



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0054788
(43) 공개일자 2024년04월26일

(51) 국제특허분류(Int. Cl.)
G06F 11/34 (2006.01) G06F 11/263 (2006.01)
G06F 11/30 (2006.01)
(52) CPC특허분류
G06F 11/3447 (2013.01)
G06F 11/263 (2013.01)
(21) 출원번호 10-2022-0135182
(22) 출원일자 2022년10월19일
심사청구일자 2022년10월19일

(71) 출원인
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
윤주범
서울특별시 송파구 충민로4길 19, 704동 401호(장지동, 송파파인타운7단지)
이영우
경기도 구리시 경춘로276번길 21, 105동 2003호(수택동, 구리역 한양수자인 리버시티)
(74) 대리인
특허법인티앤씨

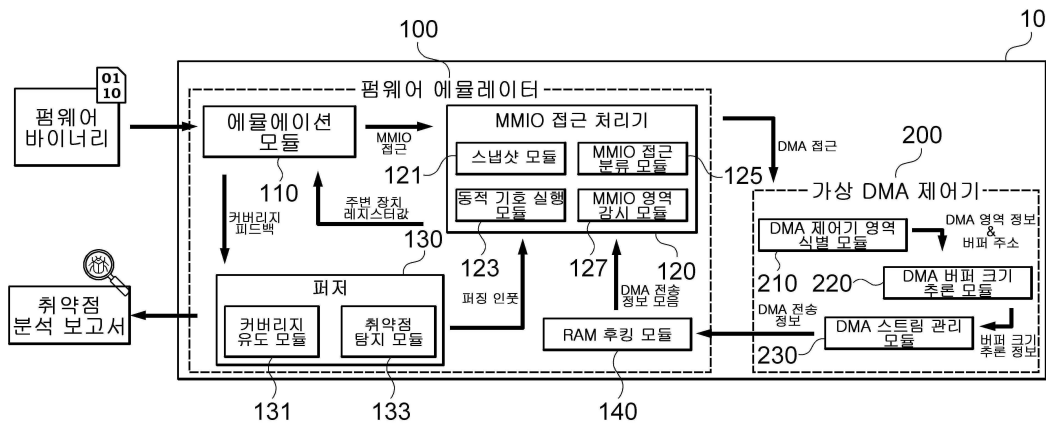
전체 청구항 수 : 총 16 항

(54) 발명의 명칭 DMA 모델링을 이용한 임베디드 장치 점검 시스템 및 방법

(57) 요약

임베디드 장치 점검 시스템 및 방법이 개시된다. 본 개시에 따른 임베디드 장치 점검 방법은, 점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적 분석하여 펌웨어에 대응하는 하드웨어 정보를 획득하는 단계, 하드웨어 정보에 기반하여 펌웨어에 대한 에뮬레이션을 시작하는 단계, 메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근 처리기를 이용하여 펌웨어의 주변 장치에 대한 MMIO 접근을 처리하는 단계, MMIO 접근의 처리 결과를 이용하여 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 모델링하는 단계, DMA 작업으로서 램(Random Access Memory, RAM) 영역에 대한 쓰기 작업을 수행하는 단계, 퍼저(Fuzzer)에서 생성되는 퍼징 입력 값에 기반하여 MMIO 접근 처리기 및 DMA 제어기를 이용하여 펌웨어를 반복 실행하는 단계, 및 사전 설정된 조건이 충족되는 경우 펌웨어에 대한 에뮬레이션을 종료하는 단계를 포함한다.

대표도



(52) CPC특허분류

G06F 11/3013 (2013.01)

G06F 11/3037 (2013.01)

G06F 11/3041 (2013.01)

G06F 11/3457 (2013.01)

(72) 발명자

김주환

서울특별시 광진구 군자로 175-2, 304호(군자동)

유지현

서울특별시 광진구 광나루로28길 18, 302호(화양동)

이 발명을 지원한 국가연구개발사업

과제고유번호 1711159975

과제번호 IITP-2022-2018-0-01423

부처명 과학기술정보통신부

과제관리(전문)기관명 정보통신기획평가원

연구사업명 대학ICT연구센터육성지원사업

연구과제명 지능형 비행로봇 융합기술 연구

기 여 율 1/1

과제수행기관명 세종대학교 산학협력단

연구기간 2022.01.01 ~ 2022.12.31

명세서

청구범위

청구항 1

점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적 분석하여 상기 펌웨어에 대응하는 하드웨어 정보를 획득하는 단계;

상기 하드웨어 정보에 기반하여 상기 펌웨어에 대한 에뮬레이션을 시작하는 단계;

메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근 처리기를 이용하여 상기 펌웨어의 주변 장치에 대한 MMIO 접근을 처리하는 단계;

상기 MMIO 접근의 처리 결과를 이용하여 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 모델링하는 단계;

DMA 작업으로서 램(Random Access Memory, RAM) 영역에 대한 쓰기 작업을 수행하는 단계;

퍼저(Fuzzer)에서 생성되는 퍼징 입력 값에 기반하여 상기 MMIO 접근 처리기 및 상기 DMA 제어기를 이용하여 상기 펌웨어를 반복 실행하는 단계; 및

사전 설정된 조건이 충족되는 경우 상기 펌웨어에 대한 에뮬레이션을 종료하는 단계를 포함하는, 임베디드 장치 점검 방법.

청구항 2

제1항에 있어서,

상기 하드웨어 정보는,

RAM 영역의 주소 및 크기, 플래시 메모리 영역의 주소 및 크기, 상기 램 영역 또는 상기 플래시 메모리 영역에 대한 접근 권한 중 적어도 일부에 대한 정보를 포함하는, 임베디드 장치 점검 방법.

청구항 3

제1항에 있어서,

상기 MMIO 접근을 처리하는 단계는,

기호 실행(symbolic execution)을 통해 상기 펌웨어의 마이크로컨트롤러 유닛(Microcontroller Unit, MCU)과 관련된 주변 장치의 MMIO 접근을 처리하는, 임베디드 장치 점검 방법.

청구항 4

제3항에 있어서,

상기 MMIO 접근을 처리하는 단계는,

기호 실행을 통해 상기 펌웨어에 대한 주변 장치의 초기 MMIO 접근 처리를 위한 분기를 탈출할 수 있는 값 정보를 식별하는 단계;

상기 퍼저를 통해 MMIO 영역의 주변 장치에 대해서 임의의 퍼징 입력이 필요한지 여부를 판단하는 단계; 및

상기 값 정보와 상기 초기 MMIO 접근 처리를 위한 명령어의 주소를 가리키는 프로그램 카운터의 쌍을 저장하되, 상기 퍼징 입력이 필요한 경우 상기 퍼징 입력에 필요한 비트의 개수 및 위치를 추가로 저장하는 단계를 포함하는, 임베디드 장치 점검 방법.

청구항 5

제1항에 있어서,

상기 DMA 제어기를 모델링하는 단계는,

MMIO 영역에서 상기 DMA 제어기의 영역을 식별하는 단계;

상기 DMA 제어기의 영역에 대응하는 DMA 영역 정보 및 DMA 버퍼 주소를 식별하는 단계;

상기 DMA 영역 정보 및 상기 DMA 버퍼 주소에 기반하여, DMA 버퍼의 크기를 추론함으로써 DMA 버퍼 크기 정보를 생성하는 단계;

상기 DMA 영역 정보, 상기 DMA 버퍼 주소 및 상기 DMA 버퍼 크기 정보에 기반하여 DMA 전송 정보를 생성하는 단계; 및

상기 DMA 제어기의 레지스터 값으로써 상기 DMA 전송 정보를 포함하는 스트림을 출력하는 단계를 포함하는, 임베디드 장치 점검 방법.

청구항 6

제1항에 있어서,

상기 펌웨어를 반복 실행하는 단계는,

상기 반복 실행에 따라 상기 펌웨어에 대한 커버리지 정보를 기록하며, 상기 반복 실행 시마다 상기 펌웨어에 대한 커버리지가 증가하는 방향으로 상기 퍼징 입력 값을 변경하는, 임베디드 장치 점검 방법.

청구항 7

제1항에 있어서,

상기 펌웨어에 대한 에뮬레이션을 종료하는 단계는,

상기 반복 실행의 횟수, 상기 반복 실행에 소요된 시간 및 상기 반복 실행에 따라 기록된 상기 펌웨어에 대한 커버리지 정보의 양 중 적어도 하나를 기초로 설정된 조건이 충족되는 경우, 상기 펌웨어에 대한 에뮬레이션을 종료하는, 임베디드 장치 점검 방법.

청구항 8

제1항에 있어서,

상기 펌웨어에 대한 에뮬레이션의 결과로서 탐지된 취약점 정보를 포함하는 보고서 파일을 생성하는 단계를 더 포함하는, 임베디드 장치 점검 방법.

청구항 9

펌웨어 에뮬레이터 및 가상의 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 포함하고,

상기 펌웨어 에뮬레이터는,

점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적 분석하여 상기 펌웨어에 대응하는 하드웨어 정보를 획득하는 하드웨어 정보 추출 모듈;

상기 펌웨어를 구동하기 위한 에뮬레이션 모듈;

상기 펌웨어의 주변 장치에 대한 메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근을 처리하는 MMIO 접근 처리기;

상기 DMA 제어기의 램(Random Access Memory, RAM) 영역에 대한 데이터 송수신 작업을 상기 펌웨어 에뮬레이터에 할당하는 RAM 후킹 모듈; 및

상기 MMIO 접근 처리기 및 상기 DMA 제어기를 이용하여 상기 펌웨어를 반복 실행하기 위한 퍼징 입력 값을 생성하는 퍼저(Fuzzer)를 포함하고,

상기 DMA 제어기는,

상기 MMIO 접근의 처리 결과를 이용하여 모델링되며, DMA 작업을 제어하는, 임베디드 장치 점검 시스템.

