Journal of Cryptographic Engineering (2023) 13:485–499
https://doi.org/10.1007/s13389-023-00323-3

REGULAR PAPER

Trojan awakener: detecting dormant malicious hardware using laser logic state imaging (extended version)

Thilo Krachenfels[1] · Jean-Pierre Seifert[1,2] · Shahin Tajik[3]

Received: 13 May 2022 / Accepted: 26 April 2023 / Published online: 29 May 2023

Abstract
The threat of (HTs) and their detection is a widely studied field. While the effort for inserting a Trojan into an (ASIC) can be considered relatively high, especially when trusting the chip manufacturer, programmable hardware is vulnerable to Trojan insertion even after the product has been shipped or during usage. At the same time, detecting dormant HTs with small or zero-overhead triggers and payloads on these platforms is still a challenging task, as the Trojan might not get activated during the chip verification using logical testing or physical measurements. In this work, we present a novel Trojan detection approach based on a technique known from (IC) failure analysis, capable of detecting virtually all classes of dormant Trojans. Using (LLSI), we show how supply voltage modulations can awaken inactive Trojans, making them detectable using laser voltage imaging techniques. Therefore, our technique does not require triggering the Trojan. To support our claims, we present three case studies on 28 nm and 20 nm SRAM- and flash-based (FPGAs). We demonstrate how to detect with high confidence small changes in sequential and combinatorial logic as well as in the routing configuration of FPGAs in a non-invasive manner. Finally, we discuss the practical applicability of our approach on dormant analog Trojans in ASICs.

Keywords Hardware security · Hardware Trojans · Optical side-channels · Hardware snapshots · LLSI

1 Introduction

Due to their reconfigurability, high performance, and a short time to market, programmable hardware, especially (FPGAs), has become the default solution in many fields. One of the main strengths of FPGAs compared with (ASICs) is that the hardware configuration can be updated and even reprogrammed during runtime. At the same time, the demand for security increases as more and more security-critical systems are based on electronics. Therefore, malicious modifications of the design, referred to as (HTs), endanger the security of many applications. On FPGAs, a Trojan might be inserted after manufacturing and testing, i.e., in the untrusted field [1, 2], for instance, by altering the entire configuration (known as bitstream) or via partial reconfiguration. Particularly if the chip foundry can be trusted, this depicts a much more powerful threat model than for ASICs. Most security-critical FPGAs rely on bitstream encryption and authentication to avoid such Trojan insertions. However, these protection schemes have shown to be vulnerable to various physical [3–6] and mathematical attacks [7], leaving them susceptible to tampering. Consequently, in critical applications, where the chip is deployed in an untrusted field or could be accessed by untrusted parties, it should be possible to check the integrity of the hardware.

Integrity checking of running applications on FPGAs in the field faces mainly two obstacles. First, while checking the configuration against a golden bitstream would reveal tampering (as proposed in [8]), it is not possible in many cases. In several defense/aerospace applications, where flash-based FPGAs [9] or SRAM-based FPGAs with preemptive decryption key zeroization [10] are deployed, no bitstream (encrypted or unencrypted) is available to the hardware testing engineer in the field for verification. In these cases, the configuration is stored inside the chip and bitstream readback

✉ Thilo Krachenfels
tkrachenfels@sect.tu-berlin.de

Jean-Pierre Seifert
jpseifert@sect.tu-berlin.de

Shahin Tajik
stajik@wpi.edu

1 Chair for Security in Telecommunications, Technische Universität Berlin, Berlin, Germany

2 Fraunhofer SIT, Darmstadt, Germany

3 Worcester Polytechnic Institute, Worcester, USA

is not possible. Even if the bitstream is available, analyzing the unencrypted bitstream is not an option since the circuit and the secret keys for bitstream decryption should be unknown even to the testing engineer. Moreover, the same bitstream can be encrypted with various keys for different FPGAs, and therefore, comparing encrypted bitstreams to each other for tampering detection might also not be feasible.

Second, while early HTs had logic triggers that could be activated by logical testing [11] under some circumstances, recently proposed HTs are classified as *stealthy* or *dormant*. In other words, the Trojan payload reacts only under extremely rare conditions, for instance, in a particular temperature, supply voltage, or frequency range [12] or after a certain amount of specific events have occurred [13]. Furthermore, under operational and testing conditions, a dormant Trojan tries to hide from physical inspection or side-channel analysis, e.g., by leveraging analog components [13], manipulating only the dopant level of the chip [14], or changing only the routing configuration on programmable hardware [12].

Several approaches based on side-channel analysis (SCA) for detecting such dormant HTs have been proposed in the literature [15–22]. However, they all face severe limitations regarding resolution and the capability to detect all types of HTs. For instance, approaches using electromagnetic (EM) backscattering side-channels are naturally limited by their resolution and can only detect larger malicious design changes [18, 19]. Furthermore, these approaches can reliably detect dormant Trojans only with a high rate of false positives. One technique that provides higher resolution is optical probing, where the chip is scanned through its backside with a laser, and the reflected light is analyzed. However, the reported approach based on electro-optical frequency mapping (EOFM) [21] is limited to detecting malicious modifications only in the sequential logic, and thus, Trojans that solely consist of combinatorial logic stay undetected.

A new optical probing technique that has recently been leveraged in the hardware security field is called *laser logic state imaging (LLSI)* [23]. It is an optical probing technique that can extract the logic states of single transistors, and therefore, more complex logic gates or memory cells [24]. In LLSI, the chip's supply voltage is modulated, which causes the light reflection originating from a laser scanning irradiation to be modulated as well. The modulation amplitude is dependent on the carrier concentration present in the silicon, for instance, inside the channel of a transistor. Consequently, the LLSI signal is highly data-dependent and provides a practically unlimited number of electro-optical probes. Hence, it should be possible to extract the configuration of an FPGA's logic fabric using LLSI, especially because the configuration is held in memory cells distributed over the chip. The logic state of these cells controls the functioning of (LUTs), multiplexers (MUXes), and pass transistors in switch boxes. In this work, we try to clarify *if small dormant HTs on state-of-the-art FPGAs—consisting of combinatorial or sequential logic—can be detected by applying LLSI.*

Our contribution We indeed positively answer the above question. First, we present how LLSI allows us to capture the state of every transistor of the logic fabrics of SRAM- and flash-based FPGAs. Based on this, we demonstrate how to partially reverse-engineer the FPGA's configuration, including the detection of changes in a single LUT. Second, we show how this new approach can detect small and dormant HTs on FPGAs. Stimulating all transistors with the power supply modulation awakens maliciously modified hardware, from which we then can take a snapshot. Therefore, the Trojan can be inactive/dormant, as our approach does not rely on any switching activity on the chip. For detecting HTs, we first capture a reference snapshot of the FPGA's logic fabric in the trusted field—when the design is known to be Trojan-free. Later, to check if the design has been altered, we capture a snapshot of the logic fabric and compare it to the reference. We show that the high resolution of optical probing allows detecting small changes of the configuration, down to changes in a single combinatorial gate.

Our approach can be applied non-invasively since almost all current FPGAs are available in flip-chip packages allowing easy access to the silicon backside. To validate our claims, we present three case studies on SRAM- and flash-based FPGAs from Xilinx (28 nm and 20 nm technology) and Microchip (28 nm technology), respectively. Although our experiments are focused on FPGAs, we discuss why LLSI is applicable for analog HT detection on ASICs.

Remarks on the extended version The original version of this work has been presented in [25]. The version at hand contains the following additional and revised content: (i) the investigation of a new target device manufactured in a 20 nm technology, including setup, results, and discussions; (ii) a more thorough explanation and discussion of the experimental setup, especially regarding the LLSI modulation frequencies; (iii) a detailed discussion of how to prepare a real-world device that should be investigated using the presented HT detection approach; and (iv) additional figures depicting the experimental setup.

2 Background

2.1 Hardware Trojans

2.1.1 Properties and taxonomy

The term hardware Trojan (HT) includes a wide range of malicious circuit modifications which, for instance, try to

leak sensitive information through side-channels, implement kill-switches and backdoors, or enforce faulty computations. HTs can be characterized by their physical properties (e.g., type and size of modifications), activation characteristics (i.e., trigger source and frequency), and action characteristics (i.e., which goal the HT serves) [26]. As diverse as the different types of HTs are, so are the potential entities that might introduce the malicious modifications [27]. During the development and production of (ICs), weak points include third-party intellectual property (IP) cores, malicious design tools, and mask layout or doping concentration modifications [28] by untrusted foundries. The platform TrustHub [29] provides several design-level HT benchmarks, primarily available as gate-level descriptions. TrustHub provides access to the automatically generated HT benchmarks presented in [30] that alter existing circuit designs by inserting malicious logic gates.

Programmable hardware devices, like FPGAs, are less prone to production-based HT insertion than ASICs. On the other hand, due to their reconfigurability, they provide the possibility for malicious modifications even after the product has been shipped to the user. It has been shown that the key used for encrypting the bitstream on recent SRAM-based FPGAs can be extracted using SCA techniques [3–6]. With the extracted key at hand, the bitstream can be decrypted, modified, and stored as a replacement for the original bitstream [17]. Although bitstream extraction from flash-based FPGAs might not be possible, the adversary could still be able to reprogram certain parts of the configuration or even replace the entire chip containing her malicious version of the design.

