1/0 Shades of UC: Photonic Side-Channel Analysis of Universal Circuits

Dev M. Mehta[1], Mohammad Hashemi[1], Domenic Forte[2] Shahin Tajik[1] and Fatemeh Ganji[1]

[1] Worcester Polytechnic Institute, Worcester, USA,
dmmehta2@wpi.edu,mhashemi@wpi.edu,stajik@wpi.edu,fganji@wpi.edu
[2] University of Florida, Gainesville, USA, dforte@ece.ufl.edu

Abstract. A universal circuit (UC) can be thought of as a programmable circuit that can simulate any circuit up to a certain size by specifying its *secret* configuration bits. UCs have been incorporated into various applications, such as private function evaluation (PFE). Recently, studies have attempted to formalize the concept of semiconductor intellectual property (IP) protection in the context of UCs. This is despite the observations made in theory and practice that, in reality, the adversary may obtain additional information about the secret when executing cryptographic protocols. This paper aims to answer the question of whether UCs leak information unintentionally, which can be leveraged by the adversary to disclose the configuration bits. In this regard, we propose the first photon emission analysis against UCs relying on computer vision-based approaches. We demonstrate that the adversary can utilize a cost-effective solution to take images to be processed by off-the-shelf algorithms to extract configuration bits. We examine the efficacy of our method in two scenarios: (1) the design is small enough to be captured in a single image during the attack phase, and (2) multiple images should be captured to launch the attack by deploying a divide-and-conquer strategy. To evaluate the effectiveness of our attack, we use metrics commonly applied in side-channel analysis, namely rank and success rate. By doing so, we show that our profiled photon emission analysis achieves a success rate of 1 by employing a few templates (concretely, only 18 images were used as templates).

Keywords: Side-channel Analysis · Photon emission microscopy · Universal circuits · FPGA.

1 Introduction

A universal circuit (UC) is a circuit, which upon giving its description as input (so-called configurations bits), can simulate any circuit of a maximum size [LMS16, AGKS20, ZYZL19, LYZ+21, CSR+20]. When first introduced, UCs have been thought of as a solution to share hardware for performing a set of distinct functions, all represented by a single circuitry [Val76]. Soon after their introduction, studies have been devoted to implementing the asymptotically optimal UC as proposed by Valiant [Val76]; nevertheless, it took more than four decades until modular and scalable implementations have been proposed [AGKS20, LYZ+21, DGS+22]. In addition to efforts put into improving the scalability and practicality of UCs, various works have discussed applications of UCs in multiple sub-fields of cryptography, for instance, software obfuscation [Zim15, GGH+16], batch execution of secure two-party computation [HKK+14, LR15], attribute-based encryption [GGHZ14], actively secure non-interactive secure computation [AMPR14], and private function evaluation (PFE), transforming any result in multiparty computation (MPC) to PFE [KS08, MS13, MSS14].

In fact, a UC can be seen as a keyed program for any circuit family [Zim15, LMS16]. Therefore, it could be tempting to apply UCs in the context of intellectual property (IP) protection, where less formal and trial-and-error methods such as logic locking have been practiced for years. In this regard, one of the closest notions to the loosely defined logic locking is PFE [CS22]. When it comes to PFE, the function embedded in the device is a private input of one party, whereas the input to the function is private to the other party. Given that the IC design is the private input of the IP owner, one could see the relationship between these notions. Nevertheless, logic locking and PFE follow two parallel tracks as explained in [CS22]. This can become evident as the insider adversary at the foundry observes "a lot of side information about the concealed IC design due to its unrestricted access to the opaque circuit and oracle access to an honestly-restored chip," cf. [CS22]. This fact reflects the definition of the adversary model in PFE, where only the output of the private function to a secret input can be revealed. Based on this, [CS22] has concluded that the locking mechanism should not be formalized in the PFE setting.

Despite this fact, [MGM+22] has attempted to formalize logic locking through formal simulation-based security. It has been further suggested that UC-based schemes can provably satisfy the simulation-based security definition of logic locking. [MGM+22] has also taken another step to implement UCs on field programmable gate arrays (FPGAs). In line with the objectives of [CS22, MGM+22], [BGH+22] (TCHES Volume 2022, Issue 2) has identified the issue with the gap between practice and formalism. Alongside this, flaws with previous formalization attempts, e.g., [DCSSY20], have been identified. Moreover, to establish a rigorous methodology for understanding the security of logic locking, new security definitions and syntax have been proposed. It has also been suggested that UCs are aligned with those new definitions. [BGH+22] has argued that formal definitions of logic locking need not model the leakage of side information. This argument overlooks not only practical probing attacks [EHP22, RTR+20], but also the rich field of cryptography, where it is known that "privacy is rarely absolute" [BHR12b]. This means that, interestingly enough, in the context of PFE, leakage of some information is acceptable; for instance, the size of the circuit or its topology can be revealed [BHR12b, BHR12a, CT16, CT19]. In addition, to capture the extent of the risks imposed by revealing further information about secure/private schemes, "auxiliary information" has been defined in the literature [BGJ+13, DKL09]. Therefore, albeit in different ways, the information leakage has been acknowledged in the relevant literature, [BGH+22] has suggested that the "goal of protecting a chip from side channel attacks seems to us orthogonal to the goal of logic locking—and indeed, these approaches could be layered," (see Section 5.4 in [BGH+22]). On the contrary, in line with observations in the literature [EHP22, RTR+20], we suggest that UC-based IP protection schemes must not be treated as a black box. In fact, UCs disclose more information than expected, which can be leveraged to extract their secret configuration bits and, consequently, the IP under IP protection scenarios. In this respect, to evaluate this in practice, our contributions are as follows.

Contributions.

- This paper presents the first-ever side-channel attack against UCs, where the adversary uses side-channels in the form of photon emission to reveal the secret configuration bits (see section 2.3 for more details). We emphasize that the potential risk posed by this attack is not limited to logic-locking approaches [BGH+22]; in contrast, any PFE scheme used to protect semiconductor IPs must be rethought to protect the function that is supposed to remain private during protocol execution.

- The proposed attack itself has several novel aspects thanks to methods derived from computer vision. These simple and off-the-shelf methods effectively reduce the cost of the attack compared to an attack that requires a more expensive setup. Furthermore, the combination of image processing and computer vision techniques allows us to

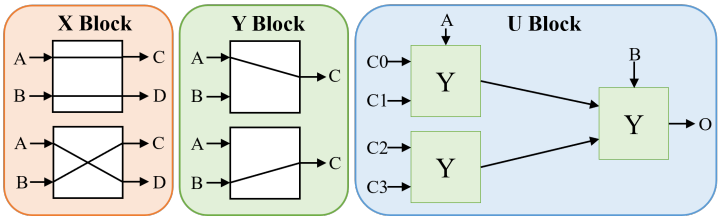


Figure 1: X-, Y-, and U-blocks are building blocks of UCs. The figure shows the possible routing configuration of each block. The X- and Y-block have 2 possible routing options based on the configuration bit. The implementation of the U-block is based on that of the Y-block. Three Y-blocks are used to create 1 U-block. The unique feature of the U-block is that the configuration bits C0-C3 are inputs of the Y-block and the selector bits in the Y-block are the inputs from the user (A & B).

reduce the cost of the attack in terms of the number of images used for profiling.

- For demonstration purposes, we present an attack against benchmark functions commonly used in the IP protection literature. These functions are converted to their associated UCs by an existing modular and scalable UC compiler [AGKS20] and implemented on FPGAs. It is noteworthy that from the point of view of our attack, any other circuit can be similarly targeted (see Section 6.4 for a discussion on this). We evaluate the cost of such attacks regarding the number of images taken from the target device during the attack phase. Depending on the specifications of the imaging setup, if the design can be captured in one image, that image is sufficient to launch the attack; otherwise, multiple images would be needed. Nevertheless, the number of images required for the attack is bounded by the square of the circuit size.

2 Background

2.1 Universal Circuits (UCs)

UCs, i.e., Valiant's UCs [Val76] are used in various applications, including secure multi-party computation protocols for private function evaluation. This is closely related to the concept of IP protection, as the idea behind these protocols is to enable two or more parties to compute a private function on their private inputs without revealing their inputs to each other. In other words, the circuit can be set up so that each party observes only its own input and the output of the circuit, not the actual function being computed or the inputs of the other parties.

More precisely, consider a (computable) Boolean function $f(x)$ that is represented as a Boolean circuit $C^g_{u,v}(x)$ with u input wires $(in_1, \cdots, in_u)$ feeding input x to the circuit, and v output wires $(out_1, \cdots, out_v$, and g gates for some u, v, and g cf. [AGKS20]. The size of this Boolean circuit is $n = u + v + g$. The UC built upon this Boolean function is a programmable circuit that can simulate *any* Boolean function up to a given size n. In order to program the UC to compute $f(x)$, configuration bits are specified as another input $c^f = \{c_1, \cdots, c_m\}$. Upon receiving c^f and x, the UC computes the result as $UC(x, c^f) = f(x)$. In this respect, without c^f provided by one of the parties, the computation cannot be performed; hence, from the IP protection perspective, c^f serves as the key to the keyed program $UC(x, c^f)$. Valiant's proposal for constructing UCs [Val76] exhibits an asymptotically size-optimal UC with size $\Theta(n \log n)$ and depth $\mathcal{O}(n)$ [Weg87]. Valiant's constructions include so-called 2-way and 4-way UCs relying on the edge-universal graphs (EUGs) that utilize recursive constructions with either 2 and 4 sub-structures (for

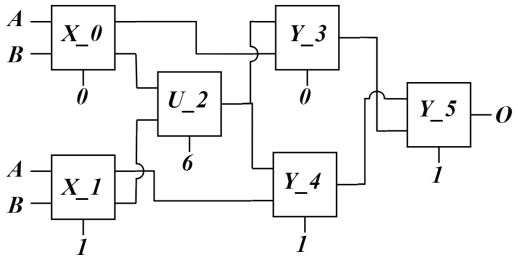


Figure 2: Schematic of a UC corresponding to a 1-bit summation. The blocks represent different UC components used to implement the summation.

more details, see, [AGKS20].