청구항 10

제9항에 있어서,

상기 하드웨어 정보는,

RAM 영역의 주소 및 크기, 플래시 메모리 영역의 주소 및 크기, 상기 램 영역 또는 상기 플래시 메모리 영역에 대한 접근 권한 중 적어도 일부에 대한 정보를 포함하는, 임베디드 장치 점검 시스템.

청구항 11

제9항에 있어서,

상기 MMIO 접근 처리기는,

기호 실행(symbolic execution)을 통해 상기 펌웨어의 마이크로컨트롤러 유닛(Microcontroller Unit, MCU)과 관련된 주변 장치의 MMIO 접근을 처리하는, 임베디드 장치 점검 시스템.

청구항 12

제11항에 있어서,

상기 MMIO 접근 처리기는,

기호 실행을 통해 상기 펌웨어에 대한 주변 장치의 초기 MMIO 접근 처리를 위한 분기를 탈출할 수 있는 값 정보를 식별하는 기호 실행 모듈;

상기 퍼저를 통해 MMIO 영역의 주변 장치에 대해서 임의의 퍼징 입력이 필요한지 여부를 판단하는 MMIO 접근 분류 모듈; 및

상기 값 정보와 상기 초기 MMIO 접근 처리를 위한 명령어의 주소를 가리키는 프로그램 카운터의 쌍을 저장하되, 상기 퍼징 입력이 필요한 경우 상기 퍼징 입력에 필요한 비트의 개수 및 위치를 추가로 저장하는 MMIO 영역 감시 모듈을 포함하는, 임베디드 장치 점검 시스템.

청구항 13

제9항에 있어서,

상기 DMA 제어기는,

MMIO 영역에서 상기 DMA 제어기의 영역을 식별하고, 상기 DMA 제어기의 영역에 대응하는 DMA 영역 정보 및 DMA 버퍼 주소를 식별하는 DMA 제어기 영역 식별 모듈;

상기 DMA 영역 정보 및 상기 DMA 버퍼 주소에 기반하여, DMA 버퍼의 크기를 추론함으로써 DMA 버퍼 크기 정보를 생성하는 DMA 버퍼 크기 추론 모듈; 및

상기 DMA 영역 정보, 상기 DMA 버퍼 주소 및 상기 DMA 버퍼 크기 정보에 기반하여 DMA 전송 정보를 생성하고, 상기 DMA 제어기의 레지스터 값으로써 상기 DMA 전송 정보를 포함하는 스트림을 출력하는 DMA 스트림 관리 모듈을 포함하는, 임베디드 장치 점검 시스템.

청구항 14

제9항에 있어서,

상기 퍼저는,

퍼징 입력 값을 생성하고, 상기 퍼징 입력 값에 기반한 상기 펌웨어의 반복 실행에 따라 상기 펌웨어에 대한 커버리지 정보를 기록하며, 상기 반복 실행 시마다 상기 펌웨어에 대한 커버리지가 증가하는 방향으로 상기 퍼징 입력 값을 변경하는 커버리지 유도 모듈을 포함하는, 임베디드 장치 점검 시스템.

청구항 15

제9항에 있어서,

상기 퍼저는,

상기 반복 실행의 횟수, 상기 반복 실행에 소요된 시간 및 상기 반복 실행에 따라 기록된 상기 펌웨어에 대한 커버리지 정보의 양 중 적어도 하나를 기초로 설정된 조건이 충족되는 경우, 상기 펌웨어에 대한 에플레이션을 종료하는, 임베디드 장치 점검 시스템.

청구항 16

제9항에 있어서,

상기 퍼저는,

상기 펌웨어에 대한 에플레이션의 결과로서 탐지된 취약점 정보를 포함하는 보고서 파일을 생성하는 취약점 탐지 모듈을 포함하는, 임베디드 장치 점검 시스템.

발명의 설명

기술 분야

[0001] 본 개시는 임베디드 장치에서 구동되는 펌웨어의 에플레이션을 통해 임베디드 장치의 취약점을 점검하는 시스템 및 방법에 관한 것이다.

배경 기술

[0002] 전자기기의 종류가 다양해짐에 따라, 임베디드 장치의 수요는 점점 증가하고 있다. 하지만 임베디드 장치의 보안은 여전히 낮은 수준에 머물러 있는데, 임베디드 장치에 사용되는 부품들은 정형화되지 않고 해당 장치의 목적에 맞게 개별적으로 설계됨에 따라, 대부분의 임베디드 장치가 별개의 하드웨어를 사용하기 때문이다.

[0003] 이러한 수많은 임베디드 장치를 개별적으로 구매하여 분석하기도 어렵고 모든 장치에 적용되는 범용적인 보안의 적용은 더욱 어려웠기 때문에, 연구자들은 각 임베디드 장치에 대해 보안 성능을 높일 수밖에 없었다. 이 같은 하드웨어 의존성 문제를 해결하고자, 몇몇 연구에서는 임베디드 장치에서 구동되는 펌웨어가 하드웨어 동작까지 재연하도록 하는 에플레이션 기술을 통해 임베디드 장치를 분석하려는 시도를 하였다.

[0004] 다만, 이러한 시도는 몇 가지 한계를 지니는데, 첫째로 임베디드 장치의 펌웨어를 에플레이션 하더라도 개별 하드웨어를 구현하기가 쉽지 않다는 것이다. 둘째로는, 분석 과정에서 코드가 진행될 수 있는 펌웨어의 경로가 상당히 제거되어 실제 취약점이 존재할지도 모르는 코드를 점검할 수 없게 된다는 것이다. 셋째로, 임베디드 장치에서의 직접 메모리 접근(Direct Memory Access, DMA)은 주변 장치인 DMA 제어기에 의해 제어되는데, DMA 제어기의 종류가 임베디드 장치의 마이크로컨트롤러(Microcontroller, MCU)에 의존적이어서 DMA 제어를 구현하는 것도 특정 임베디드 장치에 한정된다는 것이다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) 대한민국 등록특허 제10-2304861호 (2021. 09. 23. 공고)

발명의 내용

해결하려는 과제

[0006] 본 개시는 상술한 문제점을 해결하기 위해 제안된 것으로, 범용적으로 실제 장치에 근사하면서도 높은 코드 커버리지를 갖는 에플레이션을 제공하는 에플레이터를 제안하는 것을 목적으로 한다.

[0007] 또한, 본 개시는 가상의 DMA 제어기를 모델링하여 임베디드 장치 내 펌웨어의 가용성을 제고하는 것을 또 다른 목적으로 한다.

[0008] 본 개시에 의해 이루고자 하는 기술적 과제는 상기된 바와 같은 기술적 과제들로 한정되지 않으며, 이하의 실시예들로부터 또 다른 기술적 과제들이 유추될 수 있다.

