Interacting with Hardware Trojans Over a Network

Mohammed M. Farag, Lee W. Lerner, and Cameron D. Patterson
Cyber@VT
Bradley Department of Electrical and Computer Engineering
Blacksburg, VA 24061 U.S.A.
Email: {mmorsy, lwl, cdp}@vt.edu

Abstract—**Hardware Trojan horses (HTHs) are emerging threats to integrated circuits (ICs) outsourced to a global supply chain or developed with untrusted tools and intellectual property (IP). HTHs are stealthy in nature, and covert communication is their usual means of interaction and information transfer. Previous research has focused on short-range interaction via side-channels and existing IC interfaces, while remote interaction with HTHs across wired computer networks has received less attention. Generalized and non-local HTH interaction can support attacks normally associated with software Trojans. We investigate remote communication with HTHs and provide partial methods to exploit vulnerabilities in media layers of the protocol stack. Specifically, we focus on covert communication over point-to-point physical links in 10 gigabit Ethernet (10GbE) networks by exploiting loose specifications in physical- and link-layer protocols. The developed HTHs are assessed in terms of resource overhead and achieved bit rate, and demonstrate the potential for establishing high bandwidth covert channels using lightweight implanted circuits. We also describe a PUF-based IC or IP tracking attack enabled by HTH interaction across a network.**

I. Introduction

Fabrication and assembly of contemporary electronics are generally outsourced to a global supply chain. Even domestic development of modern systems is often assisted by third-party IP modules and commercial off-the-shelf (COTS) components to increase productivity and reduce design costs. High volume chip manufacturing cannot be considered trusted since ICs are vulnerable to tamper and change by untrusted parties throughout the design and fabrication process. Hardware Trojan horses (HTHs) are malicious IC inclusions or alterations to perform certain actions and functionalities not captured by design specifications [1].

HTH insertion and detection methods are emerging research topics that can partially leverage software Trojan horse (STH) ideas and experiences. However, software platform homogeneity and programmability provide a higher degree of transitive (non-direct) observability and controllability compared to HTHs, making STHs the preferred choice for attacks. For system-on-chip (SoC) ICs, development of programmable and transitive HTHs would support more powerful attacks by enhancing controllability and observability. In a HTH context, transitivity is a module's covert ability to relay information of interest from a module's input interface, with or without processing, to a certain destination linked to the module's output interface. Transitivity would extend HTH communication beyond the local node's environment.

The enabling mechanism for HTH transitivity is a means of interaction. HTHs are stealthy by nature and covert communication is their usual means of interaction. A covert channel can be defined as "an enforced, illicit signaling channel that allows a user to surreptitiously contravene the security policy and unobservability requirements of the system" [2]. System vulnerabilities leading to covert channels are a result of design oversights, weakness inherent in the system design, or underspecification of underlying protocols. System vulnerabilities must first be identified in order to provide a defense against covert data channels. Vulnerabilities caused by design oversights can be eliminated once they are discovered in the system design phase. On the other hand, it may not be possible to remove all potential weaknesses since doing so may lead to inefficient systems. Flexible protocol specifications enable application scalability even though loose specifications may lead to vulnerabilities that can be exploited to violate security specifications and establish covert channels.

We investigate development of HTHs supporting remote interaction over wired computer networks. Software-based covert communication in computer networks is an active research area examining how STHs can exploit software vulnerabilities in the protocol stack [3]. Development of HTHs exploiting network protocols vulnerabilities is an interesting new area to consider. However, it is also challenging due to the number and diversity of layers implemented with both hardware and software. Various layers of the protocol stack can be exploited to create covert data channels, but intended recipients may differ from one layer to another. To set up end-to-end (peer-to-peer) communication in different network topologies, a logical link between endpoints is established by employing a chain of physical links to a particular destination. The route to a particular destination is usually not fixed, and a logical link may be established with temporally varying physical links. Upper (host) layers of the protocol stack are typically connected to applications or users serving as the ultimate source and sink of commands and data.