A UC consists of 3 types of switching blocks, X, Y, and U blocks; see Figure 1. An X-switching block consists of two inputs and two outputs, where the input signals are forwarded to outputs switched or not switched, depending on a *configuration bit*. A Y-switching block consists of two inputs and one output, where, depending on a configuration bit, one of the input signals will be forwarded to the output. On the other hand, the U block consists of three Y-switching blocks to realize a lookup table; see Figure 1. While the X and Y blocks are used for routing the inputs, the U block implements the standard 2 input gates.

To better understand how a UC works, we provide an example of a 1-bit summation design as shown in Figure 2. Each of the blocks present in the summation design has a specific function. For example, in the summation UC, the U-block configuration bits are "0110," which is the truth table of an XOR gate. The X and Y blocks have inputs from the circuit ports, while the configuration of the blocks is decided using the configuration bits. This configuration bits of X, Y, and U blocks are stored on the FPGA and can not be changed. The routing is used to output different functions depending on the configuration bits. In this way, the design can act as different circuits based on the routing, and to get the correct functionality of the summation, the correct configuration bits need to be programmed. U-block is different as the input is configuration, whereas the selector is the input bits. The U-block is made up of 3 Y-blocks, which each act as 2-to-1 MUX to form 4-to-1 MUX. This can be seen in Figure 1 where the configuration bits (C0-C3) are used as input to the Y-block and the user inputs (A & B) are used as the selector bits of the Y-blocks.

2.2 Photon Emission Microscopy

Photon Emission Microscopy (PEM) is a popular failure analysis tool for fault localization in complementary metal-oxide-semiconductor (CMOS) integrated circuits (ICs) [BB03]. In static states of CMOS gate, where no transistor devices are switching, the current consumption of the transistor gates are minimal. However, during a switching event, a substantial current passes through the circuit, causing the transistors to enter the saturation region for a brief period. In this state, the kinetic energy of accelerated hot carriers can be released via photon emission, with n-type transistors emitting more photons than p-type transistors due to the higher mobility of electrons. The emission rate is proportional to the switching frequency of the circuit, and raising the supply voltage exponentially increases the emitted photons.

For PEM analysis, capturing the emitted photons from the front side is challenging due to the existence of multiple interconnect layers on modern IC designs, obstructing the optical path [BB03]. However, there are no obstacles on the IC backside, facilitating the photon emission analysis. Since the silicon substrate on the IC backside is only transparent

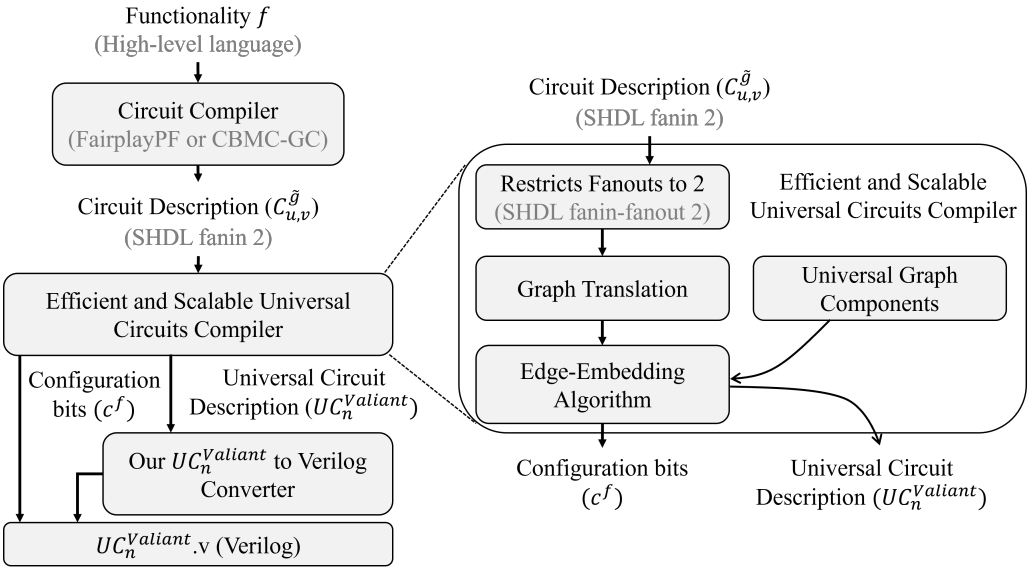


Figure 3: Flow of implementing UCs on FPGAs.

to photons with wavelengths above 1 μm, photons with the near-infrared (NIR) spectrum range can escape the silicon. To observe these photons NIR cameras, such as CCD and InGaAs, are deployed. Photon(ic) emission analysis have already been utilized as both offensive [FH08, SNK+12, KNSS13, TNH+14, TDF+14] and defensive [SSA14, SSW+14] side-channel tool in the literature.

2.3 Threat Model

For our adversary model, we consider an adversary interested in IP piracy. In the case of application-specific integrated circuits (ASICs), the adversary could be an untrusted foundry with access to all IP design details. In the case of the FPGAs, an attacker could be the IP integrator during the system design. We assume that the adversary has access to all design details of the UC circuit (e.g., its netlist, placement, and routing) except for the configuration bits. After fielding the FPGA, the configuration bits will be loaded securely into the FPGA. Otherwise, they could have been pre-programmed into user fuses or other available NVMs (e.g., in the case of Flash-based FPGAs) to configure the UC circuit. Note that stored configuration bits could be distributed all over the chip, which makes their direct readout challenging and expensive. We further assume that the fielded FPGA has proper (i.e., side-channel resistant) bitstream protection such that the adversary cannot access the plaintext bitstream for reverse engineering and IP extraction. Finally, we assume that our adversary can access a photon emission microscope and feed inputs to the UC circuit.

3 UC Implementation Flow for FPGAs

We used the compiler proposed by Alhassan et al. [AGKS20], which delivers one of the most efficient implementations of UCs. Their modular open-source implementation [Enc16] is highly scalable thanks to the hybrid UC construction combining Valiant's 2-way and 4-way constructions. Here, we briefly explain the mechanism underlying their method and how we adapt that to implement UCs on FPGAs. Figure 3 shows the high-level schematic

of our FPGA implementation flow of UCs. This flow consists of two main steps: (1) the generation of c^f and $UC_n^{Valiant}$ from the target circuit using Alhassan et al. [AGKS20] compiler, and (2) conversion of $UC_n^{Valiant}$ into a Verilog file using our $UC_n^{Valiant}$ to Verilog converter.

Generating c^f and $UC_n^{Valiant}$ from the target circuit. Given a fanin-2 circuit $C_{u,v}^{\tilde{g}}$ delivered by the circuit compiler, the configuration bits c^f, and the universal circuit description $UC_n^{Valiant}$ are generated for the high-level description of the targeted circuit $f(x)$. The steps taken in this respect are as follows.

- **Graph translation:** Involves converting a circuit into a directed acyclic graph (DAG), where each node represents a logic gate and edges represent connections between these gates. This graph formulation is important for the next steps, as it allows for the application of graph-theoretic algorithms.
- **Edge-embedding algorithm:** This step mathematically maps the edges of the original circuit's graph into a larger, pre-defined universal graph. The key here is to ensure that the mapping preserves the circuit's logical structure and connectivity.
- **UC generation:** Here, the UC is constructed based on the edge-embedded graph. This involves calculating the layout and configuration of the UC's gates and switches to make it capable of emulating any circuit of a specified size. The output of this process is $UC_n^{Valiant}$.
- **Programming the UC:** The final step involves setting the configuration bits within the UC. These bits determine the state of the UC's switches, effectively "programming" it to mimic the functionality of the original circuit. This results in obtaining the configuration bits c^f.

Our $UC_n^{Valiant}$ to Verilog converter. The generated c^f and $UC_n^{Valiant}$ cannot be implemented on FPGAs as the $UC_n^{Valiant}$ is just a representation of the circuit, which does not have a suitable Verilog or VHDL functionality to be implemented on FPGA. Therefore, we have developed our $UC_n^{Valiant}$ to Verilog converter, which parses the $UC_n^{Valiant}$ and generates the corresponding Verilog representation of UC. For this, it first parses $UC_n^{Valiant}$ and finds every $X-$, $Y-$, $U-$ blocks, and their connections. Then, it constructs those blocks using XOR and AND gates following their definitions in Alhassan et al. [AGKS20] approach. In the next step, our $UC_n^{Valiant}$ to Verilog converter makes all the connections between the XOR and AND gates generated based on UC X, Y, and U gate corresponding to the $UC_n^{Valiant}$. Finally, our $UC_n^{Valiant}$ to Verilog converter parses c^f and inserts the corresponding configuration bit in each XOR and AND gate and generates a ready-to-be-implemented UC Verilog file from c^f and $UC_n^{Valiant}$.

4 Experimental Setup

4.1 Device Under Test

We used a Genesys 2 development kit for all the experiments. This kit has an AMD/Xilinx Kintex 7 (XC7K325T-2FFG900C) FPGA manufactured with 28 nm technology. The FPGA die is packaged in a flip-chip package. Hence, by removing the fan and the heat spreader, we were able to get access to the backside silicon of the FPGA, see Fig 4(a). We did not perform any other modifications (e.g., silicon polishing or thinning) to the package or board. For all the experiments, the FPGA core is supplied by 1.0 V and the global clock operates at 555 MHz. Increasing the clock frequency could cause more frequent switching of the logic gates, leading to a higher photon emission rate.