과제의 해결 수단

- [0009] 개시되는 일 실시 예에 따른 임베디드 장치 점검 방법은, 점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적 분석하여 상기 펌웨어에 대응하는 하드웨어 정보를 획득하는 단계; 상기 하드웨어 정보에 기반하여 상기 펌웨어에 대한 에뮬레이션을 시작하는 단계; 메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근 처리기를 이용하여 상기 펌웨어의 주변 장치에 대한 MMIO 접근을 처리하는 단계; 상기 MMIO 접근의 처리 결과를 이용하여 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 모델링하는 단계; DMA 작업으로서 램(Random Access Memory, RAM) 영역에 대한 쓰기 작업을 수행하는 단계; 퍼저(Fuzzer)에서 생성되는 퍼징 입력 값에 기반하여 상기 MMIO 접근 처리기 및 상기 DMA 제어기를 이용하여 상기 펌웨어를 반복 실행하는 단계; 및 사전 설정된 조건이 충족되는 경우 상기 펌웨어에 대한 에뮬레이션을 종료하는 단계를 포함한다.
- [0010] 실시 예에 따르면, 상기 하드웨어 정보는, RAM 영역의 주소 및 크기, 플래시 메모리 영역의 주소 및 크기, 상기 램 영역 또는 상기 플래시 메모리 영역에 대한 접근 권한 중 적어도 일부에 대한 정보를 포함할 수 있다.
- [0011] 실시 예에 따르면, 상기 MMIO 접근을 처리하는 단계는, 기호 실행(symbolic execution)을 통해 상기 펌웨어의 마이크로컨트롤러 유닛(Microcontroller Unit, MCU)과 관련된 주변 장치의 MMIO 접근을 처리할 수 있다.
- [0012] 관련하여 구체적으로, 상기 MMIO 접근을 처리하는 단계는, 기호 실행을 통해 상기 펌웨어에 대한 주변 장치의 초기 MMIO 접근 처리를 위한 분기를 탈출할 수 있는 값 정보를 식별하는 단계; 상기 퍼저를 통해 MMIO 영역의 주변 장치에 대해서 임의의 퍼징 입력이 필요한지 여부를 판단하는 단계; 및 상기 값 정보와 상기 초기 MMIO 접근 처리를 위한 명령어의 주소를 가리키는 프로그램 카운터의 쌍을 저장하되, 상기 퍼징 입력이 필요한 경우 상기 퍼징 입력에 필요한 비트의 개수 및 위치를 추가로 저장하는 단계를 포함할 수 있다.
- [0013] 실시 예에 따르면, 상기 DMA 제어기를 모델링하는 단계는, MMIO 영역에서 상기 DMA 제어기의 영역을 식별하는 단계; 상기 DMA 제어기의 영역에 대응하는 DMA 영역 정보 및 DMA 버퍼 주소를 식별하는 단계; 상기 DMA 영역 정보 및 상기 DMA 버퍼 주소에 기반하여, DMA 버퍼의 크기를 추론함으로써 DMA 버퍼 크기 정보를 생성하는 단계; 상기 DMA 영역 정보, 상기 DMA 버퍼 주소 및 상기 DMA 버퍼 크기 정보에 기반하여 DMA 전송 정보를 생성하는 단계; 및 상기 DMA 제어기의 레지스터 값으로써 상기 DMA 전송 정보를 포함하는 스트림을 출력하는 단계를 포함할 수 있다.
- [0014] 실시 예에 따르면, 상기 펌웨어를 반복 실행하는 단계는, 상기 반복 실행에 따라 상기 펌웨어에 대한 커버리지 정보를 기록하며, 상기 반복 실행 시마다 상기 펌웨어에 대한 커버리지가 증가하는 방향으로 상기 퍼징 입력 값을 변경할 수 있다.
- [0015] 실시 예에 따르면, 상기 펌웨어에 대한 에뮬레이션을 종료하는 단계는, 상기 반복 실행의 횟수, 상기 반복 실행에 소요된 시간 및 상기 반복 실행에 따라 기록된 상기 펌웨어에 대한 커버리지 정보의 양 중 적어도 하나를 기초로 설정된 조건이 충족되는 경우, 상기 펌웨어에 대한 에뮬레이션을 종료할 수 있다.
- [0016] 실시 예에 따르면, 상기 임베디드 장치 점검 방법은, 상기 펌웨어에 대한 에뮬레이션의 결과로서 탐지된 취약점 정보를 포함하는 보고서 파일을 생성하는 단계를 더 포함할 수 있다.
- [0017] 개시되는 일 실시 예에 따른 임베디드 장치 점검 시스템은, 펌웨어 에뮬레이터 및 가상의 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 포함하고, 상기 펌웨어 에뮬레이터는, 점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적 분석하여 상기 펌웨어에 대응하는 하드웨어 정보를 획득하는 하드웨어 정보 추출 모듈; 상기 펌웨어를 구동하기 위한 에뮬레이션 모듈; 상기 펌웨어의 주변 장치에 대한 메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근을 처리하는 MMIO 접근 처리기; 상기 DMA 제어기의 램(Random Access Memory, RAM) 영역에 대한 데이터 송수신 작업을 상기 펌웨어 에뮬레이터에 할당하는 RAM 후킹 모듈; 및 상기 MMIO 접근 처리기 및 상기 DMA 제어기를 이용하여 상기 펌웨어를 반복 실행하기 위한 퍼징 입력 값을 생성하는 퍼저(Fuzzer)를 포함하고, 상기 DMA 제어기는, 상기 MMIO 접근의 처리 결과를 이용하여 모델링되며, DMA 작업을 제어한다.
- [0018] 실시 예에 따르면, 상기 하드웨어 정보는, RAM 영역의 주소 및 크기, 플래시 메모리 영역의 주소 및 크기, 상기 램 영역 또는 상기 플래시 메모리 영역에 대한 접근 권한 중 적어도 일부에 대한 정보를 포함할 수 있다.
- [0019] 실시 예에 따르면, 상기 MMIO 접근 처리기는, 기호 실행(symbolic execution)을 통해 상기 펌웨어의 마이크로컨트롤러 유닛(Microcontroller Unit, MCU)과 관련된 주변 장치의 MMIO 접근을 처리할 수 있다.

- [0020] 실시 예에 따르면, 상기 MMIO 접근 처리기는, 기호 실행을 통해 상기 펌웨어에 대한 주변 장치의 초기 MMIO 접근 처리를 위한 분기를 탈출할 수 있는 값 정보를 식별하는 기호 실행 모듈; 상기 퍼저를 통해 MMIO 영역의 주변 장치에 대해서 임의의 퍼징 입력이 필요한지 여부를 판단하는 MMIO 접근 분류 모듈; 및 상기 값 정보와 상기 초기 MMIO 접근 처리를 위한 명령어의 주소를 가리키는 프로그램 카운터의 쌍을 저장하되, 상기 퍼징 입력이 필요한 경우 상기 퍼징 입력에 필요한 비트의 개수 및 위치를 추가로 저장하는 MMIO 영역 감시 모듈을 포함할 수 있다.
- [0021] 실시 예에 따르면, 상기 DMA 제어기는, MMIO 영역에서 상기 DMA 제어기의 영역을 식별하고, 상기 DMA 제어기의 영역에 대응하는 DMA 영역 정보 및 DMA 버퍼 주소를 식별하는 DMA 제어기 영역 식별 모듈; 상기 DMA 영역 정보 및 상기 DMA 버퍼 주소에 기반하여, DMA 버퍼의 크기를 추론함으로써 DMA 버퍼 크기 정보를 생성하는 DMA 버퍼 크기 추론 모듈; 및 상기 DMA 영역 정보, 상기 DMA 버퍼 주소 및 상기 DMA 버퍼 크기 정보에 기반하여 DMA 전송 정보를 생성하고, 상기 DMA 제어기의 레지스터 값으로써 상기 DMA 전송 정보를 포함하는 스트림을 출력하는 DMA 스트림 관리 모듈을 포함할 수 있다.
- [0022] 실시 예에 따르면, 상기 퍼저는, 퍼징 입력 값을 생성하고, 상기 퍼징 입력 값에 기반한 상기 펌웨어의 반복 실행에 따라 상기 펌웨어에 대한 커버리지 정보를 기록하며, 상기 반복 실행 시마다 상기 펌웨어에 대한 커버리지가 증가하는 방향으로 상기 퍼징 입력 값을 변경하는 커버리지 유도 모듈을 포함할 수 있다.
- [0023] 실시 예에 따르면, 상기 퍼저는, 상기 반복 실행의 횟수, 상기 반복 실행에 소요된 시간 및 상기 반복 실행에 따라 기록된 상기 펌웨어에 대한 커버리지 정보의 양 중 적어도 하나를 기초로 설정된 조건이 충족되는 경우, 상기 펌웨어에 대한 에뮬레이션을 종료할 수 있다.
- [0024] 실시 예에 따르면, 상기 퍼저는, 상기 펌웨어에 대한 에뮬레이션의 결과로서 탐지된 취약점 정보를 포함하는 보고서 파일을 생성하는 취약점 탐지 모듈을 포함할 수 있다.

발명의 효과

- [0025] 본 개시에 따른 임베디드 장치 점검 시스템 및 방법을 통해, 가상의 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 모델링하여 임베디드 하드웨어 장치를 근사함으로써 범용적으로 사용할 수 있는 자동화 에뮬레이션을 구현할 수 있다.
- [0026] 이에 더하여, 퍼징 입력 값을 에뮬레이션에 활용하여 경로 제거 현상을 줄임으로써 종래의 에뮬레이션 방식 대비 높은 커버리지를 달성할 수 있다. 관련하여, 기호 실행(symbolic execution) 기법을 활용하여 펌웨어의 가능한 분기를 모두 탐색하기까지 소요되는 시간을 줄일 수 있다.
- [0027] 발명의 효과는 이상에서 언급한 효과로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 청구범위의 기재로부터 당해 기술 분야의 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

- [0028] 도 1은 일 실시 예에 따른 임베디드 장치 점검 시스템의 개략적인 구성도이다.
- 도 2는 직접 메모리 접근(Direct Memory Access, DMA) 제어기 영역 식별 및 전송 정보의 예시도이다.
- 도 3은 DMA 제어기의 전송 정보 및 스트림 구성을 나타낸 예시도이다.
- 도 4a 및 4b는 가상의 DMA 제어기 유무에 따른 효과를 설명하기 위한 도면이다.
- 도 5는 일 실시 예에 따른 임베디드 장치 점검 방법을 설명하기 위한 흐름도이다.
- 도 6은 일 실시 예에 따라 단계 S530을 구체적으로 설명하는 흐름도이다.
- 도 7은 일 실시 예에 따라 단계 S540을 구체적으로 설명하는 흐름도이다.
- 도 8은 추가적인 실시 예에 따른 임베디드 장치 점검 방법을 설명하기 위한 흐름도이다.
- 도 9는 일 실시 예에 따라 임베디드 장치 점검 시스템으로 기능하는 전자 장치를 설명하기 위한 블록도이다.

발명을 실시하기 위한 구체적인 내용

- [0029] 실시 예들에서 사용되는 용어는 본 개시에서의 기능을 고려하면서 가능한 현재 널리 사용되는 일반적인 용어들

을 선택하였으나, 이는 당 분야에 종사하는 기술자의 의도 또는 판례, 새로운 기술의 출현 등에 따라 달라질 수 있다. 또한, 특정한 경우는 출원인이 임의로 선정한 용어도 있으며, 이 경우 해당되는 설명 부분에서 상세히 그 의미를 기재할 것이다. 따라서 본 개시에서 사용되는 용어는 단순한 용어의 명칭이 아닌, 그 용어가 가지는 의미와 본 개시의 전반에 걸친 내용을 토대로 정의되어야 한다.

[0030] 명세서 전체에서 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있음을 의미한다. 또한, 명세서에 기재된 "...부", "...모듈" 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하고, 이는 하드웨어 또는 소프트웨어로 구현되거나 하드웨어와 소프트웨어의 결합으로 구현될 수 있으며, 도시된 예와 달리 구체적인 동작에 있어 명확히 구분되지 않을 수 있다.