2.1.2 Hardware Trojans on FPGAs

While generic Trojans, such as backdoors, can be implemented on both ASICs and programmable hardware, a few HTs especially tailored to FPGAs have been proposed. For instance, Jacob et al. have proposed an approach that exploits shared resources between the programmable logic and the embedded microcontroller on an FPGAs system on a chip (SoC) [31, 32]. By hidden functionalities in an IP design block, the programmable logic can access and manipulate shared memory locations used for storing sensitive information like cryptographic keys. Ender et al. have proposed a Trojan that is solely based on minor timing modifications on the chip [12]. They show that by operating the chip with modified signal paths at a specific frequency, the data masking scheme protecting against side-channel analysis attacks is not functional anymore, allowing the extraction of the secret key used in the protected algorithm. They show that on an FPGA, longer signal paths can be realized by instantiating route-thru LUTs, or by modifying the routing in the switch boxes, which results in zero overhead in resource usage, and therefore, is hard to detect. In another effort, Roy et al. [2] showed that the reconfigurable LUTs could be exploited to realize HTs with zero payload overheads. Finally, Ng et al. [1] demonstrated that integrated sensors inside FPGAs could be deployed as Trojan triggers.

2.1.3 Detection of hardware Trojans—related work

As already mentioned in Sect. 1, HT detection on FPGAs cannot always be carried out by checking or comparing bitstreams. Therefore, most of the HT detection techniques use different kinds of physical measurements and side-channel information obtained from the chip. Optical chip backside reflectance imaging [22], scanning electron microscopy (SEM) imaging [33], or focused ion beam (FIB) imaging [34] are not suitable for detecting HTs on FPGAs, because the physical design and layout of the chip do not depend on the actual programmed functionality. SCA techniques, such as power analysis, EM analysis [20], or backscattering analysis [18, 19], can be used for all types of ICs. By applying different clustering algorithms, the Trojan-infected chips can be separated from the non-infected chips, often without the need of a golden chip, i.e., a chip which is known to be Trojan-free. However, these techniques only offer a limited resolution, which requires the Trojan trigger logic to consist of a minimum number of gates or being separated from its input signals to a certain extent [18]. Furthermore, the clustering does only work if the set of samples contains at least one non-infected device.

SCA techniques offering higher resolution include approaches that observe the chip's operation through the silicon backside, which is transparent to near-infrared (NIR) light. For instance, photon emission (PE) analysis can be used to compare dynamic and static emissions with the chip layout [16] or emissions from a golden chip [15]. Furthermore, adding oscillators with inputs from the design that act as beacons can facilitate the detection of tampering attempts, especially when cheaper infrared imaging is used [17]. However, such an approach increases the resource consumption of the design considerably in many cases and might not be able to detect all possible changes in LUT configurations. One approach providing higher resolution and better localization capabilities is optical probing. The authors of [21] have demonstrated that using an optical probing technique, all (FFs) used in the hardware design can be located and mapped to the intended design from the FPGA's integrated development environment (IDE). In this way, malicious changes in the sequential logic can be detected reliably and in a non-invasive fashion, if the chip is packaged as flip-chip. However, combinatorial logic cannot be detected using that approach, which is the major downside of the approach.

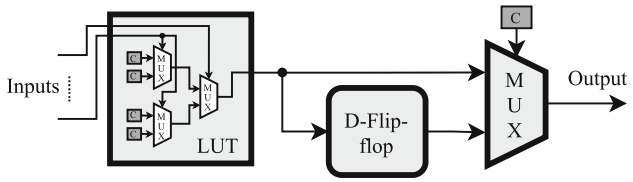


Fig. 1 Simplified schematic of an FPGA logic block. LUTs and MUXes are controlled by configuration memory cells

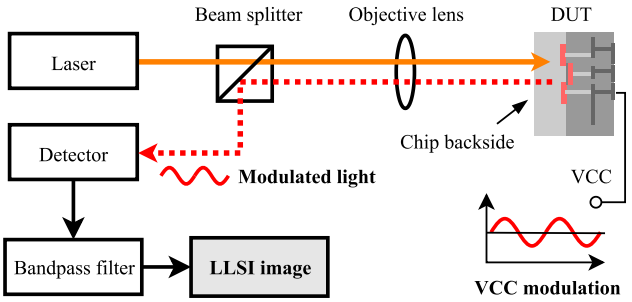


Fig. 2 Schematic of LLSI image acquisition. The DUT is scanned with a laser through the chip backside; due to a power supply (VCC) modulation, the reflected light is modulated, which can be detected

2.2 Field-programmable gate arrays (FPGAs)

The heart of an FPGA is its configurable logic fabric, consisting of an array of small configurable logic elements containing lookup tables (LUTs) and flip-flops (FFs) for implementing combinatorial and sequential logic, respectively. Configurable routing resources interconnect these blocks. Together with on-chip memories and input/output capabilities, such as transceivers, the designer can implement virtually every functionality on the FPGA. To add the software configurability of processors to FPGAs, vendors offer soft processor cores, and recently even SoCs containing both ASIC processors and an FPGA logic fabric, connected by an effective interconnection network.

Although the logic fabric architecture differs between manufacturers, the building blocks are multi-input LUTs for combinatorial logic, FFs for sequential logic, and MUXes for signal routing, see Fig. 1. The two main configuration storage types for FPGAs are volatile SRAM-based and non-volatile flash-based memories.

2.2.1 SRAM-based

The dominating manufacturers for FPGAs are Xilinx (acquired by AMD) and Intel (formerly Altera), with a combined share of more than 85% [35]. Both of them focus on SRAM-based FPGAs. The advantage of using SRAM as memory technology is that the chip can be manufactured with cutting-edge chip technologies, which allows for higher logic densities. Due to the volatile nature of SRAM cells, the FPGA's configuration is lost after every power-down. Therefore, the configuration (the bitstream) must be stored in external memory and loaded upon every reboot by the FPGA's configuration fabric. This fabric decrypts the configuration and loads it into the distributed SRAM cells on the chip, which determine the behavior of LUTs, MUXes, and routing transistors. One advantage of the volatile configuration storage is the possibility to partially reconfigure the logic fabric during runtime.

2.2.2 Flash-based

Flash-based FPGAs are offered mainly by Microchip (formerly Microsemi) and Lattice Semiconductor, with a combined market share smaller than 12% [35]. The main advantage of flash-based FPGAs over SRAM-based FPGAs is their lower power consumption. Further, the configuration is stored in a non-volatile way in distributed flash cells. One reason for the lower power consumption is that flash cells consist of fewer transistors than SRAM cells and do not need to be powered for retaining their value.

2.3 Laser-based logic readout

2.3.1 Technique

Optical probing is a powerful approach known from IC failure analysis (FA). A laser is pointed on the chip's backside, and switching activity causes the reflected laser light to modulate. More specifically, mainly the concentration of free carriers distinguishes the refraction and absorption of the laser light in silicon. When the laser scans the device and the reflected signal is fed through a bandpass filter set to a frequency of interest, all areas on the chip switching at a frequency of interest can be detected. The corresponding technique is called electro-optical frequency mapping (EOFM) or laser voltage imaging (LVI).

Using classical EOFM, only periodically switching elements on the chip can be detected. The static logic state of circuits, however, can be captured using laser logic state imaging (LLSI), which was introduced as an extension to EOFM [24]. The main idea behind LLSI is to stop the clock and induce a periodic frequency into the entire logic by modulating the power supply, see Fig. 2. This causes the free carrier concentrations to vary periodically, e.g., in the channel of transistors or in capacitors. This, in turn, modulates the reflected light, which can be detected using EOFM. Transistors that are switched on (low-ohmic channel) can thus be

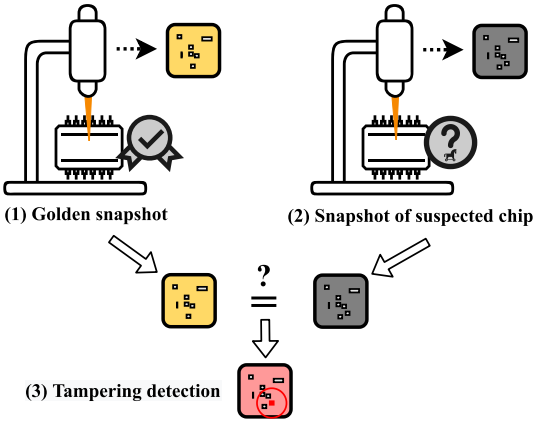


Fig. 3 Approach for detecting tampering with the FPGA logic fabric configuration

distinguished from transistors that are switched off (high-ohmic channel).

2.3.2 Related work

LLSI has been used in the hardware security field to extract the values stored in SRAM cells or FFs. The authors of [23] demonstrated that the FF content of an FPGA manufactured in a 60 nm technology can be extracted using LLSI. Using classical image recognition techniques, they show that the content can be extracted in an automated fashion. In [36], the authors demonstrate that a key stored in the SRAM of a microcontroller can be extracted using LLSI combined with deep learning techniques without the need to reverse-engineer the chip's layout. To the best of our knowledge, LLSI has neither been used to extract an FPGA's logic fabric configuration nor to detect HTs.

