Bake It Till You Make It

Heat-induced Power Leakage from Masked Neural Networks

Dev M. Mehta*[1], Mohammad Hashemi*[1], David S. Koblah[2], Domenic Forte[2] and Fatemeh Ganji[1]

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

Abstract. Masking has become one of the most effective approaches for securing hardware designs against side-channel attacks. Regardless of the effort put into correctly implementing masking schemes on a field-programmable gate array (FPGA), leakage can be unexpectedly observed. This is due to the fact that the assumption underlying all masked designs, i.e., the leakages of different shares are independent of each other, may no longer hold in practice. In this regard, extreme temperatures have been shown to be an important factor in inducing leakage, even in correctly-masked designs. This has previously been verified using an external heat generator (i.e., a climate chamber). In this paper, we examine whether the leakage can be induced using the circuit components themselves *without* making any changes to the design. Specifically, we target masked neural networks (NNs) in FPGAs, one of the main building blocks of which is block random access memory (BRAM). In this respect, thanks to the inherent characteristics of NNs, our novel internal heat generators leverage solely the memories devoted to storing the user's input, especially when frequently writing alternating patterns into BRAMs. The possibility of observing first-order leakage is evaluated by considering one of the most recent and successful first-order secure masked NNs, namely ModuloNET. ModuloNET is specifically designed for FPGAs, where BRAMs are used to store inputs and intermediate computations. Our experimental results demonstrate that undesirable first-order leakage can be observed and exploited by increasing the temperature when an alternating input is applied to the masked NN. To give a better understanding of the impact of extreme heat, we further perform a similar test on the design using an external heat generator, where a similar conclusion can be drawn.

Keywords: Side-channel Analysis · Masking · Neural Networks · Heat Generation · T-test · DPA · FPGA.

1 Introduction

Deep learning (DL) accelerators have become an integral part of Internet of things (IoT) edge devices that support image classification, speech recognition, etc. [LeC19]. In this regard, mobile and wearable devices require a low-cost neural network (NN) accelerator to support DL inference in cameras, medical devices, ground maintenance systems, video games, and so on. In these applications, an NN is handed to the users, whose preparation requires a significant investment of money and time in order to train it against a (relatively) huge training dataset and tune its hyperparameters and parameters. Hence, a malicious user may attempt to extract the NN's parameters and hyperparameters (two valuable assets of the accelerator's designer) [BBJP19].

* These authors contributed equally to this work.

Standard protections, e.g., blocking JTAG access, blocking binary readback, code obfuscation, etc., could be taken into account against reverse-engineering and binary analysis attacks [BBJP19]. However, these protections cannot prevent an attacker from extracting the assets of NNs through side-channel analysis (SCA) [BBJP19, DCA20b, XCC+20, YKO+20]. Masking schemes are among the most widely studied countermeasures to protect cryptographic primitives against SCA and have been one of the first solutions discussed in the context of protecting accelerators [DCA20b, DCA20a, DAP+22]. Masking schemes have been proven to be secure against SCA, even with higher orders, although they impose a high cost of overhead on the design, need careful construction and implementation, and may suffer from high latency as well as the fresh randomness requirement. Interestingly, recent masked NNs have overcome the challenge facing them with regard to the masking overhead and have proven that their approach is resilient against SCA in the order of million traces [DAP+22, DCA20b]. Nevertheless, despite the effort put into designing a masked scheme –irrespective of the scheme's underlying function– unexpected leakage can be exhibited when realizing the masked design [DCEM18]. This failure has been mainly attributed to the fact that leakages of different shares are no longer independent under some specific conditions, e.g., high temperature, high clock frequency, etc [DFS19]. [DCEM18] has further extended and practically evaluating the theoretical results of in [DFS19]. The work in [DCEM18] can be thought of as precise experimentation and *leakage detection* within the lab environment to give a better understanding of what has been reported before in a series of work [GOKT16, SGMT18, ZS18, RPD+18]. These studies have been devoted to (1) the impact of power consumption of circuitry placed in one region of a field programmable gate array (FPGA) on the fluctuations of the power supply voltage at other, even unrelated/unconnected, regions of the FPGA [GOKT16, DCEM18]; and (2) how this effect can be exploited to conduct SCA [SGMT18, ZS18, RPD+18]. While the latter has been extensively researched, the former topic, particularly the conditions resulting in an unexpected leakage, is yet to be sufficiently investigated.

Specifically, although the impact of high temperature on the dependency of the shares has been considered in [DCEM18] by using a climate chamber, it is interesting to explore how a masked design can be intentionally exposed to such extreme heat without using an external generator, making even remote heat-induced SCA possible. When considering active attacks, i.e., fault attacks, the devastating effect of high temperature has been widely known and studied, see, e.g., [PBR17, MLS22, BBB+22, BH22]. On the other hand, little attention has been paid to how increasing the temperature could lead to leakage, and even in those relevant studies [GOKT16, DCEM18] as discussed above, an external heat generator has been employed. This is despite the fact that internal heat generators could also be applicable. Such generators have been analyzed and developed by Happe et al. [HHAP12] and Agne et al. [AHH+14], who have demonstrated that utilizing and frequently reading/writing from/to some circuit components such as block random access memory (BRAM) and flip-flop (FF) *pipelines* cause a significant rise in circuit temperature. Therefore, a natural question to ask would be whether such heat generators *already exist and can be exploited* in masked designs to cause leakage. Our paper attempts to answer this question by considering FPGA-based accelerators, which inevitably involve BRAMs in their design as their main building blocks. Additionally, as explained before, masking has been further introduced to these accelerators, making them a viable option for our study.

Contributions: Our contributions are summarized below.

- As our first step, we focus on the design of ModuloNET as presented in [DAP+22]. Although the design has not been made available, we follow the precisely described design of modules involved in ModuloNET. Our design is further verified using one of the state-of-the-art tools, namely VERICA [RBFSG22]. During the verification phase, as a byproduct, we identify a vulnerability in the hardware implementation of Goubin's binary to arithmetic (B2A) conversion algorithm [Gou01] as used in ModuloNET, which

has been overseen in the literature. We report that writing/reading sensitive variables into/from memory can cause a power leakage, which we address by slightly changing the design of ModuloNET.[1] We emphasize that this detected vulnerability complements the set of issues with such conversion algorithms recently identified in [GPM22]. We emphasize that [GPM22] has reported glitch-based issues for straightforward implementations of Coron et al.-A2B [CGV14] in hardware; on the contrary, we report leakage in Goubin's A2B implemented in hardware [Gou01]. Using the t-test and differential power analysis (DPA) [KJJ99] also help us to demonstrate that no first-order leakage exists in our ModuloNET design after resolving the issue.

- Our second contribution is the design of the first internal heat generator, which relies *on neither additional circuitry nor an external heat generator*, but solely the design –precisely the memory used to store the inputs– and the inputs crafted by the user. Compared to the ones proposed in [AHH+14, HHAP12], our generators *do not* leverage BRAM pipelines but, rather, rely on writing into the memory in a parallel manner. In doing so, **no** changes are made in the design of FPGA-based masked accelerators under attack. To assess its efficacy, we use the BRAMs *used* in ModuloNET to store the inputs. The extreme temperature is observed by simply writing alternating "1" and "0" patterns into *single-port* BRAMs (see Section 5.2 for more details). Under this condition, the power leakage of the design is successfully changed, and at some points in time, the t-scores do not always remain within the desired threshold. Furthermore, by performing DPA, we demonstrate that the observed leakage can be successfully leveraged to extract the secret weights.

- Last but not least, We verify whether similar power leakage can be induced through external heat generators. Our results demonstrate that, indeed, this is possible; however, this would not rule out the fact that internal heat generators should be studied. Besides making remote heat-induced SCA feasible (see Section 7 for a discussion), internal heat generators can reduce the attack's cost and complexity. This is because the adversary solely needs to write into memory without modifying the design or preparing a specific setup (no temperature chamber, for instance).

2 Related Work

2.1 Attacks against NNs

SCA and fault attacks.FPGAs are extensively used to implement DL accelerators and are supported by cloud providers; however, a major security concern about them has been side-channel analysis (SCA) and fault attacks [XAQ21] [BH22]. SCA against NNs has been successful, and therefore, many protected NN designs have emerged [DAP+22, DCA20a, DCA20b]. Some of the recent work relevant to SCA against NN include [XCC+20, YMY+20, BBJP19, DCA20b, YKO+20, XAQ21, BJP22, SGMT18, ZS18, DKAA22] – just to name a few. In this series of work, regardless of whether the attacker has physical or remote access to the device, her goal is to extract the NN model and/or parameters, being the NN's assets [TG22]. This has been achieved through observing physical leakages such as timing [BBJP19], power consumption [XCC+20], and electromagnetic emanation (EM) [BBJP19]. Another category of SCA conducted against NNs is fault-induced SCA. As an example, the authors of [LGFX21] have used SCA and power-wasting circuits in conjunction with each other for their attack. The power-wasting circuit used is a look-up table- (LUT)-based combinatorial loop to bypass combinatorial loop checkers used by cloud providers that prevent similar attacks based on ring oscillators (ROs). The advantage

[1] We believe that this could have been observed and resolved by the authors of [DAP+22] since no leakage has been reported in their paper, although they could not report the vulnerability since no verification tool was applied in their study.

of this approach is that there is no need to have prior information about the NN engine under attack due to the SCA module included to schedule the attack.

SCA, as a passive attack, is not the only type of attack mounted against NNs. Voltage and clock-based faults are widely used for fault injection. This type of tampering can cause bit-flips [KGT22], timing faults [MS19] and can even reset the design [KHEB14]. It can also lead to indirect or direct physical effects on the hardware like temperature change, timing delays, etc. [KHEB14].

NN reverse engineering attacks. Researchers have developed an attack using the ROs to steal NN parameters remotely [ZYC+21]. They take advantage of the shared power resources in a cloud FPGA setup. ROs are used as sensors to measure the power consumption of different NN operations carried out by the victim design. They train a machine learning algorithm with the implementation of different kinds of NNs. This model is used to infer the parameters of the victim circuit using the power traces. Similar results were obtained by Tian et al. using a time-to-digital converter (TDC) based SCA to reverse engineer the structure of NN remotely on the FPGA implementation of the Versatile Tensor Accelerator [TMW+21]. They obtained parameters for a multi-tenant implementation of ResNet-18 and MobileNet with different layer configurations. Similarly, Moini et al. show the use of TDC in multi-tenant FPGA setup to extract image inputs for a binary NN accelerator [MTH+21]. They have shown the results for multiple FPGAs, including Ultrascale+ FPGA from Amazon AWS F1 cloud server.