[0031] 명세서 전체에서 기재된 "a, b, 및 c 중 적어도 하나"의 표현은, 'a 단독', 'b 단독', 'c 단독', 'a 및 b', 'a 및 c', 'b 및 c', 또는 'a, b, 및 c 모두'를 포괄할 수 있다.

[0032] 이하에서 언급되는 "단말" 또는 "사용자 단말"은 네트워크를 통해 서버나 타 단말에 접속할 수 있는 컴퓨터나 휴대용 단말로 구현될 수 있다. 여기서, 컴퓨터는 예를 들어, 웹 브라우저(WEB Browser)가 탑재된 노트북, 데스크톱(desktop), 랩톱(laptop) 등을 포함하고, 휴대용 단말은 예를 들어, 휴대성과 이동성이 보장되는 무선 통신 장치로서, IMT(International Mobile Telecommunication), CDMA(Code Division Multiple Access), W-CDMA(W-Code Division Multiple Access), LTE(Long Term Evolution) 등의 통신 기반 단말, 스마트폰, 태블릿 PC 등과 같은 모든 종류의 핸드헬드(Handheld) 기반의 무선 통신 장치를 포함할 수 있다.

[0033] 이하의 설명에 있어서, 신호, 메시지 또는 정보의 "전송", "통신", "송신", "수신" 기타 이와 유사한 의미의 용어는 일 구성요소에서 다른 구성요소로 정보, 메시지 또는 정보가 직접 전달되는 것뿐만이 아니라 다른 구성요소를 거쳐 전달되는 것도 포함한다.

[0034] 특히 신호, 메시지 또는 정보를 일 구성요소로 "전송" 또는 "송신"한다는 것은 그 신호, 메시지 또는 정보의 최종 목적지를 지시하는 것이고 직접적인 목적지를 의미하는 것이 아니다. 이는 신호, 메시지 또는 정보의 "수신"에 있어서도 동일하다. 또한 본 개시에 있어서, 2 이상의 데이터 또는 정보가 "관련"된다는 것은 하나의 데이터(또는 정보)를 획득하면, 그에 기초하여 다른 데이터(또는 정보)의 적어도 일부를 획득할 수 있음을 의미한다.

[0035] 또한, 제1, 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로 사용될 수 있다.

[0036] 예를 들어, 본 개시의 권리 범위를 벗어나지 않으면서 제1 구성 요소는 제2 구성 요소로 명명될 수 있고, 유사하게 제2 구성 요소도 제1 구성 요소로 명명될 수 있다.

[0037] 아래에서는 첨부한 도면을 참고하여 본 개시의 실시 예에 대하여 본 개시가 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 그러나 본 개시는 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시 예에 한정되지 않는다.

[0038] 이하에서는 도면을 참조하여 본 개시의 실시 예들을 상세히 설명한다.

[0039] 실시 예를 설명함에 있어서 본 발명이 속하는 기술 분야에 익히 알려져 있고 본 발명과 직접적으로 관련이 없는 기술 내용에 대해서는 설명을 생략한다. 이는 불필요한 설명을 생략함으로써 본 발명의 요지를 흐리지 않고 더욱 명확히 전달하기 위함이다.

[0040] 마찬가지로 이유로 첨부 도면에 있어서 일부 구성요소는 과장되거나 생략되거나 개략적으로 도시되었다. 또한, 각 구성요소의 크기는 실제 크기를 전적으로 반영하는 것이 아니다. 각 도면에서 동일한 또는 대응하는 구성요소에는 동일한 참조 번호를 부여하였다.

[0041] 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술되어 있는 실시 예들을 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시 예들에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있으며, 단지 본 실시 예들은 본 발명의 개시가 완전하도록 하고, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이며, 본 발명은 청구항의 범주에 의해 정의될 뿐이다. 명세서 전체에 걸쳐 동일 참조 부호는 동일 구성 요소를

지칭한다.

- [0042] 처리 흐름도 도면들의 각 블록과 흐름도 도면들의 조합들은 컴퓨터 프로그램 인스트럭션들에 의해 수행될 수 있음을 이해할 수 있을 것이다. 이들 컴퓨터 프로그램 인스트럭션들은 범용 컴퓨터, 특수용 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비의 프로세서에 탑재될 수 있으므로, 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비의 프로세서를 통해 수행되는 그 인스트럭션들이 흐름도 블록(들)에서 설명된 기능들을 수행하는 수단을 생성하게 된다. 이들 컴퓨터 프로그램 인스트럭션들은 특정 방식으로 기능을 구현하기 위해 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비를 지향할 수 있는 컴퓨터 이용 가능 또는 컴퓨터 판독 가능 메모리에 저장되는 것도 가능하므로, 그 컴퓨터 이용 가능 또는 컴퓨터 판독 가능 메모리에 저장된 인스트럭션들은 흐름도 블록(들)에서 설명된 기능을 수행하는 인스트럭션 수단을 내포하는 제조 품목을 생산하는 것도 가능하다. 컴퓨터 프로그램 인스트럭션들은 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비 상에 탑재되는 것도 가능하므로, 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비 상에서 일련의 동작 단계들이 수행되어 컴퓨터로 실행되는 프로세스를 생성해서 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비를 수행하는 인스트럭션들은 흐름도 블록(들)에서 설명된 기능들을 실행하기 위한 단계들을 제공하는 것도 가능하다.
- [0043] 또한, 각 블록은 특정된 논리적 기능(들)을 실행하기 위한 하나 이상의 실행 가능한 인스트럭션들을 포함하는 모듈, 세그먼트 또는 코드의 일부를 나타낼 수 있다. 또, 몇 가지 대체 실행 예들에서는 블록들에서 언급된 기능들이 순서를 벗어나서 발생하는 것도 가능함을 주목해야 한다. 예컨대, 잇달아 도시되어 있는 두 개의 블록들은 사실 실질적으로 동시에 수행되는 것도 가능하고 또는 그 블록들이 때때로 해당하는 기능에 따라 역순으로 수행되는 것도 가능하다.
- [0044] 도 1은 일 실시 예에 따른 임베디드 장치 점검 시스템의 개략적인 구성도이다.
- [0045] 도 1에 도시된 대로, 일 실시 예에 따른 임베디드 장치 점검 시스템(10)은 펌웨어 에뮬레이터(100) 및 가상의 직접 메모리 접근(Direct Memory Access, DMA) 제어기(200)를 포함한다.
- [0046] 일 실시 예에 따르면, 펌웨어 에뮬레이터(100)는 하드웨어 정보 추출 모듈, 에뮬레이션 모듈(110), 메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근 처리기(120), 퍼저(Fuzzer)(130) 및 램(Random Access Memory, RAM) 후킹 모듈(140)을 포함할 수 있다.
- [0047] 구체적으로, 하드웨어 정보 추출 모듈은 점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적 분석하여 펌웨어에 대응하는 하드웨어 정보를 획득할 수 있다. 실시 예에 따르면, 하드웨어 정보 추출 모듈에 의해 획득되는 하드웨어 정보에는 RAM 영역의 주소 및 크기, 플래시 메모리 영역의 주소 및 크기, RAM 영역 또는 플래시 메모리 영역에 대한 접근 권한 중 적어도 일부에 대한 정보가 포함될 수 있다.
- [0048] 또한, 에뮬레이션 모듈(110)은 펌웨어를 구동할 수 있고, MMIO 접근 처리기(120)는 펌웨어의 주변 장치에 대한 MMIO 접근을 처리할 수 있다.
- [0049] 이 중, MMIO 접근 처리기(120)는 실시 예에 따라 스냅샷 모듈(121), 기호 실행 모듈(123), MMIO 접근 분류 모듈(125) 및 MMIO 영역 감시 모듈(127)을 포함할 수 있다. 실시 예에 따르면, 기호 실행 모듈(123)은 기호 실행(symbolic execution)을 통해 상기 펌웨어에 대한 주변 장치의 초기 MMIO 접근 처리를 위한 분기를 탈출할 수 있는 값 정보를 식별할 수 있고, MMIO 접근 분류 모듈(125)은 퍼저(130)를 통해 펌웨어의 마이크로컨트롤러 유닛(Microcontroller Unit, MCU)과 관련된 주변 장치에 대해서 임의의 퍼징 입력이 필요한지 여부를 판단할 수 있으며, MMIO 영역 감시 모듈(127)은 기호 실행 모듈(123)에 의해 식별된 값 정보와 초기 MMIO 접근 처리를 위한 명령어의 주소를 가리키는 프로그램 카운터(Program Counter)의 쌍을 저장하되, 퍼저(130)에 의한 퍼징 입력이 필요한 경우 퍼징 입력에 필요한 비트의 개수 및 위치를 추가로 저장할 수 있다.
- [0050] 또한, 퍼저(130)는 MMIO 접근 처리기(120) 및 DMA 제어기(200)를 이용하여 펌웨어를 반복 실행하기 위한 퍼징 입력 값을 생성할 수 있으며, RAM 후킹 모듈(140)은 DMA 제어기(200)의 RAM 영역에 대한 데이터 송수신 작업을 펌웨어 에뮬레이터(100)에 할당할 수 있다.
- [0051] 이 중, 퍼저(130)는 실시 예에 따라 커버리지 유도 모듈(131) 및 취약점 탐지 모듈(133)을 포함할 수 있다. 실시 예에 따르면, 커버리지 유도 모듈(131)은 퍼징 입력 값을 생성하고, 퍼징 입력 값에 기반한 펌웨어의 반복 실행에 따라 펌웨어에 대한 커버리지 정보를 기록하며, 펌웨어의 반복 실행 시마다 펌웨어에 대한 커버리지가 증가하는 방향으로 퍼징 입력 값을 변경할 수 있다. 또한, 취약점 탐지 모듈(133)은 펌웨어에 대한 에뮬레이션의 결과로서 탐지된 취약점 정보를 포함하는 보고서 파일을 생성할 수 있다.

