This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2679183, IEEE Embedded Systems Letters

A Uniquified Virtualization Approach to Hardware Security

Greg Stitt, *Member, IEEE,* Robert Karam, *Student Member, IEEE,*
Kai Yang, *Student Member, IEEE,* and Swarup Bhunia, *Senior Member, IEEE*

***Abstract*—Virtualization has well-known security advantages for operating systems and software, but current techniques do not address increasingly important hardware-security concerns. For widely deployed systems (e.g., Internet of Things) and safety-critical systems (e.g., defense, automobiles), protecting against device tampering is critical, but is often unavoidable due to the relative ease of side-channel attacks. In this paper, we present a novel usage of virtualization that limits damage from bitstream tampering to a single instance of a deployed system by employing unique virtual architectures (i.e., overlays) on FPGAs.**

***Index Terms*—FPGA, overlay, security**

I. Introduction

VIRTUALIZATION of software, storage, and networking are widely used technologies with well-known security benefits. Although existing techniques address many security issues for software applications and operating systems, those techniques do not provide hardware security against device tampering, counterfeiting, and side-channel attacks.

Hardware security is a rapidly increasing concern due to the emergence of Internet of Things (IoT) applications, where tampering is both easier and more damaging due to billions of widespread units. Defense applications have similar concerns, where tampering or counterfeiting could result is loss of human life and billions of dollars of research.

One common characteristic of IoT, defense, and other embedded applications is the use of field-programmable gate arrays (FPGAs), which often provide performance, power, energy, and reliability advantages over other technologies [1]. However, FPGAs have security flaws that enable extraction of encryption keys [2], which allows an attacker to reverse engineer IP and modify the FPGA bitfile (i.e., tampering) for malicious purposes. Although numerous techniques have been introduced to protect against each one of these threats, current work shows that such tampering is still possible without significant effort, even on devices marketed as secure [2].

Ideally, the most effective strategy for hardware security would be to create a specialized FPGA or ASIC for every instance of a deployed application. With such an approach, even if a security flaw was exploited on one device, damage would be limited to that one instance as opposed to all deployed devices. Although this strategy clearly has prohibitive costs, we present an approach that achieves the same advantages at minimal cost by diversifying virtualized hardware across physical FPGAs, as demonstrated in Fig. 1.

Unlike the typical FPGA flow, our approach does not implement an application directly on the FPGA. Instead, it first passes the application code to an *overlay generator*, which creates an application-specialized virtual architecture overlay (e.g., [3]–[8]) that is implemented atop the FPGA. Because the only cost of creating a new overlay is the time required for FPGA compilation, our approach achieves security via *uniquification*, which modifies the generated overlay with unique bitfiles for every deployed FPGA. Finally, a compiler uses configuration keys from uniquification to compile the application into an overlay bitfile that configures each unique overlay. This helps protect against major threats to bitstream security (Fig. 2).

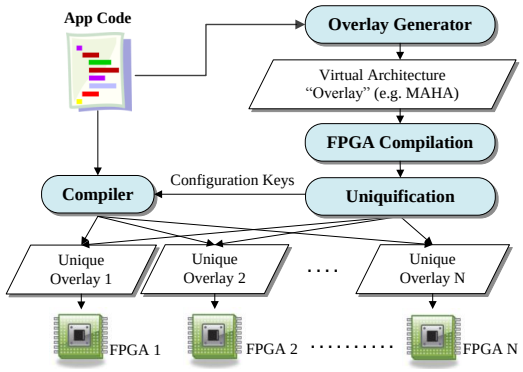

Fig. 1. Uniquified virtualization approach to hardware security.

In addition to improved security, overlays provide a number of other advantages including 10,000x faster compilation [5] than FPGA-vendor tools, 1,000x faster reconfiguration [4] than Virtex 6 FPGAs, bitfile portability across FPGAs, simplified development and debugging [7], [8], transparent high-level synthesis [4], and 1,000X smaller bitfiles [4], [5], which is critical for IoT applications. These advantages come at the cost of overhead, which for our approach ranges from 1.5X to 1.9X for execution time and energy.

