Optimized Secure VFPGA Provisioning in a Heterogenous Cloud Environment

A Research Report
Submitted to the Faculty of
The Department of Electrical and Computer Engineering
Villanova University

By

Stephen Donchez

In Partial Fulfillment
of the Requirements for the Degree of
Master of Science in Computer Engineering

VILLANOVA
UNIVERSITY

September 27, 2021

Contents

List of Figures

List of Tables

Chapter 1

Proposal

1.1 Literature Implementation

In "Cryptographically Secure Multi-Tenant Provisioning of FPGAs", Bag et al. outline a comprehensive architecture for the implementation of a cloud-compatible FPGA architecture whereby multiple discrete tenants can occupy "Virtual FPGA" (VFPGA) partitions of a larger Physical FPGA (PFPGA) device in a secured environment. In their proposal, they utilize Key Aggregation Cryptography (KAC) to facilitate the encryption of a symmetric (AES) key, which in turn can be used to decrypt tenant bitstreams within their allocated partition. In this implementation, they utilize a single KAC decryption engine at the PFPGA level, which extracts the AES key and furnishes it to individual AES engines at the VFPGA level. The authors claim that by implementing their design in this manner, the only party which the tenant is required to trust is the FPGA Vendor, who provides the public and private keys used by the KAC engine to decrypt the symmetric key.

1.1.1 Implementation Evaluation

However, this claim is not completely true. By virtue of decrypting the symmetric key at the PFPGA layer, the Cloud Service Provider (CSP) is in possession of both the encrypted bitstream and the symmetric key required to decrypt it. In essence, this is equivalent to them having access to the IP contained within the bitstream itself. Therefore, the tenant must necessarily also trust the CSP, which is not a policy compatible with the security needs of many would-be tenants.

Additionally, this implementation allocates a considerable amount of resources to the multiple identical AES decryption cores, which don't actually yield any additional security benefit since the AES key is present at the PFPGA layer alongside the bitstream

it encrypts. Furthermore, this implementation requires a predefined number of VFPGAs per PFPGA, which constrains the CSP's ability to adjust the size and number (inversely) of VFPGAs per PFPGA to adapt to demand.

1.2 Proposed implementation

The author proposes modifying this implementation to eliminate the need for trust in the CSP, along with eliminating the duplication of the AES IP. To achieve this goal, the author proposes the implementation of an attestably secured partition within the PFPGA fabric, which contains both the KAC and AES engines. By isolating both of these components in an attestably secure partition, the CSP no longer has access to the decrypted symmetric key, preventing them from decrypting the bitstream and gaining access to the IP contained therein. Meanwhile, the presence of the single AES IP in this secured partition eliminates the need for identical engines on each partition, which increases available space and also offers flexibility with regards to VFPGA provisioning, as the only fixed limit remaining is the number of VFPGA device IDs allocated by the FPGA Vendor for the device. This number can simply be made arbitrarily large relative to the realistic number of simultaneous tenants that the device can support, such that the CSP is effectively free to divide the PFPGA as they see fit.

The presence of a secure partition for decryption operations is useless without the ability to securely transport the decrypted data from said partition to the VFPGA in question. The use of the Hard Processor System (HPS), and specifically a Trusted Execution Environment (TEE) implemented therein, could conceivably facilitate the secure transfer of this information between regions of the FPGA without transiting the unsecured outer region of the programable logic. However, implementation of direct connections between partitions and the HPS such as this necessitates will require further investigation, as the mechanics of such an operation are not currently understood by the author. Alternatively, the use of direct memory access could be a route to secure transfer, although assurance must exist to prevent malicious access to the memory regions in question.

The use of the HPS to facilitate such a transfer has additional advantages, as it is conceivable on larger devices that multiple such partitions could be desired to facilitate simultaneous tenant reprogramming. In either case, the use of the HPS to schedule access to the secured partition(s) could similarly prove to be a useful design component.

1.2.1 Research Objectives

The above implementation requires a number of discrete objectives be completed and integrated into a comprehensive design. Those objectives are described below.

- Obtain or design the AES and KAC cores for the secured partition.

  - Ideally these cores should be pipelined for efficiency

- Investigate direct communication between partition and HPS (without untrusted transit)
- Investigate use of DMA to facilitate secure communication between partitions securely
- Design and Implement scheduling algorithm to allocate decryption partition to tenants
  - With multiple AES cores, it may make sense to share a single KAC engine if secured link between them can be guaranteed (the KAC has to decrypt a single key of trivial size, whereas the AES engine is in much higher demand due to need to decrypt entire bitstream)
- Implement attestably secure partition and provide mechanism for attestation and verification
- Implement Trusted Execution Environment (if using HPS for secured data transfer)
- Implement data routing and verification logic within TEE (if using HPS for secured data transfer)