On the other hand, covert data channels established with lower (media) layers of the protocol stack act as carriers across the endpoints of a physical link. Interactions between multiple HTHs along a logical link require connecting a chain of compromised physical links. Escalating covert information from a compromised lower layer to upper layers of the stack is another way of delivering covert data to intended recipients. Alteration between upper and lower layer covert channels might help to evade detection by specific defenses.

However, it may not be necessary to create an end-to-end covert channel because the attacker may be a man-in-the-middle eavesdropping on the network.

In this paper, we advance a lightweight HTH supporting two-way covert communication in point-to-point physical links by exploiting vulnerabilities in the underlying media layer protocols. Covert data channels are established by inserting HTHs exploiting unenforced, loose specifications of IP cores implementing media layer functionalities in a manner that maintains system operability. Specifically, we explore insertions in a PCS/PMA 10GBASE-X IP core implementing the physical layer functionalities of the 10GbE protocol specified in the IEEE 802.3-2008 standard [4], and in an Aurora IP core implementing the link-layer functionalities in a high-speed serial protocol. Aurora is an open source implementation of a link-layer protocol developed by Xilinx to support serial links between chips employing multi-gigabit transceivers (MGTs), with the protocol specifications adopted from the IEEE 802.3 standard [5].

The remainder of this paper is organized as follows: Section II provides a brief overview of HTH- and STH-enabled covert communication. Section III describes how HTH interactions could enable remote tracking of specific chips deployed across a computer network. HTHs supporting point-to-point covert communication in 10GbE networks are described in Section IV, while Section V focuses on HTHs targeting the Aurora protocol. Finally, Section VI provides a summary and conclusions.

II. Trojan-enabled Covert Communication

Previous research addressing covert communication using HTHs has emphasized use of hidden modules and structures to leak sensitive information from a chip. Common underlying assumptions are that a HTH can be added to critical components on the chip such as cryptographic modules, an inability to detect the HTH with existing analysis techniques, and the attacker has local access to the compromised IC. HTHs of this type commonly employ electromagnetic radiation via hidden on-chip antennas, power, or temperature side-channels to encode and leak sensitive information off a chip. Such HTHs transmit covert data over a short range, limited by the hidden nature of the Trojan, such that an attacker residing close to the compromised IC can receive and decode leaked information. Karri et al. presented several examples of covert communication using power and temperature side-channels in their Trojan taxonomy [6]. Most of the existing covert communication techniques adopting HTHs and side-channels are localized in the sense that an attacker needs physical proximity or access to the compromised device to exploit information obtained by the embedded HTH.

Adding HTHs to input/output subsystems of a compromised IC provides another means to establish covert channels. Information is covertly transferred via existing interfaces and peripherals by hiding extra data in legitimate communication and interface protocols. For example, HTHs can support covert communication by changing signaling specifications of existing hardware interfaces. Characteristics such as phase, rate, or sequencing can be modulated in unconventional way to encode covert data along with legitimate data. To evade detection by evaluations and typical run-time defenses, covert data transferred using a shared medium should mask itself in signaling or data that is generally ignored by the designer. Attributes characterizing legitimate communication should not be significantly affected by channel sharing. For example, the data rate of a legitimate interface should not be reduced by the insertion of covert information.

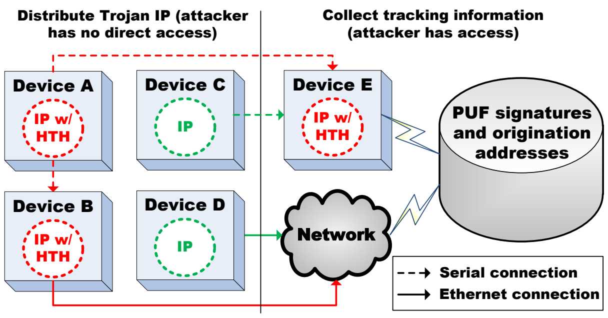
The Embedded Systems Challenge competition demonstrates leaking sensitive information from a BASYS FPGA board by changing the RS232 serial interface signaling specifications [7], [8]. Sun et al. developed a pin hijacking transceiver module exploiting idle or dead time in an I^2C interface to create two-way communication between an FPGA chip and a serial memory device [9]. They also described another covert channel encoding data in a DDR2 memory interface's phase delay attribute, which is normally used for calibration purposes. HTHs exploiting existing interfaces for covert communication are normally limited to inter-chip interactions, and we are not aware of HTHs supporting covert communication across computer networks.