2.2 Temperature-based Attacks

Heat as a covert channel. The temperature can create a covert channel between two processes on a single machine or between two machines [BDK+09, BKMN09]. In these studies, the authors have also suggested methods to create such a covert channel between circuits to attack an implementation of RSA by employing ring oscillators that generate heat.

Heat-induced faults. Here, we briefly explain how temperature changes have been leveraged by attackers targeting various designs. Faults can be injected by external temperature manipulation, as shown by Hutter et al. [HS13]. They have shown fault generation in RSA by heating the microcontroller outside its recommended temperature tolerance. Another advantage of heating is that glitches can be easily induced when the device is at a higher temperature [KHEB14]. This can increase the efficacy of the faults. External temperature-based attacks have also been mounted to inject faults in memory [GA03, Sko09]. These attacks can be translated to accelerators as they use hardware capabilities similar to FPGAs. As another example, Korak et al. showed a clock-glitching fault attack with artificial temperature control on a micro-controller platform using an external controller [KHEB14]. It results in changes in instructions, execution order, and value changes.

When it comes to temperature-based attacks against NNs, [ATG+19] can serve as an example. Writing into dual-port BRAM has been used to inject faults in NNs [ATG+19]. It has been shown that successive write-collisions lead to high voltage consumption and an increase in temperature. Using this technique, faults were injected successfully into a neural network implementation. The heating of the chip combined with voltage drop leads to bit-flips and timing violations.

Side-channel leakage dependency on temperature. Hardware masking schemes' side-channel leakages are affected by various factors such as the supply voltage, frequency, and temperature [DCEM18]. To support this claim, Moradi et al. [DCEM18] investigated the impact of such factors and demonstrated that even with a correct implementation of a masking scheme, such implementations still exhibit unexpected leakage. They performed a wide range of experiments, targeting FPGA, and reported under what circumstances a correct implementation of masked hardware shows unexpected leakage. They studied

the effect of six factors on side-channel leakage, including the number of shares, shunt resistor, voltage supply, circuit size, design frequency, and temperature. They showed that a well-designed hardware masking implementation exhibits leakage sooner than expected under specific circumstances (i.e., at a certain frequency and temperature).

Summary. There have been multiple efforts to show the weaknesses of different implementations of NNs. On the other hand, temperature-based attacks have also been discussed in various contexts, including the security of NNs, although fault attacks can be seen as the main category in this matter. To the best of our knowledge, no study has considered circuit components-based heat generation to induce leakage, let alone in the case of a masked implementation of an NN.

3 Background

3.1 Brief Introduction to Masking Schemes

A masking scheme can be seen as a secret sharing one, where secret values are split into *shares*, and operations are conducted on them in such a manner that a specific security objective is achieved. In this regard, a masking scheme aims to offer security at a given order d by making a set of assumptions on the leakage behavior of the target device.

Boolean masking. One of the most common forms of masking is Boolean masking, where binary addition is adopted to share the sensitive variables. In doing so, a sensitive value $x \in GF(2^m)$ is divided to $d+1$ shares $(x_1, \cdots, x_{d+1})$ such that $x = \bigoplus_{i=1}^{d+1} x_i$. The security requirement for this scheme is the uniformity of the shares, guaranteed by drawing shares $x_1, \cdots, x_d$ from a uniform random distribution and by choosing $x_{d+1} = x \oplus \bigoplus_{i=1}^{d} x_i$ (so-called correctness property). It is straightforward to see that linear and affine functions can be securely evaluated when applying Boolean masking cf. [DCEM18]. On the other hand, non-linear functions need further attention. This becomes more evident if we take masked multiplication as an example into account. In that case, when computing $z = xy$, a total of $(d+1)^2$ terms contribute to the output, which should be reduced to $(d+1)$ shares. For this purpose, various Boolean masking schemes have been proposed in the literature, although we focus solely on Domain-Oriented Masking [GMK16] used in the design of masked NNs [DAP+22].

Domain-Oriented Masking (DOM). To reduce the number of shares contributing to the multiplication output, DOM involves two steps. First, the cross-products are computed, and randomness is added to specific ones; for instance, for the first-order security, we obtain cf. [DCEM18]:

$$\begin{aligned} p_1 &= x_1 y_1 \\ p_2 &= x_1 y_2 \oplus r_1 \\ p_3 &= x_2 y_1 \oplus r_1 \\ p_4 &= x_2 y_2 \end{aligned} \tag{1}$$

After that, in the second phase, the terms p_i in the Equation (1) are synchronized in a register and introduced to a compression stage to reduce the $(d+1)^2$ shares to $(d+1)$ shares in the output z_i, where $z_1 = p_1 \oplus p_2$ and $z_2 = p_3 \oplus p_4$. This reduction is obtained at the cost of an extra clock cycle and the independence requirement imposed on the input shares [GMK16, GMK17, GM17]. Nevertheless, the DOM multiplier (Figure 1) is one of the commonly applied masking schemes whose security comes down to the independent power consumption of the component functions.

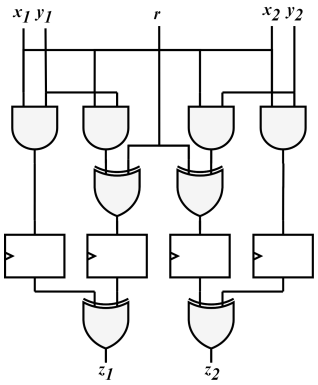

Figure 1: Domain-oriented masking (DOM) Multiplier for first-order secure computation. Here, the DOM-indep multiplier is illustrated, where the demand for fresh randomness and the area overhead in terms of gate count is significantly smaller than the DOM-dep multiplier; however, this is achieved by independently sharing the inputs.

Arithmetic masking, and conversion to Boolean masking In practice, arithmetic operations may be needed for different cryptographic (e.g., SPECK [BSS+13]) and non-cryptographic primitives (e.g., NNs) [Cor17, DMRB18, DAP+22]. Under this scenario, it can be advantageous to employ arithmetic masking. As a simple example, to compute $z = x + y \bmod 2^k$, a first-order scheme with arithmetic sharing performs the following operations. First, arithmetic shares A_1, A_2, B_1, B_2 are defined such that $x = A_1 + A_2$ and $y = B_1 + B_2$. Afterward, the shares are added separately, by letting $C_1 \leftarrow A_1 + B_1$ and $C_2 \leftarrow A_2 + B_2$ with two arithmetic shares C_1 and C_2 that can be directly added to obtain $z = x + y = A_1 + A_2 + B_1 + B_2 = C_1 + C_2$. Note that all additions and subtractions are performed modulo 2^k cf. [Cor17].

Under conditions where both Boolean and arithmetic masking schemes are needed, it is possible to convert one to the other. Specifically, a B2A conversion accepts Boolean shares and outputs arithmetic ones: Boolean shares of x that are $x_1, x_2, \cdots x_{d+1}$ should be converted to $(d+1)$ arithmetic shares a_i such that $x = a_1 + a_2 + \cdots + a_{d+1} \bmod 2^k$ without leaking any information about x. The first B2A algorithms have been introduced by Goubin [Gou01], with first-order security only (for higher-order conversion, see, e.g., [CGV14, Cor17, HT19]). The algorithm working the other way around and converting arithmetic to Boolean (A2B) has also been provided in [Gou01] and further improved in terms of complexity in [CGTV15]. As discussed in [DAP+22], conversion algorithms devised to be implemented in hardware have assumed the same field for inputs and outputs and attempted to reuse the randomness in the Boolean shares [MTMM07, Gol07]; nevertheless, [DAP+22] had to take another approach, namely concatenating each Boolean share (i.e., 1 bit) with $k-1$ fresh random bits to obtain k-bit Boolean shares. After that, the approach in [Gol07] has directly been applied as a B2A algorithm.

Leakage evaluation through t-test Test vector leakage assessment (TVLA) has been a standard test methodology used in the literature to detect side-channel leakage [DAP+22, DCA20a, DCA20b, DCEM18]. Relying on Welch's t-test, the TVLA test checks the similarity between two sets of traces captured from two populations of inputs. In particular, we use the non-specific t-test for fixed versus random leakage detection [BCD+13, GGJR+11]. Holding two sets of traces, the t-test calculates the t-score as $t = (\mu_1 - \mu_2)/\sqrt{(s_1^2/n_1^2) + (s_2^2/n_2^2)}$, where μ_1 and μ_2 are the means, s_1 and s_2 are the standard deviations, and n_1 and n_2 are the total number of the captured traces for first and second population, respectively. The null hypothesis is that the samples in the sets have been drawn from populations with the same mean, i.e., the masking is effective. On the other hand, the alternative hypothesis is that the samples have come from populations

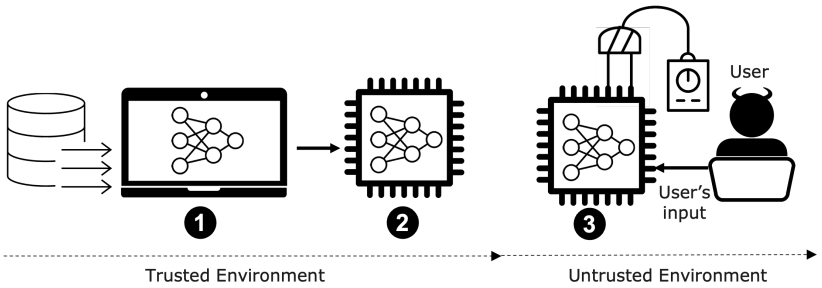

Figure 2: The adversary model considered in this work is similar to prior studies on SCA against NNs. In the trusted environment, first, the NN is trained on a given dataset. Second, the trained NN is implemented on FPGA. Third, in an untrusted environment, the device user acts maliciously and attempts to extract information about the NN's parameters and architecture by launching SCA.

with different means. To reject the null hypothesis at a particular significance level, a threshold on the t-score can be defined. For instance, the threshold ± 4.5 corresponds to 99.999% confidence [BCD+13]. Similar to the prior masking approaches, to evaluate the side-channel resiliency of our design, we choose the non-specific fixed vs. random t-test [DCEM18, DAP+22].