3 Approach

In our scenario, the supply chain from the finished product to the field cannot be trusted. In other words, an adversary might replace or change the device's functionality after it has left the trusted design house. In such a scenario, the highest efforts are paid to detect malicious hardware, e.g., in military, space, and aircraft applications. Although LLSI can capture the states of transistors and memory cells in all ICs, our goal in this work is to apply LLSI for creating snapshots of the logic fabric in FPGAs. To do so, we need to modulate the supply voltage of the logic under test, in our case, of the logic fabric, see Sect. 2.3. Furthermore, we need to halt the clock of the FPGA. To test if the FPGA's configuration manifests in the hardware snapshots, we configure the logic fabric in different ways, for instance, by altering the configuration of LUTs and the routing. We then compare the snapshot images to see if the changed configuration can be detected and at which location the change has occurred.

Once different configuration changes can be detected, the knowledge can be used to also detect malicious modifications on the chip, see Fig. 3. In our approach, we create a snapshot of the original Trojan-free design, also known as golden design, in the trusted design house (1). It typically will be necessary to create multiple snapshots to cover the entire logic fabric area with high resolution. We then assume a malicious entity that inserts a Trojan into the FPGA configuration of the product. Before using the final product in a security-critical application, the integrity of the IC should be certified. For this, we create a snapshot of the suspected chip (2). To eliminate the chance of any tampering, we compare the golden snapshot with the current snapshot (3). For comparing the snapshots, subtracting the images might be helpful. If there are differences, this indicates that the configuration has been altered, and the chip is not trustworthy. It should be noted that the state of the FPGA in step (1) and (2) should be the same, i.e., the clock should be stopped in the same cycle. We expect our approach to work on both SRAM- and flash-based FPGAs.

SRAM-based FPGAs SRAM-based FPGA configuration takes place by configuring LUTs and global/local routing via SRAM cells. In the end, all configuration SRAM cells do control MUXes, which consist of pass transistors. Since LLSI can extract the logic states of CMOS transistors, the FPGA's entire configuration should be extractable—given a sufficiently high optical resolution.

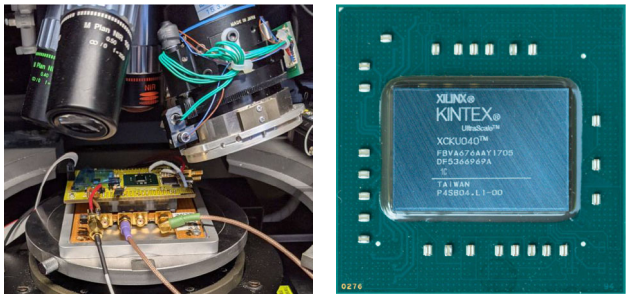
Flash-based FPGAs The configuration of flash-based FPGAs is stored in dedicated flash cells, which are distributed over the chip. They control the LUTs and global/local routing using multiplexers, which, like in SRAM-based FPGAs, consist of pass transistors. Therefore, also the configuration of flash-based FPGAs should be extractable using LLSI. If the flash cells are supplied by another voltage rail, it might be possible to see a configuration dependency by modulating that rail.

4 Experimental setup

This section first presents our measurement setup, followed by the devices under test (DUTs) and their setup for conducting LLSI.

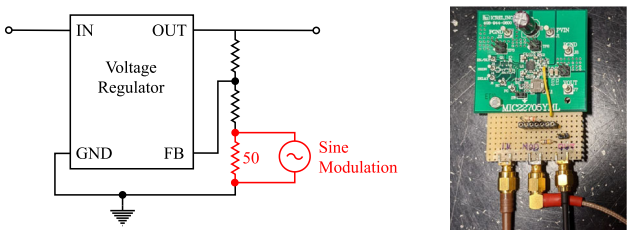
4.1 Measurement setup

As the setup for capturing the LLSI images, we use a Hamamatsu PHEMOS-1000 FA microscope, see Fig. 4a, equipped with a high-power incoherent light source (HIL) for optical probing. The microscope offers 5×, 20×, and 50× lenses and



(a) PHEMOS-1000 (b) Flip-chip package

Fig. 4 Xilinx Kintex-7 target under the PHEMOS-1000 microscope with 5× lens in use (**a**) and photography of the Xilinx UltraScale device (**b**)



(a) Schematic (b) Modulator

Fig. 5 LLSI modulation setup with the modulation regulator schematic (**a**) and the modified MIC22705YML-EV board (**b**)

an additional scanner zoom of ×2, ×4, and ×8. Due to the light source's wavelength of around 1.3 µm and the numerical aperture (NA) of our 50× lens of 0.71, the minimum beam diameter is around 1 µm. The step size of the galvanometric scan mirrors, however, is in the range of a few nanometers. For EOFM/LLSI measurements, the frequency of interest f, the bandpass bandwidth Δf, and the pixel dwell time Δt_{px} (in ms/px) can be configured in the PHEMOS software. To achieve LLSI measurements with an acceptable noise level, it is required to modulate the power rail of interest at more than around 80 kHz. LLSI image to the exact position on the chip, an optical light reflectance image can be captured alongside the measurement.

To better evaluate the LLSI signal differences and map them to a location on the optical image, we used the ImageJ application [37]. The pixel-wise subtraction of two LLSI images results in a mostly gray image with the differences displayed in white and black color. While this already shows the differences between the images clearly, the location of the changes is not intuitively visible. To superimpose the difference image on an optical image, we first remove noise by the "despeckle" functionality of ImageJ, and then merged the optical image and the difference image. To improve the visibility of the differences, we have remapped the black and white spots in the raw difference image to the colors yellow and green.

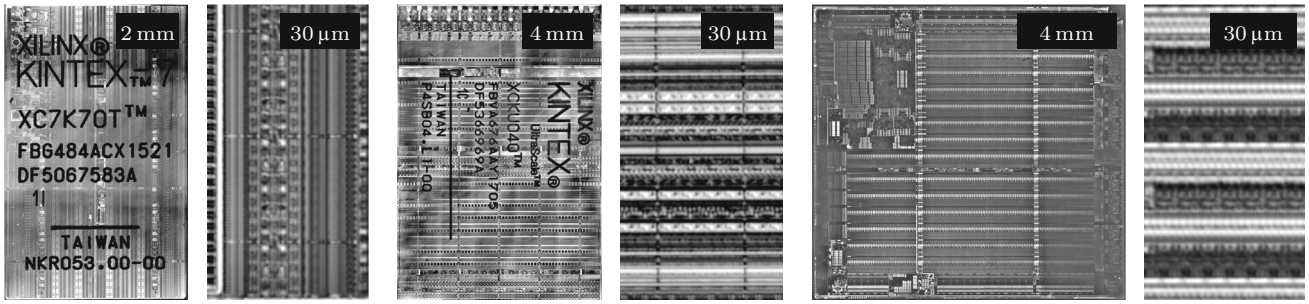
4.2 Devices under test

4.2.1 Xilinx Kintex-7 FPGA

As SRAM-based FPGA, we chose the Xilinx Kintex-7 XC7K70T, manufactured in a 28 nm technology. The chip is available in a ball grid array (BGA) bare-die flip-chip package on a Numato Systems Skoll development board. The FPGA can be programmed using the Xilinx Vivado IDE. In the Kintex-7 architecture [38], the logic fabric is comprised of (CLBs), so-called logic slices, and have a switch matrix for connecting to the global routing matrix. One slice consists of four 6-input LUTs (which can be configured as two 5-input LUTs with separate outputs each), eight FFs, as well as MUXes and arithmetic carry logic. While the slice naming uses X and Y coordinates (e.g., SLICE_X0Y0), the LUTs inside one slice are named from A5LUT/A6LUT to D5LUT/D6LUT, and the corresponding FFs from AFF/A5FF to DFF/D5FF. Next to the logic slices (2/3 of all slices), there are also memory slices usable as distributed RAM or shift registers.

To prepare the device for LLSI measurements, we disabled the onboard voltage regulator for VCC. Then, we soldered an SMA connector to the voltage rail for supplying the voltage externally via a power supply that can be modulated. For this purpose, we modified a MIC22705YML-EV voltage regulator evaluation board by replacing the resistor between the feedback pin and GND with a resistor to set the correct output voltage, in series with a 50 Ω resistor, see Fig. 5. In parallel to the latter, we connected a Keithley 3390 laboratory waveform generator to generate a sine wave. The regulator's output then provides a sine wave with a frequency of up to 300 kHz and a DC offset of the rated value for VCC of 1 V with a sufficient current drive strength. For higher frequencies, the regulator would stop functioning as intended. However, already when trying to modulate the DUT's voltage rail at low frequencies of a few kHz, no significant modulation can be measured on the printed circuit board (PCB)'s voltage rail. The reason for that is the existence of large decoupling capacitors, smoothing undesired peaks and fluctuations of the supply voltage. We desoldered all decoupling capacitors connected to VCC of 0.1 µF and larger using a hot air station to achieve a sufficiently high modulation amplitude. As a result, we could achieve a peak-to-peak modulation between 150 mV and 200 mV around the VCC offset of 1 V at a frequency f of 80 kHz.