II. MAHA Overlay

Although the presented security advantages can potentially be gained from any overlay, we evaluate the approach using a modified Malleable Hardware (MAHA) overlay [6], as shown in Fig. 3. The basic building block of MAHA is the Memory Logic Block (MLB) in Fig. 3(a). The MLB is a single-issue, single-cycle RISC-style microprocessor comprised of an instruction cache, called the schedule table, a lightweight custom datapath, a program counter, and an appropriately-sized shared data and lookup memory.

During program execution, MAHA fetches instructions from the schedule table. Depending on the instruction, MAHA will either select the datapath result or lookup table result for register write back. The lookup table enables energy-efficient evaluation of complex functions, while the lightweight datapath enables rapid execution of common functions for a domain. MAHA uses a number of MLBs interconnected in a multi-level hierarchy, the structure of which depends on the particular application. During compilation, instructions, data, and lookup

Greg Stitt, Robert Karam, Kai Yang, and Swarup Bhunia are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. Email: {gstitt,robkaram,kyang84,swarup}ufl.edu.

Manuscript received —; revised —; accepted —.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2679183, IEEE Embedded Systems Letters

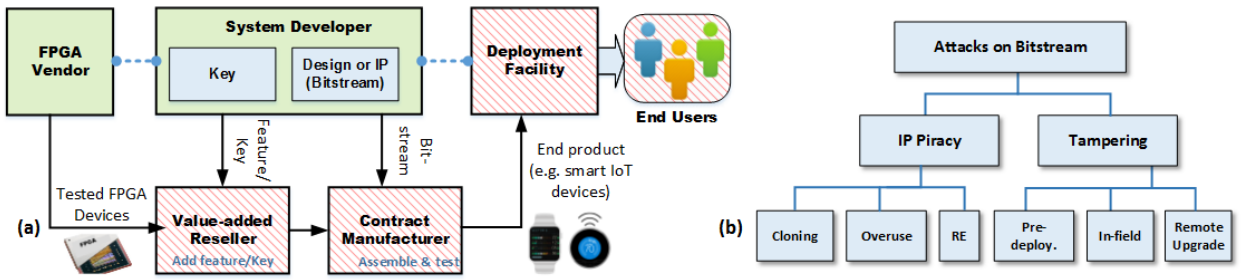

Fig. 2. (a) Typical life cycle of FPGA-based systems and stages where attacks on bitstreams can occur (marked red); (b) Brief taxonomy of major threats on bitstreams. Both unencrypted and encrypted bitstreams are susceptible to these attacks. The uniquified overlay approach aims to counter these attacks.

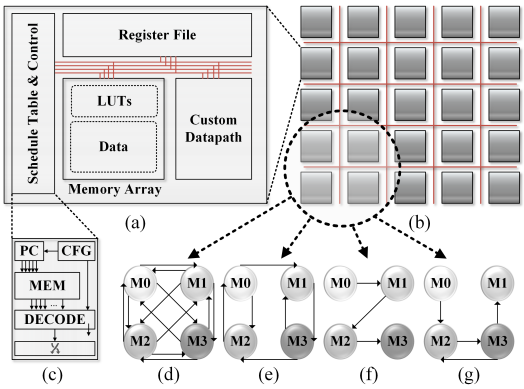

Fig. 3. (a) Overview of the MAHA architecture, with (b) a set of interconnected processing elements ([6]). (c) Individual processing elements, as well as the interconnect fabric (d-g) are highly customizable, and can be exploited for increased security.

table memory are spatially distributed among the MLBs, while local instructions within a given MLB execute temporally. The architecture is fully customizable for the target application, and can be implemented as an FPGA overlay using standard FPGA CAD toolflows.

