reload type cache side-channel attack by activating a malicious memory access logic in a custom IP exploiting RoCC instructions. Our evaluation exhibits the proposed attack can perform flush operations 9.4 times faster than traditional cache side-channel attack methods. This paper highlights the need for defense mechanisms against hardware security attacks in SoC design utilizing open-source processors.

Keywords: cache side-channel attack, RISC-V SoC, hardware Trojan, third-party IP security

· 이 성과는 2024년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2021R1C1C1012172, NRF-2022R1A2C1011469)
· 이 논문은 2024 한국컴퓨터종합학술대회에서 'RISC-V SoC 환경에서 RoCC를 이용한 캐시 부채널 공격'의 제목으로 발표된 논문을 확장한 것임

[†] 학생회원 : 고려대학교 컴퓨터학과 학생
hywp33@korea.ac.kr

^{**} 종신회원 : 고려대학교 컴퓨터학과 교수
suhtw@korea.ac.kr

^{***} 정회원 : 고려대학교 컴퓨터학과 교수(Korea Univ.)
gunjaekoo@korea.ac.kr
(Corresponding author)

논문접수 : 2024년 9월 27일
(Received 27 September 2024)

논문수정 : 2024년 11월 4일
(Revised 4 November 2024)

심사완료 : 2025년 1월 20일
(Accepted 20 January 2025)

Copyright©2025 한국정보과학회 : 개인 목적이거나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.
정보과학회논문지 제52권 제2호(2025. 2)

1. 서론

System-on-Chip(SoC)은 하나의 칩 내부에 여러 동작을 지원하는 다양한 종류의 IP 블록들을 포함하고 있다. SoC 설계 업체들은 제품의 시장 출시 시간을 단축하기 위해 사전에 설계된 서드파티 IP들을 외부 설계 업체로부터 공급받아 칩 내부에 포함시키고 있다. 그러나 이렇게 외부에서 제공된 모듈은 내부 동작을 상세히 확인하기 어렵기 때문에 보안적으로 취약하다. 즉, 신뢰할 수 없는 공급자가 서드파티 IP를 설계하는 과정에서 악성 하드웨어 모듈을 삽입하는 하드웨어 트로이목마 공격과 같은 보안 위협이 발생할 수 있다. IP 모듈 공급자는 내부 구조를 알 수 없도록 해당 IP를 공급하기 때문에 하드웨어 트로이목마는 설계 과정 중에 발견하기가 매우 어렵다. 또한 하드웨어 트로이목마 공격은 소프트웨어 코드에 많이 의존하지 않기 때문에 소프트웨어 패턴을 이용한 탐지도 거의 불가능하다.

본 논문은 현재 많은 주목을 받고 있는 오픈소스 명령어 구조(instruction set architecture, ISA) 프로세서인 RISC-V 프로세서를 포함하는 SoC 플랫폼에서 IP 블록 및 가속기 동작 제어를 위해 사용되는 RoCC(Rocket Chip Coprocessor) 명령어 인터페이스를 통한 캐시 부채널 공격을 제안한다. 기존 RISC-V ISA는 캐시 블록을 플러시(flush)하기 위한 별도의 명령어를 제공하지 않기 때문에, 이전 연구에서는 메모리를 여러 번 접근하여 플러시 동작을 구현했다. 그러나 본 논문에서는 RoCC 명령어로 제어되는 하드웨어 IP의 악의적인 동작을 통해 더 효율적으로 보안 공격이 가능하다는 것을 입증한다.

2. 배경

2.1 하드웨어 트로이목마

요즘 칩 시장에서는 System on Chip (SoC)설계에서 발생 가능한 보안적 문제의 심각성이 떠오르고 있다. 오늘날 칩 생산 시장에서는 SoC 설계과정에서 비용과 시간을 절감하기 위해 하드웨어 IP를 적극적으로 사용하고 있다[1]. 서드파티 IP에 대한 의존도가 높아지면서 신뢰할 수 없는 IP를 보안성 검증 없이 사용하기도 하는데, 이러한 하드웨어 IP에 하드웨어 트로이목마가 삽

입되어 있을 수 있다.