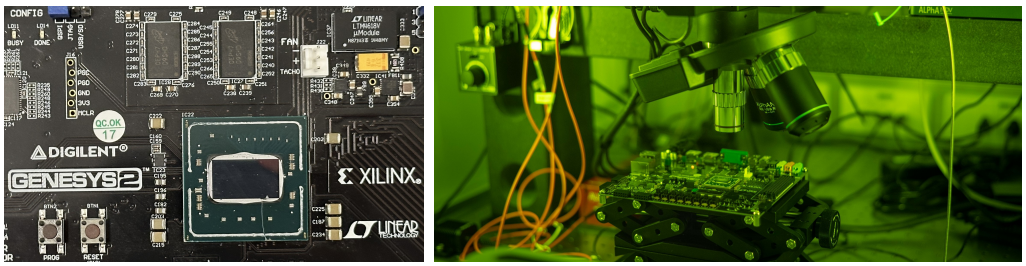


Figure 4: (a) The heatsink has been removed to get access the backside silicon of the FPGA. (b) The ALPhANOV InGaAs camera and microscope have been utilized for PEM.

4.2 Photon Emission Setup

We used an ALPhANOV S-LMS [Alp23] for NIR and emission microscopy. The microscope consists of a camera system for capturing images and a lens on an XYZ stage to focus on a region. The lens in the setup is 20x Ultra High Resolution (NA=0.6) with a typical field of 480x380 μm. The camera system consists of an InGaAs camera to detect the photon emission from the DUT. The camera is cooled down thermoelectrically to $-25\,^{\circ}\mathrm{C}$ to minimize the noise caused by the dark current. The resolution of our InGaAs camera is 640×512 px^2 where the size of each pixel is 15×15 μm^2. The combined setup is controlled using the software and hardware switches to control the XYZ stage and camera options. The software provides 2 views, an IR view of the die which is used for navigation, and a photon emission view which shows the captured photons. The software also has image processing capabilities such as overlap and dark image subtractions in addition to fine controls of the camera capture parameters, such as integration time, gain, and frame rates. The dark image subtraction is a major feature in removing the dark current noise. The integration time of the capture is kept high to get a long capture time per image. The integration time for the image was kept at 5000ms with a frame rate of 0.1Hz and a gain of 400. This allows to capture high amount of photon emission.

4.3 Hardware Implementation

The FPGA and the emission microscope with controlling devices form our experimental setup as shown in Figure 5. First, the FPGA needs to be configured by a bitstream. The placement of the designs for all the bitstreams is known apriori. The bitstream is programmed using Vivado 2021 [Xil21] from a computer. An Arduino kit is connected to the FPGA for changing inputs to the design using PMOD pins. The signals from PMOD pins activate routing circuits to either connect ring oscillators (flipping input generator) or constant input drivers to the UC design input. We send control messages to the Arduino using the computer over UART for this purpose. The DUT is now ready for capture. The microscope is moved and focused on the region of interest for capturing using the XYZ controls. Once focused, the software is used to capture images. These images are post-processed using MATLAB. To achieve consistency in the images, the microscope is not moved in XY dimension for a UC design once the experiment starts. However, refocusing might be needed from time to time due to the vibration present in the environment.

The process for the identification of the configuration bits is accomplished using the emission from LUT and routing. To make it efficient and accurate, an image processing pipeline for the captured raw images has been implemented. A detailed explanation of the capturing process and image processing pipeline is provided in the next section.

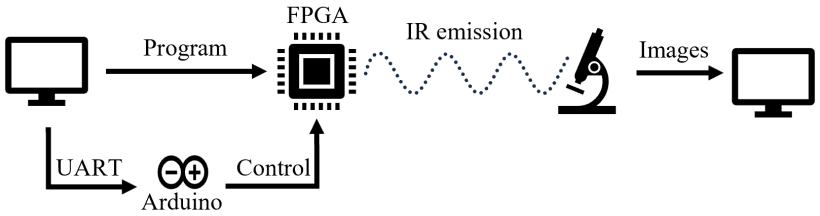


Figure 5: The experimental setup. This includes the PC used to program the bitstream, Arduino board to control the inputs, microscope to capture photons, and another PC to post-process the images.

Table 1: The input patterns used for the photon emission analysis and their impact on the output of the blocks. Each block has 2 inputs A and B and hidden configuration bits. The "F" denotes the flipping pattern. For U-blocks, the outputs are configuration bits that alternate every clock cycle. Hereafter, we refer to these patterns by their numbers, e.g., input pattern 1 means that the input B is set to "0", whereas input A is flipping.

Input Pattern	A	B	X output		Y output		U output
			c = 0	c = 1	c = 0	c = 1	
1	F	0	(F,0)	(0,F)	F	0	C0 & C2
2	F	1	(F,1)	(1,F)	F	1	C1 & C3
3	0	F	(0,F)	(0,F)	0	F	C0 & C1
4	1	F	(1,F)	(F,1)	1	F	C2 & C3
5	F	F	(F,F)	(F,F)	F	F	C0 & C3
6	F	*F*	(F,*F*)	(*F*,F)	F	*F*	C1 & C2

5 Attack Approach

The process of bits extraction has two main components. First, we need to acquire images from the chip backside and then process the images through approaches borrowed from image processing. The processed images are then employed to extract the configuration bits through computer vision.

5.1 Capturing Images

For PEM, we first start with capturing the images, followed by post-processing those images to find the configuration bits. We begin with programming the FPGA with the desired design. There are two types of design, namely, profiling and attack designs. The profiling set consists of designs used to form a dataset of the emission fingerprint of each block. The attack designs are the UC designs under attack by our approach.

The PEM captures the activity of switching transistors. To perform our attack, we force the transistors to switch on the FPGA by toggling the inputs of the UC. We consider 2-input patterns as all the blocks used in the UC are 2-input blocks. The X and Y blocks are used for routing, i.e., the output values will be connected to specific input-based configuration bits. For example, for a Y block, if input A is connected to the flipping bit and the output also shows flipping values, the configuration bit is 0. Otherwise, the configuration bit is 1 as seen in Table 1.

Similar methods can also be used for X-block to predict its behavior, as illustrated in Table 1. This leads to patterns where both inputs are flipped alternatively and the other input is kept constant, making up for 2 patterns.

However, U-block configuration bits are not that easy to identify. A U-block acts as a universal gate where the 4 configuration bits act as the input of the MUX, and the 2 inputs act as the selector. This behavior is very similar to LUTs in an FPGA. Therefore, flipping

the input does not translate to a flipping behavior at the output, and thus, emission is not guaranteed in contrast to the case of X and Y blocks. Thus, more input patterns need to be added for the identification of the configuration bits in a U-block. Hence, we add 4 more patterns to the 2 existing patterns, making a total of 6 input patterns for our experiments, as shown in Table 1. In the table, the last column shows all the configuration bits that would alternate for a particular input pattern. All 6 input patterns give unique configuration bits switching at the output of the U-block. Therefore, we have unique emission patterns for each case, which provides better profiling for all the cases of U-block. There is no guarantee of an emission for the U-block in each input pattern case. For example, if the configuration bits are '1010' then pattern 1 would not show any emission at the output, the output is alternating between 1 (C0) and 1 (C2), while input pattern 3 would show emission at the output. In some cases, it is also possible that the output might not have flipping values, but the intermediate Y-blocks might have flipping values. In this case, the LUT corresponding to that Y-block shows emissions in the LUT and their local output. With the added input patterns, repetition is observed for the X and Y blocks.

The Xilinx implementation floor plan is used as a guide in finding the point of interest on the FPGA using the XYZ stage of the microscope. The navigation view of the NIR camera is used to achieve precise focus and navigation. This process can be assisted by placing ring oscillators (RO) around the design, as the ROs are very bright and easier to spot. This could be done on an extra FPGA with the same die as the victim FPGA. Once the design is located, we use the input patterns discussed above to get 6 different images for the design. The inputs are controlled through the Arduino board using a PC. These photon emission prints are dependent on the type of LUT used for the implementation. For all the experiments, we used the same type of LUTs to form the profiling set of photon emission fingerprints. These LUTs are the same as the ones used in the attack designs, too. Note that while the emission patterns do not depend on LUT locations, they depend on the orientation of the LUTs. For example, some LUTs could be fabricated and placed mirrored to others (as previously observed on Intel/Altera FPGAs [TNH+14]), and thus in the post-processing rotation has to be applied to get the images in the same orientation. For the profiling phase, we change the configuration bits and get 6 images for each configuration bit. However, for the attack phase, we only need 6 images for the whole design. Now that we have all the images we require, we explain the post-processing steps required to extract the configuration bits.

5.2 Image Processing

The workflow of our computer vision-based algorithm includes cropping, filtering, contrast enhancement, feature extraction, and a bag of feature-based image retrieval, as shown in Figure 7. Each step is taken by applying off-the-shelf algorithms to emphasize how straightforward the attack is. For this purpose, in each step, we also provide an example of the command given in Matlab [Mat23]. Here, we briefly explain what each of the steps is used for.

