CPU 하드웨어 Trojan 에 대비한 신뢰성 확보를 위한 이질시스템 연구

김한이, 이보선, 서태원
고려대학교 컴퓨터교육학과
e-mail : hanyeemy@korea.ac.kr

A Study on Heterogeneous Systems Against CPU Hardware Trojan for Enhancing Reliability

Hanyee Kim, Bosun Lee, and Taeweon Suh
Dept. of Computer Science Education, Korea University

요 약

하드웨어 Trojan 은 악의적인 목적으로 전자 회로망에 수정을 가한 회로로, Trojan 설계자의 목적에 따라 특정 환경에서 동작(Trigger) 되어 전체 시스템에 심각한 보안문제를 초래할 수 있다. 일반적으로 Trojan 은 동작 시 시스템의 방화벽이나 보안 장치 등의 시스템 일부를 하드웨어적으로 무력화 시켜 제 기능을 상실시키며 심각한 경우 시스템 전반에 걸쳐 모든 기능을 마비시킬 가능성이 있다. 본 연구에서는 군사 시설과 같이 고도의 보안 및 정확성이 요구되는 시스템 분야에서 신뢰성 향상에 초점을 두고, 서로 다른 프로세서에서 같은 연산을 처리하여 이를 비교할 수 있는 Vote Counter 를 탑재한 이질 시스템(Heterogeneous system)을 제안한다.

1. 서론

기존의 컴퓨터의 보안 공격은 버퍼 오버플로우(Buffer overflow)나 DDoS(Distributed denial of service) 와 같은 소프트웨어, 혹은 네트워크 프로토콜의 취약점을 이용하였다. 최근에는 보다 다양한 방식으로 컴퓨터에 대한 보안공격이 이루어지고 있는데, 반도체 생산 공정이 고도화되고 분업화되면서 시스템 공격의 방법으로 하드웨어 Trojan 이 이슈화되고 있다. 시장조사기관 IHS iSuppli [1] 에 따르면 위조된 하드웨어가 중요 군사 시설에 많이 포함되어 있으며, 그 수는 급격히 증가하는 추세라고 한다. 2007 년에 일어난 시리아의 핵 시설에 대한 이스라엘의 미사일 공격은 하드웨어 Trojan 피해 사례 중 하나이다. 이 때 관찰 레이더가 오동작하여 핵 시설이 파괴되었는데, 그 원인으로 레이더에 사용된 마이크로프로세서가 지목되었다. 프로세서 내의 하드웨어 Backdoor 가 집적되어 있었을지도 모른다는 의혹이 제기되었다. 지목된 Backdoor 는 특정코드가 수행될 때 반응하여 일시적인 기능 마비를 한 것으로 추측되었다. 또한 익명의 미국방성 관리에 의하면, 한 유럽의 IC 제조사가 원격 제어 가능한 Kill switch 를 집적한 프로세서를 제조하고 있다고 한다 [2].

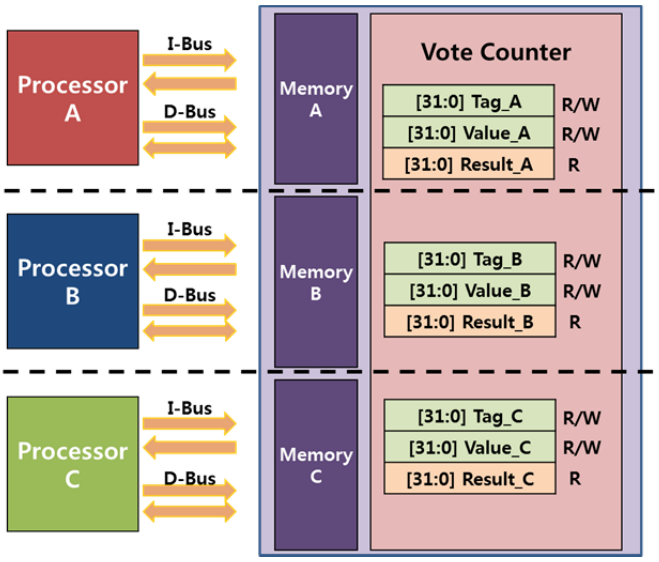
이 논문은 2012 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2012-008231)