하드웨어 트로이목마는 설계 회사나 파운드리에서 제작 중 신뢰할 수 없는 설계자 또는 설계도구를 사용하여 IC에 악의적인 회로를 주입하는 공격이다. 이 공격은 트리거(trigger)와 페이로드(payload)라는 2가지 요소를 포함한다 (그림 1). 트리거는 하드웨어 트로이목마가 깨어날 타이밍을 결정하며 페이로드는 트리거된 트로이목마가 어떤 일을 실행할지를 결정한다. 이러한 공격은 내부적인 SoC 테스트 중에는 발견하기 어렵고 대부분 제품을 장시간 사용할 때 발견된다. 하드웨어 트로이목마 공격으로 칩의 오작동이 발생되거나 정보가 유출될 수 있다. 특히, 정보 유출은 군사, 경제 거래 등 보안에 큰 주의가 필요한 분야에서는 매우 큰 위협으로 작용한다 [2]. 최근에는 SoC에 포함되는 회로가 점점 복잡해질 뿐만 아니라 악성 회로 또한 점점 은밀해지면서 탐지가 더욱 어려워지고 있는 상황이다. 그렇지만, 복잡한 SoC 개발을 위해서는 매우 많은 종류의 서드파티 IP 활용이 불가피하기 때문에, 신뢰할 수 없는 공급자에 의한 하드웨어 트로이목마 및 이를 이용한 보안 공격을 막기 위한 연구는 매우 중요하다[3].

2.2 캐시 부채널 공격

캐시 부채널 공격은 공격자가 캐시 블록에 특정 데이터를 미리 준비한 후, 피해자가 프로그램을 실행하게 하여 캐시 접근 패턴을 분석함으로써 피해자의 중요한 정보를 유출할 수 있는 보안 공격 기법이다. 캐시 부채널 공격은 주로 캐시 hit과 miss에 따른 접근 시간 차이를 이용해 수행된다. 이러한 공격은 대체로 다음과 같은 세 가지 단계로 이루어진다[4].

- ① 공격자는 피해자가 실행하기 전에 캐시 블록에 데이터를 미리 준비해 둔다.
- ② 피해자가 비밀 데이터를 사용하여 메모리에 접근하고, 이로 인해 캐시 내용이 변경되기를 공격자는 기다린다.
- ③ 공격자는 자신이 미리 준비한 캐시 블록에 접근하여 접근 시간을 측정함으로써, 피해자의 비밀 데이터를 추정해낸다.

대표적인 캐시 부채널 공격 기법 중 하나인 Flush+Reload 공격은 멀티 코어 구조에서 공격자와 피해자가 공유 캐시에 접근할 수 있는 상황을 이용해 진행된다[4]. 공격자는 ①단계와 같이 캐시 블록을 무효화하기 위해 x86 ISA의 cflush 명령어와 같은 캐시 제어 명령어를 사용하여 캐시를 플러시한다. 이후, 피해자가 메모리에 저장된 비밀 데이터에 접근하기를 기다린다(②단계). 마지막으로, 공격자는 첫 번째 단계에서 플러시한 캐시 블록에 접근하여 접근 시간을 측정한다(③단계). 이 과정에서 공격자는 캐시 hit과 miss에 따른 접근 시간 차이를 이

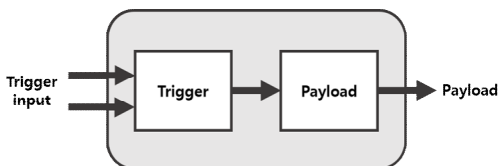


그림 1 하드웨어 트로이목마
Fig. 1 Hardware Trojan

용해 비밀 데이터를 추정할 수 있다.

그러나 Flush+Reload 공격을 RISC-V 프로세서에서 시도하기에는 어려움이 있다. 이는 RISC-V에는 clflush 명령어와 같은 캐시 라인을 플러시 하는 명령어가 없기 때문이다. 이를 해결하기 위해 이전 연구에서는 디미 주소를 여러 번 접근하여 저장되어 있던 L1캐시의 데이터를 모두 삭제하는 방법으로 Flush+Reload 공격을 시도하였다[5]. 다만 이 방법은 L1 캐시 사이즈의 4배의 크기를 가지는 메모리 접근을 실행해야 높은 확률로 L1 캐시의 데이터를 삭제할 수 있다고 언급하고 있다[5]. 즉, 이러한 방법은 너무 많은 메모리 접근을 요구하므로 플러시 속도가 느려서 공격의 효율성이 떨어진다. 하지만 커스텀 IP와 CPU 코어가 캐시를 공유하는 SoC구조라면 공격자가 커스텀 IP에 하드웨어 트로이목마를 삽입하여 공유하는 캐시를 원하는 대로 제어하는 시나리오가 가능할 수 있다. 따라서 이 논문에서는 RISC-V SoC내에서 새로운 플러시 방법을 만들어 빠르게 캐시를 조작하여 피해자의 기밀 정보를 유출하는 공격 방법을 제시할 것이다.