Registration. Registration is an image processing technique to align figures and find the region of interest. This is used to crop each block to be processed to find the configuration bit. Specifically, registration is done to select LUTs for each block present in the design. To decide the region of interest and identify LUTs corresponding to the block, we use the implementation view in the Vivado. The relative placement of the LUTs is consistent with the flow, so one LUT location leads to the rest. In order to crop images and extract the region of interest, it is necessary to crop it out of solely one image (i.e., an image taken with one input pattern). This cropped image is used to find coordinates for cropping the rest of the images in the set using registration of the image. In MATLAB [Mat23], it is achieved using the normalized 2-D cross-correlation (*normxcorr2*) [Lew95]. The normalized

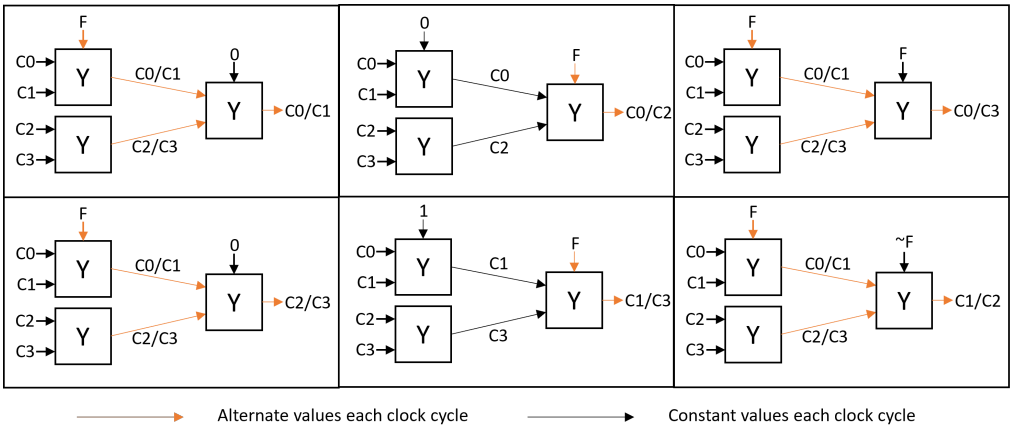


Figure 6: Each sub-figure illustrates the effect of giving one input pattern to the U-block. The orange lines represent alternate values for every clock cycle. "F" shows a flipping input of 0/1s. Y-blocks act as 2-to-1 MUX in a U-block which are controlled by the circuit inputs. C0-C3 denotes the configuration bits of the U-block.

cross-correlation compares the cropped base image to the original image from which the cropped image is derived to provide us with the exact coordinates that can crop the same portion from the rest of the image set.

Noise removal. The raw images exhibit a high noise level, interfering with the feature extraction algorithm. Therefore, the next step is to remove the noise using filtering techniques. The noise in the images is salt and pepper, i.e., randomly occurring white and black pixels, in nature (see Figure 7). So, we use median filtering in combination with bilateral filtering to take care of it. Median filtering is a non-linear digital filtering technique [Lim90]. It uses a window-based algorithm, where the median of neighboring pixel values is used as the new value for each pixel in an image. The window size could be changed to achieve different intensity of filtering. This is an adequate filter, especially when the salt and pepper noise should be removed, but the edges need to be preserved. On the other hand, bilateral filtering is a non-linear filter that is mainly used to smooth the images [TM98]. This filter also preserves the edges while smoothing the points in the image using Gaussian-based filtering. MATLAB uses a *degree of smoothing* parameter to change the intensity of smoothing. This parameter is based on the variance of noise calculated from a noisy patch of the image selected by the user. It is useful in our case as it preserves the emission spots while merging the noisier white dots with the black background. It also has other parameters related to the intensity threshold, such as *spatial sigma*, which can be used to further eliminate those smaller white dots in a controlled manner. This helps tailor the smoothing effect of the filter to our needs. We have implemented a pipeline with weak bilateral filtering, median filter, and then again, more robust bilateral filtering. We also tried the Weiner filter [Lim90], a linear time-invariant filter, and watershed segmentation algorithm [Mey94]. However, the results produced by those techniques were not satisfactory in terms of enhancing the features.

Contrast adjustment. Once the image is cleaned from noise, enhancing the intensity of LUTs before employing feature extraction is helpful. There are two reasons for this: the original contrast is not that high, and the filters will dull it even more. For this, the intensity parameter can be adjusted. The available command in Matlab [Mat23] sets the contrast limits for the grayscale image to enhance the contrast. The limits are set manually based on the filtering outputs. This will also improve the image's dynamic range, making it easy to distinguish between the remaining small amount of noise and the actual emission.

Now, the image is ready to perform feature extraction.

Feature extraction. The features are extracted from the pre-processed images. These features mainly include white dots of a specific size, representing the emissions points. For this purpose, the scale-invariant feature transform (SIFT) algorithm is purposefully chosen as it can extract features for spot detection, whereas popular algorithms like speeded-up robust feature (SURF) algorithm [BTVG06] work better with edge detection. SIFT first identifies the key points using the local intensity extrema and descriptors calculated using local image information around those key points. It is also invariant to the image scale and rotation. This is of utmost importance as the LUT emission is usually seen as circle-like light spots on a black background. The extracted features can be seen in Figure 7 marked with green circles. The settings for the SIFT extractor need to account for several parameters like contrast threshold settings and a maximum number of features. The algorithm mainly selects all the spots above a certain threshold, and the contrast enhancement in the previous step plays a vital role in avoiding noise or outliers.

Image retrieval using a customized bag of features. During profiling, the features extracted by the SIFT algorithm from the images are used to form a bag of features for each of the X/Y/U blocks with all possible input and configuration bit combinations (see Table 1). The profiling bag of features is captured once and can be applied to all the image sets with different settings. A bag of features is a set of visual words that makes a dataset for features of each image [CDF+04, NS06]. It is inspired by the document retrieval system using the bag of words. It is a common technique for implementing a content-based image retrieval system. Instead of the words used in documents, a bag of feature implementation uses the features extracted from the images. This bag is a dataset for identifying the target image (image from the circuit under attack) and its configuration bit using the image retrieval function during the attack [PCI+07].

The target image used to disclose the configuration bit also undergoes the same processing pipeline to obtain a processed clean image; however, the bag of features is *not* created for target images as that is part of the profiling step. The image retrieval process finds 2 similar images, one from the bag of features dataset (profiling set) and the other is a block cropped from the UC under attack. In doing so, the features extracted from each image are utilized to return scores, sorted from best to worst. Each image in the bag of feature datasets has a configuration bit label assigned to it. Based on the match, we disclose the configuration bits of that particular block in the attack image. This is repeated for each block of the UC in the attack image *automatically*.

Next, we look at a detailed understanding of the ranks and the metrics used to show the results.

5.3 Metrics

The metric used to measure the effectiveness of our experiment is the success rate (SR), which is in line with studies on SCA. To perform profiled SCA, the adversary utilizes a set of profiling and attack images of sizes N_P or N_A, respectively. In doing so, she computes a score vector per secret that depends on N_P or N_A. In our scenario, N_P refers to how many images with different input patterns are taken into account as a template. N_A corresponds to how many images are taken from the design to extract the configuration bit, which is $N_A = 1$ in our attacks. Suppose that following the divide-and-conquer approach, the configuration bits $c^f = \{c_1, \cdots, c_m\}$ are divided into m single configuration bits to be guessed after launching the attack. After the attack, a bit guessing vector $b = [b_i, b_j, \cdots, b_t]$ that has a decreasing order of probability, i.e., b_i corresponds to the i^{th} image template with the highest probability of correctness. Here, $\mid b \mid$ equals the number of templates. The rank R_i $(1 \leq i \leq m)$ of the correct configuration bit is defined as its position in b. For

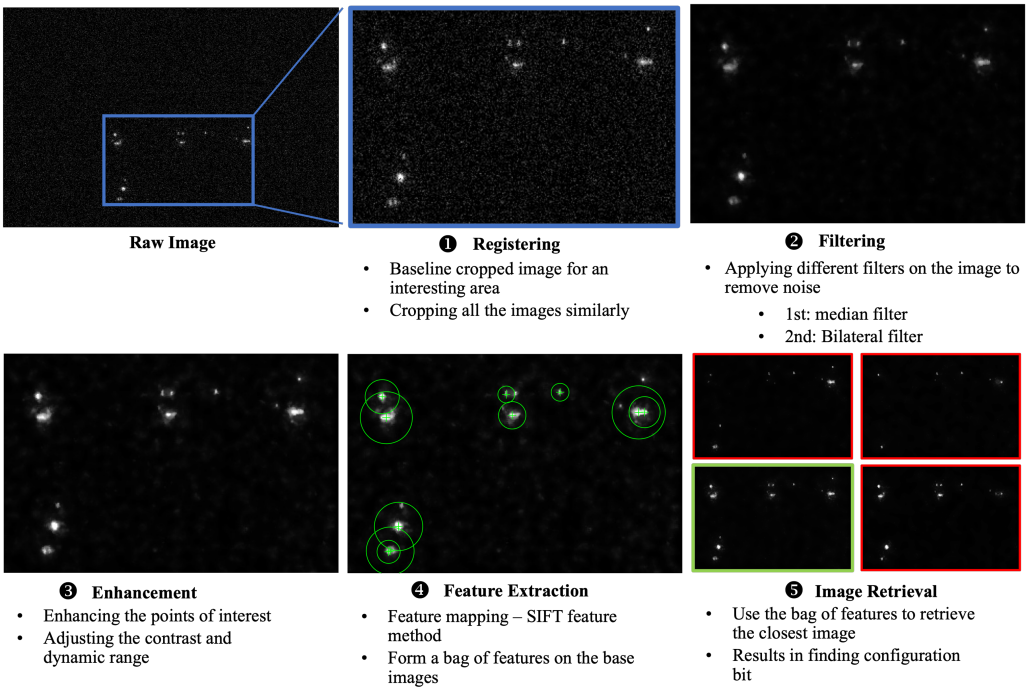


Figure 7: Steps taken by applying image processing and computer vision-based techniques to extract configuration bits.

each configuration bit, the success rate (SR) of order ORD is denoted by SR_i^{ORD} and defined as

$$SR_i^{ORD}(N_P) = \Pr[R_i \leq ORD], \quad (1 \leq i \leq m). \tag{1}$$

In other words, SR_i^{ORD} is the probability that the i^{th} correct configuration bit is ranked among the Ord first most probable configuration bits. If all the correct configuration bits are ranked first, the full key is trivially recovered. Otherwise, in order to determine the success rate of the attack for all configuration bits, following the methodology in SCA, different approaches can be taken [CPS16]. A straightforward method is a key enumeration algorithm to output all configuration bit strings (possible combination of configuration bits) from the most probable to the least probable one [LMM+16]. As the number of templates $\mid b \mid$ is small, key enumeration can be performed fast and with a much-reduced memory cost. An example, as provided below, can clarify this.