3.2 VERICA

VERICA [RBFSG22] is a tool for the formal verification of hardware. Most verification tools focus on either SCA or fault attacks (FA), but VERICA has incorporated verification of a design under both attacks. It mainly uses the probing model introduced in [ISW03] to check security as one of the methods, especially for a masked design. This allows it to formally verify the circuits by taking in a netlist of a design and forming binary decision diagrams (BDD) for them. The tool also uses many other models like glitch-robust probing model [FGDP+18], active security [DN20], extended fault model [RBSG22], etc. It is claimed to perform better than SILVER [KSM20] and FIVER [RBSS+21] as it not only gives the verification for SCA and FA individually but can also verify combined SCA/FA attacks. It also has the ability to verify composability properties like PINI [CS20], SNI [BBD+16], FNI [DN20], CINI [RBFSG22], etc. This enables the tool to verify gadgets and help reduce the development time for secure designs. It also uses a more advanced BDD engine and has a modular interface for testing different module versions. The modular interface is achieved using a JSON file to annotate the shares for the netlist instead of editing the netlist. This way, new annotations do not need to be generated with the same port for a module, even for different implementations. This allows for faster and more efficient experimentation.

3.3 Adversary Model

First and foremost, we stress that our adversary model is the same as what has been considered in [DAP+22] and various studies devoted to SCA against NNs. As illustrated in Figure 2, the NN provider trains the model offline, and the adversary is the user performing the inference. In this regard, valuable assets of NNs consist of their architectures and the parameters critical to achieving reasonable accuracy [BBJP19]. The countermeasure developed in [DAP+22] has not been concerned with the former and attempted to solely protect the parameters in NNs (e.g., weights). Therefore, without loss of generality, our adversary attempts to induce leakage by using the circuit components themselves. For this purpose, she collects power/EM traces from the device that she possesses either via direct

access or remotely, see, e.g., [SGMT18, ZS18, DKAA22]. The adversary follows a chosen-plaintext-type attack model, where she sends her inputs to the device to be classified and captures multiple traces, being further used to launch power/EM SCA [XCC+20, YMY+20, BBJP19, DCA20b, YKO+20, XAQ21, BJP22]. Notice that none of the voltage/EM fault injection and template/profiled attacks has been in the scope of [DCA20a].

Moreover, the countermeasure proposed in [DCA20a] aims to ensure that the information about parameters never leaks during any intermediate computation through a *first-order* power/EM-based SCA. In other words, the proposed protection scheme masks all intermediate computations. Furthermore, the t-probing model [ISW03] and robust-probing model [FGDP+18] provide security guarantees for their proposed masking scheme. The former model takes into account an adversary who observes the values of at most t wires in the masked circuit. The security is achieved if and only if the value on each of those t wires can be simulated using solely randomness. To reflect the impact of physical faults in hardware, such as glitches, transitions, and coupling, the probing model is enhanced by considering glitch-extended probes [RBN+15] to obtain the robust-probing model. As their name implies, the glitch-extended probes are relevant to the notion of glitches. The probes leak the value of the probed wires as well as all the wires in the fan-in until the last synchronization point.

4 FPGA-based Accelerators and ModuloNET

Before elaborating on our heat generation method, this section gives insight into aspects of modern FPGA-based accelerators that are crucial to understanding why our heat generation method can be applied to masked FPGA-based accelerators in practice. In the second part of this section, an example of masked NNs implemented on FPGAs has been given, namely ModuloNET [DAP+22]. We stress that this example is selected because of its intrinsic characteristics, including being theoretically sound, impressive in terms of side-channel resilience, and lightweight enough to be implemented on FPGAs.

4.1 FPGA-based Accelerators: Pros and Cons

As a result of decades of study and practice, implementation of NNs on FPGAs has become pervasive in various research fields and commercial applications and achieved satisfactory products [WGY+16]. Parallelism, modularity, and dynamic adaptation are some of the main computational features of NNs, which can be met when implementing them on FPGAs [MHS08]. Although there are challenges to face, including scalability and precision, compared with graphics processing unit (GPU) acceleration, FPGA accelerators can achieve at least moderate performance with lower power consumption. The latter is of great importance since the ever-growing data volume has led to exceedingly high power consumption (and consequently, temperature) in data centers [USA22]. Nonetheless, FPGAs have relatively limited computing resources, memory, and input/output (I/O) bandwidths; hence, a great deal of attention needs to be paid to developing complex and massive FPGA-based accelerators. In this respect, multiple approaches have been devised to optimize the design of FPGA-based accelerators in terms of throughput and latency. Here, throughput means that more data can be analyzed in a given amount of time, whereas the latency should be within the range specified by a service objective.

Challenge 1: Throughput. To optimize the design of NNs by considering the throughput as a metric, the data should be accessed every clock cycle and fed into the network. Batching (i.e., processing a batch of multiple input samples together) is a technique taken as a step towards this, although it increases processing latency and implementation complexity. Therefore, in practice, using large batch sizes is not practical; see, e.g., [NSS+16]. In response to this, processing a stream of input data on FPGA accelerators

is required, in particular for streaming applications, including image/video processing applications, real-time vision algorithms, and network packets encryption algorithms that are all FPGA-friendly data-intensive applications [RBK19, RHCM+16, RHL+18]. In traditional applications of FPGA accelerators, e.g., for computation-intensive tasks, access of the user to the memory shared between her and the FPGA has been restricted by, e.g., employing a hierarchy of dynamic random access memory (DRAM)/BRAMs in OpenCL platforms [SGS10]. This has, obviously, dramatically impacted the throughput streaming application; hence, methods have been developed to allow FPGA BRAMs to transfer data point to point every clock cycle [RHL+18]. Clearly, when the user has direct access to the FPGA (no memory hierarchy like the one in OpenCL), achieving such a high throughput is even more straightforward, see, e.g., [ZSZ+17, CSJC10]. Therefore, accelerators attempt to store inputs inside the chips into memory resources before any calculation takes place [SFM17, GYSC17, ZLS+15]. As an example, Shen et al. [SFM17] have used 1108 BRAMs to implement SqueezeNet on Xilinx Virtex-7 FPGA, which leads to 38% BRAM utilization. In another approach, Li et al.[LFJ+16] have utilized 1913 BRAMs (65.07% of available Xilinx VC709 BRAMs) for their AlexNET hardware accelerator. Zhang et al. [ZLS+15] have utilized 1024 BRAMs out of 2060 available BRAMS (50% BRAM utilization) in the Xilinx Virtex-7 FPGAs to implement their CNN accelerator on the FPGA. This is indeed helpful to reduce *latency* as well.

Challenge 2: Latency. Latency can refer to the *inference* latency, namely the time taken to process one unit of data given that only one unit of data is processed at a time; however, another aspect of data processing is more critical from the perspective of our study: the memory access latency. What has been suggested in the literature is to generally balance the computation throughput and memory bandwidth [SSEM18, ZP17]. In doing so, using external memories to store weights, especially for convolutional neural networks (CNNs), cannot be recommended due to how the throughput of such a design is limited by the external memory bandwidth [MVZ+21, CLL+14, LFJ+16, CLL+14, DFC+15]. Additionally, frequent access to the off-chip memory also introduces high energy consumption and, consequently, higher temperature [Hor14]. Although one might think that careful operation scheduling can significantly reduce external memory access, the widely accepted remedy, data buffering, imposes another difficulty: the limited buffer size. In fact, data buffering has often been paired with using external memories, aiming to tackle the issue of limited on-chip memory.

The challenges discussed above are generic in that designers of FPGA accelerators must tackle them irrespective of the security issues, such as resiliency to SCA. In fact, designing an FPGA accelerator with optimized throughput and latency is a hard-to-attain objective, let alone how this could be securely handled in the case of a masked NN. ModuloNET [DAP+22] is one of those proposals attempting to tackle all these aspects, including side-channel resiliency, together. At least for the networks showcased in their paper, no external memory has been used to store the weights so as not to cause harm to the throughput. Moreover, although masking imposes some additional cycles and consequently increases the latency, it is argued that the percentage of this increase is insignificant, thanks to the already high latency of the sequential design. At the time of writing this paper, the design of ModuloNET has not yet been made publicly available; therefore, we have had to follow the instructions and methodology given in [DAP+22] to implement it. This, however, even helps improve the design of ModuloNET[2].

4.2 ModuloNET: An Example of Masked NNs

The implementation uses masking for provable security against first-order attacks in the t-probing and glitch-extended probing models. Their binary NN, i.e., with binary weights

[2]Our traces and tools will be available upon acceptance of the paper.

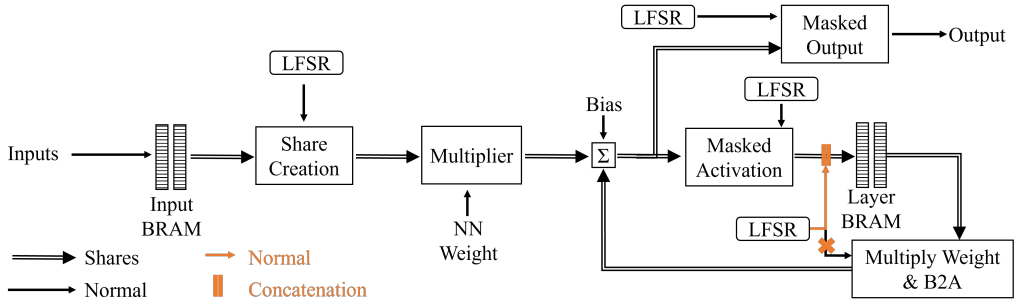

Figure 3: Design of ModuloNET cf. [DAP+22]. Inputs are directly stored in the BRAM, which are then used to create shares for the computation whereas layer BRAM is placed to store values per layer for computation. The orange-colored design shows changes that we made to the original design.

and activation function (AF) [CHS+16], includes five layers, one input layer, three hidden layers, and one output layer. All the layers are fully connected, making the design a multi-layer perceptron (MLP). The input layer consists of 784 neurons, each hidden layer has 1024 neurons, and the output layer has 10 neurons, which is compatible with the size of images in the MNIST data set. The design incorporates calculations in both binary and arithmetic-sharing schemes. Therefore, the design uses masking of both types and applies conversions between them. The masking scheme used in [DAP+22] is domain-oriented masking, where 2-input DOM-indep AND gates (refer to Section 3.1) are used in various places in the design.

Each neuron of the layer is shown in Figure 3. The design includes two BRAM-based memories, which store values used in computations for every neuron. In this way, the design sequentially calculates the values of the neurons one after another. Every neuron calculation includes Summation, Masked Activation, and B2A converter. The input layer neurons will additionally use the input BRAM, the Multiplier (shown in Figure 3), and the output layer neurons will use the Masked Output layer. Each of these modules has been discussed briefly below. Complete details regarding the operations and theoretical security proof can be found in [DAP+22].

