FP7-ICT-SEC-2009-5
Contract no.: 258754
www.TAMPRES.eu

TAMPRES

Deliverable D4.2

Implementation of lightweight memory protection mechanisms

Editor:	Aurélien Francillon
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public
Contractual delivery date:	31 June 2012
Actual delivery date:	31 June 2012
Suggested readers:	Consortium, European Commission
Version:	1.0
Total number of pages:	11
Keywords:	Memory protection, virtualization, MPU

Abstract

This deliverable reports on the current state of work on memory protection within the TAMPRES project.

Disclaimer

Impressum

TAMPRES- Tamper Resistant Sensor Node

TAMPRES

WP4 “System Design and Partitioning”

Editor: Aurélien Francillon, Eurecom

List of authors

Company	Author
Eurecom	Aurélien Francillon
Eurecom	Andrei Costin

Contents

List of Figures

1 Introduction

This document is accompanying the source code package provided as a implementation deliverable of D4.2. It briefely presents the implementation choices and how to perform basic simulations.

2 Implementation Choices

The TAMPRES project partners choose the OpenMSP430 variant of the MSP430 microcontroller as a basis of chip to used. The main reasons of this choice are the following:

- The OpenMSP430 is available under a BSD License, which puts almost no restrictions on the uses of the resulting work, whether we later decide to publish it as an open source project, license or produce it commercially. This also simplifies cooperation between members of the projects.
- It's code organization and quality is very good, in particular it has a very good automated test suite.
- T he OpenMSP430 has a very good debug support, that is usable with of the shelf tools. Being binary compatible with the original MSP430 it also benefits from the commercially of open source toolchains.

3 Code Organization

We keep the code in a git repository, where we imported the original repository of the OpenMSP430. We then regularly import important changes from the parent project. Code Contributions can be summarized as follows (which however also includes some automatically generated files and therefore cannot be directly translated into an effort).

```
project   : openmsp430
repo age  : 3 years ago
active    : 107 days
commits   : 330
files     : 1107
authors   :
  136 olivier.girard          41.2%
   75 Frank Vater             22.7%
   63 Aurélien Francillon     19.1%
   44 Aurélien Francillon     13.3%
    8 Marcel Medwed           2.4%
    3 Andrei Costin           0.9%
```

Olivier Girard being the original author of the OpenMSP430 and Aurélien Francillon appearing once for ETHZ and once for Eurecom.

Figure 1 presents the root directory of the project is organized as follows:

-Tools

-RAM RAM blocks for IHP Technology.

-fpga verilog and vendor specific files for test on FPGAs. This is mainly used to validate the prototype on a real platform, as well as to validate the debug connection.

-doc Documentation of the OpenMSP430.

-core This includes the main OpenMSP430 code.

-synthesis scripts and sources for synthesis.

-sim scripts for simulation, which includes a suite of scripts for testing the OpenMSP430.

-rtl_sim for simulation at the RTL level.

-syn_sim for post synthesis simulation.

-rtl contains the main RTL files

-Verilog contains the main RTL files for the modified OpenMSP430 core, especially the integrity checks in SMART are performed in omsp_mem_backbone.v.

-VHDL contains the VHDL files for the cryptographic blocks

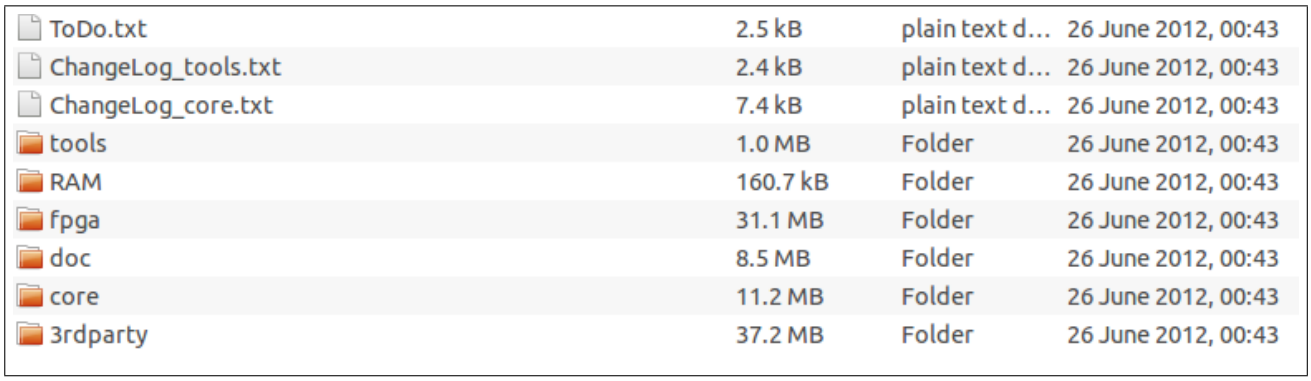

Figure 1: Listing of the root directory of the TAMPRES Chip

4 Architecture Overview

As can be seen in Figure 4, modifications to the architecture of the OpenMSP430 are located in the memory backbone and to the memories the SMART ROM and the KEY ROM. In this preliminary prototype, the SMART ROM is implemented as an SRAM, to enable flexibility in testing, while the KEY ROM is implemented as Latches. The changes to the memory backbone consists in adding the memory checks, tracking memory accesses and resetting the processor when a violation occurs.