Suppose that after conducting image retrieval, when guessing the i^{th} configuration bit, the correct configuration bit is at 1^{st} position in b; hence, $R_i = 1$. If for all other configuration bits $j \neq i$ $(1 \leq j \leq m)$, $R_j = 1$, clearly $SR^1 = 1$. Now, if $R_i \neq 1$ $(1 \leq i \leq m)$, image scores delivered by the image retrieval algorithm can be mapped to integer weights required by the key enumeration in [LMM+16]. Given those weights (image scores), the rank R is defined as the number of configuration bits with a weight less than or equal to m multiplied by ORD. With this, $SR^{ORD}(N_P) = \Pr[R \leq ORD]$. Obviously, $SR^1 = 1$ if the scores (weights) of the correct images are 1.

It is noteworthy that to improve the efficiency of our attack further, we restrict the number of indices (i.e., matches) retrieved by the image retrieval algorithm to 5. This is due to the fact that the other matches sorted below 5 have low scores, thus being not useful. As a result, if the algorithm cannot find the correct configuration bits within the top 5 guesses, no match will be reported (denoted by "–" in Section 6).

6 Experimental Results

This section covers the results for two different cases: summation and c17 circuit from the ISCAS85 circuit benchmark set to test our approach [HYH99]. ISCAS85 is composed of netlists associated with industrial designs. As their high-level design is unknown, they are good candidates for random logic circuits. While the summation circuit represents a small design with a limited number of X/Y/U blocks, the second example, C17 from the ISCAS85 set, demonstrates how, for a large circuit, a divide-and-conquer strategy can be deployed to extract secret configuration bits. Before looking at the results for the attack phase, we discuss the emission results for the individual blocks used in the profiling phase and their important features.

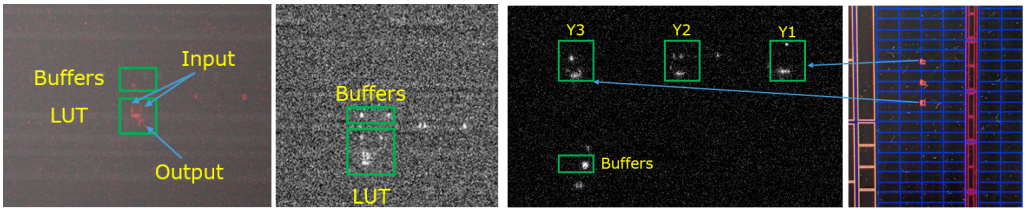
6.1 Profiling

There are 3 types of blocks in a UC, as mentioned in Section 2. We started by capturing the images from each block individually to understand the elements of the implementations present in the images as part of profiling. This helps understand their behavior in terms of LUT emissions. The raw images from the X-block and Y-block are shown in Figure 8(a). As seen in this figure, we have one instance of each block to analyze the components that emit photons. The difference in the color is because, for the sake of demonstration, the image on the left is taken with the overlay of the die image, whereas the image on the right is a raw photon image. For both blocks, we can observe the LUT and buffers of input/output of the LUT with some routing emission on the right-hand side of the LUT. The image on the left is of a Y-block with the input pattern 5 as listed in Table 1 and configuration bit of '0". Therefore, both inputs of the block are flipping, and the output is also flipping as the Y-block routes the input A to the output. This can be seen in the image with a "Y" looking shape with 2 lines on the top marked as the input and output marked at the bottom. The central horizontal illuminating bar represents the LUT containing the logic for the Y-block. We also see a few other emission points around the LUT: routing points and buffers for the LUT input/output. On the right side, we have an X-block emission image, where we can identify the LUT emissions and the buffer. Compared to the Y-block, the inputs and outputs are not that clearly distinguishable. This is due to the noise and changes in LUT logic for the X-block. For an X-block, both input and output are flipping with input pattern 6 (see Table 1) irrespective of the configuration bits. Therefore, the emission pattern can be differentiated from the Y-block.

The U-block is formed out of three Y-block, so it exhibits a similar emission pattern, but with 3 LUTs making up a block with a slight difference in buffers and routing as seen in Figure 8(b) on the left side. Figure 8(b) (right) shows the placement of the 3 Y-blocks building a complete U-block in the Vivado implementation view. All the 3 Y-blocks are captured in a single image for the profiling set, as seen in Figure 8(b) (left). We can see the difference in the routing and the buffers for the U-block, as the block only has a single output for 3 LUTs. As explained in Section 5, these images taken from the basic blocks with all possible combinations of input and configuration bits form our profiling image set. Next, we look at how the attack is carried out using the profiling set.

6.2 Example 1: Summation Circuit

The summation design includes a 1-bit summation using an XOR gate. The design, when given to the UC compiler, is converted to a 6-block design including 2 X-block, 1 U-block, and 3 Y-block as seen in Figure 2. As illustrated in the schematic, the inputs can be used to provide switching activity to all of the blocks in the design. The X_0 and X_1 blocks are directly connected to inputs, whereas the U_2 block is indirectly connected to both inputs based on the configuration bits; however, Y-blocks are not directly connected to



(a) (b)

Figure 8: (a) Y-block (left) and X-block (right) are building blocks of UCs. The figure shows raw images taken by the microscope to compare the emission from different components like LUT and buffer in blocks. (b) Raw image showing the photon emission from the U-block (left) and U-block implementation and placement from Vivado (right). The relationship between the placement of LUTs on the FPGA and the actual emission on the die can be observed here. The image is rotated when put under the microscope because of the physical placement of the device in a different position.

both inputs. Y_3 and Y_4 each have one connection to inputs and one to the output of the U-block. Lastly, Y_5 is connected to the Y_3 and Y_4.

A top module is developed to grant control of the 2 inputs using the Arduino connection on the PMOD. Then, place and route are performed to know the location of the LUTs, and the design is ready to be put under the microscope. In a real-world scenario, one could reach this stage when the netlist and the placement are known or by reverse engineering the bitstream [KEL22] to gather this information (see Section 2.3).

After capturing the images, to disclose the configuration bits of block, e.g., X_0, we perform registering on the X_0 LUT images taken from all possible input combinations (6 images in the image set). The registered images are given to the image retrieval algorithm, which matches each image to an image in the dataset (see Section 5.2). We can extract the configuration bit from the matches based on the image scores delivered by the image retrieval algorithm (see Section 5.3. This process is repeated for the X_1. For X_0, we get rank 1 in 4 of the images corresponding to different input patterns, while for X_1, the best rank is 2 as seen in Table 2. This is due to the image set quality.

The inputs are routed to the U-block through X_0 and X_1 blocks. Thus, the flipping input given to the UC reaches the U-block. Hence, we can repeat the process for disclosing the configuration bit of the next layer's U-block (U_2), i.e., employing a bag of features for image retrieval. The output of the U-block is routed to two Y-blocks. The other inputs to the Y-blocks are the outputs of the X-block. We repeat the image retrieval process for this layer using the bag created for the Y-blocks. Y_3 and Y_4 configuration bits reveal that the Y_3 routes the U-block output, and the Y_4 routes the X_1 output to the last block. Lastly, Y_5 routes the U-block output as the output of the UC with the summation functionality. The ranks for each block can be found in Table 2 with the SR of disclosing the configuration bits of the circuit. According to our results, $SR^1(1)$ and $SR^2(1)$ are at most 0.66, indicating that with only one input pattern, the SR is not satisfactory, even if the guesses ranked second are taken into account. Increasing the number of input patterns and the number of templates to 6 helps boost the success rate and achieve $SR^2(6) = 1$. Interestingly, $SR^1(6)$ has not improved much, indicating that the quality of images was not good enough. We will see in the next experiment that different sets with different contrast and image quality result in different results.

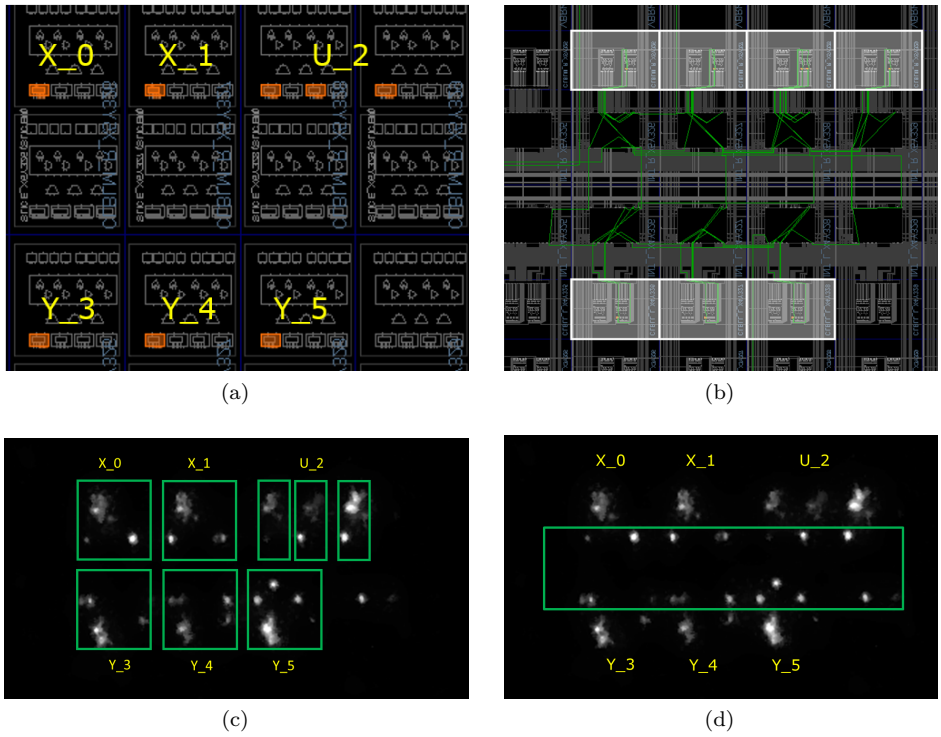


Figure 9: Configuration of the summation UC circuit for Kintex 7 FPGA board. (a) The LUTs used by each block are shown in orange. (b) The routing between the blocks. (c) and (d) show the corresponding emission of the LUTs with routing and mainly the routing nodes themselves, respectively.