Input share creation. Pixels are 8-bit inputs for the NN. As the design is a masked implementation, the inputs must be converted into shares. The shares are created using a linear-feedback shift register (LFSR) by subtracting the random value from the pixel value. So, one share is the random value, and the other share is the subtraction of those. These shares are arithmetic shares because they are integer values. These shares can be generated on the fly and used for each neuron calculation.

Input weight multiplication. First, we have the weight multiplication for the input layer. The weights here are binary weights. The multiplication with the weight results in either the same value or the complement value. This functionality is implemented as a multiplexer (MUX). Also, to keep the shares independent, the shares are calculated using a parallel implementation of the same modules.

Summation. After multiplication, shares are aggregated by the summation module. *In every cycle*, a new pair of shares is read from BRAM to be summed. This summation occurs for all the inputs for the current neuron, and then the bias is added. The summation module is also connected to the B2A module for the hidden layer and output layer computations and receives the input from the previous layer.

Masked activation. The summation results for the neuron are given as input to the activation layer. The activation layer is a non-linear function; therefore, it needs to be implemented using DOM gates. For this particular NN, we only need the carry-out of the summation for the result of the AF (more details about the AF used here can be

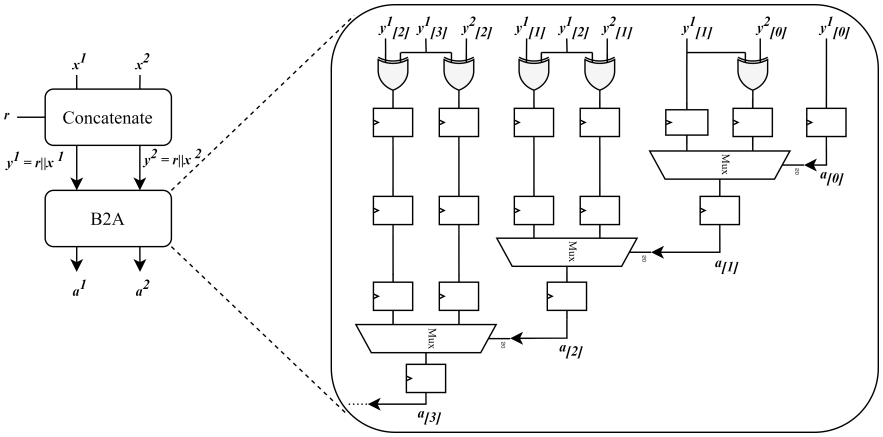

Figure 4: Boolean to arithmetic converter module in [DAP+22]. The B2A module comprises the concatenate module and Golic's B2A converter [Gol07].

found in [DAP+22]). This is accomplished by using a Kogge-Stone adder. The adder computes both the input shares and outputs the shares of carry-out. These output shares are Boolean shares and are stored in the layer BRAM. Layer BRAM is filled with the output of the AF for all the neurons in a layer before calculating the next layer.

B2A. Now, for the next layer, the input is the output from the previous layer, i.e., the values stored in the layer BRAM. For multiplication on these layers, XNOR-POPCOUNT is performed, which includes the B2A module. The B2A module converts the Boolean shares generated by the AF to arithmetic shares for summation. To convert the 1-bit Boolean share to 15-bit arithmetic shares, concatenation is performed before using the conversion algorithm as shown in Figure 4. Here, 1-bit Boolean share, x^1 *and* x^2 are concatenated with 14-bit random number, r to form 15-bit numbers, y^1 *and* y^2. These are converted to 15-bit arithmetic shares, a^1 *and* a^2 using Golic's protected design [Gol07]. After the conversion of shares, they are left shifted and fed to the summation module. This process is repeated for all the layers.

Masked output layer. For the output layer, the change is that instead of the masked activation module, the summation outputs are processed by the masked output layer. Similar to the masked activation layer, the output layer AF is a non-linear function. Thus, it also needs masking to keep the output shares independent. Three modules and a register file achieve this. Next, we will discuss all of these modules.

First comes the A2B Converter, which converts the Arithmetic summation shares to Boolean shares. This conversion is essential as the masked output layer performs the binary calculation. The values are stored in the register file. Before processing the output layer, we need values for all the output layer neurons. It is the reason for the register file after the A2B converter. Once all the values are ready, the output layer process starts with the threshold module.

For the NN, the threshold module is used to check if the scores are below a certain threshold (details on how the threshold is decided can be found in [DAP+22]). If it crosses the threshold, the output is 0. This is implemented with AND gates replaced by DOM gates to make it masked. The output is given to the masked comparator module using DOM gates to protect the module. The comparator module checks which class has the highest confidence score, so we compare it with the current global and local max results. The higher confidence score-based values are selected, and indexes for the same are also saved into local/global max registers. Masked MUX is used to select these values, which use DOM gates for masking. This flow is used to select a class in the output layer.

5 Temperature-induced Leakage from NNs

This section first explains why temperature variation can lead to inducing leakage in a design. Next, methods for generating heat within the NN designs are discussed.

5.1 Thermal Impact on FPGA Operational Dynamics

Delay attributes in FPGA elements.A comprehensive analysis of FPGAs' delay sources and their consequent effects on system functionality is imperative for developing robust systems [Kal13, GWWT12]. The architecture of FPGAs includes various elements like LUTs, FFs, and interconnects, each introducing unique propagation delays. The propagation delays in LUTs are predominantly attributed to the duration of logic function execution [ATF16], subject to variability based on the LUT's dimensions and structural complexity [ATF16]. On the other hand, FFs are responsible for managing sequential logic. The propagation delay in FFs is mainly determined by the clock-to-output delay, referring to the time that an FF takes to change its output after the clock edge [OH11].

Interconnections within an FPGA are critical in linking FPGA components. The delays in these interconnects depend on several parameters, including the length of wires, the degree of routing congestion, and their capacitance [NHCB02]. Additionally, variation in the manufacturing process in FPGA production can result in differing propagation delays, even among identical model units [SC06]. This study specifically delves into how temperature variations influence the delay of two key FPGA components, namely LUTs, and FFs, which are massively utilized in ModuloNET's modules.

Interplay between temperature and delays in LUTs and FFs. FPGAs employ digital complementary metal-oxide semiconductor (CMOS) technology, making the impact of temperature fluctuations on circuit performance a significant aspect to be considered. Temperature changes can have profound effects on the characteristics and behavior of CMOS circuits, a subject extensively covered in various studies [Kal13, HAM+86, WFKP98, MBS+24]. As an example, Kalra et al. [Kal13] have explained the temperature-sensitive parameters of MOSFETs. These parameters include carrier mobility, threshold voltage, saturation velocity, and the resistance at the parasitic drain/source, all of which can modulate the circuit's speed and power usage. In a similar vein, Gag et al. [GWWT12] have evaluated the effect of carrier mobility and threshold voltage on the timing characteristics of circuits. Equation 2 indicates how the threshold voltage of CMOS devices decreases by increasing the temperature T, depending on Vt_0 (threshold Voltage at temperature T_0) and T_0 (initial temperature) [Tsi87, GWWT12, FA01].

$$Vt = Vt_0 + \alpha_{Vt} \cdot (T - T_0) \tag{2}$$

Here the factor α_{Vt} varies between 2 to 4mV/K (see Equation 2), where its exact value depends on the doping level of the actual semiconductor. This results in higher drain currents at higher temperatures; consequently, the circuit is getting faster.

The carrier mobility, μ also varies by changing the temperature as formulated in Equation 3 [GWWT12, FA01, LS94].

$$\mu = \mu_0 \cdot \left(\frac{T}{T_0}\right)^{\alpha_\mu} \tag{3}$$

Here α_μ is a constant, and μ_0 denotes the carrier mobility at the initial temperature. α_μ depends on doping and varies from -1.5 to -2.5 in literature [GWWT12]. The higher the temperature, the lower the drain current is; hence, the device should be slower. Among the two conflicting factors -the carrier mobility and the threshold voltage- the latter dominates if the supply voltage is close to the threshold voltage. In such a scenario, the design will be

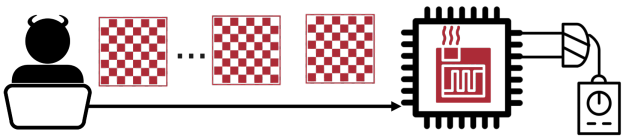

Figure 5: The adversary relies on the fact that at high temperatures, the power consumption associated with different shares is no longer independent of each other. In this regard, the adversary takes advantage of the memory allocated to store the inputs and, by writing alternating '0' and '1' patterns, attempts to increase the operating temperature of the FPGA and detect first-order leakage.

faster; otherwise, the temperature increase makes the design slower. Regardless of whether it is faster or slower, the gates' delay varies with temperature.

Now, the question is whether the variation of gates' delay can affect the effectiveness of masking. This has been explored in the literature for a long time. As a prime example, [MPG05] has explained that varying propagation delay of the gates leads to glitches in the circuit and, thus, a significant impact on the circuit's power consumption. Therefore, the variation of gates' delay due to an increase in temperature clearly changes the circuit's power consumption pattern. Following the argument in [DCEM18], consider the first-order Boolean masking, where secret value x is represented by two shares x_1 and x_2 with their corresponding exclusive power consumption $P_1(x_1)$ and $P_2(x_2)$, respectively. When the sub-circuits operating on each of the shares work in parallel, thanks to the variation of the delays, these sub-circuits influence each other's power consumption. Even though the assumption of their independent power consumption may hold at room temperature (i.e., the condition under which the circuit is designed and tested against SCA), this assumption may be violated at higher temperatures.

5.2 Inducing Leakage through Internal Heat Generators

Given the above discussion, it is tempting to increase the temperature of the circuit to induce leakage. This can be achieved using a climate chamber to operate the device at higher temperatures as practiced in [DCEM18]. Nevertheless, an internal heat generator (HG) has various advantages, namely its use in remote attacks and its cost-effectiveness. Seen from another perspective, even if the internal HG is enabled unintentionally (i.e., the operation of the device causes high temperature), it is interesting to study how effect the masking will remain under such conditions. As argued below, such conditions are highly probable for NNs, which *heavily* utilize BRAMs (see Section 4.1)- let alone the masked NNs with even higher utilization thanks to storing shares.

