Secure Processor Design

by

Shayan Taheri

University of Central Florida
Orlando, Florida
Fall 2015

Overview

1. Introduction
2. Related Work
3. Questions

Introduction

- **Security** → A new processor design parameter → Why?
  - Protection of trusted code and data
  - Integrity verification of computations
  - Prevention of unauthorized access to system and network

Introduction …

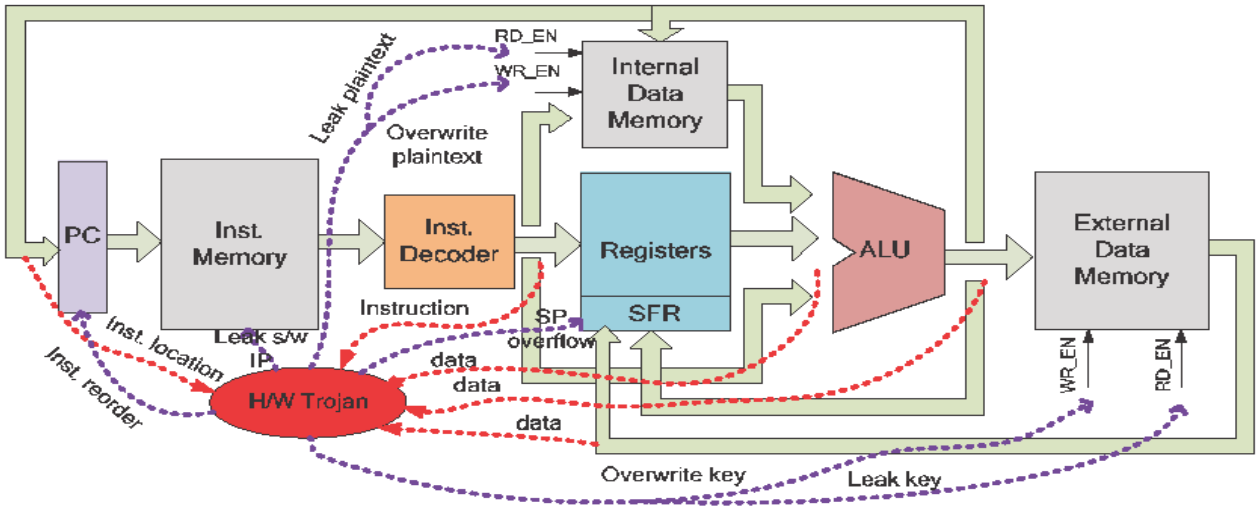
- **Security and Processor**
  - **Processor for Security** → Provision of security from processor
    - Example: Cryptographic Processors
  - **Security of Processor** → Provision of security for processor
    - Why? → Possible attacks and threats for processors
    - Example: Malicious modifications, bugs, virus, and etc.

Introduction …

- **Countermeasures for Attacks**
  - **Physical Analysis** (After Fabrication)
    - Destructive → Reverse-engineering
    - Nondestructive → Functional Verification and Side-Channel Analysis
  - **Design Analysis** (Before Fabrication)
    - Information Flow Tracking
    - Functional Verification

Introduction …

- **Functional Verification**
  - Formal Verification
  - Model Checking → VLSI Testing and Verification
  - Run-Time Verification



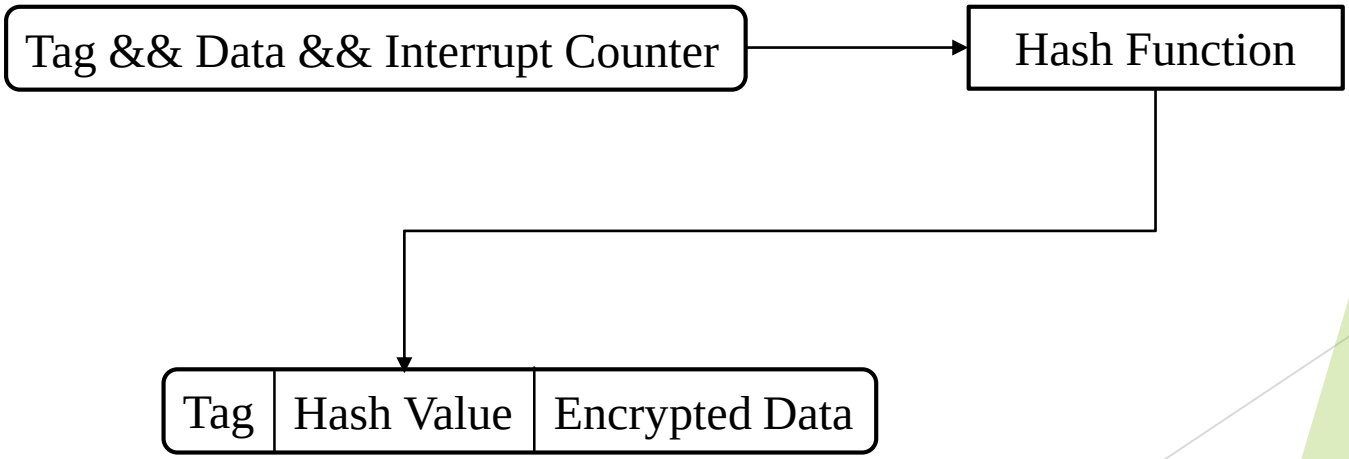
Trojan Insertion inside Processor

Related Work

- **Security Checkers: Detecting Processor Malicious Inclusions at Run-time**
  - A run-time verification technique
  - Property Specification Language → A standard assertion language
  - Inclusion of synthesizable assertion functions into hardware design
  - Experiment → The Plasma CPU → A 32-bit MIPS architecture
    - Memory write verification
    - Delay → Additional one clock cycle for every write operation
    - Area Overhead → + 0.1%