5 Running Tests

The minimal set of tools required to test RTL code is :

- Iverolog simulator
- msp430 GCC toolchain
- GTKWave
- Other tools are also supported by the test framework (e.g. Modelsim, Cadence...).

With those tools installed tests can be run automatically from the folder openmsp430/core/sim/rtl_sim/run. Original tests can be run as follows, to ensure that no modification introduced problems:

```
> ./run_all
Cleanup...
 ====================================================
| Start simulation:             two-op_mov
 ====================================================
Compile, link & generate IHEX file (Program Memory: 32768 B, Data Memory: 16384 B,
Convert IHEX file to Verilog MEMH format...
Start Verilog simulation...
 ================================================
|                 START SIMULATION               |
```


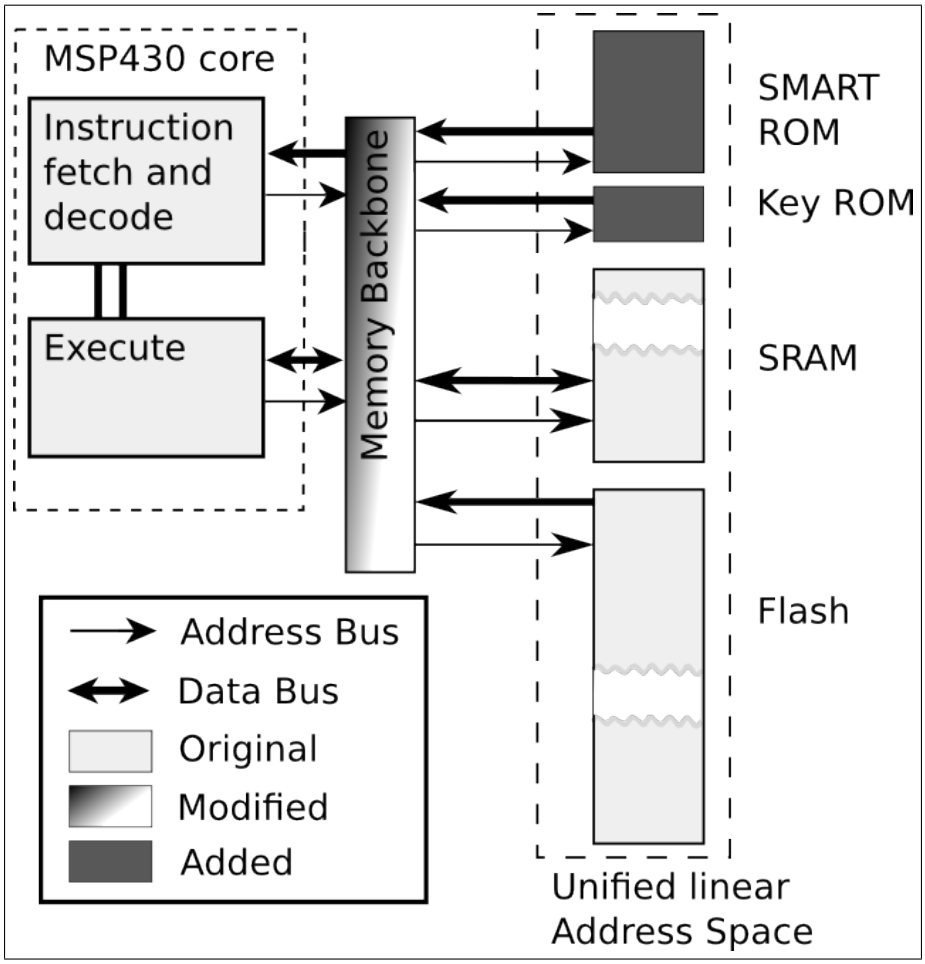

Figure 2: MSP430: Memory backbone was modified to control access to ROM and Key. Since MSP430 is based on Von Neumann architecture, concurrent access can occur to different memory parts (e.g., instruction fetch and read data). In that case, memory backbone arbitrates bus access and temporarily saves/restores data.

```
 ================================================
 ================================================
|               SIMULATION PASSED                |
 ================================================
Cleanup...
 ============================================================
| Start simulation:             two-op_mov-b
 ============================================================


[...]
 ================================
| Number of passed patterns : 60
| Number of failed patterns : 0
| Number of skipped patterns: 13
|--------------------------------
| Number of patterns:        73
 ================================
```

Furthermore, a dedicated test suite can be run to test the memory access control checks as follows:

```
> ./run_srom

[...]
 ============================================================
| Start simulation:             skey_valid_access
 ============================================================
Compile, link & generate IHEX file (Program Memory: 32768 B, Data Memory: 16384 B,
do mac function is at: 0x00006000
Convert IHEX file to Verilog MEMH format...
Start Verilog simulation...
 ================================================
|                 START SIMULATION               |
 ================================================
VCD info: dumpfile tb_openMSP430.vcd opened for output.
 ================================================
|               SIMULATION PASSED                |
 ================================================
Cleanup...
[...]

 ================================
| Number of passed patterns: 9
| Number of failed patterns: 0
|--------------------------------
| Number of patterns:        9
 ================================
 Make sure passed == total
```

Those tests perform several checks on the SMART implementation, for example verifying that no code other than code in the SMART ROM memory may access the Key, checks for all the assumptions listed in the design document are performed.

6 Conclusion

In this document accompanying the code deliverable we briefly introduced the project code organization, it's architecture and the organization of its test suite.

[end of document]