- [0052] 한편, 일 실시 예에 따르면, 가상의 DMA 제어기(200)는 MMIO 접근의 처리 결과를 이용하여 모델링되는 것으로, DMA 작업을 제어할 수 있다. 실시 예에 따라, DMA 제어기(200)는 DMA 제어기 영역 식별 모듈(210), DMA 버퍼 크기 추론 모듈(220) 및 DMA 스트림 관리 모듈(230)을 포함할 수 있다.
- [0053] 구체적으로, DMA 제어기 영역 식별 모듈(210)은 MMIO 영역에서 DMA 제어기(200)의 영역을 식별하고, DMA 제어기(200)의 영역에 대응하는 DMA 영역 정보 및 DMA 버퍼 주소를 식별할 수 있다.
- [0054] 또한, DMA 버퍼 크기 추론 모듈(220)은 DMA 영역 정보 및 DMA 버퍼 주소에 기반하여, DMA 버퍼의 크기를 추론함으로써 DMA 버퍼 크기 정보를 생성할 수 있다.
- [0055] 또한, DMA 스트림 관리 모듈(230)은 DMA 영역 정보, DMA 버퍼 주소 및 DMA 버퍼 크기 정보에 기반하여 DMA 전송 정보를 생성하고, DMA 제어기(200)의 레지스터 값으로써 DMA 전송 정보를 포함하는 스트림(stream)을 출력할 수 있다.
- [0056] 상기의 DMA 제어기(200)와 관련하여 추가적으로, 도 2 및 3을 참조하여 후술한다.
- [0057] 도 2는 DMA 제어기 영역 식별 및 전송 정보의 예시도로서, 보다 자세하게는 MMIO 영역을 모니터링한 결과와 DMA 제어기 영역 식별과 전송에 관한 정보를 도시한 것이다.
- [0058] 도 2의 마지막 열은 레지스터의 주소와 해당 주소에 어떤 값이 쓰였는지 나타낸다. 도 2에서 박스 표시된 부분은 예시적으로, 0x40026094 주변 장치 레지스터에 0x20004de8의 값이 기록되었음을(write, w) 의미한다. RAM 영역을 제어할 수 있는 주변 장치인 DMA 제어기(200)의 영역을 식별하기 위해, RAM 영역에 대한 쓰기 명령을 식별한다. 이에, 0x40026XXX의 영역을 DMA 제어기(200)의 영역으로 정의하고, 이때 0x40004404가 포함된 주변 장치를 데이터복사의 주체로 설정한다. 결과적으로, DMA 전송 정보로 확인되는 출발지는 0x40004XXX 영역의 주변 장치이며, 목적지인 DMA 버퍼는 0x20004de8로 식별될 수 있다.
- [0059] 도 3은 DMA 제어기의 전송 정보 및 스트림 구성을 나타낸 예시도이다. DMA 제어기(200)의 MMIO 영역에 매핑된 주소가 0x4002 6000이라고 할 때, 일반적인 제어 또는 상태 레지스터가 먼저 매핑되어 있고, 그 이후 DMA 스트림 정보가 매핑되어 있다. DMA 스트림은 각각 DMA 작업에 대한 전송 정보를 포함하며, 도 3에 예시적으로 도시된 바에 따르면 각 스트림마다 20바이트의 간격을 두고 스트림이 나누어질 수 있다. DMA 스트림에 포함되는 DMA 전송 정보는 데이터의 복사를 위한 출발지(Source), 목적지(Destination), 버퍼의 크기(Size), 반복 횟수(Count) 등을 포함할 수 있다. 또한, 하나의 스트림은 주어진 시간에 하나의 DMA 전송만을 지원할 수 있다.
- [0060] 도 4a 및 4b는 가상의 DMA 제어기 유무에 따른 효과를 설명하기 위한 도면이다. 도 4a는 가상의 DMA 제어기가 없는 경우를, 도 4b는 가상의 DMA 제어기가 존재하는 경우를 비교하여 코드 커버리지의 변화를 시각적으로 도시한 예시도이다.
- [0061] 도 4a에 도시된 바와 같이, 가상 DMA 제어기(200)가 없는 상태에서 펌웨어의 코드 커버리지는 특정 분기에서 진행될 수 없어, 경로 제거 현상이 발생한다. 즉, 일반 주변장치 레지스터로의 코드 진행은 가능할지라도, DMA 레지스터 주소로의 코드 진행 시에는 무한 루프에 빠지거나 과도한 시간이 소모되며, 특히 DMA 특성상 알 수 없는 펌웨어 코드 영역에 대해서는 경로 제거 현상이 발생하여 취약점을 발견할 수 있는 가능성이 줄어든다.
- [0062] 반면 도 4b에 도시된 바와 같이, 가상 DMA 제어기(200)가 있는 상태에서는 코드의 탐색이 원활하게 진행될 수 있다. DMA 제어기(200)가 구현됨에 따라 제거될 수도 있던 경로에서 새로운 주변 장치 레지스터를 발견할 수 있고, 이에 따라 임베디드 장치 점검 시스템(10)은 이를 다시 모델링하는 작업을 반복할 수 있다. 더욱이, DMA 제어기(200)가 존재할 때 탐색할 수 있는 분기에서 DMA 관련 하드웨어를 사용하는 펌웨어의 코드가 존재할 가능성이 높으므로, 이를 통해 에뮬레이션 시 높은 커버리지를 갖는 상승 효과를 얻을 수 있다.
- [0063] 이하 도 5 내지 8을 참조하여 후술할 임베디드 장치 점검 방법은, 상술한 임베디드 장치 점검 시스템(10)에 의해 수행될 수 있고, 구조적으로 임베디드 장치 점검 시스템(10)의 기능을 수행하는, 도 9에 도시된 전자 장치(1)에 의해 수행될 수 있다.
- [0064] 또한, 이하에 도시된 흐름도에서는 임베디드 장치 점검 방법을 복수 개의 단계로 나누어 기재하였으나, 적어도 일부의 단계들은 순서를 바꾸어 수행되거나, 다른 단계와 결합되어 함께 수행되거나, 생략되거나, 세부 단계들로 나뉘어 수행되거나, 또는 도시되지 않은 하나 이상의 단계가 부가되어 수행될 수 있다.
- [0065] 도 5는 일 실시 예에 따른 임베디드 장치 점검 방법을 설명하기 위한 흐름도이다.
- [0066] 단계 S510에서, 임베디드 장치 점검 시스템(10)은 점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적

분석하여 펌웨어에 대응하는 하드웨어 정보를 획득한다.

- [0067] 단계 S520에서, 임베디드 장치 점검 시스템(10)은 하드웨어 정보에 기반하여 펌웨어에 대한 애플리케이션을 시작한다.
- [0068] 단계 S530에서, 임베디드 장치 점검 시스템(10)은 메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근 처리기를 이용하여 펌웨어의 주변 장치에 대한 MMIO 접근을 처리한다. 구체적으로, 임베디드 장치 점검 시스템(10)은 기호 실행(symbolic execution)을 통해 펌웨어의 마이크로컨트롤러 유닛(Microcontroller Unit, MCU)과 관련된 주변 장치의 MMIO 접근을 처리할 수 있다.
- [0069] 단계 S540에서, 임베디드 장치 점검 시스템(10)은 MMIO 접근의 처리 결과를 이용하여 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 모델링한다.
- [0070] 단계 S550에서, 임베디드 장치 점검 시스템(10)은 DMA 작업으로서 램(Random Access Memory, RAM) 영역에 대한 쓰기 작업을 수행한다.
- [0071] 단계 S560에서, 임베디드 장치 점검 시스템(10)은 퍼저(Fuzzer)에서 생성되는 퍼징 입력 값에 기반하여 MMIO 접근 처리기 및 DMA 제어기를 이용하여 펌웨어를 반복 실행한다. 실시 예에 따르면, 임베디드 장치 점검 시스템(10)은 펌웨어의 반복 실행에 따라 펌웨어에 대한 커버리지 정보를 기록하며, 반복 실행 시마다 펌웨어에 대한 커버리지가 증가하는 방향으로 퍼징 입력 값을 변경할 수 있다.
- [0072] 단계 S570에서, 임베디드 장치 점검 시스템(10)은 사전 설정된 조건이 충족되는 경우 펌웨어에 대한 애플리케이션을 종료한다. 실시 예에 따르면, 임베디드 장치 점검 시스템(10)은 펌웨어의 반복 실행의 횟수, 펌웨어의 반복 실행에 소요된 시간 및 펌웨어의 반복 실행에 따라 기록된 펌웨어에 대한 커버리지 정보의 양 중 적어도 하나를 기초로 설정된 조건이 충족되는 경우, 펌웨어에 대한 애플리케이션을 종료할 수 있다.
- [0073] 도 6은 일 실시 예에 따라 단계 S530을 구체적으로 설명하는 흐름도이다.
- [0074] 단계 S610에서, 임베디드 장치 점검 시스템(10)은 기호 실행을 통해 펌웨어에 대한 주변 장치의 초기 MMIO 접근 처리를 위한 분기를 탈출할 수 있는 값 정보를 식별한다.
- [0075] 단계 S620에서, 임베디드 장치 점검 시스템(10)은 퍼저를 통해 MMIO 영역의 주변 장치에 대해서 임의의 퍼징 입력이 필요한지 여부를 판단한다.
- [0076] 단계 S630에서, 임베디드 장치 점검 시스템(10)은 단계 S610을 통해 식별한 값 정보와 초기 MMIO 접근 처리를 위한 명령어의 주소를 가리키는 프로그램 카운터의 쌍을 저장하되, 단계 S620의 판단 결과 퍼징 입력이 필요한 경우 퍼징 입력에 필요한 비트의 개수 및 위치를 추가로 저장한다.
- [0077] 도 7은 일 실시 예에 따라 단계 S540을 구체적으로 설명하는 흐름도이다.
- [0078] 단계 S710에서, 임베디드 장치 점검 시스템(10)은 MMIO 영역에서 DMA 제어기의 영역을 식별한다.
- [0079] 단계 S720에서, 임베디드 장치 점검 시스템(10)은 DMA 제어기의 영역에 대응하는 DMA 영역 정보 및 DMA 버퍼 주소를 식별한다.
- [0080] 단계 S730에서, 임베디드 장치 점검 시스템(10)은 DMA 영역 정보 및 DMA 버퍼 주소에 기반하여, DMA 버퍼의 크기를 추론함으로써 DMA 버퍼 크기 정보를 생성한다.
- [0081] 단계 S740에서, 임베디드 장치 점검 시스템(10)은 DMA 영역 정보, DMA 버퍼 주소 및 DMA 버퍼 크기 정보에 기반하여 DMA 전송 정보를 생성한다.
- [0082] 단계 S750에서, 임베디드 장치 점검 시스템(10)은 DMA 제어기의 레지스터 값으로써 DMA 전송 정보를 포함하는 스트림(stream)을 출력한다.
- [0083] 도 8은 추가적인 실시 예에 따른 임베디드 장치 점검 방법을 설명하기 위한 흐름도이다.
- [0084] 단계 S810에서, 임베디드 장치 점검 시스템(10)은 점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적 분석하여 펌웨어에 대응하는 하드웨어 정보를 획득한다.
- [0085] 단계 S820에서, 임베디드 장치 점검 시스템(10)은 하드웨어 정보에 기반하여 펌웨어에 대한 애플리케이션을 시작한다.