The core idea underlying the heat-induced leakage is to generate heat inside the FPGA by flipping the input image (i.e., writing alternating '0' and '1' input patterns into memory) frequently to toggle the corresponding BRAMs, thereby increasing dynamic power consumption and, subsequently, chip temperature. This simple but effective concept is realized by an adversary who solely feeds the inputs to the design in each and every clock cycle, see Figure 5. This type of adversary is entirely agnostic about the design of the NN, but leverages the *internal* HG to make the leakage from the masked NNs detectable. The foundation of this has been laid by Happe et al. [HHAP12] and Agne et al. [AHH+14], who have indicated that one of the primary sources of heat in FPGAs is a significant number of BRAMs and FF pipelines. In addition, they showed that reading/writing from BRAMs and FFs also generate extreme heat. To understand the effect of such HGs on masked NNs, we consider ModuloNET [DAP+22].

Similar to many other accelerators [CLL+14, DFC+15, DCA20a, DCA20b, DCA20a], input images and masked AF outputs in ModuloNET have been stored in BRAMs; therefore, ModuloNET meets the needs of the adversary for generating heat on FPGAs. As mentioned

in Section 3.3, the adversary cannot control the input/output of the masked AF when the FPGA is operating. However, the adversary can continuously feed flipping images into the FPGA, as shown in Figure 5, which are stored in BRAMs. To make the most of the HG, it is necessary to change multiple bits every cycle. This can be achieved by writing alternating '1' and '0' patterns into BRAMs; hence, the adversary crafts image pixels with such a pattern to obtain so-called flipping images. We emphasize that, in contrast to HGs in [HHAP12, AHH+14], we do not write into memory in a pipeline fashion, but in parallel, in compliance with the design of NNs.

Now the question is why exploiting such an internal HG would impact the masked design. In fact, the heat generated inside the FPGA results from high power consumption, which directly affects how (in)dependent the power consumption of the shares (and the functions being operated on them) are on each other. This has been explored in [DCEM18], where it is shown that "the power consumption of a function operating on a share influences the amount of power consumption of other functions simultaneously operating on other shares." De Cnudde et al. [DCEM18] have empirically examined this (in fact, in "an artificial lab environment") that such a phenomenon is observable within the temperature range between 50 °C and 70 °C. These are the ranges selected in [DCEM18] to express the extreme effect of temperature on first-order leakage. While they could enjoy the freedom of selecting these by employing the climate chamber, we attempt to determine if the internal HG executed by solely feeding the alternating inputs could influence the first-order leakage from the design. In the same vein as [DCEM18], we aim to pinpoint the existence of leakage in correctly implemented masked NN.

We examine whether, utilizing the BRAMs for storing NN inputs, the heat can be generated by flipping the value of inputs (e.g., image pixels) frequently.

Comparison with the most relevant heat generation methods. First and foremost, we stress that De Cnudde et al. [DCEM18] have investigated the impact of extreme temperature on masking in general. To this end, a climate chamber has been used to operate the device at higher temperatures. While the concept has been well understood, no practical heat generation has been proposed to increase the temperature internally to make the masked implementation leak. On the other hand, the heat generation in [ATG+19] has leveraged a large number of simultaneous write-collisions to cause voltage drop and, consequently, a significant temperature rise. Here, write-collision refers to the scenario where both ports of BRAMs in *dual-port* BRAMs are writing different data to the same memory address. It has been observed that FPGAs (i.e., including series crafted by Xilinx, Intel, Microsemi, etc.) contain dual-port random-access memory (RAMs) with concurrent writing possibility, which can be turned into a serious vulnerability if opposite logic values are written to an address to create a transient short circuit. While our heat generation relies on writing flipping pixels into BRAMs, as opposed to [ATG+19], single-port BRAMs are utilized in our case. The implication of this is that our heat generation is not restricted to designs with dual-port BRAMs. More importantly, [ATG+19] has aimed to *inject* faults into NNs, whereas our study demonstrates the possibility of inducing *side-channel leakage* in masked NNs.

6 Experimental Results

6.1 Measurement Setup

To evaluate how effective our heat generator (HG) is and understand its impact on first-order leakage, ModuloNET is implemented on Artix-7 FPGA, embodied within the CW305 board. We have also used Vivado 2021 [Xil21], whose IP catalog is applied to generate BRAMs. To maintain the independence of shares and the security of the design, we had to tune some of the synthesis and implementation parameters. We disabled the LUT

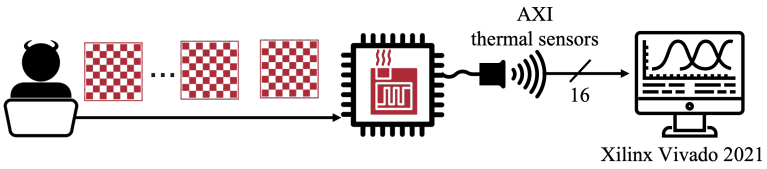

Figure 6: Experimental setup used to perform the thermal test.

Table 1: Hardware resource allocation in our implementation of ModuloNET used to perform experiments on BRAM-based HG.

Resource	Used	Utilization (%)	Available
LUT	15807	24.93	63400
FF	7782	6.13	126800
BRAM	**131**	**97.03**	135

sharing and the optimization option as well as enabled `keep_equivalent_registers` in combination with using the `DONT_TOUCH` attribute. This setting forces Vivado to place the shares in different locations, as specified in the design, and not optimize them. We have used an external low-noise DC power supply, BK Precision 9130, to ensure a stable and consistent power supply at different temperatures. CW305 target board is connected to the power supply using banana pins to give 1V to the board, i.e., the default voltage for the FPGA sitting on the CW305 board.

Power traces have been captured using CW Husky and Lite using synchronous capturing feature. To accelerate the capturing process, we have also used the segmentation feature of the CW husky board. It allows us to capture 8 traces with a single communication from the computer. The traces are collected with no shunt resistor on the CW305 target board and low-noise amplifier of the CW305 board. Moreover, the mini-circuits SLP-30+ low-pass filter is attached to the SMA cable connecting the CW305 and CW Husky. The design frequency is set to 10 MHz which is the same as the capturing frequency, i.e., synchronous capturing. This helps alleviate timing issues in the trace collection, such as clock jitter-based noise during capturing.

External heat generation. Testequity model 107 temperature chamber is used to control the temperature of the target. The chamber can maintain a temperature between $-42\,°C$ to $130\,°C$ with a tolerance of $\pm 0.5\,°C$. The chamber is used to conduct experiments to show the impact of external heat generation on the design.

In all experiments, regardless of whether we used our proposed internal BRAM-based HG or the temperature chamber, we have monitored the output of the PRNG and have not observed any irregularity in its behavior.

6.2 Our implementation of ModuloNET

Our ModuloNET is implemented with regard to the description provided in [DAP+22] and includes the modules explained in Section 4.2. As mentioned in [DAP+22], the share creation during the t-test calculation would lead to leakage due to input correlation. To avoid this, they calculated the inputs and corresponding shares before the t-test calculation. We have employed a similar approach where we calculate the shares on the device and store them in the input BRAM before starting the t-test. Dubey et al. [DAP+22] have implemented a multi-layer Perceptron (MLP) with 784 input nodes, 3 hidden layers with 1024 nodes, and 10 output layer nodes to argue about the scalability of their approach; however, to accelerate the trace collection process, the number of nodes in the hidden layers is reduced to 64. It is also justified that the leakage assessment is generalizable to the larger NN thanks to the repetitive nature of NN computations and the sequential

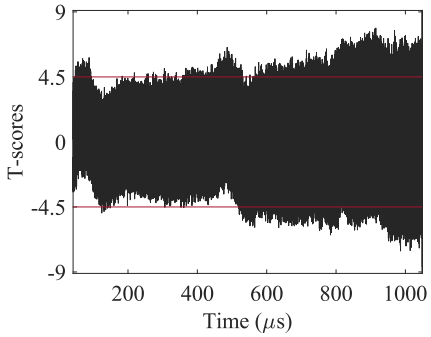

Figure 7: T-test results for the entire design, including all modules, explained in Section 4.2 for $100K$ traces captured from the original design (as depicted in Figure 8 in [DAP+22]).

design (i.e., computing one node at a time). Similarly, we take a reasonable size of the NN into account to reduce the number of clock cycles and, consequently, the time taken to collect a trace, as well as the size of the data stored for a single trace. Our network has 100 input nodes, 64 nodes in the hidden layer, and 10 nodes in the output layer. This reduced design takes 10,400 clock cycles per trace and around 100kB of data per trace. We stress that if more input nodes were considered, the operating temperature would be even higher than observed in our experiments; hence, our results give an optimistic estimate of what can happen in practice.

Since we have considered 100 input nodes, we need 100 pairs of shares of all the inputs. We have allocated 2 BRAM, one for each share, with a size of 78,400 (corresponding to 784 pixels in MNIST images and each with 100 input nodes), with a 15-bit width. The 15-bit width is the result of padding the 8-bit pixels to 15, as performed in the original design in [DAP+22]. While 2 BRAMs (FIFO36) are consumed by layer BRAM (see Figure 3), 69 BRAMs (FIFO36) are used to store only 100 pixels per image to accelerate the trace collection process further.

Moreover, calculating the t-scores requires capturing traces from a set of fixed and a set of random inputs, which prohibits us from having the luxury of flipping input images entirely during the t-test process. Hence, we have considered a set of 120 FIFO18 BRAMs (half the size of FIFO36 BRAMs) in the design and flipped them simultaneously with the t-test process. Writing into memory occurs every cycle, and thus, the address counter stays constant at zero. We stress that adding these BRAMs help us mimic the effect of the flipping input images without interfering with the trace-capturing process. Note that the total number of bytes in flipping and t-test inputs is much less than the number of input bytes that can be flipped by the adversary in a real-world scenario, where the actual input size of the MNIST image is 784 pixels. Furthermore, since the t-test inputs are not changed once the t-test is started, the change in the operating temperature reflects the situation where the adversary does not take full advantage of the HG. The complete list of resources used by the design when performing the experiments is provided in Table 1. In terms of the number of LUTs and FFs, although we tried to follow the original design as closely as possible, we have higher resource utilization compared to ModuloNET [DAP+22] (5635 LUTs and 5009 FF). Nonetheless, since we compare the operating temperature and the leakage solely by considering our design under different conditions, this does not cause any issues.