3. RISC-V SoC 환경에서의 공격 방법

3.1 RISC-V SoC 플랫폼

그림 2는 보안 공격 실험에 사용된 RISC-V SoC 플랫폼 환경을 보여준다. RISC-V SoC 플랫폼에 포함된 가속기는 RISC-V의 RoCC 명령어 포맷을 통해 제어할 수 있다. RoCC 명령어는 RISC-V 명령어 포맷에서 연산코드(operation code) 필드를 통해서 가속기에 특화된 새로운 명령어 정의가 가능하며 RISC-V의 명령어 구조에 포함되어 있다. 그러므로, RISC-V를 포함하는 범용 IP에서도 폭넓게 RoCC 명령어 지원이 가능하다[6].

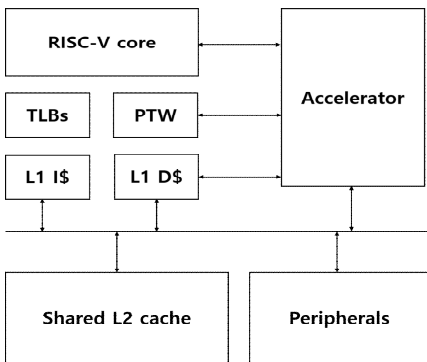


그림 2 RISC-V SoC 플랫폼 구조
Fig. 2 RISC-V SoC platform architecture

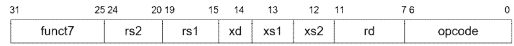


그림 3 RoCC 명령어 포맷
Fig. 3 RoCC instruction format

```

1 #define ROCC_INSTRUCTION_0_R(x, rs1, rs2, func7){
2   asm volatile(
3     ".insn r " STR(CAT(CUSTOM_, x)) " ", "
4     STR(0x3) " ", " STR(func7) " ", x0, %0, %1"
5     :
6     : "r"(rs1), "r"(rs2));
7 }
    
```

그림 4 RISC-V 명령어 매크로 코드
Fig. 4 RISC-V instruction macro code

RISC-V 코어는 L1 캐시, translate lookaside buffer(TLB), 그리고 페이지 테이블 워커(page table walker)를 포함하고 있다. RISC-V 코어와 가속기는 인터커넥션 네트워크를 통해 L2 공유 캐시에 접근할 수 있다.

RISC-V 코어는 RoCC 명령어를 사용하여 연결된 가속기 IP를 제어한다. RoCC 명령어는 그림 3에서 보듯이 표준화된 RISC-V 명령어 포맷을 사용한다[7]. RISC-V 코어는 RoCC 명령어를 통해 가속기 내부의 레지스터 값을 업데이트하거나 CPU의 레지스터 값을 전달할 수 있다. 이 명령어를 사용해 가속기 내부의 레지스터 값을 변경함으로써, RISC-V 코어는 간단한 명령어 실행을 통해 가속기의 동작을 제어할 수 있다.

그림 4는 RISC-V 코어가 RoCC 명령어 포맷을 이용해 가속기로 제어 명령을 전달하는 코드를 보여준다. 이 명령어는 RISC-V 코어의 rs1과 rs2 레지스터에 저장된 값을 읽어 가속기의 레지스터에 저장하는 과정을 나타낸다. 이를 통해 RISC-V 코어는 가속기의 내부 레지스터 값을 변경하여 가속기의 동작을 제어할 수 있으며, 가속기의 레지스터 값을 읽어 그 상태를 모니터링하는 것도 가능하다.