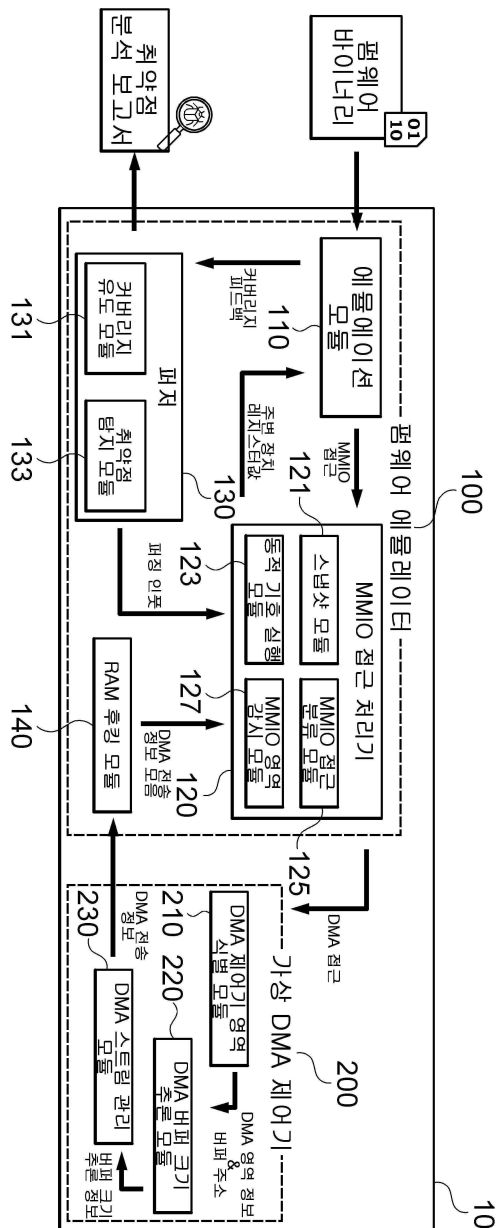
- [0086] 단계 S830에서, 임베디드 장치 점검 시스템(10)은 메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근 처리기를 이용하여 펌웨어의 주변 장치에 대한 MMIO 접근을 처리한다.
- [0087] 단계 S840에서, 임베디드 장치 점검 시스템(10)은 MMIO 접근의 처리 결과를 이용하여 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 모델링한다.
- [0088] 단계 S850에서, 임베디드 장치 점검 시스템(10)은 DMA 작업으로서 램(Random Access Memory, RAM) 영역에 대한 쓰기 작업을 수행한다.
- [0089] 단계 S860에서, 임베디드 장치 점검 시스템(10)은 퍼저(Fuzzer)에서 생성되는 퍼징 입력 값에 기반하여 MMIO 접근 처리기 및 DMA 제어기를 이용하여 펌웨어를 반복 실행한다.
- [0090] 단계 S870에서, 임베디드 장치 점검 시스템(10)은 사전 설정된 조건이 충족되는 경우 펌웨어에 대한 에뮬레이션을 종료한다.
- [0091] 단계 S880에서, 임베디드 장치 점검 시스템(10)은 펌웨어에 대한 에뮬레이션의 결과로서 탐지된 취약점 정보를 포함하는 보고서 파일을 생성한다.
- [0092] 도 9는 일 실시 예에 따라 임베디드 장치 점검 시스템으로 기능하는 전자 장치를 설명하기 위한 블록도이다.
- [0093] 전자 장치(1)는 일 실시 예에 따라, 프로세서(11) 및 메모리(13)를 포함할 수 있다. 일 실시 예에서, 전자 장치(1)는 트랜시버나 통신 인터페이스를 통하여 외부 서버 또는 데이터베이스와 연결되고, 데이터를 교환할 수 있다.
- [0094] 프로세서(11)는 도 1 내지 도 8을 통하여 전술한 적어도 하나의 방법을 수행할 수 있다. 메모리(13)는 도 1 내지 도 8을 통하여 전술한 적어도 하나의 방법을 수행하기 위한 정보를 저장할 수 있다. 메모리(13)는 휘발성 메모리 또는 비휘발성 메모리일 수 있다.
- [0095] 프로세서(11)는 프로그램을 실행하고, 정보를 제공하기 위해 전자 장치(1)를 제어할 수 있다. 프로세서(11)에 의하여 실행되는 프로그램의 코드는 메모리(13)에 저장될 수 있다.
- [0096] 프로세서(11)는 메모리(13)와 연결되어, 점검 대상인 펌웨어에 대응하는 펌웨어 바이너리 파일을 정적 분석하여 펌웨어에 대응하는 하드웨어 정보를 획득하고, 하드웨어 정보에 기반하여 펌웨어에 대한 에뮬레이션을 시작하고, 메모리 맵 입출력(Memory-Mapped I/O, MMIO) 접근 처리기를 이용하여 펌웨어의 주변 장치에 대한 MMIO 접근을 처리하고, MMIO 접근의 처리 결과를 이용하여 직접 메모리 접근(Direct Memory Access, DMA) 제어기를 모델링하고, DMA 작업으로서 램(Random Access Memory, RAM) 영역에 대한 쓰기 작업을 수행하고, 퍼저(Fuzzer)에서 생성되는 퍼징 입력 값에 기반하여 MMIO 접근 처리기 및 DMA 제어기를 이용하여 펌웨어를 반복 실행하며, 사전 설정된 조건이 충족되는 경우 펌웨어에 대한 에뮬레이션을 종료할 수 있다.
- [0097] 도 9에 도시된 전자 장치(1)는 본 실시 예와 관련된 구성요소들만 도시되어 있다. 따라서, 도 9에 도시된 구성요소들 외에 다른 범용적인 구성요소들이 더 포함될 수 있음을 본 실시 예와 관련된 기술분야에서 통상의 지식을 가진 자라면 이해할 수 있다.
- [0098] 전술한 실시 예들에 따른 장치는 프로세서, 프로그램 데이터를 저장하고 실행하는 메모리, 디스크 드라이브와 같은 영구 저장부(permanent storage), 외부 장치와 통신하는 통신 포트, 터치 패널, 키(key), 버튼 등과 같은 사용자 인터페이스 장치 등을 포함할 수 있다. 소프트웨어 모듈 또는 알고리즘으로 구현되는 방법들은 상기 프로세서상에서 실행 가능한 컴퓨터가 읽을 수 있는 코드들 또는 프로그램 명령들로서 컴퓨터가 읽을 수 있는 기록 매체 상에 저장될 수 있다. 여기서 컴퓨터가 읽을 수 있는 기록 매체로 마그네틱 저장 매체(예컨대, ROM(read-only memory), RAM(random-access memory), 플로피 디스크, 하드 디스크 등) 및 광학적 판독 매체(예컨대, 시디롬(CD-ROM), 디브이디(DVD: Digital Versatile Disc)) 등이 있다. 컴퓨터가 읽을 수 있는 기록 매체는 네트워크로 연결된 컴퓨터 시스템들에 분산되어, 분산 방식으로 컴퓨터가 판독 가능한 코드가 저장되고 실행될 수 있다. 매체는 컴퓨터에 의해 판독 가능하며, 메모리에 저장되고, 프로세서에서 실행될 수 있다.
- [0099] 본 실시 예는 기능적인 블록 구성들 및 다양한 처리 단계들로 나타내어질 수 있다. 이러한 기능 블록들은 특정 기능들을 실행하는 다양한 개수의 하드웨어 또는/및 소프트웨어 구성들로 구현될 수 있다. 예를 들어, 실시 예는 하나 이상의 마이크로프로세서들의 제어 또는 다른 제어 장치들에 의해서 다양한 기능들을 실행할 수 있는, 메모리, 프로세싱, 로직(logic), 룩업 테이블(look-up table) 등과 같은 직접 회로 구성들을 채용할 수 있다. 구성 요소들이 소프트웨어 프로그래밍 또는 소프트웨어 요소들로 실행될 수 있는 것과 유사하게, 본 실시 예는