Covert communication in computer networks uses protocols such as TCP/IP for information transfer instead of the payload data used in steganography. The vast amount of data, large number of existing protocols, and the ease of eavesdropping on computer networks provide many opportunities for high bandwidth covert communication. Zander et al. surveyed a number of software-based covert channels in network protocols, and possible countermeasures [10]. Covert channels may be classified as storage channels involving a shared medium accessed by both the transmitter and receiver, and timing channels involving modulation of certain characteristics.

Various protocols and layers have been exploited by STHs to create covert channels over networks. Examples of such exploits include unused header bits, header extensions and padding, TCP initialization sequences, IP identification and framing offset, checksum fields, and many other vulnerabilities. For example, a TCP initial sequence number is used to coordinate between transmitter and receiver, and may be optionally selected by the client according to loose rules. As reported in [10], a covert channel may encode information in this field while maintaining a uniform data distribution. Previous research addressing covert channels in computer networks has emphasized software methods, with less attention to hardware exploits and countermeasures.

III. Example Attack Scenario

In this section we present an attack scenario to illustrate communicating useful information with HTHs. Our attack makes use of Physically Unclonable Functions (PUFs), which when challenged provide identifier responses unique to the devices on which they are implemented due to subtle variations in the fabrication process [11]. PUFs have been proposed for device and IP authentication across an untrusted supply



Distribute Trojan IP (attacker has no direct access)
Collect tracking information (attacker has access)
Device A
IP w/ HTH
Device C
IP
Device E
IP w/ HTH
PUF signatures and origination addresses
Device B
IP w/ HTH
Device D
IP
Network
Serial connection
Ethernet connection

Fig. 1. Trojan IP system tracking attack scenario.

chain [12]. It has also been suggested that PUF-sourced unique identifiers might be useful for device tracking [13]. Building on this idea, our attack utilizes PUFs to perform reconnaissance on target systems post-deployment.

Fig. 1 illustrates conceptually how a PUF with additional Trojan circuitry inserted into a third-party IP module might be used to track systems. IP cores must often be obtained from several vendors because both hard (e.g. process-tailored PHYs) and soft (e.g. link-layer netlist) blocks are needed. After this point, the attacker may have no visibility into which systems use the IP. However, this attack does not require PUF signatures to be linked to devices beforehand as the attacker would still likely be able to generally target a specific project and may be able to infer system information from recovered PUF identifier information when the system comes online. A PUF is particularly useful for this attack because it enables a single Trojan IP core to be implemented on any number of devices while providing unique information for each. In Fig. 1, devices A, B, and E make use of the Trojan IP while devices C and D do not. The attack is considered successful when the HTH communicates its PUF-created unique identifier, and perhaps an origination address copied from observed data packets, back to the attacker. The attacker must ultimately have some level of observability over a part of the cyber system in order to recover HTH communications.

This paper develops possible ways for the distributed Trojan IP to communicate tracking information to the attacker. Section IV explores point-to-point covert data channels created over Ethernet links, illustrated in Fig. 1 as the connection between device B and the computer network. This enables tracking information to be recovered by eavesdropping on the network with which the system communicates. More deeply embedded HTHs may require intra- or inter-device point-to-point links to propagate information out of a system. Section V therefore explores covert channels over serial links such as the connection between devices A and E. This also enables tracking information to be recovered by monitoring a device in a HTH chain. The attacker can leverage access to device E to recover tracking information sourced from device A.

IV. HTH Interaction Across 10GbE Physical Links

We first consider how to propagate covert information across 10GbE links, enabling a PUF signature to be transmitted from a device to a network. 10GbE offers a more efficient and less expensive approach to moving data on backbone connections between network switches while also providing a consistent technology end-to-end. It uses the IEEE 802.3 MAC sublayer, connected through a 10 gigabit media independent interface (XGMII) to the 10GBASE physical layer entity specified in IEEE 802.3 clause 48. The physical layer of 10GbE consists of the physical coding sublayer (PCS), the physical medium attachment (PMA), and the physical medium dependent (PMD) sublayers [4].