The major security advantages of MAHA are obtained via uniquification using the following modifications:

- Instruction Set Randomization (ISR): modification of the instruction encoding
- Instruction Order Randomization (IOR): permuting of the schedule table and program counter sequence
- Interconnect Randomization (ICR): modifying the inter-MLB communication network

Of these, ISR has been applied to standard microprocessors for the purpose of software security [9], whereas IOR is more difficult to implement on a standard microprocessor due to the requirement of instruction caching. Instruction caching is not an issue with the MLB for the following reasons: 1) the schedule table size can be modified to match application requirements, and 2) applications can be mapped spatially, using additional MLBs as necessary up to the FPGA's physical resource limit. Finally, the ICR implementation depends on the MLB interconnect structure, which can vary between implementations of the overlay, and has a direct impact on the possible application mappings for that instance. The security implications are discussed in Section IV.

These uniquification approaches can be implemented within each MLB by making small modifications to the MLB structure. Specifically, ISR requires that either 1) a permutation network is used to permute the order of inputs to certain functional units, which does *not* require resynthesis, but does increase area and delay, or 2) the encoding is uniquified at the RTL level, which does not affect timing, but does require resynthesis. The time required for resynthesis can vary depending on the size of the design, but in general this can be mitigated by leveraging incremental compilation to only resynthesize the small fraction of the design performing instruction decoding in each MLB. An analysis of the recompilation time with and without partitioning MAHA's instruction encoding module demonstrate an average 24% decrease in compilation time, which can make resynthesis more practical.

IOR can be implemented in a number of ways. The important aspect of IOR is to introduce randomness into the instruction order sequence, such that no two MLBs have the same program counter activity, and no two overlays share the same instruction ordering among all MLBs. While this can be achieved using a cryptographically secure sequence generator such as a stream cipher, it can be accomplished more simply using a nonlinear feedback shift register (NLFSR) with period $2^n - 1$. There are $2^{2^{n-1}-n+1}$ different n-bit NLFSRs [10], providing a source of diversity which complements the other randomization approaches.

Finally, ICR can be implemented by beginning with a fully-connected network of MLBs, and cutting specific connections between MLBs by modifying connection and switch box configuration bits in the FPGA. As long as the overlay bitfile mapping tool, which generates the instructions for the MLBs, is cognizant of the modifications (ISR, IOR, and ICR), it can create a functionally correct and latency-aware mapping with little variation in performance across all uniquified overlays.

Because a uniquified bitstream is not portable among devices, each FPGA must provide a device ID. Each device has a corresponding randomly-generated configuration key that specifies all ISR, IOR, and ICR modifications. The key / device ID association is known only to the manufacturer. During an update, the ID can be retrieved from the target device, and the corresponding key can be looked up in the manufacturer's database. The compiler can then guarantee by construction the correctness of the mapped application because it is aware of the modifications made to the overlay for that device.

III. Overhead Analysis

In this section, we evaluate the energy, performance, and area on a 40 nm Stratix IV FPGA, both with and without the MAHA overlay. Vector-based dynamic/total power and clock frequency were measured using Altera Quartus II. To evaluate the overlay, we chose 10 benchmarks from image / signal processing and security domains, then verified functional correctness of each benchmark using Altera ModelSim

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2679183, IEEE Embedded Systems Letters