Figure 6a shows optical (light reflectance) images of the entire chip and a section of the logic fabric. A raw LLSI image from the Kintex-7 logic fabric indicates that the modulation of VCC influences the light reflection almost everywhere, see Fig. 7.



(a) Xilinx Kintex-7 (b) Xilinx UltraScale (c) Microchip PolarFire SoC

Fig. 6 Laser reflection images of the DUTs: entire chip (left) and zoom into the logic fabric (right)

4.2.2 Xilinx UltraScale FPGA

As a second SRAM-based FPGA, we chose the Xilinx UltraScale XCKU040, manufactured in a 20 nm technology. The chip is available in a flip-chip bare-die package, see Fig. 4b, on an AVNET development board (model AES-KU040-DB-G). Similar to the Kintex-7 architecture (Sect. 4.2.1), the UltraScale logic fabric is comprised of CLBs. Each CLB contains one slice providing eight 6-input LUTs (which can also be configured as two 5-input LUTs with separate outputs), sixteen FFs, as well as MUXes and arithmetic carry logic. The slices are named using X and Y coordinates, whereas the LUTs and FFs are named with capital letters (A5LUT/A6LUT to H5LUT/H6LUT and AFF/AFF2 to HFF/HFF2). Next to the logic slices, there are memory slices that can be used as distributed RAM or shift registers. Figure 6b shows optical images of the entire chip and a section of the logic fabric.

To modulate the voltage rail of the UltraScale target, we used the same external modulation circuit as for the Kintex-7 (see Fig. 5). First, we disabled the onboard voltage regulator for VCC (0.95 V) by desoldering the coil at the regulator's output. Then, we soldered an SMA connector to the corresponding pad for supplying VCC externally. Furthermore, we desoldered all decoupling capacitors connected to VCC of 0.1 µF and larger from the PCB for being able to modulate the voltage rail at a sufficiently high frequency. For the experiments, we used a peak-to-peak modulation of around 150 mV at a frequency f of 80 kHz with a VCC offset of 0.95 V.

4.2.3 Microchip PolarFire SoC FPGA

As flash-based FPGA, we chose the Microchip PolarFire SoC MPFS250T-FCVG484EES, manufactured in a 28 nm technology. The configuration is stored in distributed flash cells manufactured in Microchip's SONOS technology [39], consisting of two floating-gate transistors. The chip is available on the PolarFire SoC FPGA Icicle Kit in a BGA flip-chip package with a lid. After cooling down the device in a typical household freezer, we could pry off the lid using a knife to access the chip backside. The FPGA can be programmed using the Microsemi Libero IDE. In the PolarFire architecture [40], the logic fabric is comprised of arrays of logic clusters (LCs) that are connected by interface logic (IL). Each LC consists of 12 logic elements (LEs), whereas each LE contains a 4-input LUT, a FF, and a MUX. Next to a connection to the IL, the individual LEs inside one LC are connected by a carry chain. Next to the LCs, there are other blocks, such as dedicated math and memory blocks, connected via the IL.

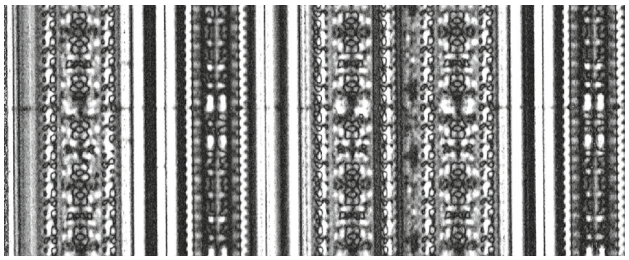


Fig. 7 LLSI raw image from the logic fabric on the Kintex-7 FPGA. 50× (×2) zoom, $\Delta t_{\text{px}} = 2.1$ ms/px, $\Delta f = 300$ Hz

We could use the onboard MIC22705YML voltage regulator for modulating VDD of this target. Via a jumper, the resistor in the feedback path can be changed to create a 1.0 V or 1.05 V supply voltage. By removing the jumper and connecting our own resistors, we could create the same modulation capabilities as shown in Fig. 5a. To increase the LLSI signal's amplitude, we desoldered all decoupling capacitors connected to VDD of 0.1 µF and larger from the PCB. We used a peak-to-peak modulation of approximately 170 m V around the VDD offset of 1 V. A modulation frequency f of 83.5 kHz led to the highest LLSI signal amplitude. Note that the SONOS cells are not supplied by VDD but VDD25, which is supplied by a 2.5 V regulator. To modulate the VDD25 voltage, we soldered a jumper to disable the onboard regulator and added an SMA connector to supply VDD25 via our external modulator circuit. However, as we could not detect any benefit over modulating VDD, we only used the

VDD modulation for the experiments presented in this paper. Figure 6b shows optical images of the entire chip and a part of the logic fabric.

5 Results

5.1 Detecting changes in the logic fabric

To investigate the capabilities of LLSI for detecting changes in the logic fabric configuration, we first tried to detect small changes within one logic element, i.e., changes in the LUT configurations and FF logic states. Although the number of different configurations is high, we aimed at creating a good coverage of detectable changes.

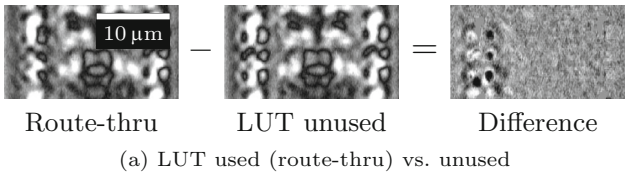
5.1.1 SRAM-based (Kintex-7)

LUT used versus unused We compared implementations where once the LUT is unused and once a route-thru LUT is implemented. We assumed a route-thru LUT to be the configuration with minimal differences compared to the unused LUT, as the input of the LUT is directly routed to the output of the SLICE. Nevertheless, the differences can be clearly identified, see Fig. 8a.

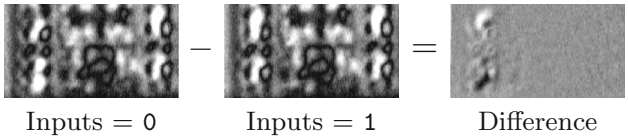
LUT inputs 0 versus 1 When changing the values of LUT inputs, which originate from the output of another LUT or a FF, the change is clearly visible as well, see Fig. 8b. As could be expected, we observed fewer changes if fewer input values are changed. Still, we could detect changes also if only one input value is changed.

LUT configuration value changes The smallest possible change we could imagine is the manipulation of single bits in the LUT configuration. We observed that the number of bits changed in the LUT configuration `INIT` value does not necessarily determine how significant the difference in the LLSI response is, see Fig. 8c, d. We assume that not the SRAM cell holding the configuration produces the LLSI signature, but the actual multiplexers and pass transistors. If a configuration change causes—due to the applied LUT inputs—more multiplexers to change their states (cf. Fig. 1), there will be a bigger difference between the LLSI images.

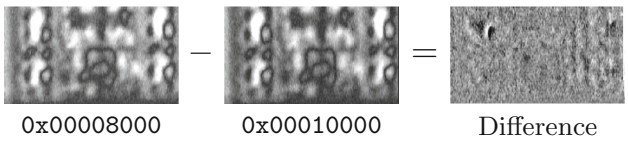
FF value 0 versus 1 Finally, we designed a bit more complex design, which contains two FFs and one LUT residing in different logic slices, see Fig. 9. We have subtracted the LLSI images of two consecutive clock cycles. While the difference for the LUT is concentrated in a single small area, there are many different spots for the FFs. This might be explained by the fact that the input buffers, the actual memory cell, the output buffers, and the clock buffers have changed their values by advancing a clock cycle as well. Interestingly, although the two registers were instantiated in exactly the same way in the IDE, different changes can be observed between them.



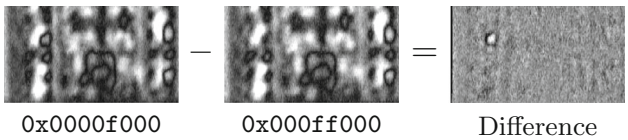
(a) LUT used (route-thru) vs. unused



(b) 5-input LUT with all inputs set to one value



(c) 1-bit LUT configuration (`INIT`) value change



(d) 4-bit LUT configuration (`INIT`) value change

Fig. 8 Kintex-7 LLSI results for different lookup-table configurations. 50× (×4) zoom, $\Delta t_{\text{px}} = 3.3\,\text{ms/px}$, $\Delta f = 100\,\text{Hz}$

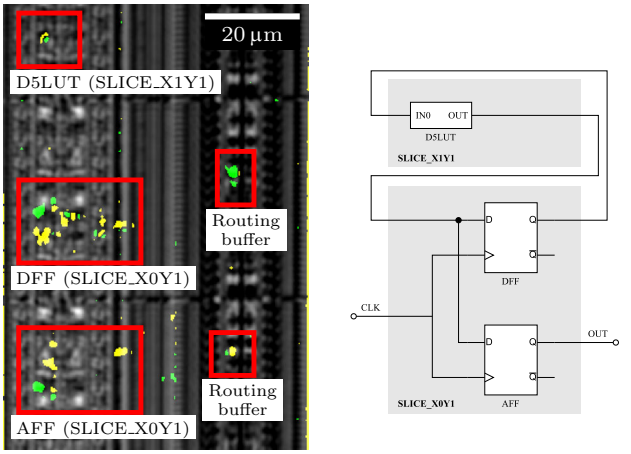
This might be caused by the different output configurations of the FFs or an asymmetric ASIC design of the CLB. For instance, the clock buffers or some intra-CLB routing capabilities, which are invisible in the IDE for the designer, might reside close to DFF. Finally, we could observe differences in the (assumed-to-be) routing areas, supposedly interconnecting the two slices X0Y1 and X1Y1.