10GBASE-X is a serial interface IP block implementing the PCS and PMA functionalities between the XGMII MAC and the PHY layers. The 10GBASE-X PCS maps XGMII data and control characters to/from a stream of code groups according to an 8B/10B transmission code. The PCS is responsible for data encoding/decoding, lane synchronization and alignment, conversion of XGMII idle control characters to/from a randomized sequence of code groups, and PHY clock rate compensation achieved by embedding special non-data code groups in the idle stream. Clock recovery and serializing / deserializing data are performed in the PMA. The PMD layer consists of four lanes employing high speed serial transceivers running at 3.125 GHz. Fig. 2 illustrates the block diagram of the PCS/PMA 10GBASE-X IP core.

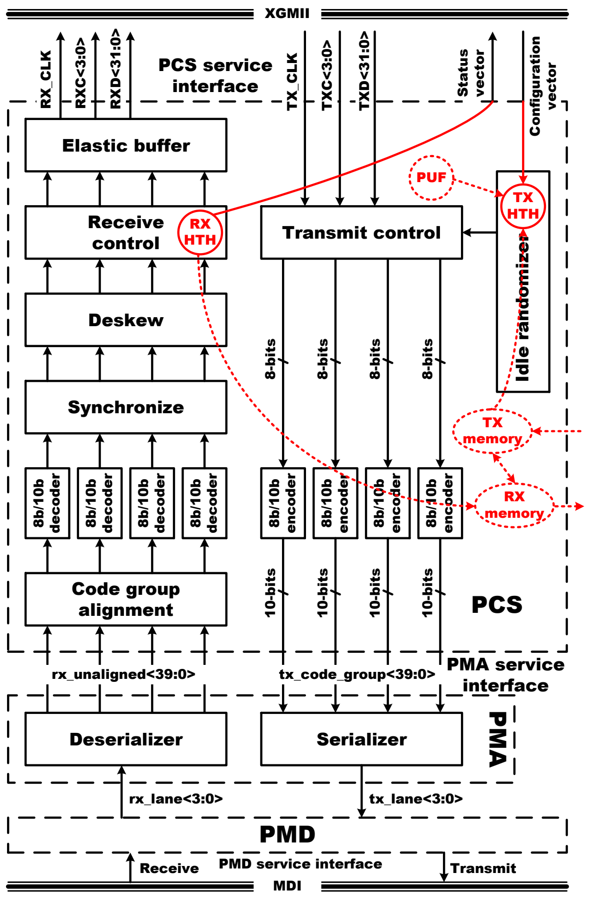


Fig. 2. PCS/PMA 10GBASE-X IP core.

The PCS transmit process continuously generates code groups based upon transmit data (TXD) and control (TXC) signals, while the PCS receive process continuously accepts code groups from the PMA service interface and generates receive data (RXD) and control (RXC) on the XGMII. All received idle code groups are replaced with idle characters before forwarding to the XGMII. The 8B/10B transmission code as well as the rules by which the PCS encode and decode code groups are specified in IEEE 802.3 clause 36. 10GBASE-X PCS ordered sets consists of combinations of special and data code groups of length four beginning in lane 0. The PCS defines special code groups for control purposes, and provides capabilities such as synchronization, deskew, and error detection.

A. Idle Sequence-based Covert Channels

In this section, we sketch HTHs supporting covert communication in 10GbE physical links by changing signaling specifications of the underlying PCS layer. Idle ordered sets ||I|| are transmitted in full columns whenever the XGMII is idle. The idle sequence (ISQ) provides a continuous fill pattern to establish and maintain lane synchronization, perform lane-to-lane deskew, and achieve PHY clock rate compensation. An ||I|| sequence consists of one or more consecutive sync_column ||K||, skip_column ||R||, or align_column ||A|| ordered sets. Some of the rules governing ||I|| ordered set sequencing are:

- Each ||A|| is sent after r non-||A|| columns where r is a randomly distributed number between 16 and 31.
- When not sending an ||A||, either ||K|| or ||R|| is sent with a random uniform distribution between the two.