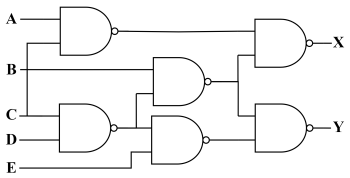


Figure 10: ISCAS85 - C17 schematic. The circuit consists of 6 NAND gates interconnected to each other to give 2 outputs.

6.3 Example 2: C17 in ISCAS85

First, we should stress that this circuit solely serves as an example of a large circuit whose UC-based implementation is targeted to extract the configuration bits. The core idea underlying our attack is that the adversary is capable of taking images from multiple portions of the design, regardless of its size, to achieve her goal. Therefore, any other benchmark circuit could have been considered.

The C17 consists of 6 NAND gates, as Figure 10 illustrates. It has 5 inputs and 2 outputs. The 6-gate circuit is converted into an 81-block design using the UC compiler. We have looked at 16 blocks connected to the first U-block in the circuit. This is done due to the camera's limitations, namely, the camera's aperture and focal length, resulting in the impossibility of capturing all the LUTs into one frame. In this part of the circuit, we are only interested in two inputs connected to the first U-block. This sub-circuit is

Table 2: The SR of disclosing the configuration bits of the UC constructed for a summation circuit. Each row represents results for an image corresponding to an input pattern. Here, "–" indicates that there was no correct guess of the configuration bit *within the top 5 guesses*.

							All Blocks			
Input pattern	**u_2**	**x_0**	**x_1**	**y_3**	**y_4**	**y_5**	SR^1**(1)**	SR^2**(1)**	SR^1**(6)**	SR^2**(6)**
1	–	1	3	1	2	2	0.33	0.66	0.66	1
2	3	–	3	1	–	3	0.18	0.18		
3	1	3	3	1	2	2	0.33	0.66		
4	–	1	2	1	2	4	0.33	0.66		
5	3	1	2	1	4	1	0.5	0.66		
6	–	1	4	1	3	1	0.5	0.5		

shown in the Figure 11(a). In this sub-circuit, 6 Y-blocks, 9 X-blocks, and 1 U-block are used. These connections are shown using the brown and yellow wires, whereas the purple wire is connected to a constant "0" to have activity in the sub-circuit we are interested in. The U-block of interest is highlighted in blue at the top right corner of the schematic; see Figure 11(a). The placement of the circuit is shown in Figure 11(b), and the resulting photon emission for input pattern 6 is shown in Figure 11(c). As can be seen in the placement, the LUTs connected to the U_24 block are placed in sequence. To determine the configuration bits of each block, we employ the same flow for post-processing and image retrieval.

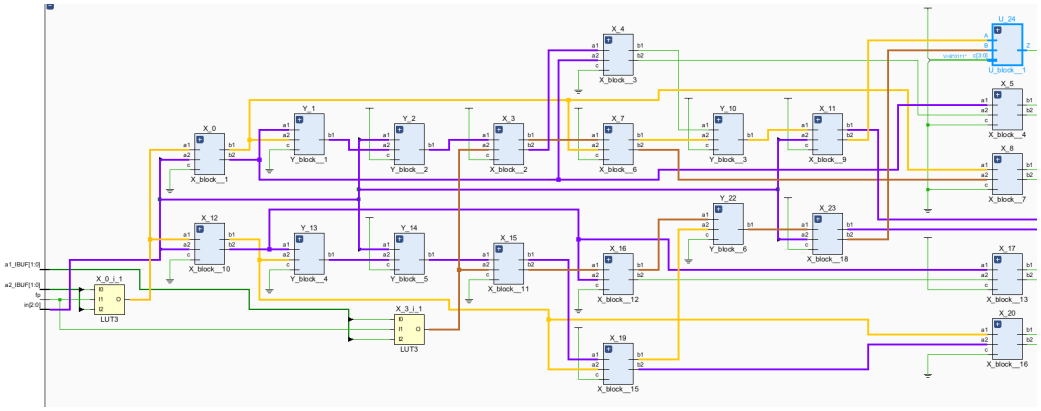
To examine how the quality of templates can lead to different results, we took 3 sets of images with each input pattern, a total of 18 images (templates). We calculate the success rate for each image individually and for each set. Interestingly enough, in our experiments, not all LUTs emit photons when flipping the inputs, thanks to the routing. This is noticed in the Figure 11(a) and (c). The schematic shows that LUTs such as Y_1 are not directly connected to the inputs that the adversary can flip. Because of this reason, some of the LUTs do not light up in the photon emission images. Moreover, in input patterns where only one of the inputs is flipping, the photon emission would not be present for some LUTs. These cases and the cases where a match could not be found are marked by "–" in Table 3. Nevertheless, as can be seen in this table, other input patterns can make LUTs emit photons in all blocks and image sets. Here, for the second and third image sets (second and third rows in the table), we get SR^1 of 1. These sets consist of images taken with various dynamic ranges (the ratio between the brightest and darkest parts of an image, adjustable for the microscope), where image set 1 and 3 exhibit the lowest and highest ratio, respectively. When considering the first image set, $SR^1 = 0.9$. We observed that one of the LUTs, corresponding to x_3, can not be recognized with a high score. The SR increases as the dynamic range increases for the sets, leading to $SR^1(6) = 1$. This shows that all the blocks can be correctly identified with proper imaging, even in a complex circuit. The next section will discuss the efficiency and complexity of successfully recovering configuration bits in different circuits.

6.4 Efficiency and Complexity

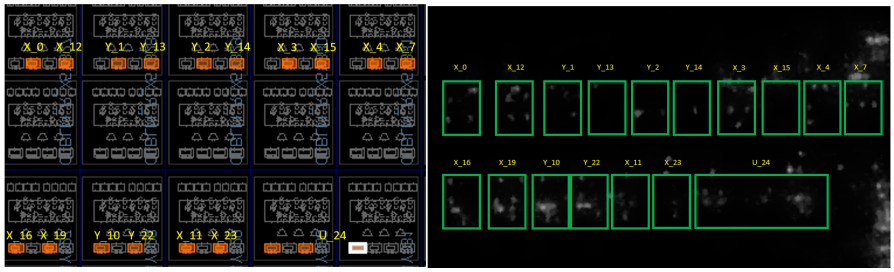
We have seen that the photon emission-based SCA can break the security of the universal circuits. In this section, we will discuss the efficiency of the attack. The attack's efficiency can be mainly attributed to the circuit size, setup, and, consequently, the image quality. The image quality can depend on both the setup and the skills of subject matter experts. Nonetheless, for a given setup, the effect of image quality on efficiency can, to some extent, be reduced through proper post-processing. As discussed in Section 6.3, our algorithm can become agnostic about the circuit size by deploying a divide-and-conquered strategy. Hence, its impact can be understood as the number of images to take. Here, we consider the number of images needed to find all the configuration bits as the metric for discussing

Table 3: The success rate of disclosing the configuration bits of the UC-based implantation of C17 sub-circuit. Each row presents results for an image taken with an input pattern as introduced in Table 1. Each image set is a repetition of the experiment with a different dynamic range of the images (the ratio between the brightest and darkest parts of an image). The ratio is smallest in the image set 1, so it is harder to distinguish between noise and the interested LUTs. The ratio increases from set 1 to 3, increasing the intensity of the LUT emission and contrast ratio of the image. Here, "–" indicates no successful guess of the configuration bit in the top 5 guesses.

													X Block		Y Block		All Blocks		
Image set	**Input Pattern**	**u_24**	**x_0**	**x_3**	**x_11**	**x_12**	**x_15**	**x_16**	**x_19**	**x_23**	**y_10**	**y_22**	SR^1**(1)**	SR^2**(1)**	SR^1**(1)**	SR^2**(1)**	SR^1**(1)**	SR^2**(1)**	SR^1**(6)**
1	1	3	–	–	1	1	1	4	1	2	–	2	0.50	0.63	0	0.50	0.36	0.55	0.91
	2	–	–	–	1	2	1	–	1	2	–	1	0.38	0.63	0.50	0.50	0.36	0.55	
	3	4	2	4	1	1	1	1	2	–	1	1	0.50	0.75	1.00	1.00	0.55	0.73	
	4	–	1	4	3	3	3	1	1	1	3	1	0.50	0.50	0.50	0.50	0.45	0.45	
	5	1	1	4	2	2	2	4	–	1	1	2	0.25	0.63	0.50	1.00	0.36	0.73	
	6	–	1	4	1	–	1	2	–	1	3	1	0.50	0.63	0.50	0.50	0.45	0.55	
2	1	2	–	–	1	5	1	–	1	1	–	1	0.50	0.50	0.50	0.50	0.45	0.55	1
	2	1	–	–	1	1	5	1	1	1	–	2	0.63	0.63	0	0.50	0.55	0.64	
	3	5	–	2	1	1	2	1	2	3	–	1	0.38	0.75	0.50	0.50	0.36	0.64	
	4	2	1	4	1	1	1	2	2	2	2	–	0.50	0.88	0	0.50	0.36	0.81	
	5	–	1	2	1	1	3	1	1	2	2	1	0.63	0.88	0.50	1.00	0.55	0.81	
	6	–	2	1	3	1	1	2	1	3	1	3	0.50	0.75	0.50	0.50	0.45	0.64	
3	1	–	–	–	1	4	1	–	–	1	–	1	0.38	0.38	0.50	0.50	0.36	0.36	1
	2	–	–	–	–	2	3	–	1	1	–	1	0.25	0.38	0.50	0.50	0.27	0.36	
	3	–	1	5	1	1	1	1	1	–	2	–	0.75	0.75	0	0.50	0.55	0.64	
	4	1	1	4	2	2	2	1	–	–	2	2	0.25	0.63	0	1.00	0.27	0.73	
	5	–	1	5	1	1	1	1	1	3	2	1	0.75	0.75	0.50	1.00	0.64	0.73	
	6	–	1	1	2	1	1	1	–	5	1	1	0.63	0.75	1.00	1.00	0.64	0.73	



(a)



(b) (c)

Figure 11: ISCAS85 - C17 placement of the sub-circuit connected to the first U-blocks (marked in blue at the top right corner): (a) shows the paths for 2 inputs connected to the U-block (orange and purple paths); (b) illustrates the placement of the circuit using the Vivado design suite, implementation view; (c) depicts the photon emission results of the LUTs in the circuit for the input pattern 6 (see Table 1).

the attack's efficiency.