5.1.2 SRAM-based (UltraScale)

To investigate if similar results can be achieved on a DUT manufactured in a smaller technology, we conducted the same experiments on the UltraScale FPGA.

LUT used versus unused Although the technology node size of the UltraScale series is around 28% smaller than of the Kintex-7 series, the difference between a route-thru LUT and a completely unused LUT is clearly visible, see Fig. 10a. Due to the technology size reduction, the affected area is smaller but can still be resolved using our optical setup. Furthermore, the difference image looks more blurry than for the Kintex-7 FPGA. One explanation for this might be the lower modulation amplitude achievable on the UltraScale board.

LUT inputs 0 versus 1 Flipping the LUT's inputs values can be detected reliably as well, see Fig. 10b. Interestingly, the affected area seems to be as large as in the previous exper-



(a) LLSI difference (b) Logic schematic

Fig. 9 Kintex-7 LLSI difference superimposed over an optical image for FF values 0 versus 1 with CLB inputs and outputs connected. Yellow and green colors correspond to the black and white spots in the raw difference image. 50× (×2) zoom, $\Delta t_{\text{px}} = 2.1$ ms/px, $\Delta f = 300$ Hz

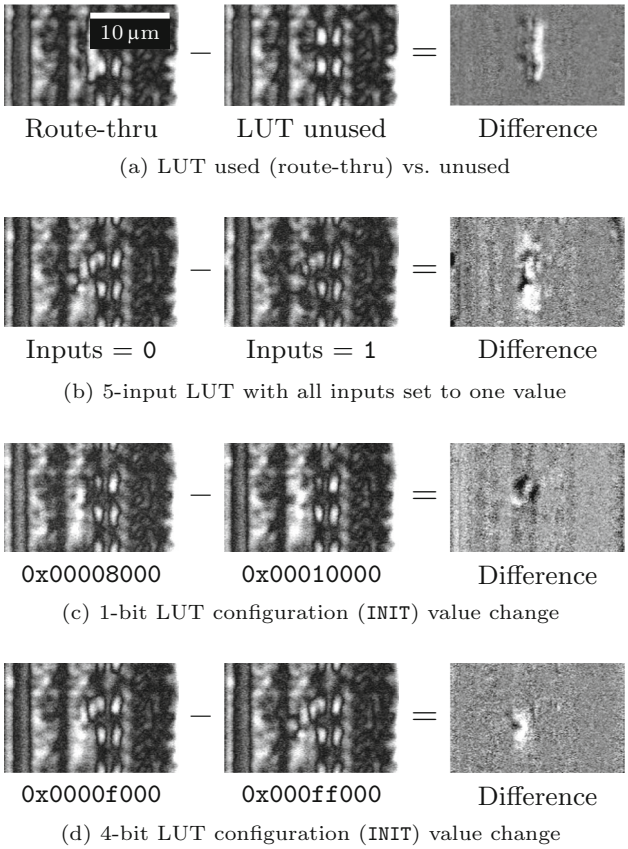
iment on used vs. unused LUT. The reason might be that we cannot control the routing of signals and which values are applied to unused inputs.

LUT configuration value changes We could clearly detect the same LUT configuration changes that we could detect on the Kintex-7, see Fig. 10c, d. For this target, the affected area neither reflects the number of bits changed in the configuration. This observation supports the hypothesis that the LUT's multiplexers and not the memory cells for the configuration contribute most to the LLSI signal.

FF value 0 versus 1 When investigating an entire CLB with one LUT and two FFs in use, multiple areas with differences in the LLSI image can be observed, see Fig. 11. Again, we subtracted the LLSI images of two consecutive clock cycles. From the knowledge gained in the previous experiments, we could identify the changes in the LUT and map two areas with similar changes to the two FFs. Despite these distinctly allocable changes, many other areas with clear differences appear in the image. These changes seem to belong to the CLB's MUXes (left of the LUTs and FFs) and routing resources, such as buffers (right side of the image). However, since the chip's layout is unknown, these assumptions cannot be verified further.

5.1.3 Flash-based (PolarFire SoC)

To investigate whether configuration changes can also be detected on the flash-based FPGA, we conducted similar experiments on the PolarFire SoC FPGA.



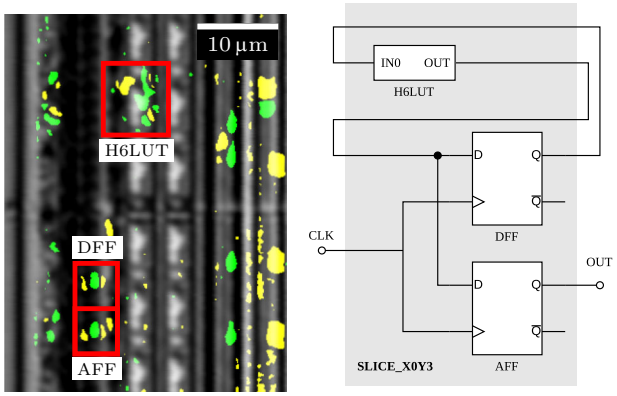
(a) LUT used (route-thru) vs. unused

(b) 5-input LUT with all inputs set to one value

(c) 1-bit LUT configuration (INIT) value change

(d) 4-bit LUT configuration (INIT) value change

Fig. 10 UltraScale LLSI results for different lookup-table configurations. 50× (×4) zoom, $\Delta t_{\text{px}} = 2.1$ ms/px, $\Delta f = 300$ Hz



(a) LLSI difference (b) Logic schematic

Fig. 11 UltraScale LLSI difference superimposed over an optical image for different FF values and LUT inputs. 50× (×4) zoom, $\Delta t_{\text{px}} = 2.1$ ms/px, $\Delta f = 300$ Hz

LUT used versus unused For this target, we compared the configuration for a route-thru LUT with an unused LUT as well, see Fig. 12a. The LLSI responses show a clear difference, although the corresponding area is smaller than on the Xilinx FPGAs. The reason might be that the LUTs on Kintex-7 and UltraScale have up to 6 inputs, while they only have 4 inputs on PolarFire, resulting in a significant difference in the number of contained MUXes.

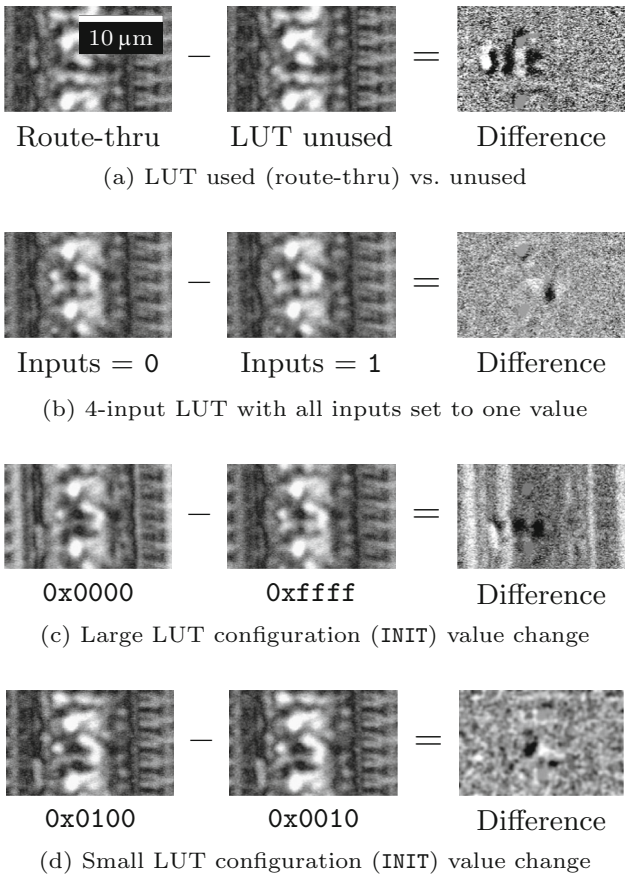
LUT inputs 0 versus 1 The area of differences when only the LUT inputs change are smaller than the differences between a used and unused LUT—as can be expected, see Fig. 12b.

