MeXT-SE: A System-Level Design Tool to Transparently Generate Secure MPSoC

Md Jubaer Hossain Pantho
University of Florida, Gainesville, USA
mpantho@ufl.edu

Christophe Bobda
University of Florida, Gainesville, USA
cbobda@ece.ufl.edu

***Abstract*—This paper presents the MeXT-SE (Multiprocessor Exploration Tool with Security Extension), an FPGA-based MPSoC development tool capable of generating platform-independent MPSoCs with enforced hardware access control mechanism from a high-level abstraction.**

I. MeXT-SE:Design Flow

Figure 1(a) illustrates the design flow of our MeXT-SE tool. The tool starts with a user-defined abstract and concrete representation of a system and ends up generating a final hardware design with enforced countermeasures for preventing unauthorized access of hardware IPs in the SoC. The user-defined specifications include the number of processors, memory size, hardware IPs, operating system, target device, etc. The tool uses this knowledge to generate the appropriate hardware design by setting up the communication structure of different components. While setting up the communication

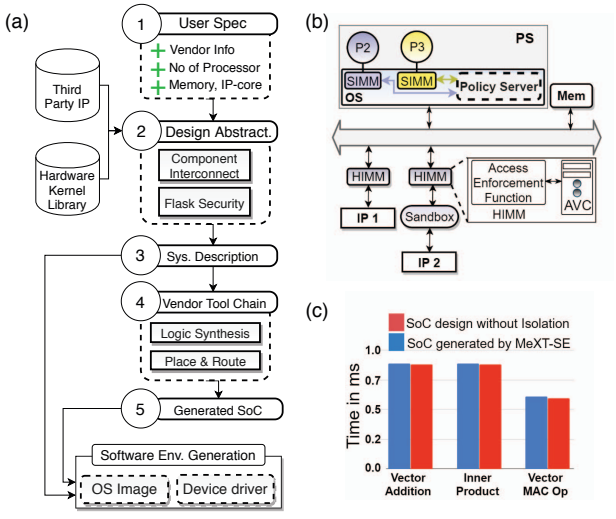

Fig. 1. (a) MeXT-SE Design Flow. (b) Generated Secure SoC. (c) Computation Overhead

network, an access control mechanism that inherits MAC-based authentication policies, found in the flask security architecture, is enforced directly in the hardware design [1]. Abstract specifications produced by MeXT-SE are generic and can be used for the implementation of different MPSoCs, regardless of the technology. The final script generated at the end of step 4 (Figure 1(a)) comprises concrete specifications. It can be invoked within a vendor toolchain (Vivado, Quartus, etc.) to generate the final bitstream of the secure SoC. Besides, MeXT-SE provides mechanisms to generate the appropriate device drivers for the IP cores with flask security enforced.

The flask security framework is implemented as a decentralized hardware/software architecture with the Hardware IP Management Module (HIMM) governs access control at the IP level, while the Software IP Management Module (SIMM) manages policies inheritance of security contexts and queries the host kernel security server for associated permissions (Shown in figure 1(b)).

The HIMM consists of an internal enforcement function and an Access Vector Cache component (AVC) to cache the last queries and ensure that policies check are handled locally. Each HIMM maintains a map of security context labels for the corresponding hardware module it manages. It implements a custom circuit, the "Access Enforcement Function," which guards access to the hardware modules according to the host kernel MAC policy. Upon receiving an access request to the hardware module, the SIMM module on the host CPU queries the host kernel security server for associated permissions. The server consults its MAC policy and returns associated permissions, which are then sent to the respective HIMM module for future access requests.

II. Results & Conclusion

We tested the feasibility of our approach by generating reconfigurable hw/sw designs for Xilinx FPGAs. The results suggest that the added isolation framework contributes little to the performance overhead of the generated SoC. The comparison of computation time for three different designs is shown in figure 1(c).

MeXT-SE enables FPGA-accelerators to inherit at run-time, software security policies of the processes calling them. This capability allows system security enforcement mechanism to propagate access control privilege boundaries expressed at the kernel level, down to individual IP. The work was supported by Air Force Research Lab & MIT Lincoln Lab.

References

[1] P. Loscocco and S. Smalley, "Integrating flexible support for security policies into the linux operating system," in *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*. USA: USENIX Association, 2001, p. 29–42.

2576-2621/20/$31.00 ©2020 IEEE
DOI 10.1109/FCCM48280.2020.00047