Both ||A|| spacing as well as ||K||, ||R||, or ||A|| selection are based on a random integer r generated by a PCS idle randomizer employing a pseudo-random binary sequence generator. We exploit the ISQ ordered sets to encode covert data between the endpoints of the physical link by adding a HTH to the standard idle randomizer. To satisfy ISQ constraints, only ||K|| and ||R|| ordered sets are used to encode data as shown in Fig. 3. Spacing between consecutive ||A||'s is fixed at 32 characters, which does not degrade lane synchronization or clock compensation and only slightly decreases lane-to-lane deskew robustness. Other techniques such as Huffman encoding enable uniformly distributed covert data using the three ordered sets to increase entropy.

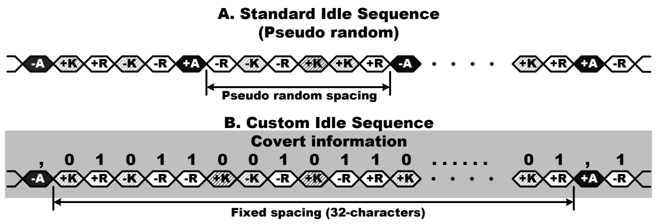

A. Standard Idle Sequence
(Pseudo random)
-A +K +R -K -R +A -R -K -R +K +K +R -A . . . . +K +R +A -R
Pseudo random spacing
B. Custom Idle Sequence
Covert information
, 0 1 0 1 1 0 0 1 0 1 1 0 ...... 0 1 , 1
-A +K +R -K -R -R +K -K -R +K -R +R +K . . . . +K +R +A -R
Fixed spacing (32-characters)

Fig. 3. Standard and custom idle sequence timing diagram.

As shown in Fig. 2, the TX HTH is a binary encoder instantiated inside the PCS idle randomizer, and the RX HTH is a binary decoder instantiated inside the receive control module. The TX and RX HTHs could be selectively enabled by a rarely occurring data sequence in the transmit and receive directions. In our example scenario, a tracking PUF provides the TX HTH with the IP or device signature to be sent over the ISQ covert channel. Configuration and status vectors are possible interfaces for covert information flowing to and from the core. Portions of these interfaces may be used in the development and test phases while kept idle during normal operation. These interfaces could be employed by the TX and RX HTHs to forward covert data to upper layers of the protocol stack. Store and forward techniques using hidden memory modules might also also be used. Table I shows the incremental resource utilization of the HTHs on an FPGA.

TABLE I
XILINX VIRTEX-7 FPGA RESOURCE UTILIZATION FOR HTHS IN A 10GBASE-X IP CORE (% OF PCS/PMA CORE).

	# Regs	**# LUTs**
TX HTH	16 (0.7%)	13 (0.5%)
RX HTH	5 (0.23%)	5 (0.2%)
PCS/PMA IP	2146	2473

V. HTH INTERACTION IN MULTI-GIGABIT TRANSCEIVERS

We now consider how to propagate covert information across high-speed serial links, enabling a PUF signature to be transmitted between devices. MGTs are the preferred means of transferring high-speed data between ICs since the sender and receiver do not share a global or transmitted clock signal. Parallel data words are serialized by the MGT transmitter, and the receiver performs the reverse function. Low voltage current mode logic supports signaling rates up to 28 Gbps over a single differential pair of conductors. Compared to parallel data transfer, MGTs reduce pin count, electromagnetic interference, ground bounce due to simultaneous switching outputs, and power. An MGT integrates clock, data, and control in a single bitstream transferred over a point-to-point serial link [14].