LUT configuration value changes Changes in the LUT configurations can be detected as well. For a large change in the configuration, i.e., by flipping all bits, the change with the largest area is visible, see Fig. 12c. As for the other FPGAs, the reason might be the different number of MUXes affected by the configuration change, under the assumption that the inputs of the LUT stay constant. For a 2-bit change in the `INIT` value, a smaller difference is visible, see Fig. 12d. Moreover, we observed that when all LUT inputs are set to `0`, the difference for changed `INIT` values is larger than when all inputs are set to `1`. Since in our experiment the output of the LUT was not changed by applying the different inputs (due to the configured `INIT` value), we suppose that a different number of multiplexers changed their states depending on the LUT inputs.

FF value 0 versus 1 Similar to the experiments on the SRAM-based FPGAs, we created snapshots of a larger area of the logic fabric, on the one hand, to observe the LLSI response differences for a FF, and on the other hand, to learn about the detectability of buffers and routing transistors. Figure 13 shows the difference of two LLSI responses captured in two consecutive clock cycles. The state change of the FF is clearly visible on the top right of the image. The three LUTs receive the output of the FF as inputs, and therefore, their responses differ, too. Differences can also be observed in between the rows of logic elements. These areas presumably belong to the routing logic, thus containing data and clock buffers.

5.2 Detecting changes in routing

The authors of [12] propose malicious modifications in the signal runtime on the FPGA by using either route-thru LUTs or manipulating the routing to take longer paths. We have already shown that the insertion of route-thru LUTs can be detected; see Sect. 5.1. To test the capability of our approach



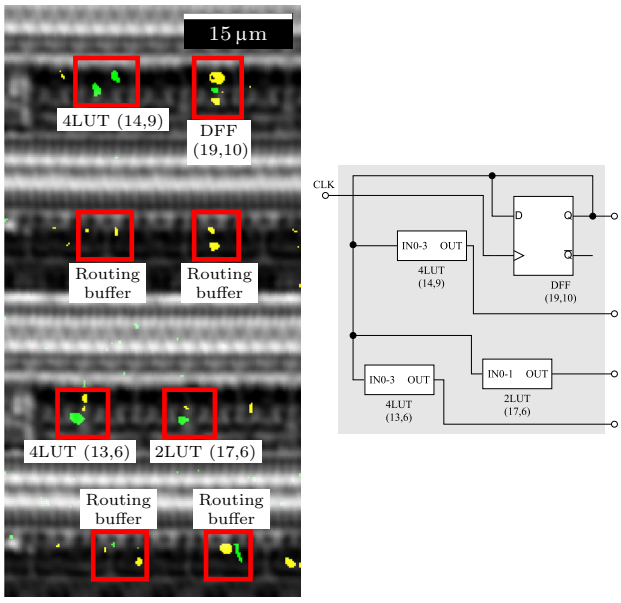
(a) LUT used (route-thru) vs. unused

(b) 4-input LUT with all inputs set to one value

(c) Large LUT configuration (`INIT`) value change

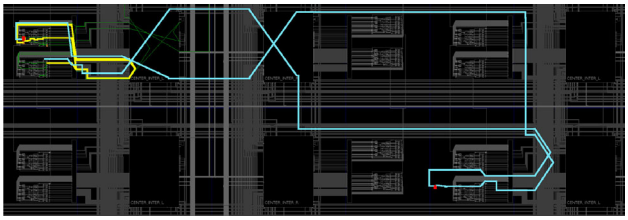
(d) Small LUT configuration (`INIT`) value change

Fig. 12 PolarFire SoC LLSI results. Images rotated by 90 degrees, 50× (×4) zoom, $\Delta t_{\text{px}} = 3.3$ ms/px, $\Delta f = 100$ Hz

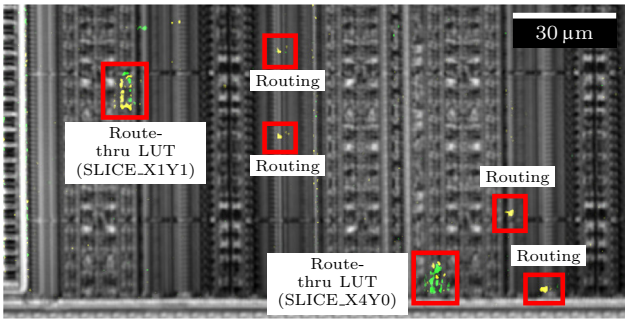


(a) LLSI difference (b) Logic schematic

Fig. 13 PolarFire SoC LLSI difference superimposed over an optical image for different FF values and LUT inputs. 50× (×2) zoom, $\Delta t_{\text{px}} = 3.3$ ms/px, $\Delta f = 100$ Hz



(a) Placement schematic of design with LUT in SLICE_X1Y1 (yellow) and in SLICE_X4Y0 (blue)



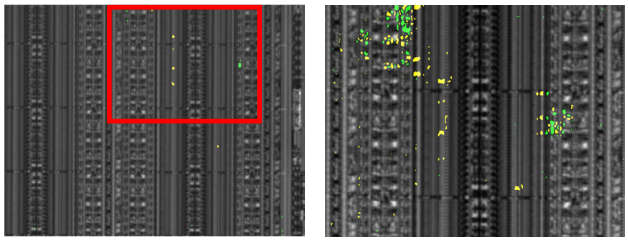
(b) LLSI difference

Fig. 14 Difference in routing configuration on Kintex-7 when moving a route-thru LUT from SLICE X1Y1 to X4Y0 while keeping the signal source and destination in SLICE X1Y1 and X0Y1. 50× (×2) zoom, $\Delta t_{\text{px}} = 2.1$ ms/px, $\Delta f = 300$ Hz

to detect changes in the routing, we created a design for the Kintex-7 FPGA that contains one route-thru LUT, whose location we change between two measurements. Thereby, the signal is forced to be routed differently. For the first snapshot, the LUT is placed in SLICE_X1Y1, while for the second snapshot, it is placed in SLICE_X4Y0, see Fig. 14a. The signal source and sink are kept at the same location (in SLICE_X0Y1 and X1Y1). Figure 14b clearly shows not only the differences in the LLSI response for the changed LUT placement but also for the routing logic. Consequently, one can also detect changes in signal routing with our approach.

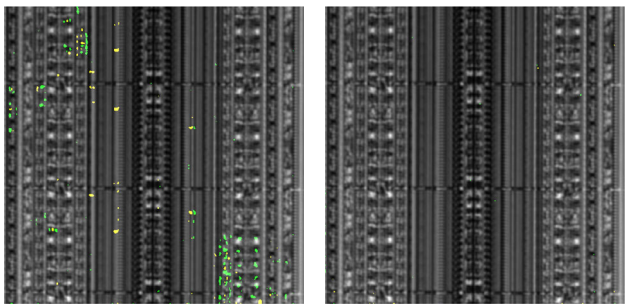
5.3 Trojan benchmarks

The previous results have already shown that small changes, down to single bit changes in the LUT configuration and small changes in the routing configuration, can be detected using our method. Therefore, we have demonstrated that LLSI can detect the malicious modifications proposed in [12] introducing changes in the signal path delays. To demonstrate that we can also detect other HTs proposed in the literature, we exemplarily implemented HT benchmarks generated using the TRIT framework [30], which can be found on TrustHub [29]. We implemented two benchmarks on the Kintex-7 DUT, one consisting only of combinatorial HT logic (from TRIT-TC) and one also containing sequential logic (from TRIT-TS). All provided benchmarks generated using



(a) 50× zoom (b) 50× (×2) zoom

Fig. 15 Combinatorial Trojan benchmark (c2670_T071) section on Kintex-7. (**a**) $\Delta t_{\text{px}} = 5ms/px$, (**b**) $\Delta t_{\text{px}} = 3.3$ ms/px, $\Delta f = 100$ Hz



(a) Free vs. Trojan (b) Free vs. Free, different runs

Fig. 16 Sequential Trojan benchmark (s1423_T607) section on Kintex-7. 50× (×2) zoom, $\Delta t_{\text{px}} = 3.3ms/px$, $\Delta f = 100Hz$

TRIT introduce additional logic gates and/or FFs. We fixed the location and routing placement of all logic components and the routing that does not belong to the HT trigger or payload to keep the changes of the implementation minimal.

5.3.1 Combinatorial Trojan

The c2670_T071 HT benchmark introduces six additional logic gates. Figure 15 only shows a part of the logic fabric area consumed by the implementation. However, already in this section of the design, clear differences can be observed. As can be seen, zooming into an area with suspicious differences can highlight the changes more clearly.

5.3.2 Sequential Trojan

Next to combinatorial gates, the s1423_T607 benchmark contains a counter with 15 states implemented using FFs. Figure 16a indicates that many changes can be detected both in the CLBs and routing areas. As expected, when capturing two LLSI images of the same area from the Trojan-free design, no clear differences can be observed, see Fig. 16b. This proves that the previously observed differences are not only caused by noisy measurements.

6 Discussion

In this section, we first discuss further research directions continuing our approach. Subsequently, we talk about the applicability of our approach and discuss potential limitations.

6.1 Further research directions

6.1.1 Application to ASICs

Regarding the applicability of our approach to ASIC implementations, a few things have to be kept in mind. Generally, it should be possible to detect the locations of all transistors and then overlay the layout file. In this way, irregularities and deviations from the intended designs can be detected, even without having a golden chip. One drawback is that modifications that only affect the metal layers cannot be detected if the changes do not manifest in the light reflection. However, we think that detecting analog HTs, such as capacitor-based and dopant-level Trojans, should be possible using LLSI. Since these HTs use analog properties of the chip and are pre-silicon modifications, we could not investigate them. However, in the following, we explain why our approach should be able to detect such HTs.