본 연구에서는 CPU 하드웨어 Trojan 에 대비하여 서로 다른 프로세서에서 같은 연산을 진행하고 그 결과를 비교해 연산 결과의 신뢰성을 보장하는 이질 시스템의 구성을 제안한다. 기존의 이질 시스템은 OpenCL[3]이나 CUDA[4]와 같은 프로그래밍 모델을 이용하여 고속의 그래픽 처리나, 병렬 처리로 복잡한 연산을 처리하는 성능 향상에 중점을 두고 개발되었다. 하지만 이질 시스템은 서로 다른 아키텍처 구조를 가지므로 동시에 같은 연산을 수행하고 결과를 비교하면 연산 결과값에 보다 높은 신뢰성을 기대할 수 있다. 이질적 환경에서의 신뢰성 향상을 위한 연구는 분산 시스템 분야에서 소프트웨어 적으로 이미 시작되었다[5]. 하지만, 현재 하드웨어적으로 보안 향상을 위해 진행된 연구는 미비한 실정이다. 본 연구에서는 신뢰성을 보장하는 이질 시스템과 이를 하드웨어적으로 지원하는 메모리 맵 (Memory map) 기반 장치인 Vote Counter 를 제안한다. 본 논문의 구성은 시스템에 대한 소개, 동작원리와 사용 방법, 실험 방법에 대하여 설명한다. 이와 더불어 현재 연구의 한계와 향후 연구 방향으로 구성하였다.

2. 시스템 소개

본 연구에서 제안하는 시스템은 독립된 메모리 공간과 서로 다른 아키텍처를 갖는 이질 시스템 환경이다. 시스템 내에서는 주요 변수를 비교할 수 있는 메모리 맵 기반의 장치 Vote Counter 를 가지며, 각 프로세서는 Vote Counter 를 활용하여 다른 프로세서와

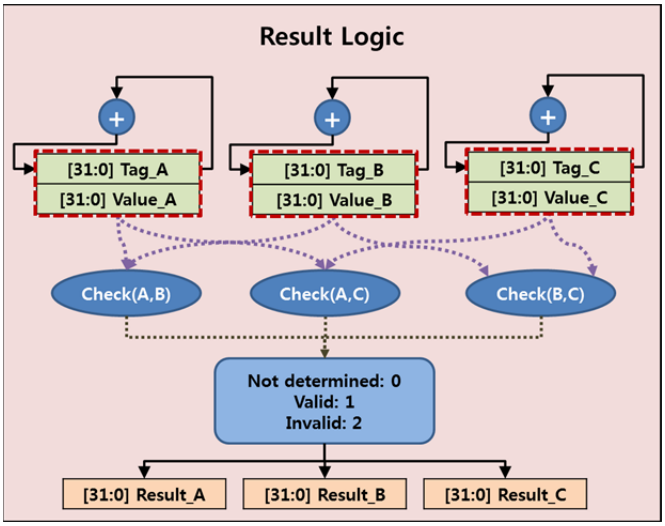
결과 값이 일치하는지 알아 볼 수 있다. Vote Counter 는 서로 다른 아키텍처, 다른 클락 사이클(Clock cycle) 에서도 결과 값을 비교할 수 있도록 설계 되었다. (그림 1)과 같이 Vote Counter 는 프로세서 별로 저장한 데이터 값의 순서를 표기하는 Tag 레지스터와 해당 데이터를 저장할 수 있는 Value 레지스터, 각 프로세서가 저장한 Value 값들을 비교하여 투표 결과를 알려주는 Result 레지스터로 구성된다. 각 프로세서는 순차적으로 비교할 값들을 Vote Counter 의 Value 레지스터에 저장하고, Tag 레지스터는 이에 따라 Tag 값을 갱신한다. Tag 값이 같으면 Vote Counter 는 Value 값들을 비교하여 그 결과를 Result 레지스터에 반영한다.



(그림 1) Vote Counter 를 내장한 시스템 개요

3. Vote Counter 의 사용 방법 및 동작원리

Tag 레지스터의 값이 일치해야 Vote Counter 가 값 비교에 들어가므로, Vote Counter 를 사용하기 위해서는 Tag 값들을 모두 같은 값으로 초기화해야 한다. Tag 레지스터는 Value 레지스터에 새로운 값이 쓰일 때마다 자동으로 1 씩 증가하므로 초기화 후에는 Tag 값을 갱신할 필요가 없다. 다만, Vote Counter 사용시에 Value 값을 갱신할 때마다 Tag 값이 증가하는 것에 유의해서 프로그램을 작성해야 한다. 만약 프로그램의 특성상 Value 레지스터에 값을 쓰는 횟수가 다르다면 Value 레지스터와 함께 Tag 레지스터에 값을 같이 갱신하는 방법을 사용하여 Vote Counter 를 사용할 수 있다. (그림 2)와 같이 Result 레지스터는 Tag 값과 Value 에 따라 0 부터 2 사이의 값을 갖는다. 사용자는 Tag 와 Value 값의 일치 여부를 Result 레지스터를 통해서 확인이 가능하며, 만약 Tag 값이 하나라도 다르면 Not determined 로 취급되어 모든 Result 레지스터가 0 이 된다. Tag 값이 모두 같은 상태에서는 Valid 값의 비교를 통해 Result 값이 결정된다. Value 값이 서로 일치하는 프로세서는 Result 가 1 로 세팅되며, 이와 다른 Value 값을 가지는 프로세서는 Result 가 2 로 세팅된다. 만약 모든 Value 값이 다르면 각 Result 레지스터가 2 로 설정된다. 모든 Result 레지스터는 하드웨어적으로 동기화되어 동시에 값이 갱신된다.