가속기가 활성화되면 가속기는 내부에 구현된 하드웨어 로직에 따라 필요한 데이터를 메모리에서 가져오고 가져온 데이터는 가속기 내부의 버퍼나 스트래치패드 메모리에 저장된다. 그림 2에서 볼 수 있듯이 RISC-V 코어와 가속기는 L2 캐시를 공유하기 때문에 가속기가 메모리에서 읽어온 데이터는 자동으로 L2 캐시에 할당된다. 가속기의 작업이 완료되면 가속기는 내부 상태 레지스터를 업데이트하고, RISC-V 코어는 이 레지스터 값을 읽어 가속기의 상태를 파악한다.

3.2 가속기 IP를 이용한 캐시 조작

이번 문단에서는 RISC-V SoC 플랫폼에서 RoCC 인터페이스로 연결된 가속기 IP가 어떻게 악의적인 캐시

부채널 공격에 활용될 수 있는지에 대해 설명한다. 가속기가 RoCC 명령어에 의해 활성화되면 가속기는 지정된 메모리 공간에 접근한다. 그러나 RISC-V 코어는 가속기 내부에서 일어나는 메모리 접근 동작을 모니터링할 수 없다. 만약 가속기에 악의적인 하드웨어 로직이 삽입되어 있다면 가속기는 지정된 데이터뿐만 아니라 임의의 데이터에도 접근할 수 있다. 이러한 경우 가속기가 수행하는 메모리 접근은 공유된 L2 캐시를 통해 이루어지므로 가속기가 임의로 메모리에 접근하여 L2 캐시의 내용을 조작할 수 있다. 비록 가속기가 코어와 직접 연결된 L1 캐시를 직접적으로 변경하지는 않지만, 오픈소스 RISC-V SoC 설계 플랫폼인 Chipyard는 inclusive 캐시 정책을 채택하고 있기 때문에 L2 캐시의 변경이 L1 캐시에도 영향을 미칠 수 있다. 즉, 가속기를 통해 L2 캐시의 데이터를 플러시하게 되면 inclusive 캐시 정책에 따라 L1 캐시에 있는 해당 데이터가 무효화된다.

가속기가 활성화되기 전에는 해당 데이터가 L1과 L2 캐시에 모두 존재한다. 그러나 가속기가 악의적인 메모리 접근을 통해 해당 데이터가 저장된 L2 캐시 세트를 채우게 되면, 그 데이터는 L2에서 제거되고(L2 eviction) 동시에 L1에서도 무효화된다. 이로 인해 해당 데이터에 대한 접근 시간이 더 길어지는 현상이 발생할 수 있다.

3.3 악의적인 가속기 IP를 이용한 Flush+Reload 공격

앞서 설명한 바와 같이, 공격자는 RoCC로 연결된 가속기 IP에 하드웨어 트로이목마를 삽입하여 캐시 부채널 공격을 수행할 수 있다. 즉, 가속기가 활성화될 때, L2 캐시를 의도하지 않은 데이터로 채워 L1 캐시에 저장된 여러 블록을 무효화시키고, 이를 통해 Flush+Reload 방식의 공격이 다음과 같이 가능해진다.

① 가속기에 삽입된 트로이목마 로직이 가속기가 활성화되었을 때 지속적으로 메모리에 접근하여 L1 캐시에 있는 블록들을 무효화시킨다.

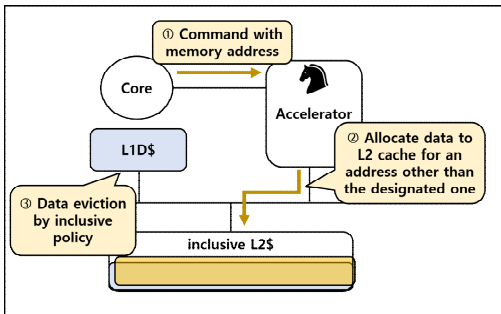


그림 5 공격 시나리오
Fig. 5 Attack scenario

표 1 실험 구성
Table 1 Configuration

Core	Rocket core
L1 cache	16KB, 64B/block
L2 cache	512KB, 64B/block
Accelerator	Gemmini
Scratchpad capacity	256 KB

② 공격자는 피해자가 비밀 데이터를 사용해 메모리에 접근하여 캐시의 내용이 변경되기를 기다린다.