As said above, the increase in circuit size will generally increase the number of images needed for the attack. More accurately, it depends on the actual placement of the LUTs on the FPGA. For example, if blocks are concentrated within a single region on the FPGA that the setup can capture, then a single image per input pattern is required, i.e., 6 images in total. On the other hand, if a large number of blocks should be placed or they are spread over the FPGA, then multiple images would be required per input pattern depending on the camera's sensor size with a particular lens.

Furthermore, for a given circuit size, the number of images can vary based on the properties of the lens, e.g., zoom and focal length. The configuration of our setup has been good enough to capture the images required to break the security of UCs studied in this paper, namely the summation circuit and part of the C17 circuit. We would have needed more than 6 images if not all the blocks had been captured in one image.

The differences in placement, circuit size, and image quality can be observed in Figure 12. For examples given in this paper, all images could be captured in about an hour once the bitstreams are ready. This is possible thanks to how we control the inputs using the Arduino setup. This enabled us to find the region of interest, set up specific capturing settings, and take 6 images with different inputs without changing the microscope settings. This workflow makes it very efficient to capture images.

After the images are taken, post-processing efficiency is essential. In the current

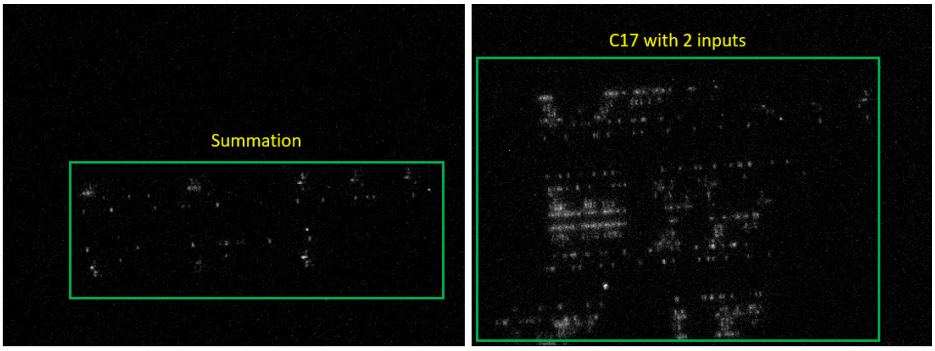


Figure 12: Raw images without cropping with the 20x zoom lens using the ALPhANOV microscope. The sub-figure on the left side shows the emission captured from the summation circuit, while the one on the right side shows the emission from the C17 circuit. The size and placement density of both circuits are visible clearly. The summation is sparsely placed, while the c17 is densely packed.

workflow, we manually extract the location of each block. These locations are constant for all images in each image set. Once this is prepared, the image retrieval algorithm can be run through the image set and find the configuration bits. This process is time-efficient; for instance, it took approximately 15 minutes for 16 blocks involved in the C17 circuit. The time for both steps would increase linearly with the number of blocks [PCI+07]. It is worth noting that for different image sets, the location might change, and location extraction might have to be repeated.

There is one more factor based on which the requirements for collecting image sets per circuit are decided. The examples considered in this paper have 2 inputs. Note that for the C17 example, we have considered 2 out of 5 circuit inputs. We should consider each possible pair of inputs to find configuration bits of other blocks present in the circuit. This ensures that all blocks in a circuit are covered, as each block has 2 inputs. Therefore, the total number of images needed will be given by the following equation:

$$N_P = 6 \times C(n, 2) = \Theta(n^2), \tag{2}$$

where $C(n, 2)$ refers to the total number of combinations for choosing 2 out of n, where n is the total number of inputs to the circuit. Here, $\Theta(\cdot)$ denotes the big-theta notation, i.e., N_P is bounded both above and below by n^2 asymptotically. The multiplication by 6 is because, for each pair, we still need to capture images for 6 input patterns (see Table 1). Using this equation as an example, if we need to find configuration bits for 81 blocks of C17, we would need 10 image sets with 6 images each.

7 Discussion

Resoution and Measurement Time. While our current setup has some limitations in terms of resolution and measurement time, both parameters could easily be improved drastically. One could use a higher magnification objective lens with a higher numeric aperture (NA) (e.g., 50X, NA=0.65) to improve the resolution for smaller technology sizes. Moreover, solid immersion lenses (SILs) can be deployed to adapt the system for sub-10 nm technology sizes. To reduce the measurement time, on the other hand, one option would be to reduce the dark current of the NIR camera to reduce the noise and increase sensitivity. While our camera can theoretically be cooled down only to -25°C, more sophisticated InGaAs cameras can be cooled down to -70°C or even below this temperature

Table 4: The table shows the comparison between the U-block implementation using 3 Y-block from the UC compiler vs. a single LUT-based implementation proposed in [DGS+23]. Here, the U-block is used in the summations circuit as in Example 1. By applying input patter 6, the configuration bit of the U-block proposed in [DGS+23] is extracted.

Input pattern	U [AGKS20]	U [DGS+23]
1	-	-
2	3	2
3	1	3
4	-	2
5	3	-
6	-	1

using water or nitrogen cooling in addition to thermoelectrical cooling. Moreover, one could increase the supplied voltage and increase the switching frequency to increase the photon emission rate. Finally, reducing the vibrations of the microscope stage, specifically for higher magnification objective lenses, could significantly improve both the resolution and measurement time, as it requires fewer iterations to capture images.

Vulnerability of a recent UC variant. Recently, [DGS+23] has proposed a more efficient implementation of UCs specifically designed for FPGAs. They have used the LUT structure directly as a U-block implementation instead of 3 Y-blocks to implement a U-block. This reduces the number of LUTs used in a circuit in most cases. They have also added support for multi-input and multi-output LUTs. We mounted our attack against this implementation by replacing the U-block in the summation circuit with a single 2-to-1 LUT U-block as suggested in [DGS+23]. After profiling this U-block implementation, we were able to achieve similar results for targeting this version of UCs as presented in Table 4. Here, the ranks for the X- and Y-blocks have not been re-calculated as no changes have been made to their implementations. This shows the susceptibility of the new version of UCs to our attack.

Potential Countermeasures. As presented throughout the paper, the nature of photonic side-channel leakage from UC circuits is directly caused by the physical placements and configuration-dependent emission fingerprints of X, Y, and U blocks. Therefore, a fundamental approach to resolving such leakage is real-time refreshing of block placements. Hardware randomization schemes [GM11, Men17, KGFT22] present methods including partial reconfiguration as hiding and moving target defenses, to secure FPGAs against more conventional SCAs, such as power and EM. Similar approaches [HPG+19] could be deployed during runtime to prevent photonic side-channel leakage. Countermeasures could also be applied at the level of the package to protect the IC backside. Otherwise, package protection schemes, such as active optical layers [AKH+20] and enclosures [VNK+15] or capacitive enclosures [IOK+18] could hide the IC package and actively monitor any attempt for removal of the enclosure.

8 Conclusion

This paper demonstrates SCA against UCs, where photon emission from the target device leaks the secret configuration bits of the UC. To the best of our knowledge, we are the first to uncover and exploit the vulnerability successfully. We take advantage of the existing modular design of UCs and off-the-shelf algorithms in our attacks. For the sake of demonstration, we recovered the configuration bits of summation and the ISCAS85-C17 designs, serving as benchmark functions, implemented in the UC format. We reported that complex large circuits can be targeted by applying the same technique and deploying a divide-and-conquer strategy. Another novel aspect of our attack is the application of computer vision-based techniques alongside image processing, which allows us to disclose the configuration bits with a minimum effort in terms of profiling.

9 Acknowledgments

This work has been supported partially by Semiconductor Research Corporation (SRC) under Task IDs 2991.001 and 2992.001 and NSF under award numbers 2117349 and 2138420.

References

[AGKS20] Masaud Y Alhassan, Daniel Günther, Ágnes Kiss, and Thomas Schneider. Efficient and scalable universal circuits. *Journal of Cryptology*, 33(3):1216–1271, 2020.

[AKH+20] Elham Amini, Tuba Kiyan, Norbert Herfurth, Anne Beyreuther, Christian Boit, and Jean-Pierre Seifert. Second generation of optical ic-backside protection structure. In *2020 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, pages 1–5. IEEE, 2020.

[Alp23] AlphaNOV. S-lms. [Online]https://www.alphanov.com/en/products-services/single-laser-fault-injection [Accessed: Dec.1, 2023], 2023.

[AMPR14] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 387–404. Springer, 2014.

[BB03] Michael R Bruce and Victoria J Bruce. Abcs of photon emission microscopy. *Electronic Device Failure Analysis*, 5:13–22, 2003.

[BGH+22] Peter Beerel, Marios Georgiou, Ben Hamlin, Alex J Malozemoff, and Pierluigi Nuzzo. Towards a formal treatment of logic locking. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (2):92–114, 2022.

[BGJ+13] Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 316–334. Springer, 2013.

[BHR12a] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18*, pages 134–153. Springer, 2012.

[BHR12b] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proc. of the 2012 ACM Conf. on Computer and communications security*, pages 784–796, 2012.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 404–417. Springer, 2006.

[CDF+04] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[CPS16] Marios O Choudary, Romain Poussier, and François-Xavier Standaert. Score-based vs. probability-based enumeration–a cautionary note. In *Progress in Cryptology–INDOCRYPT 2016: 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings 17*, pages 137–152. Springer, 2016.