Aurora 8B/10B is a simple example of a link-layer protocol defining the packet structure, communication channel initialization and validation, error handling, and clock compensation [5]. Fig. 4 shows the top-level block diagram of the Aurora IP core, including a brief description of individual modules. Aurora shares characteristics and basic functionalities of other media layers such as 10GBASE-X PCS and XAUI [15]. Aurora protocol specifications are drawn from the IEEE 802.3 standard. Fig. 4 depicts insertion points and top-level interfaces of the developed HTHs. The HTHs are evaluated in terms of performance measured by covert communication bit rate, cost estimated by resources usage, productivity assessed by implementation difficulty, and impact on the main communication channel.

A. Clock Correction-based Covert Channels

The tight jitter requirements on a transmission clock prevent an MGT from using the recovered clock as a reference, and

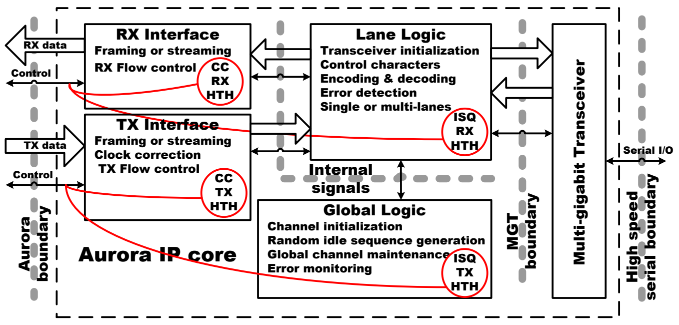


Fig. 4. Aurora IP core structure.

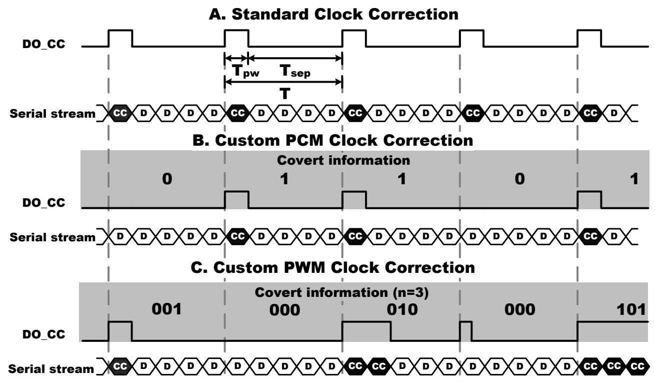


Fig. 5. Standard, PCM, and PWM clock correction timing diagram.

communicating MGTs usually have different oscillators. Small deviations between the transmitter and receiver clock frequencies result in either overflow or underflow of the receiver FIFO. Most MGTs tolerate slight differences in oscillator frequencies with built-in clock correction (CC) compensation that uses a unique symbol or sequence of symbols not found elsewhere in the data stream. On the transmitter side, the Aurora protocol implements CC by periodically inserting sequences of multiple /CC/ control characters into idle patterns or user data. On the receiver side, the PCS looks for and drops the next CC sequence if the FIFO is getting close to full. If the FIFO is getting close to empty, the PCS writes the next CC sequence twice into the FIFO [5]. Aurora uses a CC control module respecting the following rules:

- CC sequences should last at least two cycles to ensure they are recognized by the receiver.
- Duration and period should be precisely assigned to correct for the difference between oscillators frequencies.
- The minimum separation time between consecutive CC sequences is eight clock cycles.

CC can source covert information with sequence variation instead of standard periodic initiation. The CC is performed by asserting a DO_CC control signal which stalls data and relays a sequence of repeated /CC/ characters. Different pulse modulation techniques can be used for covert data encoding. We employ pulse code modulation (PCM) and pulse width modulation (PWM) to encode data in the CC, as illustrated in Fig. 5. In a PCM CC cycle, asserting DO_CC for a certain period denotes a logic 1 while releasing it denotes a logic 0. In a PWM CC cycle, n bits of data can be encoded in the DO_CC signal hold time (pulse width). The CC TX HTH is a custom encoder driving the DO_CC signal as shown by Fig. 4. The CC TX HTH is a custom decoder instantiated inside the Aurora RX interface module to detect received CC sequences carrying covert information.