(그림 2) Vote Counter 의 Result Logic

좀 더 구체적인 사용방법은 <표 1>의 예제 코드를 통해 확인해 볼 수 있다. 예제코드를 살펴보면 먼저 Tag 레지스터를 0 으로 초기화 하는 것으로 시작한다. 프로세서는 Value 레지스터에 중요 연산의 결과 값을 저장하고, 연산된 결과를 Result 레지스터를 통해 지속적으로 읽게 하였다.

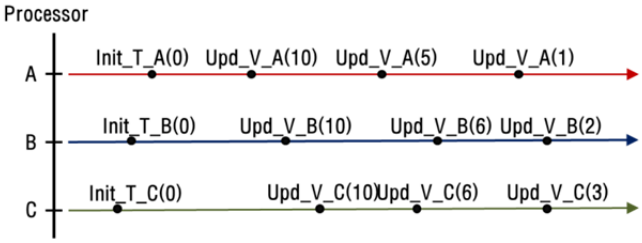
<표 1> Vote Counter 동작을 위한 예제

Example code

```
int wait(*result);
…
*tag = 0;
…
*value = Critical_func(x);
Result_handler(wait(*result));
*value = Critical_func(y);
Result_handler(wait(*result));
*value = Critical_func(z);
Result_handler(wait(*result));
…
int wait(*result)
{
        while (*result==0)
        ;
        return (int) *result;
}
```

(그림 3)은 <표 1>의 코드를 각각 서로 다른 세 프로세서에서 실행시켰을 때 Vote Counter 레지스터들의 값 변화를 시간에 따라 묘사한 예제이다. 각 프로세서의 화살표 부분을 보면, 코드와 같이 Tag 레지스터를 초기화 한 후 Value 레지스터의 값을 세 번 갱신하는 것을 알 수 있다. 시간의 흐름에 따라 (그림 3) 하단의 레지스터 값들의 변화를 살펴보면, 각 프로세서의 Result 레지스터는 모든 프로세서의 Value 레지스터

가 갱신되어 Tag 값이 같아진 후에야 동시에 갱신된다. 처음의 연산에 대한 Result 값은 Value 값이 10 으로 모두 동일하기 때문에 모두 1 로 세팅된다. 두 번째 경우에는 B 와 C 의 연산결과가 같고 A 가 다르기 때문에 Result Logic 의 투표 알고리즘에 따라 B 와 C 의 Result 레지스터가 1 로 세팅되고, A 의 Result 레지스터는 2 로 세팅된다. 세 번째의 경우에는 각각 서로 다른 Value 값을 가지게 되므로, 모든 Result 레지스터는 2 로 세팅된다. 모든 Result 레지스터는 가장 먼저 새로운 Value 값을 갱신하는 프로세서에 의해 다시 0 으로 세팅되며, 다른 프로세서의 Value 레지스터가 갱신될 때까지 그 값을 유지한다.



Tag_A	-	0	1	2	3
Value_A	-		10	5	1

Tag_B	-	0	1	2	3
Value_B	-		10	6	2

Tag_C	-	0	1	2	3
Value_C	-		10	6	3

Result_A	0	1	0	2	0	2
Result_B	0	1	0	1	0	2
Result_C	0	1	0	1	0	2

(그림 3) 시간에 따른 Vote Counter 레지스터의 값

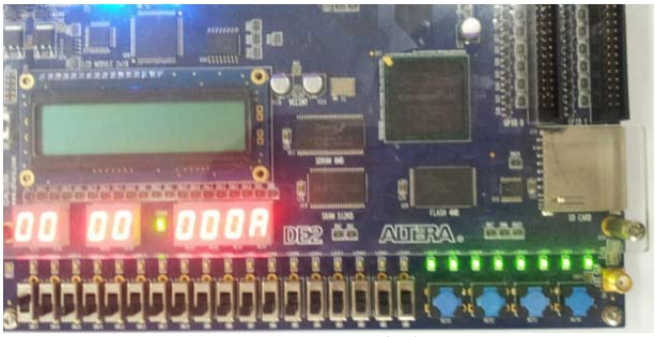
4. Vote Counter 의 실험 방법

Vote Counter 및 전체 시스템은 Verilog code 로 작성되었으며 Modelsim Simulator 를 통한 검증과 FPGA (Field-programmable gate array)보드를 통한 검증을 통해 동작을 확인하였다. 자세한 실험 환경은 <표 2>와 같다.