③ 공격자는 무효화된 캐시 블록들에 접근해 접근 시간을 측정함으로써, 피해자의 비밀 데이터를 추정한다.

4. 실험

4.1 실험 환경 구성

본 연구에서는 오픈소스 RISC-V SoC 설계 플랫폼인 Chipyard를 사용하여 RISC-V 코어와 가속기 IP로 이루어져 있는 SoC 개발 환경을 구성하였으며 Verilator를 사용하여 시뮬레이션을 수행하였다[8].

실험에 사용한 Chipyard 환경의 구성은 표1에서 확인할 수 있다. Chipyard 환경에서는 RISC-V Rocket 코어를 CPU로 이용하고 코어와 연결되는 가속기로 Gemmini systolic array 가속기를 사용하였다. 실험에서는 Gemmini 가속기에 하드웨어 트로이목마가 삽입되었다고 가정한다.

메모리 접근 시간 측정은 RISC-V CSR 명령어인 rdcycle을 사용하여 측정하였다(그림 6). rdcycle 명령어는 csrrs rd, cycle, x0 명령어로 실행되며, 프로그램이 시작될 때부터 클럭 사이클을 측정하는 CPU 내부 하드웨어 카운터에서 측정된 값을 결과 레지스터(rd)에 쓴다. 이를 통해 각 메모리 접근의 시간을 정밀하게 측정할 수 있다.

4.2 실험 결과

섹션 3에서 설명한 방법을 따라 Flush+Reload 공격을 구현했으며, 그림 7에서는 그 결과를 제시하고 있다. 그림에서 알 수 있듯이 여러 인덱스 중 특정 인덱스에 서만 메모리 접근 지연 시간이 짧게 나타나는 것을 확

```

1 For i from 0 to L1SIZE-1
2   addr=&array[i*LINE];
3   time1=rdcycle();
4   junk = *addr;
5   time2=rdcycle()-time1;
6 End
    
```

그림 6 메모리 접근 시간 측정 코드
Fig. 6 Memroy access time measurement code

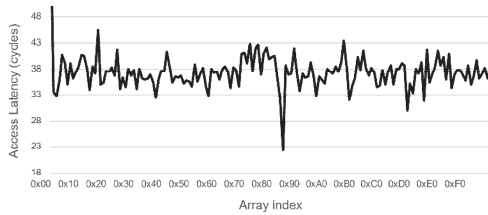


그림 7 인덱스 별 접근지연시간
Fig. 7 Access latency by indexes

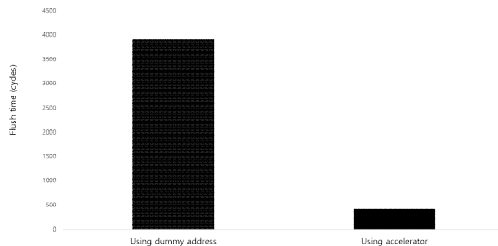


그림 8 더미주소와 가속기 사용 시 플러시 속도 차이
Fig. 8 Cache flush time between dummy address accesses and RoCC instructions

인할 수 있었다. 이는 공격자가 L1 캐시를 플러시한 후, 피해자가 해당 인덱스에 접근했음을 의미하며, 공격자는 각 캐시 라인에 대해 사이클 시간을 측정하여 피해자가 특정 인덱스에 접근했음을 추론할 수 있다.

이전에 언급했듯이 RISC-V 아키텍처에는 캐시 라인을 플러시하는 표준 명령어가 없기 때문에 이전의 연구에서는 더미 메모리 주소를 여러 번 접근하여 기존에 있던 L1 캐시의 데이터를 모두 제거시켰다[5]. 이러한 공격 방법은 캐시 사이즈를 크게 넘어가는 크기의 캐시 접근을 필요로 하여 공격 속도가 느리며 비정상적인 캐시 접근으로 검출될 수도 있다. 하지만 본 논문에서는 공격자가 RoCC 명령어를 사용하여 손쉽게 하드웨어 트로이목마를 트리거할 수 있기 때문에 보다 짧은 시간 내에 L1 캐시의 데이터를 모두 플러시시킬 수 있다. 그림 8에서 확인할 수 있듯이 기존의 더미 주소를 이용하여 L1 캐시의 데이터를 플러시 했을 때보다 가속기 IP와 RoCC 명령어를 이용하여 L1 캐시를 플러시 했을 때 클럭 사이클 소모가 9.4배 빨라진 것을 확인할 수 있다. 플러시 속도가 빠르기 때문에 공격자는 이전보다 reload 동작을 더 많이 진행할 수 있고 자연스럽게 사용자의 접근 패턴을 파악하는 것이 쉬워져서 공격 성공

확률이 증가하게 된다. 이는 공격자가 Flush+Reload 공격을 더 효율적으로 수행할 수 있다는 것을 의미하며, 공격 진행 속도가 빨라져 방어도 어려워진다.