Detecting capacitor-based Trojans Results from [24] indicate that decoupling capacitors can be imaged using LLSI. Since these capacitors are connected between VCC and GND, the power supply modulation will modulate the electric field and charge density of the capacitor, which influences the light reflection. Therefore, LLSI might also be applicable to detect HTs that only introduce changes in the capacitance to create a stealthy trigger mechanism (e.g., A2 Trojans [13]).

Detecting dopant-level Trojans The investigations in [41] and [42] show that the light reflection for optical probing depends on the doping level of the silicon. Therefore, malicious modifications in the doping concentration to alter the functionality of logic gates [14] might be detectable using LLSI.

6.1.2 Reverse-engineering the FPGA configuration

As already shown in this work, the configuration of the FPGA logic fabric is contained in the LLSI snapshots. Although the resolution seems to be insufficient to extract the exact configurations manually, machine learning approaches might be able to solve that task. The advantages of employing deep learning techniques have already been demonstrated in [36] for data extraction from dedicated on-chip memories. Such configuration extraction can also facilitate the structural and functional reverse engineering of bitstreams in proprietary formats.

6.2 Applicability of LLSI

We have shown that our approach using LLSI can detect a wide range of changes in the FPGA logic fabric configuration. In the following, we discuss the practical applicability of LLSI.

6.2.1 Chip access

For our approach, we need access to the silicon backside of the chip. Since all FPGAs used in this work are only available in flip-chip packages, this requirement can be easily met. Moreover, due to performance, size, cost, and environmental compatibility reasons, chips are predominantly delivered in flip-chip packages [43]. While many of such packages have a lid installed—which we could easily remove for the PolarFire SoC—there are also bare-die packages available, like the one of our Kintex-7 and UltraScale DUTs. Consequently, if a customer would like to have the opportunity to test the chip for HTs using an optical probing approach, he or she should choose a bare-die package to facilitate testing. Thinning or polishing the silicon backside is not necessary for optical probing, as shown in this work.

6.2.2 PCB modifications

In order to reach modulation frequencies of 80 kHz and higher, we had to replace the voltage regulator on the Kintex-7 and UltraScale DUTs with an external one. However, on the PolarFire DUT, we could leverage the on-PCB regulator for the modulation, requiring no modifications on the PCB. Consequently, by using a suitable voltage regulator on the PCB, there is no need to provide the modulated voltage from an external source.

During our investigations, we observed that a higher modulation of the supply voltage produces a clearer LLSI image, and consequently, a shorter pixel dwell time is sufficient. Moreover, a higher modulation frequency can further reduce the pixel dwell time, leading to faster scan times. The PCB and the die interposer PCB, however, are designed to compensate spikes and smooth undesired peaks and fluctuations of the supply voltage. For this purpose, decoupling capacitors of different sizes are connected between the supply voltage rail and ground, effectively acting as low-pass filters.

To achieve the desired modulation amplitude of the power rail at frequencies above 80 kHz, we had to remove the decoupling capacitors of 0.1 µF and larger from the PCB. Due to the existence of other capacitive and inductive elements in the circuit, a higher modulation frequency results in a lower modulation amplitude and, therefore, a lower LLSI signal level. Consequently, there is a tradeoff between the noise ratio in the LLSI images, the scan time, and the electrical preparation of the DUT. Due to practical reasons, we did not remove

smaller capacitors. Furthermore, we did not remove capacitors from the interposer PCB, as there is no documentation on potential effects available. Nevertheless, a device that is ready for use in a practical application must have installed all capacitors due to reliability and stability constraints. One way to still enable the measurements required by our approach is the installation of jumpers or other switches on the PCB to disable the capacitors on demand.

6.2.3 Optical stability

In our experiments, we observed that the optical focus was slightly drifting during the LLSI measurements due to mechanical instabilities in the setup. Since the LLSI signal heavily depends on the focus position, there are small differences between LLSI images that are not caused by design modifications. However, the stability of our setup was sufficient to produce reliable and significant results for detecting malicious changes in the design. Nevertheless, the image quality will improve if the mechanical stability is enhanced, for instance, by operating the setup in a tempered room and a shock-absorbing building.

6.2.4 Optical resolution

The optical resolution of laser-assisted side-channel techniques has been discussed extensively by the research community in numerous publications, e.g., in [4, 21, 44–47]. We discuss the most important and new insights in the following.

Both FPGAs used in this work were manufactured in 28 nm and even 20 nm technologies. Although the minimum width of our setup's optical beam is around 1µm, it should be kept in mind that the technology size does distinguish neither the minimum size of a transistor nor the typical distance between transistors. An important fact is that the laser scanner has a step size in the range of a few nanometers. Therefore, while scanning with the laser over the DUT, the beam covers one specific point on the chip multiple times. Consequently, if the beam covers multiple nodes of interest, the LLSI image shows a different position-dependent superposition of the same nodes at different adjacent pixel locations. However, due to the Gaussian intensity distribution of the beam, it might still be possible to extract the logic state. This explains why optical probing delivers meaningful results also on structures that are smaller than the beam diameter.

Moreover, a so-called solid immersion lens (SIL) can be used to increase the optical resolution down to 250 nm [48], which is sufficient to resolve individual transistors in a 14 nm technology [49]. Accordingly, Intel has shown that LLSI can be applied on very small devices, such as single inverters, on a test chip manufactured in a 14 nm technology [24].

Even if it might not be possible to resolve single SRAM cells used for configuration storage in future technologies, the FFs, MUXes, and other pass transistors are influenced by the configuration and contribute to the LLSI image as well. This is supported by the observation that even on the 20 nm FPGA, the different LUT configurations could be detected. Furthermore, typical HTs in benchmarks alter the design by inserting or modifying multiple logic gates or FFs, resulting in huge changes, which we could detect reliably.

7 Conclusion

Dormant hardware Trojans that introduce only tiny malicious hardware modifications pose a severe threat in security-critical applications. In this work, we have demonstrated a detection approach for dormant HTs using the laser-assisted optical probing method LLSI. By modulating the power supply of the chip, even inactive logic is visible on the logic snapshots. By awakening the potential Trojan in this way, no malicious modification of the FPGA's configuration stays undetected. We have demonstrated that our approach is applicable to recent SRAM- and flash-based FPGAs on the market in a non-invasive manner. It did not make a significant difference whether the FPGAs were manufactured in a 28 nm or 20 nm technology. Finally, we have explained why our framework should also be suitable for detecting stealthy HTs on ASICs.

Acknowledgements The authors would like to acknowledge Hamamatsu Photonics K. K. Japan and Germany for their help and support on the PHEMOS system.

Funding Open Access funding enabled and organized by Projekt DEAL. The authors from Technische Universität Berlin have been supported in part by the Einstein Foundation (EP-2018-480), and in part by the Deutsche Forschungsgemeinschaft (DFG—German Research Foundation) under the priority programme SPP 2253, Grant Number 439918011. The author from Worcester Polytechnic Institute has been supported in part by National Science Foundation (NSF) under the Grant Number 2117349 and in part by Massachusetts Technology Collaborative (MassTech).

Data availability The datasets generated during and analyzed during the current study are available from the corresponding author on reasonable request.