TABLE I
ENERGY, PERFORMANCE, AND SIZE OF MAPPED APPLICATIONS, WITH AND WITHOUT MAHA OVERLAY

	Execution Time (μs)			Dynamic Energy (nJ)			Total Energy (nJ)			Bitstr. Size (b)		Comb. ALUTs		Total Mem. (b)	
	MAHA	FPGA	Ratio	MAHA	FPGA	Ratio	MAHA	FPGA	Ratio	MAHA	FPGA	MAHA	FPGA	MAHA	FPGA
CI	0.24	0.29	0.83	3.98	2.24	1.78	8.04	6.76	1.19	4368	10.9M	1227	87	36216	1920
FIR	0.21	0.20	1.05	3.16	1.58	2.00	6.96	4.47	1.56	3440	10.9M	887	101	29384	1920
AES	5.23	2.56	2.04	96.67	41.82	2.31	210.01	102.08	2.06	7456	10.9M	824	771	60016	1920
DES	4.51	2.50	1.80	77.32	37.45	2.06	182.81	98.43	1.86	13392	10.9M	831	104	97280	1920
DCT	0.78	0.44	1.77	11.05	5.67	1.95	25.92	13.06	1.98	3728	10.9M	1690	90	39108	1920
SHA	12.26	5.13	2.39	179.43	80.0	2.24	351.37	200.75	1.75	5828	10.9M	1073	959	20480	1920
TM	0.11	0.10	1.10	1.53	0.98	1.56	3.66	2.48	1.48	2288	10.9M	754	60	19208	1920
LFSR	0.12	0.09	1.33	1.43	0.74	1.93	2.99	2.34	1.28	1136	10.9M	378	90	10056	1920
BF	3.10	2.11	1.47	56.39	31.39	1.80	140.06	75.14	1.86	4784	10.9M	800	199	38952	1920
THR	0.19	0.20	0.95	3.06	1.91	1.60	67.99	47.05	1.45	3344	10.9M	89	89	27640	1920
Avg	2.66	1.36	1.47	43.4	20.39	1.92	99.98	55.26	1.65	4976	10.9M	855	255	37843	1920

with 10,000 input vectors. For each benchmark, we generated specialized overlays by 1) adding/removing MLBs (up to 8), 2) optimizing the bitstream size by customizing the lookup table and schedule table sizes, and 3) specializing the datapath block. Our compiler generated all instructions for each specialized instruction set. The execution time of each benchmark was calculated based on the number of cycles needed, multiplied by the clock period as determined by TimeQuest Timing Analyzer.

Table I compares the MAHA overlay with direct FPGA implementations. On average, the MAHA overlay obtained its security advantages at a cost of 1.5X execution time, 1.9X energy, and 1.7X power. The energy-delay-product increase from the MAHA overlay was 2.8X. MAHA reduced bitstream sizes by an average of 2,190X, while using 855 LUTs on average, compared to 255 for the FPGA—an average LUT increase of 3.4X. Although this ratio may seem significant, it is inflated due to the tiny sizes of the FPGA circuits. Also, the FPGA circuits are application specific, whereas the overlay has flexibility to implement other applications, which suggests not all these extra LUTs are overhead. Furthermore, the Total Mem. column refers to the total memory bits used in the design; for MAHA, this varies with the number of LUTs and instruction memory bits used by the particular application, while for FPGA, these bits are used as buffer memory when streaming data into the design.

Although space constraints prohibit an analysis of scalability, earlier overlay studies have shown that overhead is similar, and can even decrease, for larger overlays [3]–[5].

IV. SECURITY ANALYSIS

Qualitatively, the overlay acts as an obfuscation layer which makes it more difficult for the attacker to reverse engineer the design. Namely, an adversary needs the following information to attack the system:

1) access to the uniquified FPGA bitstream, and knowledge of its format (e.g. location and meaning of LUT content bits and routing information)
2) access to the overlay bitfile, and knowledge of its format (e.g. instruction format, location of LUT content), which varies among all devices

In this section, we analyze the potential impact from the proposed diversification approaches.

A. Security against Brute Force Attacks

Quantitatively, the level of security against brute force attacks, defined as the number of brute force attempts required to reverse engineer the IP mapped to the overlay, is fairly straightforward for ISR and IOR:

- ISR: 2^n, where n is the instruction bitwidth
- IOR: $m!$, where m is the size of the schedule table

This is because there are 2^n possible binary encodings for an n-bit instruction, and the order of the m instructions in a given MLB can be permuted in $m!$ ways. Furthermore, the particular implementations of ISR and IOR used can differ among all K MLBs in the system. During application mapping, nodes from the control/dataflow graph (CDFG) are distributed among all available MLBs. A functionally correct mapping can only be realized with proper execution within each MLB, so the overall security from these two approaches is an exponential function of the number of MLBs, as shown in Eqn. 1:

$$S_1(n, m, K) = \prod_{j=1}^{K} 2^n \times m_j! \tag{1}$$

This expression assumes that the size of the schedule table may differ between MLBs, but instruction widths are constant.