5. 결론

본 논문에서는 RoCC 인터페이스를 적용한 가속기 IP를 활용하여 RISC-V SoC 플랫폼에서 캐시 부채널 공격이 발생할 수 있는 시나리오를 제시하였다. 서드파티 IP를 적극적으로 활용하는 SoC 플랫폼에서는 하드웨어 트로이목마와 같은 보안 공격이 매우 심각한 위협이 되고 있다. 본 연구에서는 많은 관심을 받고 있는 오픈소스 프로세서와 SoC 개발 플랫폼을 사용하여 캐시 부채널 공격이 가능함을 입증하였다. 특히, 플러시 속도의 향상으로 인해 공격자가 기밀 정보를 유출할 가능성이 더욱 높아질 수 있음을 확인하였다. 이를 통해 현재의 오픈소스 SoC 플랫폼에서 이러한 보안 공격을 방어하기 위한 대책이 필요하다는 점을 강조하였다.

References

- [1] Bhunia, Swarup, et al. "Hardware Trojan attacks: Threat analysis and countermeasures." *Proc. of the IEEE*, 102.8, pp. 1229-1247, 2014.
- [2] Kounelis, Fotios, Nicolas Sklavos, and Paris Kitsos. "Run-time effect by inserting hardware trojans, in combinational circuits." *2017 Euromicro Conference on Digital System Design (DSD)*. IEEE, 2017.
- [3] Beaumont, Mark, Bradley Hopkins, and Tristan Newby. "Hardware trojans—prevention, detection, countermeasures (a literature review)." (2011).
- [4] Briongos, Samira, et al. "(RELOAD+ REFRESH): Abusing Cache Replacement Policies to Perform Stealthy Cache Attacks." *The 29th USENIX Security Symposium (USENIX Security 20)*. 2020.
- [5] Gonzalez, Abraham, et al. "Replicating and mitigating spectre attacks on an open source risc-v microarchitecture." *Third Workshop on Computer Architecture Research with RISC-V (CARRV)*. 2019.
- [6] HPCA 2015 Bootcamp, RISC-V Rocket Chip Tutorial. [Online]. Available: <https://hPCA2015.org/riscv-rocket-chip-tutorial-bootcamp-hPCA2015.pdf>
- [7] Waterman, Andrew, et al. "The RISC-V instruction set manual." Volume I: User-Level ISA, version 2, pp. 1-79. 2014.
- [8] Amid, Alon, et al. "Chipyard: Integrated design, simulation, and implementation framework for custom socs." *IEEE Micro*, 40.4, pp. 10-21, 2020.



황 예 원

2024~현재 한국전자통신연구원 석사후
연수연구원. 2022~2024 고려대학교 컴
퓨터학과 (석사). 2018~2022 한국교통대
학교 전자공학과 (학사)



서 태 원

2008~현재 고려대학교 정보대학 컴퓨터
학과 교수. 2007~2008 Systems Engineer,
Intel Corporation, Hillsboro, Oregon
1998~2001 하이닉스반도체 선임연구원
1995~1998 LG 종합기술원 주임연구원
2021~2006 Georgia Institute of
Technology ECE (박사). 1993~1995 서울대학교 전자공학과
(석사). 1989~1993 고려대학교 전기공학과 (학사)



구 건 재

2020~현재 고려대학교 컴퓨터학과 부교수
2018~2020 홍익대학교 전자전기공학부
조교수. 2003~2011 LG전자 책임연구원
2011~2018 University of Southern
California EE (박사). 2001~2003 서울
대학교 전기컴퓨터공학부 (석사) 1997~
2001 서울대학교 전기공학부 (학사)