Verifying the design using VERICA. As seen in Figure 7, the t-test results for the original design as described in [DAP+22] showed leakage. To figure out the cause of the leakage, we used a design verification tool, namely VERICA [RBFSG22]. For this purpose, first, we need to generate a gate-level netlist for the individual modules using the Synopsys design compiler [Syn20]. The tool command language (TCL) script has to follow

constraints for the VERICA tool. Next, we need to define the security annotations for the input and output, such as shares, refresh bits, and control signals. This information has to be defined in a JSON file, which can be used to test different versions of the same module, which is a significant improvement over SILVER [KSM20]. We tested different modules using VERICA and found that the B2A module failed the test. In fact, the B2A module passed the d-probing test but failed the glitch-extended probing test. There was one probing point inside the B2A module, where the leakage of the Boolean shares was detected. By looking at the netlist, we found the cause of the problem at the point where Boolean shares were written/read into/from BRAM. The point is that the prerequisite for Goubin's B2A algorithm is the field being the same for inputs and outputs. Therefore, if the Boolean shares have less than k bits (see Section 3.1), fresh random values can be used to pad them (concatenating with the random values as done in [DAP+22]). If this padding is performed after reading the values from BRAM, the glitch-extended probing test fails. We could resolve the problem by performing padding before writing the Boolean shares into the layer BRAM, as highlighted in orange in Figure 3. The t-test results for 2M traces collected from the enhanced design are depicted in Figure 9(a-bottom). We stress that this is the amount of measurements for designs with long execution times as used in [DAP+22].

6.3 Die Temperature Measurement

The CW305 target board is a customized board for effective side-channel evaluation purposes; unfortunately, it lacks the system monitoring sensors. Hence, we have used the PYNQ-Z1 board with Artix-7 FPGA (package number FTG256) to perform thermal tests. Figure 6 illustrates our temperature evaluation experimental setup. We have used the XADC thermal sensors and read the 16-bit temperature channel analog to digital converter (ADC) via Xilinx Vivado 2021 local server. To communicate with the XADC module, we used the PYNQ-Z1 AXI streaming port, which has a refreshing period of 1s. To establish communication between the personal computer (PC) and the XADC and store the sensor data, Happe et al. [HHAP12] used the MicroBlaze processor embedded alongside the design. However, as the MicroBlaze processor generates heat on FPGA, we have used the Xilinx Vivado TCL to store XADC temperature sensor data with the bandwidth of 1s to prevent heat generation from other sources rather than the design.

To evaluate the impact of our internal HG, we have done experiments in two cases, namely giving (a) a normal (not flipping) image and (b) one with alternating "0" and "1" patterns (flipping image). Note that the former corresponds to the heat generated when processing the t-test inputs stored in the memory. In both cases, the XADC sensor collects 3,600 temperature samples over $3,600s$. Then, we let the board cool down for 60 minutes (m), even more than the suggested cool-down time recommended in [HHAP12, AHH+14], and provided ModuloNET with flipping images. Figure 8 shows the heat generation results in these two experiments.

It is observable in Figure 8 that giving the normal images to ModuloNET with BRAM input storage generates heat up to a maximum of 42.1 °C. While with the flipping images, read/write from/to HG BRAMs, its operating temperature reaches the maximum of 72.9 °C in an even shorter period.

Using Xilinx Power Estimator (XPE) tool to validate the emulation results. The XPE tool [Adv24] can be used for estimating power consumption and die temperature in Xilinx SoCs and FPGAs. It stands out for its ability to assess die temperature by considering the FPGA bitstream and ambient temperature. This feature is particularly valuable for designers to understand thermal performance under various conditions. To verify the results of our emulation, we also used this feature of XPE. In doing so, when setting the ambient temperature to the room temperature (25 °C), the XPE simulates the die temperature to be equal to 40.2 °C, confirming the similarity of CW305 ChipWhisperer

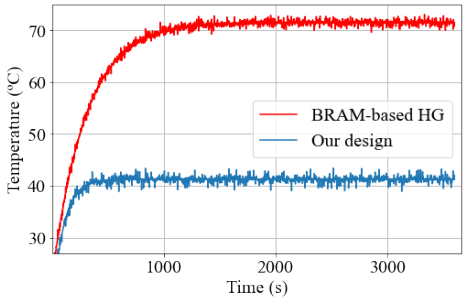

Figure 8: Temperature testing results for our implementation of ModuloNET when giving a normal (not flipping) image and flipping inputs are written into BRAMs (BRAM-based HG). Note that the HG is indeed part of the design, specifically, the BRAMs.

board operational die temperature and PYNQ-Z1 XADC sensor according to Figure 8. Furthermore, if the die temperature reaches 70 °C, equal to the die temperature after using our HG (see Figure 8), the ambient temperature should be set to 61.5 °C.

An important point to mention is that we have decreased the number of the input pixels of the ModuloNET to be implemented on Artix-7 FPGA with a limited BRAM support, as explained in 6.2. However, the temperature will increase even more if an FPGA has sufficient BRAM to support a larger input size, e.g., 784 pixels. As an example, a defense-grade VIRTEX-7Q FPGA has $1,500$ BRAM, sufficient to implement the ModuloNET with the larger inputs requiring around 1048 BRAMs. In this case, the results indicated that even at room temperature, 25 °C, the die temperature can increase up to 64.2 °C.

6.4 Temperature Influence on Delay of FPGA Components

This study concentrates on understanding how temperature variation can induce leakage. As discussed in Section 5.1, variation of delays can cause glitches with an impact on the power consumption of shares. Here, we validate experimentally how increasing the temperature affects the propagation delay of FFs and LUTs in FPGAs. For this purpose, on the Artix-7 FPGA of the CW305 board, we have implemented (1) a single FF connected to a variable bit source and (2) a series of 10 inverters, each mapped individually to a distinct LUT. The output pins of the FPGA operate within a voltage range of $0V$ to $3V$. We selected $2.5V$ as our logical "1", noting that pin output values surpassing this threshold exhibited instability, oscillating around $3V$. A threshold above $2.5V$ was avoided due to the prevalence of false peaks at voltages nearing $3V$. We have used WavePro 254HD 2.5 GHz High Definition Oscilloscope [TL23] with 20 GS/s resolution to capture the changes in output pin values in both cases.

The delay in the FF was analyzed by measuring the clock-to-Q delay [OH11]. This involves monitoring the disparity in timings between two outputs: one from the clock source of the FF and the other from the FF's output. To reduce the influence of interconnect delay between the clock source and the FF input, the FF was placed near a clock source.

For analyzing the LUT delay, we have crafted a sequence of 10 inverters, initiating the chain with a connection to the clock source. The reason behind choosing 10 LUTs instead of one is that the delay of one LUT is close to the error of our measurement setup, WavePro 254HD 2.5 GHz High Definition Oscilloscope [TL23], and it might not be possible to measure the delay accurately. The delay is measured by focusing on the time difference between the output of the clock and that of the chain's final inverter. We ensured the proximity of the first inverter in the chain to the clock source and arranged the LUT chain to minimize spatial separation.

Table 2 details the delay measurements for a single FF and a chain of 10 LUTs at room temperature (25 °C) and 61.5 °C (see Section 6.3). The delay values in Table 2 represent

Table 2: Component delays at different ambient temperatures.

Temperature (°C)	Delay (ps)	
	Single FF	Chain of 10 LUTs
25	793	85
61.5	783	77

the time difference between the moment the clock source output pin and the FF's Q-pin (column 2) or the final LUT in the chain's output pin (column 3) exceed $2.5V$, measured in picoseconds (ps). To mitigate noise in the measurements, the values reported are averages from 1000 repeated experiments, employing a noise reduction approach akin to [SLL+08].

An interesting observation from Table 2 is that the propagation delays for both the FF and the LUT chain do not linearly correlate with temperature increments. Notably, the FF's propagation delay decreases by $8ps$ when the ambient temperature rises from 25 °C to 61.5 °C. A similar trend is observed in the LUT chain, with a reduction of $10ps$ in delay over the same temperature range. As the device is faster, one conclusion is that the effect of the threshold voltage is dominant. Irrespective of whether the effect of the threshold voltage or the carrier mobility is dominant, variation of delay as shown in Table 2 plays a role in inducing the leakage at higher temperatures.

6.5 Leakage Detection

After verifying the impact of giving flipping images on the operating temperature of the FPGA embodying ModuloNET, we investigate how the resulting temperature rise can affect the first-order leakage. The goal of experiments done in this regard is to understand whether a first-order secure design, i.e., our design of ModuloNET, exhibits first-order leakage if flipping images are fed into it and, thus, increases the operating temperature[3]. For this purpose, we compare the t-scores calculated for traces collected from the design when providing normal (not flipping) and flipping images to BRAM-based input storage. In these experiments, before collecting traces, we first wrote the "0" and "1" alternating patterns into the memory to reach a high die temperature. More precisely, after about 20min. (1,200s), the temperature becomes almost stable and reaches its maximum value (see Figure 8). Note that an adversary interested in detecting the leakage needs neither this information nor access to the sensor/monitoring system, as she can simply give the flipping images for hours to ensure the die temperature is high.

Furthermore, we should highlight that we start with several tens of thousands of traces to see how many traces of the first-order leakage are detectable after capturing. In this regard, we collected 2M traces from the device in our experiments. Note that throughout this section, the number of traces refers to the total number of fixed and random traces, i.e., collecting 2M traces means that 1M fixed and 1M random traces are collected. The effect of activating the HG, i.e., giving a flipping image to ModuloNET, can be seen in Figure 9. As can be seen in Figure 9(b), under the scenario where flipping inputs are fed into ModuloNET, the t-scores calculated for 2M traces are much increased compared to Figure 9(a), where normal inputs (not flipping inputs) are considered. Notice that no first-order leakage has been seen for ModuloNET when evaluating 2M traces as depicted in Figure 9(a). Further, Figure 9(b) illustrates a sample trace and the average of 1,000 traces showing how the power characteristics of the design are affected by the extreme heat. Moreover, as marked in Figure 9(b) (see the bottom row), the t-scores do not always fall within the desired threshold, implying the rejection of the null-hypothesis with the confidence of 99.999%.

[3]For results obtained by using an external heat generator, see Section 6.7.