<표 2> Vote Counter 검증을 위한 환경

Tool Type	Tool Name
Simulation Tool	Modelsim Altera 10.0.c
Emulation Tool	Qurtus 11.1
FPGA Board	Altera EP2C35F672C6
Candidate Architecture	MIPS, ARM
Compiler	mips-elf-gcc, arm-elf-gcc

실험은 Vote Counter 가 서로 다른 프로세서와의 구성에서도 호환성을 보이는지에 대해 중점적으로 확인하였으며, 시스템을 구성한 아키텍처로는 MIPS, ARM 프로세서를 사용하였다. 시스템 검증을 위해 크로스 컴파일러(Cross compiler)를 사용하여 각 메모리에 예제 코드를 내장시켰으며, 본 실험을 위해 Vote Counter 레지스터들은 0xFFFF0000 에서 0xFFFF0008 까지의 메모리 맵 주소에 Tag, Value, 그리고 Result 순서로 설정하였다. FPGA 를 통한 검증에서는 가시적인 FPGA 상의 동작 확인을 위해 추가적인 조합회로를 시스템에 포함시켜 보드에 연결된 7-세그먼트(segment) 및 컬러 LED 등을 활용하였다. (그림 4)는 FPGA 상에서 예제 코드를 동작시켜 정상 동작여부를 검증하는 부분이다. FPGA 의 8 개의 7-세그먼트는 선출된 Value 레지스터 값을 4-bit 씩 나눠 16 진법으로 표기하도록 하였으며, 오른쪽 하단 부분의 LED 는 각 세 프로세서의 Result 값에 따라 온오프 (on/off) 가 결정되도록 하였다. (그림 4)는 예제코드의 첫 번째 투표가 완료된 상황을 나타내는 사진이다. (그림 3)과 같이 첫 투표 때 선출된 Value 값은 10 이므로 7-세그먼트는 A 로 표기되며, LED 는 연산 결과가 모두 일치하므로, 모두 켜져 있는 상태이다.



(그림 4) Altera FPGA 상의 검증

5. 결론 및 향후 연구방향

본 연구에서 Vote Counter 는 다양한 아키텍처를 포용하여 이질 시스템에서의 연산결과 신뢰성을 향상시켰다. 하지만, 이로 인해 발생되는 성능 저하는 고려할 사항이다. 아키텍처 별로 연산이나 기능에 따라 성능 차이가 나타난다. 현재 Vote Counter 의 인터페이스는 (그림 3)과 같이 가장 속도가 느린 프로세서의 속도에 맞춰 Result 레지스터 값을 변환시키는 한계가 있다. 만약 시스템을 구성하는 어느 한 프로세서의 성능이 지나치게 떨어진다면, 프로그램의 구성에 따라 전체 시스템의 심각한 성능 저하를 초래할 수 있다. 또 다른 Vote Counter 의 한계점으로 입출력 프로토콜이다. Vote Counter 는 현재 대부분의 시스템이 지원하는 인터럽트(Interrupt) 방식이 아닌 폴링(Polling) 방식을 사용한다는 점이다. 폴링 방식은 인터럽트 방식에 비해 지속적인 확인이 요구된다. 그러므로, 프로세서의 연산 수행 능력을 떨어트릴 수도 있고 사용자가 Vote Counter 를 사용하는데 있어서의 편의성을 떨어트릴 수 있다. 향후 연구에는 Vote Counter 가 인터럽트 방식을 이용해 시스템에 적용될 수 있도록 수정할 예정이며, 부가적으로 프로세서간의 성능 차에 따른 전체 시스템 성능 저하를 줄이기 위한 연구를 진행하여 고도의 보안이 요구되는 분야에서 적극 활용될 수 있도록 지원할 계획이다.

참고문헌

[1] eetimes. Available from: http://www.eetimes.com/electronics-news/4236345/Counterfeit-parts-putting-military-at-risk-.
[2] Adee S. The Hunt For The Kill Switch. Spectrum, IEEE. 2008;45(5):34-9.
[3] Stone JE, Gohara D, Shi G. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. Computing in Science & Engineering. 2010;12(3):66-73.
[4] Sanders J, Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming: Addison-Wesley Professional; 2010. 312 p.
[5] Dogan A, Ozguner F, editors. Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing. Parallel Processing, 2000 Proceedings 2000 International Conference.