[CS22] Animesh Chhotaray and Thomas Shrimpton. Hardening circuit-design ip against reverse-engineering attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1672–1689. IEEE, 2022.

[CSR+20] Rosario Cammarota, Matthias Schunter, Anand Rajan, Fabian Boemer, Ágnes Kiss, Amos Treiber, Christian Weinert, Thomas Schneider, Emmanuel Stapf, Ahmad-Reza Sadeghi, et al. Trustworthy ai inference systems: An industry research view. *arXiv preprint arXiv:2008.04449*, 2020.

[CT16] Henry Carter and Patrick Traynor. Opfe: Outsourcing computation for private function evaluation. 2016. https://eprint.iacr.org/2016/067. doi:10.1504/ijics.2019.103052.

[CT19] Henry Carter and Patrick Traynor. Outsourcing computation for private function evaluation. *International Journal of Information and Computer Security*, 11(6):525–561, 2019.

[DCSSY20] Giovanni Di Crescenzo, Abhrajit Sengupta, Ozgur Sinanoglu, and Muhammad Yasin. Logic locking of boolean circuits: Provable hardware-based obfuscation from a tamper-proof memory. In *Innovative Security Solutions for Information Technology and Communications: 12th International Conference, SecITC 2019, Bucharest, Romania, November 14–15, 2019, Revised Selected Papers*, pages 172–192. Springer, 2020.

[DGS+22] Yann Disser, Daniel Günther, Thomas Schneider, Maximilian Stillger, Arthur Wigandt, and Hossein Yalame. Breaking the size barrier: Universal circuits meet lookup tables. Cryptology ePrint Archive, Paper 2022/1652, 2022. https://eprint.iacr.org/2022/1652. URL: https://eprint.iacr.org/2022/1652.

[DGS+23] Yann Disser, Daniel Günther, Thomas Schneider, Maximilian Stillger, Arthur Wigandt, and Hossein Yalame. Breaking the size barrier: Universal circuits meet lookup tables. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–37. Springer, 2023.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 621–630, 2009.

[EHP22] Susanne Engels, Max Hoffmann, and Christof Paar. A critical view on the real-world security of logic locking. *Journal of Cryptographic Engineering*, 12(3):229–244, 2022.

[Enc16] Encryptogroup. v2021.1. [Online]https://github.com/encryptogroup/UC [Accessed: Oct.15, 2023], 2016.

[FH08] Julie Ferrigno and M Hlaváč. When aes blinks: introducing optical side channel. *IET Information Security*, 2(3):94–98, 2008.

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure attribute based encryption from multilinear maps. *Cryptology ePrint Archive*, 2014.

[GM11] Tim Güneysu and Amir Moradi. Generic side-channel countermeasures for reconfigurable devices. In *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13*, pages 33–48. Springer, 2011.

[HKK+14] Yan Huang, Jonathan Katz, Vladimir Kolesnikov, Ranjit Kumaresan, and Alex J. Malozemof. Amortizing garbled circuits. In *Annual Cryptology Conf.*, pages 458–475. Springer, 2014.

[HPG+19] Benjamin Hettwer, Johannes Petersen, Stefan Gehrer, Heike Neumann, and Tim Güneysu. Securing cryptographic circuits by exploiting implementation diversity and partial reconfiguration on fpgas. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 260–263. IEEE, 2019.

[HYH99] M. C. Hansen, H. Yalcin, and J. P. Hayes. Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. *IEEE Design Test of Computers*, 16(3):72–80, 1999. doi:10.1109/54.785838.

[IOK+18] Vincent Immler, Johannes Obermaier, Martin König, Matthias Hiller, and Georg Sig. B-trepid: Batteryless tamper-resistant envelope with a puf and integrity detection. In *2018 ieee international symposium on hardware oriented security and trust (host)*, pages 49–56. IEEE, 2018.

[KEL22] Sahand Kashani, Mahyar Emami, and James R Larus. Bitfiltrator: A general approach for reverse-engineering xilinx bitstream formats. In *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, pages 01–08. IEEE, 2022.

[KGFT22] David S Koblah, Fatemeh Ganji, Domenic Forte, and Shahin Tajik. Hardware moving target defenses against physical attacks: Design challenges and opportunities. In *Proceedings of the 9th ACM Workshop on Moving Target Defense*, pages 25–36, 2022.

[KNSS13] Juliane Krämer, Dmitry Nedospasov, Alexander Schlösser, and Jean-Pierre Seifert. Differential photonic emission analysis. In *Constructive Side-Channel Analysis and Secure Design: 4th International Workshop, COSADE 2013, Paris, France, March 6-8, 2013, Revised Selected Papers 4*, pages 1–16. Springer, 2013.

[KS08] Vladimir Kolesnikov and Thomas Schneider. A practical universal circuit construction and secure evaluation of private functions. In *Financial Cryptography and Data Security: 12th International Conference, FC 2008, Cozumel, Mexico, January 28-31, 2008. Revised Selected Papers 12*, pages 83–97. Springer, 2008.

[Lew95] John P Lewis. Fast normalized cross-correlation. *Vision Interface, 1995*, 95:120, 1995.

[Lim90] Jae S Lim. Two-dimensional signal and image processing. *Englewood Cliffs*, 1990.

[LMM+16] Jake Longo, Daniel P Martin, Luke Mather, Elisabeth Oswald, Benjamin Sach, and Martijn Stam. How low can you go? using side-channel data to enhance brute-force key recovery. *Cryptology ePrint Archive*, 2016.

[LMS16] Helger Lipmaa, Payman Mohassel, and Saeed Sadeghian. Valiant's universal circuit: Improvements, implementation, and applications. *Cryptology ePrint Archive*, 2016.

[LR15] Yehuda Lindell and Ben Riva. Blazing fast 2pc in the offline/online setting with security for malicious adversaries. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 579–590, 2015.

[LYZ+21] Hanlin Liu, Yu Yu, Shuoyao Zhao, Jiang Zhang, Wenling Liu, and Zhenkai Hu. Pushing the limits of valiant's universal circuits: simpler, tighter and more compact. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*, pages 365–394. Springer, 2021.

[Mat23] MathWorks. version 9.14.0.2239454 (r2023a). [Online]https://www.mathworks.com/products/matlab.html [Accessed: Jan.15, 2024], 2023.

[Men17] Nele Mentens. Hiding side-channel leakage through hardware randomization: A comprehensive overview. In *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 269–272. IEEE, 2017.

[Mey94] Fernand Meyer. Topographic distance and watershed lines. *Signal processing*, 38(1):113–125, 1994.

[MGM+22] Elisaweta Masserova, Deepali Garg, Ken Mai, Lawrence Pileggi, Vipul Goyal, and Bryan Parno. Logic locking - connecting theory and practice. Cryptology ePrint Archive, Paper 2022/545, 2022. https://eprint.iacr.org/2022/545. URL: https://eprint.iacr.org/2022/545.

[MS13] Payman Mohassel and Saeed Sadeghian. How to hide circuits in mpc an efficient framework for private function evaluation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 557–574. Springer, 2013.

[MSS14] Payman Mohassel, Saeed Sadeghian, and Nigel P Smart. Actively secure private function evaluation. In *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20*, pages 486–505. Springer, 2014.

[NS06] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. Ieee, 2006.

[PCI+07] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[RTR+20] M Tanjidur Rahman, Shahin Tajik, M Sazadur Rahman, Mark Tehranipoor, and Navid Asadizanjani. The key is left under the mat: On the inappropriate security assumption of logic locking schemes. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 262–272. IEEE, 2020.

[SNK+12] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple photonic emission analysis of aes: photonic side channel analysis for the rest of us. In *Cryptographic Hardware and Embedded Systems–CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14*, pages 41–57. Springer, 2012.

[SSA14] Franco Stellari, Peilin Song, and Herschel A Ainspan. Functional block extraction for hardware security detection using time-integrated and time-resolved emission measurements. In *2014 IEEE 32nd VLSI Test Symposium (VTS)*, pages 1–6. IEEE, 2014.

[SSW+14] Franco Stellari, Peilin Song, Alan J Weger, Jim Culp, A Herbert, and Dirk Pfeiffer. Verification of untrusted chips using trusted layout and emission measurements. In *2014 IEEE international symposium on hardware-oriented security and trust (HOST)*, pages 19–24. IEEE, 2014.

[TDF+14] Shahin Tajik, Enrico Dietz, Sven Frohmann, Jean-Pierre Seifert, Dmitry Nedospasov, Clemens Helfmeier, Christian Boit, and Helmar Dittrich. Physical characterization of arbiter pufs. In *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16*, pages 493–509. Springer, 2014.

[TM98] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 839–846. IEEE, 1998.

[TNH+14] Shahin Tajik, Dmitry Nedospasov, Clemens Helfmeier, Jean-Pierre Seifert, and Christian Boit. Emission analysis of hardware implementations. In *2014 17th Euromicro Conference on Digital System Design*, pages 528–534. IEEE, 2014.

[Val76] Leslie G Valiant. Universal circuits (preliminary report). In *Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 196–203, 1976.

[VNK+15] Michael Vai, Ben Nahill, Josh Kramer, Michael Geis, Dan Utin, David Whelihan, and Roger Khazan. Secure architecture for embedded systems. In *2015 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–5. IEEE, 2015.

[Weg87] Ingo Wegener. *The complexity of Boolean functions*. John Wiley & Sons, Inc., 1987.

[Xil21] Inc. Xilinx. v2021.1. [Online]https://www.xilinx.com/products/design-tools/vivado.html [Accessed: Jan.11, 2023], 2021.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 439–467. Springer, 2015.

[ZYZL19] Shuoyao Zhao, Yu Yu, Jiang Zhang, and Hanlin Liu. Valiant's universal circuits revisited: an overall improvement and a lower bound. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 401–425. Springer, 2019.