A CC-based TX HTH reduces the data rate of the main channel. The bit rate R is the number of data bits transferred per second via the covert channel, while the channel utilization ratio U is a function of the CC pulse width T_{PW}, period T, and occurrence probability P_{CC}:

$$R = \frac{\text{\# bits encoded per CC cycle}}{\text{CC period}} \tag{1}$$

$$U = P_{CC} \cdot T_{PW}/T \tag{2}$$

As shown by Fig. 5(B), the CC TX HTH encodes a single bit per CC cycle, where the CC period and pulse width are constant values. The CC probability is the likelihood of a 1-bit occurring in the covert data, or 0.5 in a DC-balanced data stream. The maximum bit rate and channel utilization are determined using CC constraints on the pulse width and period:

$$R_{PCM} = 1/(T_{pw} + T_{sep}) \tag{3}$$

$$U_{PCM} = 0.5 \cdot T_{pw}/(T_{pw} + T_{sep}) \tag{4}$$

$$R_{PCM\_max} = f_{ref\_clk}/10 \tag{5}$$

$$U_{PCM\_max} = 10\% \tag{6}$$

The PWM TX module encodes n bits of data by asserting the DO_CC signal for a variable number of clock cycles, and fixing the CC period as shown in Fig. 5(C). The CC period is arbitrarily assigned to a value larger than the maximum pulse width. When the period is twice the maximum pulse width, the maximum bit rate and channel utilization are:

$$R_{PWM\_max} = n/(2^{n+1}) \cdot f_{ref\_clk} \tag{7}$$

$$U_{PWM\_max} = 25\% \tag{8}$$

CC-based covert communication is experimentally validated using the RocketIO transceivers on a Xilinx Virtex-5 FX130T FPGA. The Xilinx CoreGen tool provides HDL for the Aurora core. Two MGTs running at 2.5 Gbps line rate and 125 MHz reference clock frequency are connected on a custom MGT interface board attached to a Xilinx ML510 evaluation platform. The maximum bit rates of the PCM and the PWM side-channel communication are 12.5 and 23.43 Mbps (for n=3), respectively, at 10% and 25% channel utilization. Moreover, the bit rate and channel utilization are parametric functions with tunable design parameters that can be adjusted to achieve significant data rates at channel utilizations still satisfying the

TABLE II
XC5VFX130T FPGA RESOURCE UTILIZATION FOR HTHS IN MULTI-GIGABIT TRANSCEIVERS (% OF AURORA CORE).

	# Regs	**# LUTs**
PCM HTH (TX, RX)	24, 20 (1.4%, 1.2%)	27, 31 (1.4%, 1.6%)
PWM HTH (TX, RX)	22, 17 (1.3%, 1%)	34, 29 (1.8%, 1.5%)
ISQ HTH (TX, RX)	16, 5 (1%, 0.3%)	13, 5 (0.7%, 0.3%)
Aurora IP	1630	1870

covert requirement. Established pulse modulation and digital encoding practices simplify the design effort.

B. Idle Sequence-based Covert Channels

MGTs must incorporate a number of functions to permit high line rates. ISQs are ordered sets of control characters used to perform word boundary alignment and channel bonding during initialization. During operation, ISQs are inserted during wait states to keep the channel active. The Aurora protocol's ISQ uses /A/, /K/, and /R/ control characters applied in a pseudo-random sequence subject to the Aurora constraints drawn from IEEE 802.3 [5]. We change the signaling specification of the Aurora protocol to encode covert information in ISQs in a manner similar to Section IV.

The ISQ TX HTH is a binary encoder instantiated inside the Aurora core and the ISQ RX HTH is a binary decoder instantiated inside the Aurora lane logic module. ISQ TX HTH encodes binary data in /K/ and /R/ characters between separating /A/ characters as illustrated by Fig. 4. The ISQ TX HTH bit rate depends on the number of bytes per lane, where every byte corresponds to an idle character. The covert channel bandwidth B is a function of the achieved bit rate and the MGT wait state time percentage $I_{\%}$:

$$R_{ISQ} = 31/32 \cdot \text{Bytes/lane} \cdot f_{ref\_clk} \tag{9}$$

$$B_{ISQ} = \text{Bytes/lane} \cdot f_{ref\_clk} \cdot I_{\%} \tag{10}$$

In a test with a 125 MHz reference clock frequency, 4-byte lane width, and assuming 10% idle time, the covert channel communication bit rate is 600 Mbps while the bandwidth is 60 Mbps without decreasing the main channel bandwidth. Table II shows the absolute and relative resource utilization of TX and RX HTHs.