Meanwhile, the security provided by ICR depends not only on the number of MLBs, but also on their particular interconnect configuration, variations in which will result in different mappings and different power/performance/area tradeoffs. To demonstrate this, consider K identical, fully connected MLBs, and an application which is perfectly parallelizable. For each application mapping, the total number of possible *placements* is $K!$ because for any given mapping, a particular subgraph of the original CDFG may be placed into any MLB, and only the routing, encoded within the instructions, needs to be updated to match. In other words, the fully connected network of identical MLBs is isomorphic, and therefore the placement algorithm is free to assign any subgraph to any MLB. For the purpose of security, the isomorphism is not ideal, because the overlay bitfiles will not differ significantly.

However, other mappings based on different interconnections are possible. This will change not just the routing portion of the instructions, but also the application mapping itself. For example, in Fig. 3(e), there exists a path from $M1 \leftrightarrow M3 \rightarrow M2 \leftrightarrow M0$. Compared with an equivalent mapping on Fig. 3(d), removal of the $M1 \rightarrow M0$ adjacency will either change the mapping entirely, or modify it by using $M3$ and $M2$ to pass required data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2679183, IEEE Embedded Systems Letters

Given that different interconnect configurations will result in different mappings, ICR has profound implications for system security through overlay diversification, as long as there exists at least one functionally correct mapping for the various ICR-based interconnect configurations. If this is true, then the total number of possible mappings would be equal to the number of interconnect configurations for K MLBs. Computing this is nontrivial, but is given by $S_2(K) = A[K]$, where A is the OEIS sequence A035512 [11], the 13th term of which is roughly 2^{123}. We assume the digraph is unlabeled, because as with the example of K fully connected MLBs, isomorphic configurations do not contribute significantly to security.

In fact, it can be shown that for every interconnect configuration, there exists at least one functionally correct application mapping, given that the particular configuration satisfies the requirements for a strongly connected digraph, and the per-MLB schedule and LUT memory size constraints are relaxed. By extension, if the interconnect is only weakly connected, this holds as long as there exists a path from the MLB processing the CDFGs primary input (PI) to the MLB processing its primary output (PO), as shown in Fig. 3(f) and (g).

To prove this, we first consider the case of one MLB ($K = 1$). By definition, a single MLB is a connected graph, and with sufficient schedule table and lookup table memory, the entire application can be mapped into the single MLB. For $K > 1$, it follows that either 1) a portion of the application can be parallelized, or 2) that the application is implemented in a pipeline fashion. In the first case, the particular subgraphs can be mapped to any available MLB, as long as partial or intermediate results may be communicated between any two MLBs (even over multiple cycles), which is true if the network is strongly connected. If instead multiple MLBs are used for pipelining, then the application can be divided into sequential subgraphs, each of which can be placed in adjacent MLBs along the direction of the given edge. Thus, pipelining requires only a weakly connected network of MLBs with an extant path from PI to PO. Therefore, regardless of the application properties, there is at least one functionally correct mapping for every interconnect configuration.

B. Security Tradeoffs in Design Mapping

From Eqn. 1, it follows that, for highly parallelizable applications, the overall level of brute force security may decrease as more MLBs are added when only ISR and IOR are used. This is because an increase in K will usually result in an overall reduction in the size of m, since a smaller schedule table will be needed when instructions are distributed among a greater number of MLBs, causing a reduction in brute force security for certain values of m and K. For example, $S_1(32, 56, 1) = 2^{281}$. Assuming that the application is not perfectly parallelizable, we can divide it into two MLBs, each of which has 30 (instead of 28) instructions. This gives us $S_1(32, 30, 2) = 2^{279}$. Similarly, ICR is effective against brute force only when $K \geq 14$; below that, the number of brute force attempts will be below 2^{128}. Also, in cases where an application is mapped to a network which is a subset of another, without ISR and IOR, the overlay bitfile will function on both devices. For example, the application mapped to Fig. 3(e) would function properly on Fig. 3(d), but the converse is not necessarily true, as the connections in Fig. 3(e) are a subset of those found in Fig. 3(d). However, combining the three approaches is multiplicative, since the connectivity is independent of the internal MLB security expressed in Eqn. 1; thus, combining these architectural modifications gives us