데이터 구조, 프로세스들, 루틴들 또는 다른 프로그래밍 구성들의 조합으로 구현되는 다양한 알고리즘을 포함하여, C, C++, 자바(Java), 어셈블러(assembler) 등과 같은 프로그래밍 또는 스크립팅 언어로 구현될 수 있다. 기능적인 측면들은 하나 이상의 프로세서들에서 실행되는 알고리즘으로 구현될 수 있다. 또한, 본 실시 예는 전자적인 환경 설정, 신호 처리, 메시지 처리, 및/또는 데이터 처리 등을 위하여 종래 기술을 채용할 수 있다. “매커니즘”, “요소”, “수단”, “구성”과 같은 용어는 넓게 사용될 수 있으며, 기계적이고 물리적인 구성들로서 한정되는 것은 아니다. 상기 용어는 프로세서 등과 연계하여 소프트웨어의 일련의 처리들(routines)의 의미를 포함할 수 있다.

[0100] 전술한 실시 예들은 일 예시일 뿐 후술하는 청구항들의 범위 내에서 다른 실시 예들이 구현될 수 있다.

도면

도면1

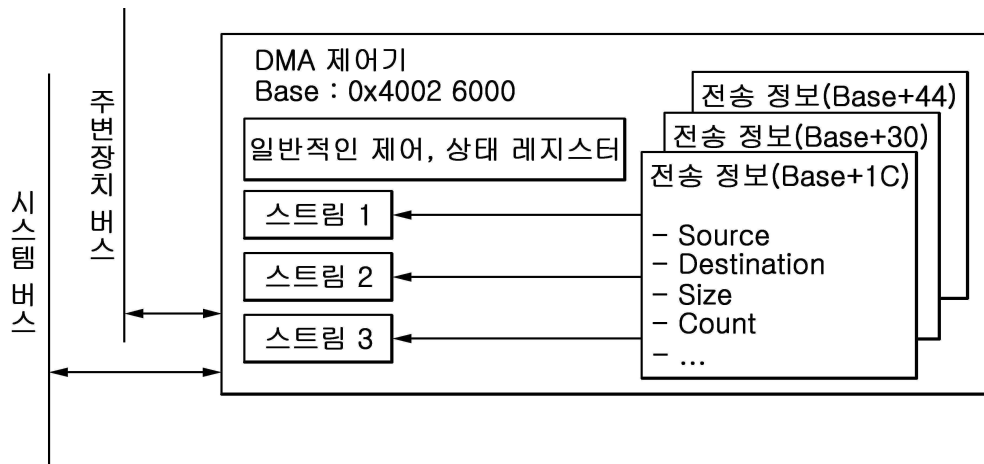


도면2

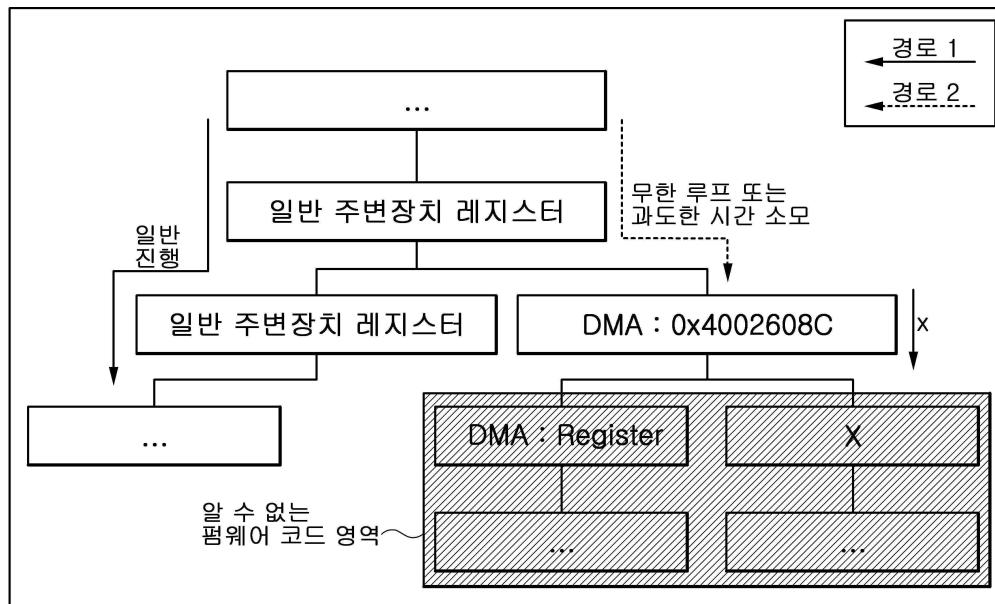
```

e7aa: 8000250 80017dd r 4 8 0 0x4002609c : 0
e7b6: 8000270 80017dd w 4 0 0 0x4002609c : 0
e7bb: 8000278 80017dd w 4 0 0 0x4002608c : 8
e7bf: 8000280 80017dd w 4 0 0 0x40026090 : 40004404
e7c3: 8000288 80017dd w 4 0 0 0x40026094 : 20004de8
e7ce: 8000318 80017e5 r 4 8 0 0x40026088 : 8020500
e7d0: 8000320 80017e5 w 4 0 0 0x40026088 : 8020501
    
```