VI. CONCLUSIONS AND FUTURE WORK

Computer networks are a natural medium to investigate non-local interaction with HTHs. We introduced novel HTHs for covert communication in point-to-point 10GbE and high-speed serial links by exploiting loose specifications in the IP core controlling the physical communication medium. A detailed evaluation and experimental testing of the developed HTHs was presented for MGTs controlled by the Aurora link-layer protocol. We also described how HTH interaction could enable remote tracking of ICs over a wired network. To enable the proposed PUF attack, we are developing methods that do not require support from host layers. All covert information insertion, communication, and retrieval can take place entirely within network adapters and switches.

Thwarting covert interactions exploiting interface under-specification is another focus of our future work. System and module specifications can be necessarily loose to enhance portability or enable certain applications such as autonomic computing. As shown in this paper, such ambiguities can be leveraged to violate system-level security policies while maintaining legal operation as defined by design specifications. The developed HTHs also illustrate security threats arising from the use of third-party IP to assemble computing platforms. Verification of functional specifications is not always a complete method of evaluating and providing system security, and even specifications with rigorous provisions to enhance security may be insufficient to anticipate all possible attacks. We are investigating use of run-time monitoring and enforcement guards to prevent covert communication through underspecified interfaces.

REFERENCES

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *Design Test of Computers, IEEE*, vol. 27, no. 1, pp. 10–25, Jan–Feb 2010.

[2] The Common Criteria Recognition Arrangement, "Common Criteria for Information Technology Security Evaluation," Common Criiteria, Tech. Rep., Sep 2006.

[3] I. S. Moskowitz, R. E. Newman, D. P. Crepeau, and A. R. Miller, "Covert channels and anonymizing networks," in *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, ser. WPES'03. ACM, 2003, pp. 79–88.

[4] "IEEE standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements Part 3: CSMA/CD access method and physical layer specifications," *IEEE Std 802.3-2008*, pp. 1–586, 2008.

[5] *Aurora 8B/10B Protocol Specification*, Xilinx, Apr 2010.

[6] R. Karri, J. Rajendran, and K. Rosenfeld, "Trojan taxonomy," in *Introduction to Hardware Security and Trust*. Springer New York, 2012, pp. 325–338.

[7] Y. Jin, N. Kupp, and Y. Makris, "Experiences in hardware trojan design and implementation," in *Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust*. IEEE Computer Society, 2009, pp. 50–57.

[8] A. Baumgarten, M. Steffen, M. Clausman, and J. Zambreno, "A case study in hardware trojan design and implementation," *International Journal of Information Security*, vol. 10, pp. 1–14, 2011.

[9] J. Sun, R. Bittner, and K. Eguro, "FPGA side-channel receivers," in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA'11. ACM, 2011, pp. 267–276.

[10] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *Communications Surveys Tutorials, IEEE*, vol. 9, no. 3, pp. 44–57, 2007.

[11] E. Simpson and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, ser. LNCS. Springer, 2006, vol. 4249, pp. 311–323.

[12] J. Graf and J. Hallman, "Trust in the FPGA supply chain using physically unclonable functions," in *The* 35^{th} *Annual Government Microcircuit Applications & Critical Technology Conference (GOMACTech)*, 2010.

[13] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ser. LNCS. Springer, 2007, vol. 4727, pp. 63–80.

[14] A. Athavale and C. Christensen, *High-Speed Serial I/O Made Simple*. Xilinx, 2005.

[15] P. Noel, F. Zarkeshvari, and T. Kwasniewski, "Recent advances in high-speed serial I/O trends, standards and techniques," in *Electrical and Computer Engineering, 2005. Canadian Conference on*, May 2005, pp. 1292–1295.