References

1. Ng, X.T., et al.: Integrated sensor: a backdoor for hardware trojan insertions? In: 2015 Euromicro conference on digital system design (2015). https://doi.org/10.1109/DSD.2015.119
2. Roy, D.B., et al.: The conflicted usage of RLUTs for security-critical applications on FPGA. J. Hardw. Syst. Secur. **2**, 162–178 (2018). https://doi.org/10.1007/s41635-018-0035-4
3. Moradi, A., Schneider, T.: Improved side-channel analysis attacks on Xilinx bitstream encryption of 5, 6, and 7 series . In: International workshop on constructive side-channel analysis and secure design (2016)
4. Tajik, S., Lohrke, H., Seifert, J.-P., Boit, C.: On the power of optical contactless probing: attacking bitstream encryption of FPGAs . In: 2017 ACM SIGSAC conference on computer and communications security (CCS) (2017)
5. Lohrke, H., Tajik, S., Krachenfels, T., Boit, C., Seifert, J.-P.: Key extraction using thermal laser stimulation. In: Conference on cryptographic hardware and embedded systems (CHES) (2018)
6. Hettwer, B., Leger, S., Fennes, D., Gehrer, S., Güneysu, T.: Side-channel analysis of the Xilinx Zynq ultrascale+ encryption engine. In: Conference on cryptographic hardware and embedded systems (CHES) (2021)
7. Ender, M., Moradi, A., Paar, C. The unpatchable silicon: a full break of the bitstream encryption of Xilinx 7-series FPGAs. In: 29th USENIX security symposium (USENIX security 20) (2020)
8. Zhang, Z., Njilla, L., Kamhoua, C.A., Yu, Q.: Thwarting security threats from malicious FPGA tools with novel FPGA-oriented moving target defense. IEEE Trans. VLSI Syst. **27**(3), 665–678 (2019). https://doi.org/10.1109/TVLSI.2018.2879878
9. Microchip Technology, Inc. UG0753 User guide PolarFire FPGA security (2021)
10. Xilinx, Inc.: Developing tamper-resistant designs with Zynq UltraScale+ devices (2018)
11. Salmani, H., Tehranipoor, M., Plusquellic, J.: New design strategy for improving hardware trojan detection and reducing trojan activation time . In: 2009 IEEE international workshop on hardware-oriented security and trust (HOST) (2009)
12. Ender, M., Ghandali, S., Moradi, A., Paar, C.: The first thorough side-channel hardware trojan. In: Advances in Cryptology—ASIACRYPT 2017. **10624**, 755–780 (2017). https://doi.org/10.1007/978-3-319-70694-8_26
13. Yang, K., Hicks, M., Dong, Q., Austin, T., Sylvester, D.: A2: analog malicious hardware. In: 2016 IEEE symposium on security and privacy (SP) (2016). https://doi.org/10.1109/SP.2016.10
14. Becker, G.T., Regazzoni, F., Paar, C., Burleson, W.P.: Stealthy dopant-level hardware trojans. In: Conference on cryptographic hardware and embedded systems (CHES) (2013). https://doi.org/10.1007/978-3-642-40349-1_12
15. Song, P., et al.: MARVEL—malicious alteration recognition and verification by emission of light. In: 2011 IEEE international symposium on hardware-oriented security and trust (HOST) (2011). https://doi.org/10.1109/HST.2011.5955007
16. Stellari, F., et al.: Verification of untrusted chips using trusted layout and emission measurements. In: 2014 IEEE international symposium on hardware-oriented security and trust (HOST) (2014). https://doi.org/10.1109/HST.2014.6855562
17. Duncan, A., et al.: FLATS: filling logic and testing spatially for FPGA authentication and tamper detection. In: 2019 IEEE international symposium on hardware oriented security and trust (HOST) (2019). https://doi.org/10.1109/HST.2019.8741025
18. Nguyen, L.N., Cheng, C.-L., Prvulovic, M., Zajic, A.: Creating a backscattering side channel to enable detection of dormant hardware trojans. IEEE Trans. VLSI Syst. **27**(7), 1561–1574 (2019). https://doi.org/10.1109/TVLSI.2019.2906547
19. Adibelli, S., Juyal, P., Nguyen, L.N., Prvulovic, M., Zajic, A.: Near field backscattering based sensing for hardware trojan detection. IEEE Trans. Antennas Propag. (2020). https://doi.org/10.1109/TAP.2020.3000562
20. He, J., Ma, H., Liu, Y., Zhao, Y.: Golden chip-free trojan detection leveraging trojan trigger's side-channel fingerprinting. ACM Trans. Embed. Comput. Syst. **20**(1), 1–18 (2020). https://doi.org/10.1145/3419105
21. Stern, A., Mehta, D., Tajik, S., Farahmandi, F.,Tehranipoor, M.: SPARTA: a laser probing approach for trojan detection. In: 2020 IEEE international test conference (ITC) (2020)
22. Zhou, B., et al.: Hardware trojan detection using backside optical imaging. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. (2020). https://doi.org/10.1109/TCAD.2020.2991680
23. Krachenfels, T., Ganji, F., Moradi, A., Tajik, S., Seifert, J.-P.: Real-world snapshots vs. theory: questioning the t-probing security model . In: 2021 IEEE symposium on security and privacy (SP) (2021). https://doi.org/10.1109/SP40001.2021.00029
24. Niu, B., et al.: Laser logic state imaging (LLSI) (2014). In: 40th international symposium for testing and failure analysis ISTFA (2014)
25. Krachenfels, T., Seifert, J.-P., Tajik, S.: Trojan awakener: detecting dormant malicious hardware using laser logic state imaging. In: 5th workshop on attacks and solutions in hardware security (2021). https://doi.org/10.1145/3474376.3487282
26. Wang, X., Tehranipoor, M., Plusquellic, J.: Detecting malicious inclusions in secure hardware: challenges and solutions. In: 2008 IEEE international workshop on hardware-oriented security and trust (2008). https://doi.org/10.1109/HST.2008.4559039
27. Bhunia, S., Hsiao, M.S., Banga, M., Narasimhan, S.: Hardware trojan attacks: threat analysis and countermeasures. Proc. IEEE **102**(8), 1229–1247 (2014). https://doi.org/10.1109/JPROC.2014.2334493
28. Becker, G.T., Regazzoni, F., Paar, C., Burleson, W.P.: Stealthy dopant-level hardware trojans. In: Conference on cryptographic hardware and embedded systems (CHES) (2013)
29. Shakya, B., et al.: Benchmarking of hardware trojans and maliciously affected circuits. J. Hardw. Syst. Secur. **1**(1), 85–102 (2017). https://doi.org/10.1007/s41635-017-0001-6
30. Cruz, J., Huang, Y., Mishra, P., Bhunia, S.: An automated configurable trojan insertion framework for dynamic trust benchmarks (2018). In: 2018 design, automation & test in Europe conference & exhibition (DATE). https://doi.org/10.23919/DATE.2018.8342270
31. Jacob, N., Rolfes, C., Zankl, A., Heyszl, J., Sigl, G.: Compromising FPGA SoCs using malicious hardware blocks (2017). In: Design, automation test in Europe conference exhibition (DATE). https://doi.org/10.23919/DATE.2017.7927157
32. Jacob, N., Heyszl, J., Zankl, A., Rolfes, C., Sigl, G.: How to break secure boot on FPGA SoCs through malicious hardware. In: Conference on cryptographic hardware and embedded systems (CHES) (2017)
33. Vashistha, N., et al.: Trojan scanner: detecting hardware trojans with rapid SEM imaging combined with image processing and machine learning. In: 44th international symposium for testing and failure analysis (ISTFA) (2018)
34. Sugawara, T., et al.: Reversing stealthy dopant-level circuits. In: International workshop on cryptographic hardware and embedded systems (2014)
35. Doug Black.: Xilinx says its new FPGA is world's largest (2019). https://www.enterpriseai.news/2019/08/21/xilinx-says-its-new-fpga-is-worlds-largest/

36. Krachenfels, T., Kiyan, T., Tajik, S., Seifert, J.-P.: Automatic extraction of secrets from the transistor jungle using laser-assisted side-channel attacks. In: 30th USENIX security symposium (USENIX security 21) (2021)
37. Rueden, C.T., et al.: Image J2: ImageJ for the next generation of scientific image data. BMC Bioinform. **18**(1), 529 (2017). https://doi.org/10.1186/s12859-017-1934-z
38. Xilinx, Inc.: 7 series FPGAs configurable logic block user guide (UG474) (2016)
39. Microsemi Corporation: White paper: PolarFire non-volatile FPGA family delivers ground breaking value: cost optimized, Lowest Power, EU immunity, and high-security (2017). https://www.microsemi.com/document-portal/doc_download/1243174-polarfire-fpga-white-paper
40. Microchip Technology, Inc.: UG0680 user guide PolarFire FPGA fabric (2021)
41. Kindereit, U., et al.: Quantitative investigation of laser beam modulation in electrically active devices as used in laser voltage probing. IEEE Trans. Device Mater. Reliab. **7**(1), 19–30 (2007). https://doi.org/10.1109/TDMR.2007.898074
42. Kindereit, U.: Investigation of laser-beam modulations induced by the operation of electronic devices. Ph.D. thesis, Technische Universität Berlin (2009). https://depositonce.tu-berlin.de//handle/11303/2440
43. Tong, H., Lai, Y., Wong, C.: Advanced Flip Chip Packaging. Springer (2013)
44. Boit, C., et al.: From IC debug to hardware security risk: the power of backside access and optical interaction. In: 23rd international symposium on the physical and failure analysis of integrated circuits (IPFA) (2016). https://doi.org/10.1109/IPFA.2016.7564318
45. Lohrke, H., Tajik, S., Boit, C., Seifert, J.-P.: No place to hide: contactless probing of secret data on FPGAs. In: Conference on cryptographic hardware and embedded systems (CHES) (2016). https://doi.org/10.1007/978-3-662-53140-2_8
46. Rahman, M.T., et al.: Physical inspection attacks: new frontier in hardware security. In: 2018 IEEE 3rd international verification and security workshop (IVSW) (2018). https://doi.org/10.1109/IVSW.2018.8494856
47. Rahman, M.T., Tajik, S., Rahman, M.S., Tehranipoor, M., Asadizanjani, N.: The key is left under the mat: on the inappropriate security assumption of logic locking schemes. In: IEEE international symposium on hardware oriented security and trust (HOST) (2020)
48. Hamamatsu Photonics K.K.: NanoLens-SHR (2015). https://www.hamamatsu.com/resources/pdf/sys/SSMS0053E_Nanolens-SHR.pdf
49. Von Haartman, M., et al.: Optical fault isolation and nanoprobing techniques for the 10 nm technology node and beyond. In: 41st international symposium for testing and failure analysis (ISTFA) (2015)