Related Work …

- **Architectural Support for Copy and Tamper Resistant Software**
  - A secure processor design technique → XOM Architecture
  - Targeting processor interactions with memory
  - Implementation of cryptographic algorithms on these interactions

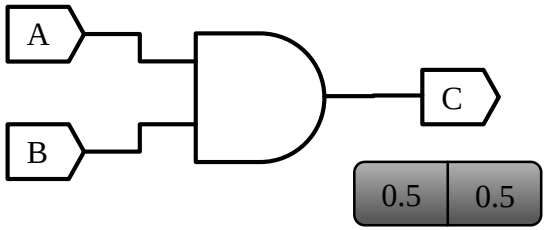


Message Format from Processor

Related Work …

- **FANCI: Functional Analysis for Nearly-unused Circuit Identification**
  - A model checking technique
  - **Goal**: Finding possible infected sites in a digital IC
  - **How?** → The input wire(s) effect of each component on its output wire
    - Having a vector of "**Control Values**" for each output wire
  - **Outcome** → Suspicious wires → Nearly stuck-at fault wires
  - My Implementation using C++ Language → 1574 Number of Lines!

Related Work …



Input “A” Fixed at **0**

A	B	C
0	0	0
0	1	0

Input “A” Fixed at **1**

A	B	C
1	0	0
1	1	1

$$\text{Control Value} = \frac{\text{Number of Conflictions}}{\text{Number of Rows in Each State}}$$

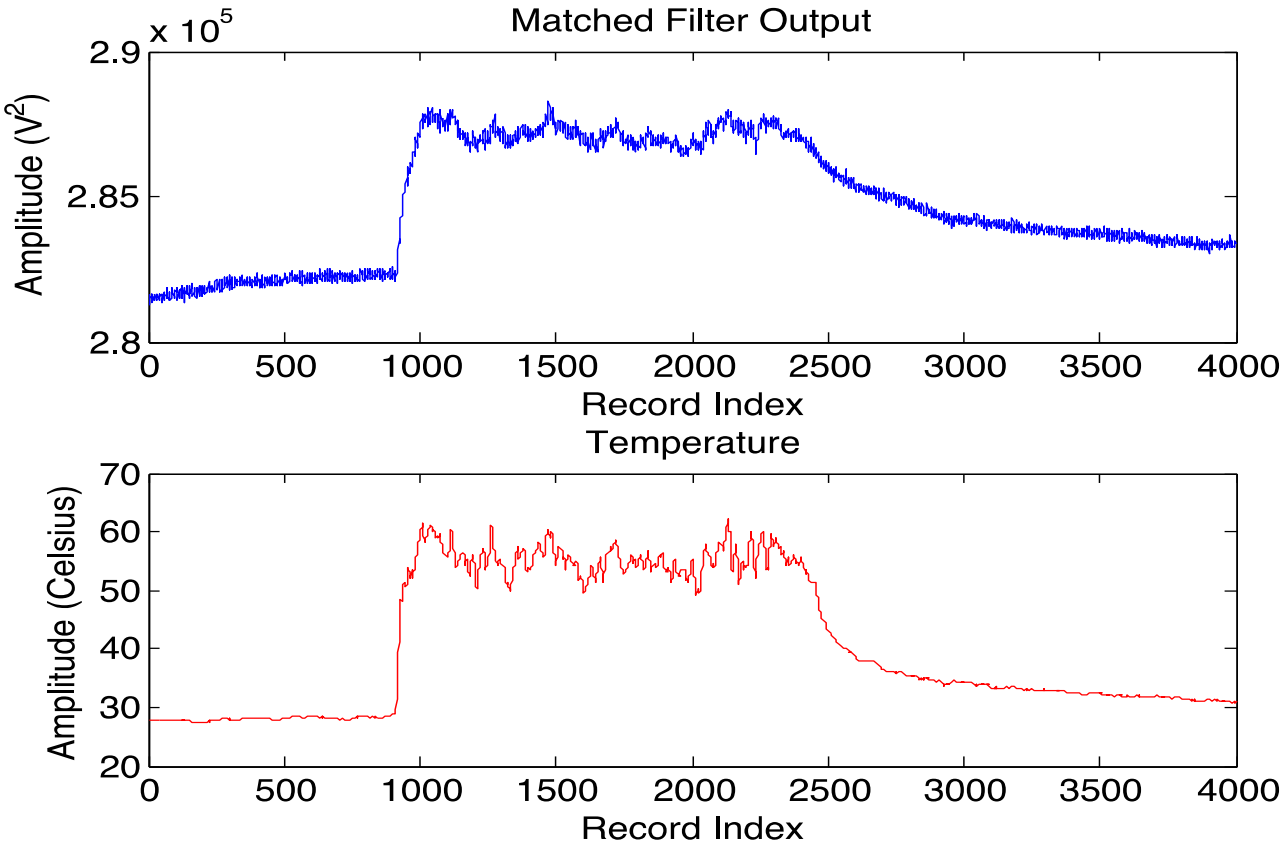
Infected NoC Router

Method Name	Run Time (Minute : Second)	$\frac{\text{Suspicious Wires}}{\text{Total Wires}}$ (%)
Average	5:30	3.7225
Median	5:24	3.9590

Related Work …

- Physical Layer Identification (PLI)
  - A side-channel analysis technique
  - Targeting processor interactions with networking devices
  - Using the first layer of the OSI model for device identification
  - PLI Methodology
    - Identify and acquire a device's signal (i.e. Fingerprint)
    - Extract a set of meaningful features from the signal
    - Compare the test feature set with the reference feature set
    - Determination of the device identity

Related Work …



Device Tampering by Temperature Variation

Questions