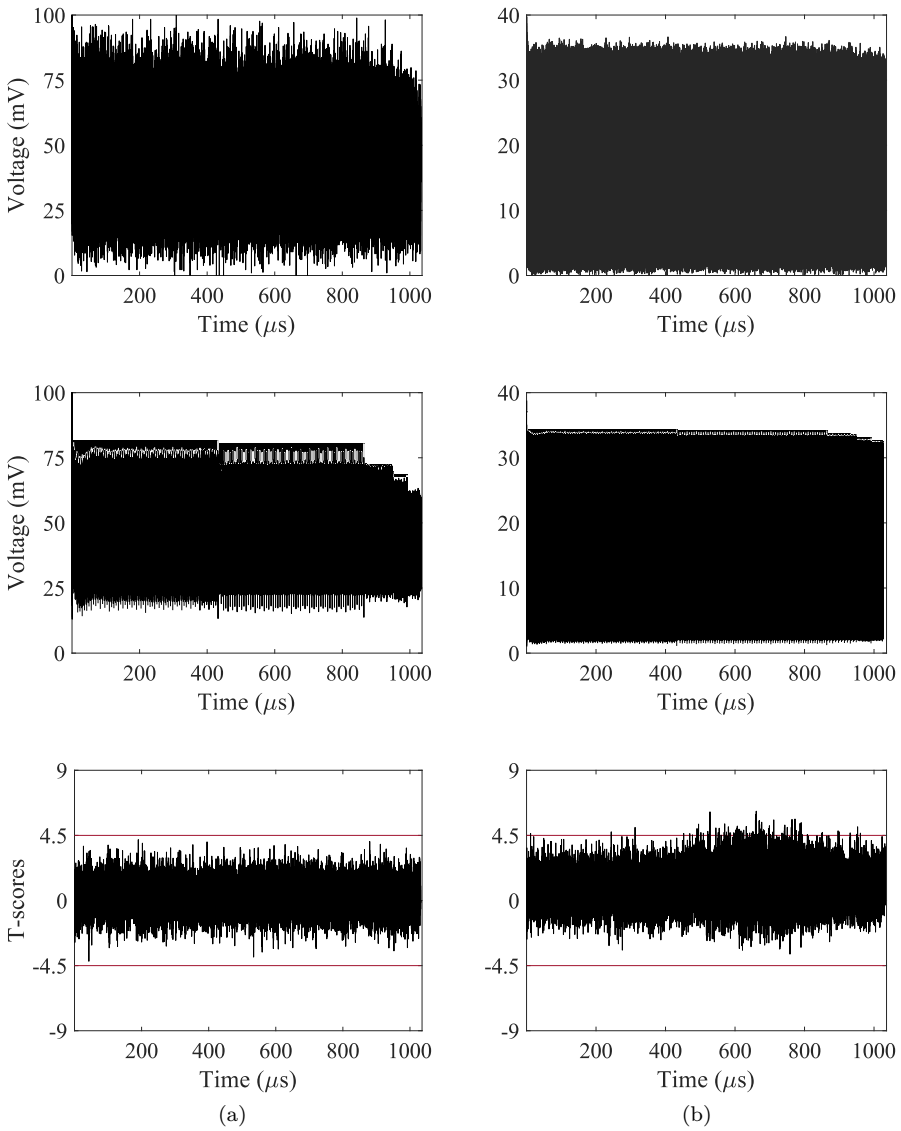

Figure 9: Results for ModuloNET with (a) no HG, and (b) with HG enabled. Sub-figures present (top) sample trace, (Middle) average of 1000 randomly chosen traces, and (bottom) t-test results for $2M$ traces.

6.6 Leveraging the Leakage

After detecting the leakage, the natural question is whether the adversary can leverage the leakage. To answer this question, we apply differential power analysis (DPA). DPA stands as a prominent side-channel attack wherein attackers exploit the power consumption patterns of a device to extract its secret [KJJ99]. To find out if the leakage induced through our HG can be exploited in an attack, we launched four cases of DPA attacks against ModuloNET. First, we investigate whether a first-order DPA can break the security of ModuloNET in the absence of PRNG without enabling the HG. This serves as a baseline to examine if the first-order DPA against ModuloNET would be effective, i.e., if masking is effectively implemented. After ensuring that, we shift our focus to how HG can induce leakage that can be exploited by DPA. To understand if enough number of traces is used in DPA, we also launched a second-order DPA against ModuloNET. The distinguisher

used in all cases is Pearson correlation with a confidence level of 99.99% as suggested in [MOP08]. We have obtained the intermediate value from ModuloNET stored in layer BRAMs and registers between bias addition and masked output for the hidden layer and output layer, respectively; see Figure 3. These attack vectors for launching DPA are chosen according to the points where the t-score exceeds the threshold, namely around $500\mu s$ for the hidden layer and $750\mu s$ for the output layer in Figure 9.

DPA with PRNG off. By turning off the PRNG, all the masked values are unmasked, i.e., making a first-order attack successful. This is verified in Figure 10(a) and Figure 11(a). At the bottom of these sub-figures, the results for DPA with $500K$ traces against hidden layer and output layers are depicted. It is observable in Figure 10(a-bottom) that the DPA correlation exceeds the threshold around $540\mu s$, whereas for the output layer, the correlation exceeds the threshold around $830\mu s$ as shown in Figure 11(a-bottom). This point is where ModuloNET starts multiplying the secret weights with the image in the hidden layer. This correlation peak confirms the vulnerability of the unprotected design, unmasked ModoloNET [DAP+22]. Figure 10(a-top) and Figure 11(a-top) show the results for evaluating the correlation coefficient for the chosen window marked by dotted line rectangles, where the number of traces increases for a trace range of 100 to $500K$. It is observable that if the adversary focuses on a window around the correlation peak, it is possible to successfully launch a DPA with $230K$ and $70K$ traces against the hidden and output layers, respectively.

First-order DPA with PRNG on and HG off. Next, we evaluate how effectively the masking is employed by using the same attack vectors from the case of PRNG off, but this time, the PRNG is turned on. Still, the HG is off in this case. Figure 10(b-bottom) and Figure 11(b-bottom) show the results for $500K$ traces against ModuloNET hidden and output layer, respectively. In contrast to the case where the PRNG is off, the correlation has never exceeded the threshold. This validates the resiliency of the protected ModuloNET against first-order DPA attacks for up to $500K$ traces for both hidden and output layers. Moreover, Figure 10(b-top) and Figure 11(b-top) confirm that even if the adversary focuses on the same window used in first-order DPA with PRNG being off, the protected ModuloNET still shows resiliency to the first-order DPA.

Second-order DPA with PRNG on and HG off. To understand whether we have used a sufficient number of traces to perform a first-order DPA when the PRNG is turned on and the HG is off, we launch a second-order DPA. We use the same attack vectors as in the first-order DPA and follow the method in [OMHT06, Mes00]. Figure 10(c-bottom) and Figure 11(c-bottom) show the results for second-order DPA with $500K$ traces against the hidden and output layers, respectively. In addition, focusing on the window in which the correlation exceeds the threshold, it can be observed in Figure 10(c-top) and Figure 11(c-top), an adversary can successfully launch a second-order DPA with $200K$ against the hidden and the output layers. This confirms that $500K$ traces are sufficient for a first-order attack when the HG is off (see Figure 10(b) and Figure 11(b)). Moreover, when conducting a second-order DPA with $500K$ traces, the time points, near which the second-order leakage happens, are $540\mu s$ and $830\mu s$ for hidden and output layers, respectively. These time points are close to the leakage points in Figure 10(a) and Figure 11(a), respectively (see the bottom figures). This confirms the point of interest on which an adversary can launch a DPA against the hidden and output layer of ModuloNET.

First- and second-order DPA with PRNG on and HG on. After performing the first- and second-order DPA against the design without enabling the HG, we study how our proposed HG can facilitate these attacks. We start with the first-order DPA, whose result for the hidden layer is illustrated in Figure 10(d). The bottom figure shows the results for DPA with $1M$ traces. It is noteworthy that DPA with $1M$ traces failed under the scenario, where the HG was not enabled.

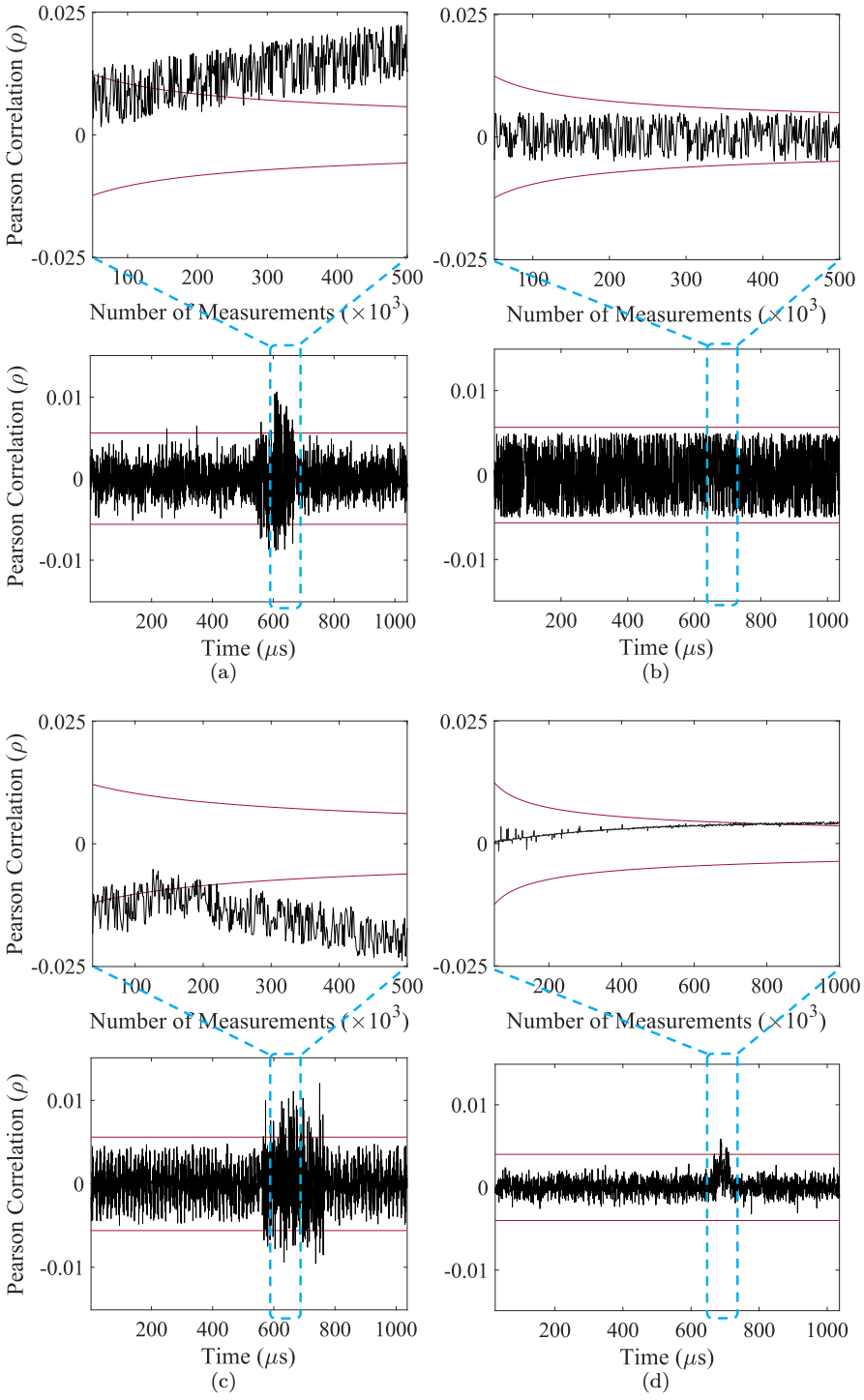

Figure 10: Results for DPA against ModuloNET's hidden layer in different cases. In all sub-figures, the results in the bottom image show the correlation vs. time, whereas the top image indicates how many traces are required to launch the attack in a time window around the peak in the bottom image (see dotted line rectangles). (a) first-order DPA for $500K$ with HG: off and PRNG: off; (b) first-order DPA for $500K$ with HG: off and PRNG: on; (c) second-order DPA for $500K$ where HG: off and PRNG: on; and (d), first-order DPA for $1M$ traces when HG: on and PRNG: on.