$$S_3(n, m, K) = S_1(n, m, K) \times S_2(K) \tag{2}$$

In short, the system security for small values of K requires both ISR and IOR, and the potentially reduced security from ISR and IOR for small values of m is partially compensated by the presence of ICR. Therefore, the proposed architectural modifications provide a powerful and versatile tool for security through diversification for the FPGA overlay.

C. Security against Side-Channel Attacks

The typical goal for a side channel attack is to obtain secret information, such as an encryption key, by carefully observing certain time-varying system properties, such as power consumption or electromagnetic radiation. The reason such attacks are effective is that these side channels inadvertently leak information because certain operations take more or less power, depending on if the bits involved are 1 or 0. By comparison, the overlay does not rely on operations with secret keys; instead, the particular modifications are encoded into the architecture of the overlay itself, and many aspects of the overlay (e.g. IOR) can be changed each time it is mapped, providing a moving target defense which makes the overlay approach highly effective against side-channel attacks.

V. Conclusion

We have presented a flexible FPGA overlay which is amenable to architectural diversity for the purpose of system security, making it more difficult for an attacker to reverse engineer intellectual property mapped to the overlay, and limiting bitstream tampering to single devices. The power, performance, and area overhead for the overlay are moderate, but provide provably robust protection against common attacks against FPGA bitstreams.

References

[1] P. Cooke *et al.*, "A Tradeoff Analysis of FPGAs, GPUs, and Multicores for Sliding-Window Applications," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 8, no. 1, pp. 2:1–2:24, Mar. 2015.

[2] W. Luis, G. R. Newell, and K. Alexander, "Differential Power Analysis Countermeasures for the Configuration of SRAM FPGAs," in *Military Communications Conference*, Oct 2015, pp. 1276–1283.

[3] J. Coole and G. Stitt, "Intermediate Fabrics: Virtual Architectures for Circuit Portability and Fast Placement and Routing," in *International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES/ISSS '10. ACM, 2010, pp. 13–22.

[4] J. Coole and G. Stitt, "Fast, Flexible High-Level Synthesis from OpenCL using Reconfiguration Contexts," *IEEE Micro*, vol. 34, no. 1, pp. 42–53, Jan 2014.

[5] J. Coole and G. Stitt, "Adjustable-Cost Overlays for Runtime Compilation," in *FCCM*, May 2015, pp. 21–24.

[6] S. Paul *et al.*, "MAHA: An Energy-Efficient Malleable Hardware Accelerator for Data-Intensive Applications," *IEEE TVLSI*, vol. 23, no. 6, pp. 1005–1016, September 2014.

[7] A. K. Jain, S. A. Fahmy, and D. L. Maskell, "Efficient Overlay Architecture Based on DSP Blocks," in *FCCM*. IEEE, 2015, pp. 25–28.

[8] D. Capalija and T. S. Abdelrahman, "A High-performance Overlay Architecture for Pipelined Execution of Data Flow Graphs," in *FPLA*. IEEE, 2013, pp. 1–8.

[9] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering Code-injection Attacks with Instruction-Set Randomization," in *CCS*. ACM, 2003, pp. 272–280.

[10] E. Dubrova, "A List of Maximum Period NLFSRs," *IACR Cryptology ePrint Archive*, vol. 2012, p. 166, 2012.

[11] "Number of Unlabeled Strongly Connected Digraphs with *n* Nodes." [Online]. Available: https://oeis.org/A035512