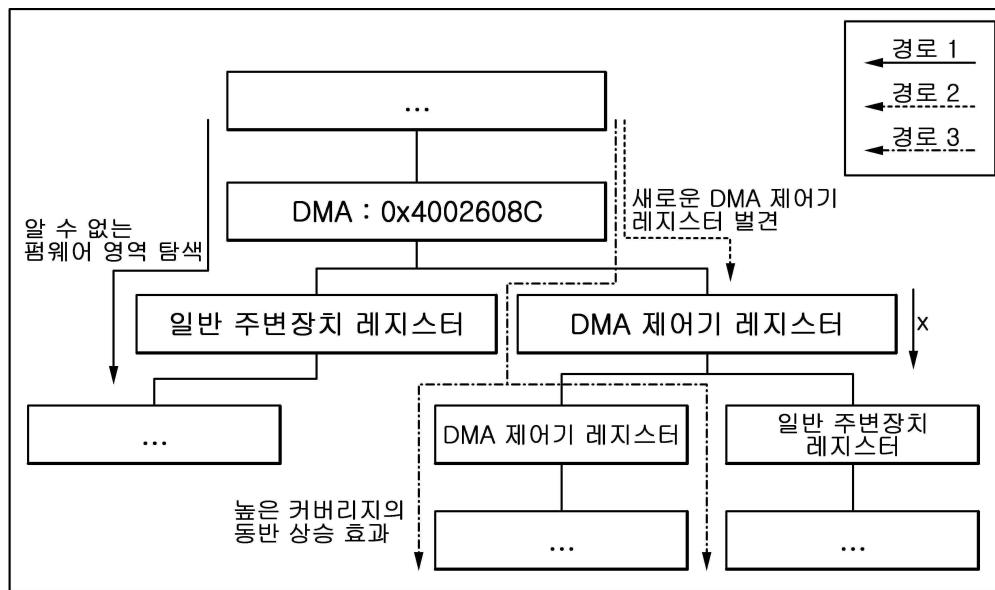
도면3



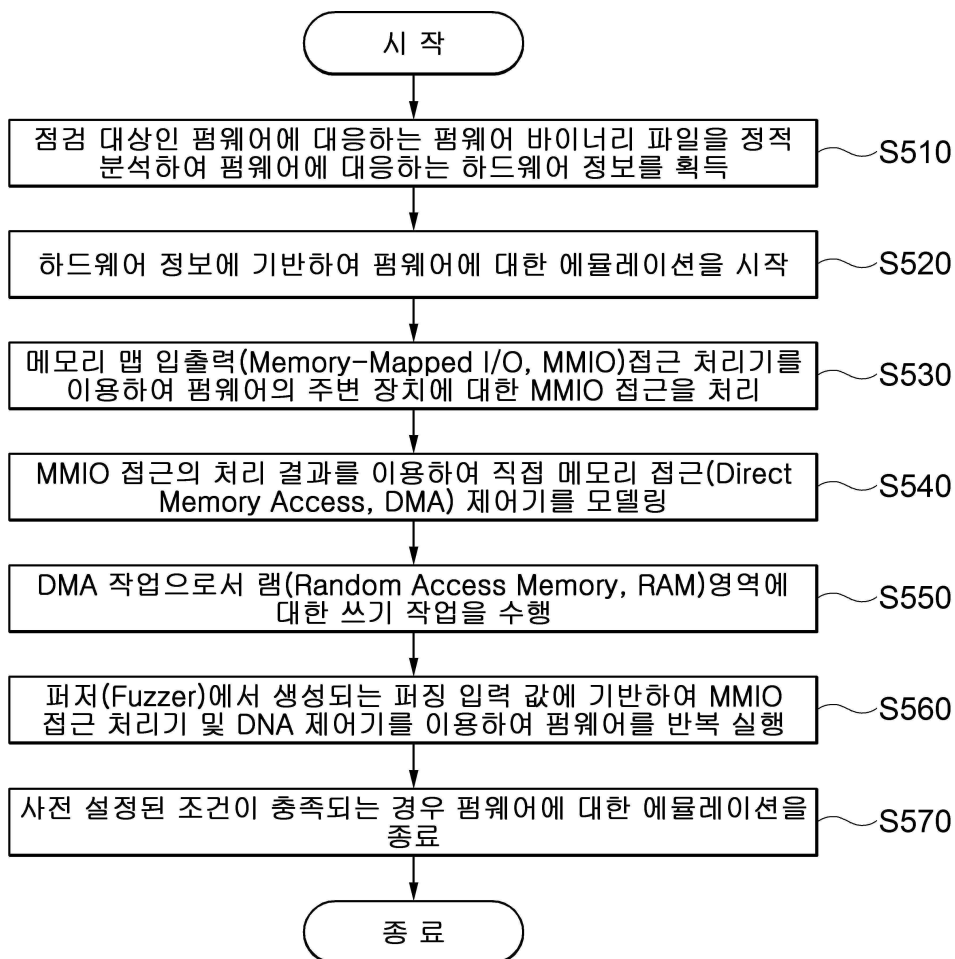
도면4a



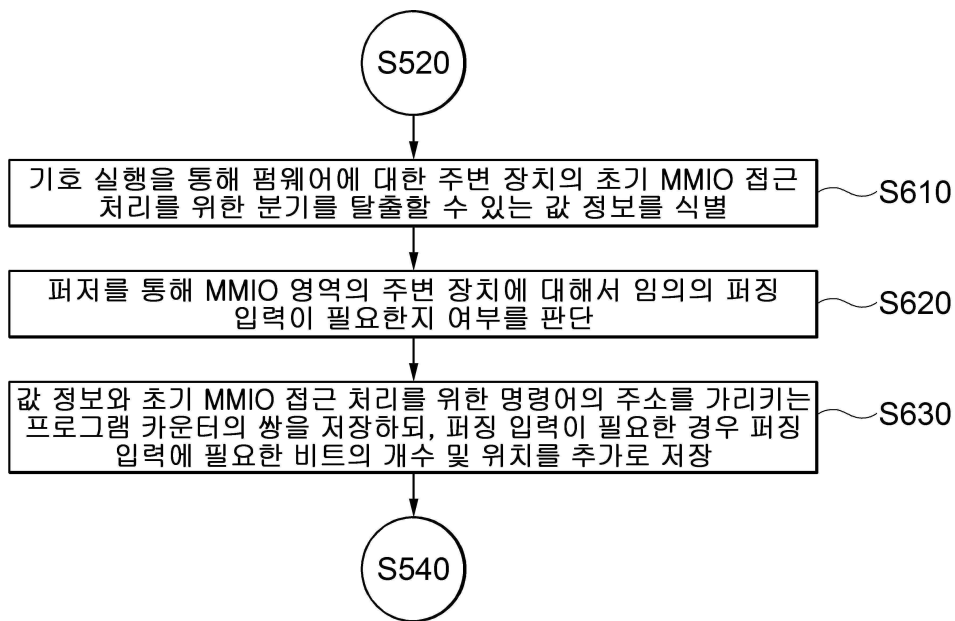
도면4b



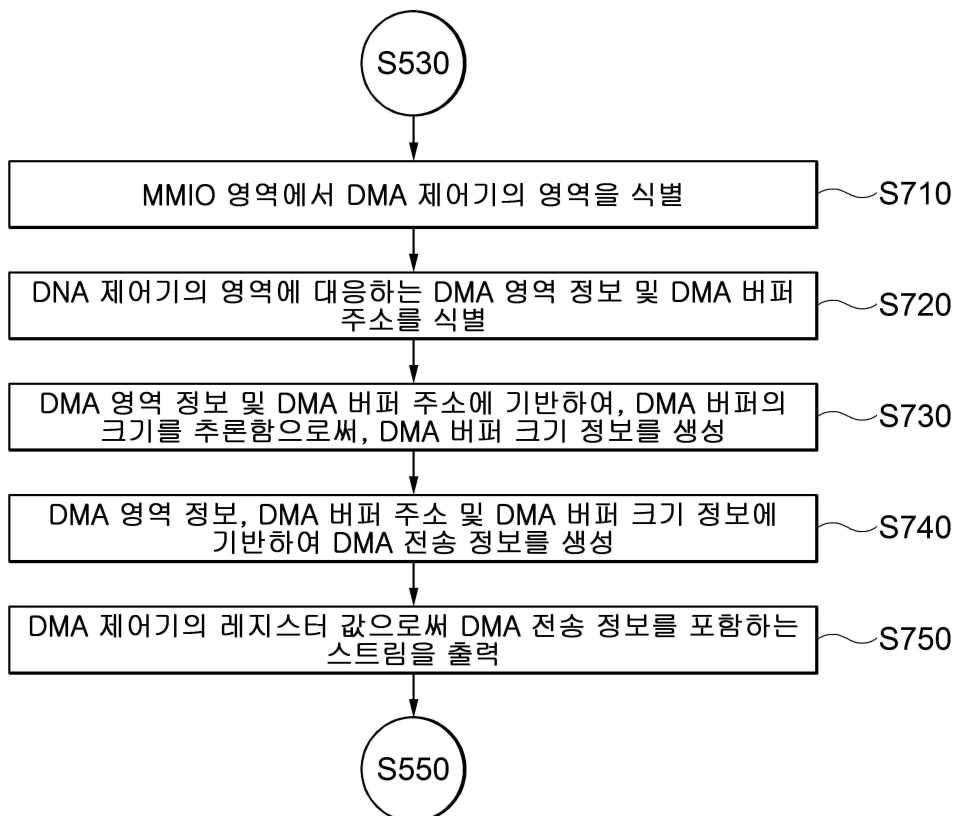
도면5



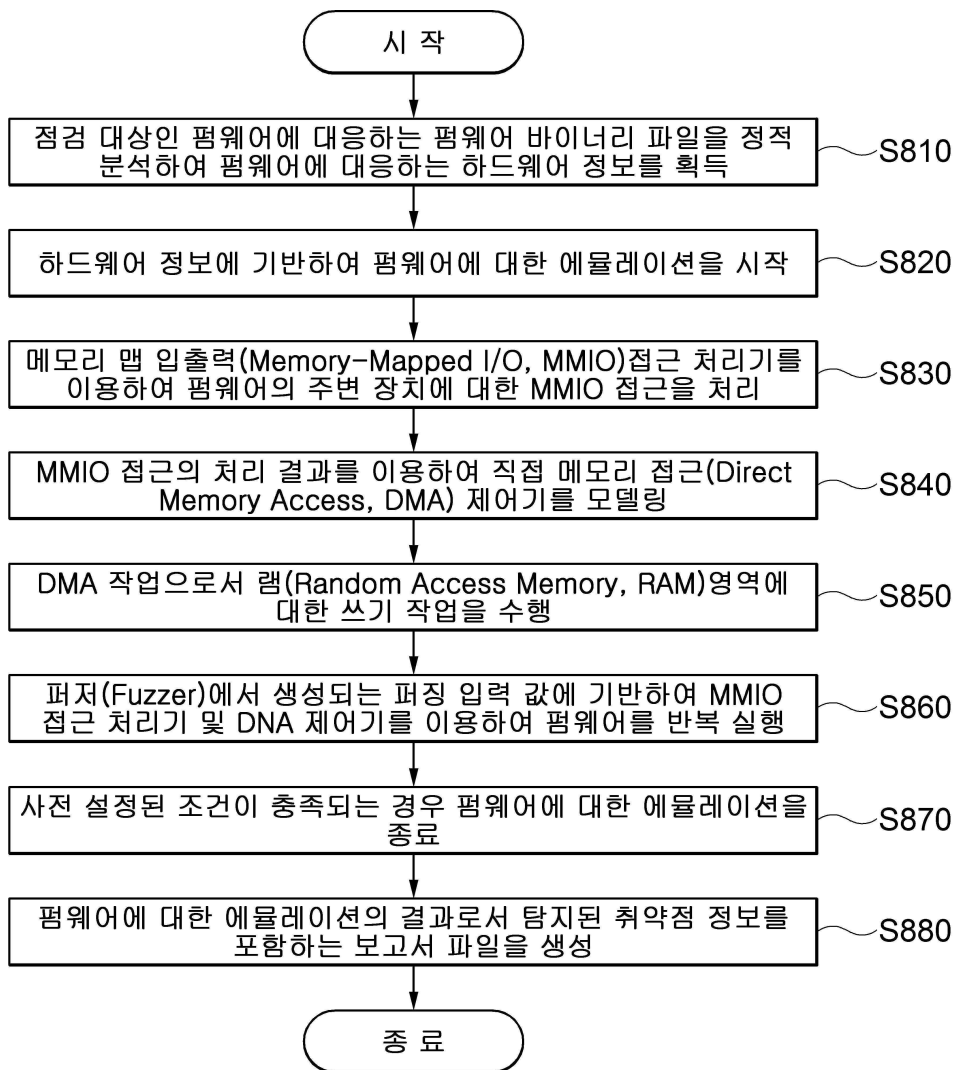
도면6



도면7



도면8



도면9