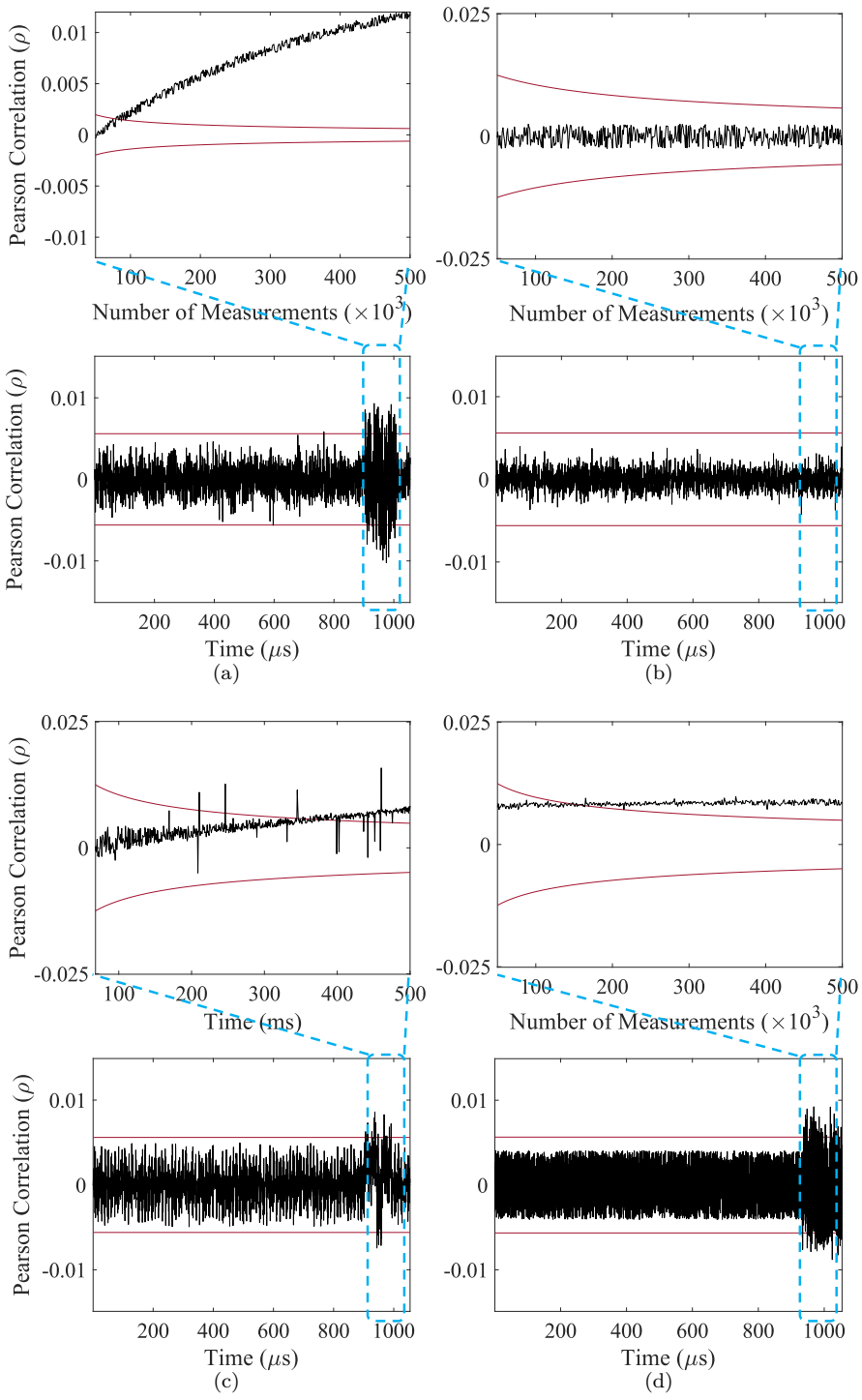

Figure 11: Results for DPA against ModuloNET's output layer in different cases. In all sub-figures, the results in the bottom image show the correlation vs. time, whereas the top image indicates how many traces are required to launch the attack in a time window around the peak in the bottom image (see dotted line rectangles). (a) first-order DPA for $500K$ with HG: off and PRNG: off; (b) first-order DPA for $500K$ with HG: off and PRNG: on; (c) second-order DPA for $500K$ where HG: off and PRNG: on; and (d), first-order DPA for $500K$ traces when HG: on and PRNG: on.

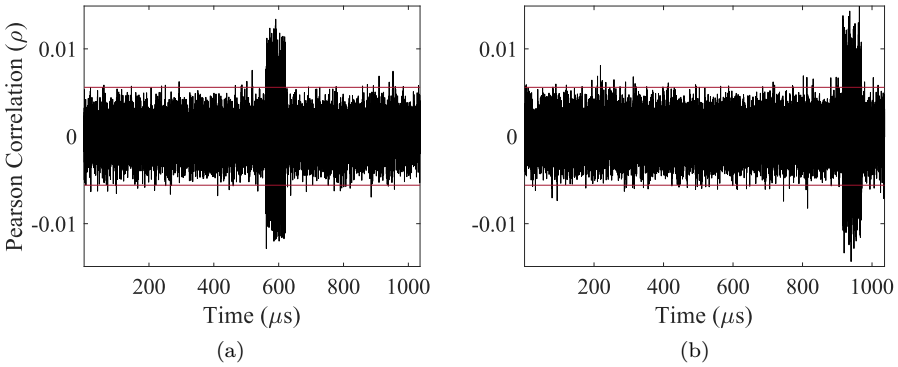

Figure 12: The second-order DPA for $500K$ traces against the (a) hidden and (b) output layer of ModuloNET with HG on.

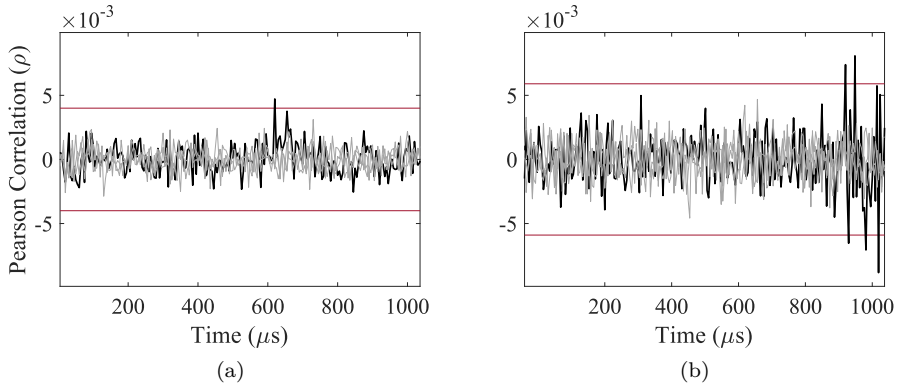

Figure 13: The first-order DPA against ModuloNET with HG on at (a) hidden for $1M$ traces and (b) output layer for $500K$ traces (gray lines for the wrong weight guesses and black line for the correct weight guess.)

This validates the effectiveness of our HG in breaking the masking protection of ModuloNET. Furthermore, focusing on the window in which the correlation exceeds the threshold in Figure 10(d-bottom), an adversary can successfully launch a DPA with $880K$ traces. The same holds for second-order DPA with HG on, where the leakage from both the hidden and output layers becomes more noticeable after comparing the results in Figure 12 with Figures 10(c-bottom) and 11(c-bottom).

We repeated this analysis for the output layer, whose result for $500K$ traces is shown in Figure 11(d-bottom). In contrast to the case where the HG was disabled (Figure 11(b-bottom)), the attack is successful after enabling the HG. Compared to the attack against the hidden layer (Figure 10(d-bottom)), DPA succeeds with a fewer number of traces. In line with this, as it is shown in Figure 11(d-top), if the adversary focuses on a window around the correlation peak, it is possible to launch a first-order DPA with approximately $200K$ traces successfully. In a nutshell, based on these results, we can claim that our HG enables a successful first-order DPA against ModuloNET.

First-order DPA against ModuloNET with HG on for the correct and wrong weight guesses. After enabling the HG, we also launched the first-order DPA against ModuloNET for four weight guesses at the output and hidden layers (i.e., the same attack vectors as in the above paragraphs). Figure 13(a)-(b) show the results, where the correct and wrong weight guesses are highlighted in black and gray, respectively. According to Figure 13, the threshold is passed around $500\mu s$ and $875\mu s$ for the correct weight guess at

the hidden and output layers, respectively. This is aligned with the results presented in Figures 10 and 11 (c-bottom). For all wrong weight guesses at both hidden and output layers, it is observable that the correlation remains under the threshold for the entire time window. This again confirms that the first-order DPA is successfully mounted after enabling the HG.

6.7 Inducing Leakage through External HG

This section aims to study the impact of temperature on the first-order side-channel resiliency of ModuloNET when the leakage is induced using an external heat generator, i.e., the thermal chamber.

Setting the ambient temperature. As mentioned earlier, the CW305 target board does not support the XADC sensors to monitor the die temperature while it is operating; therefore, we have used XPE to find out the proper ambient temperature in the thermal chamber to match the die temperature as in the scenario where the internal HG is enabled; see Figure 8. For this, we gave the XPE tool the ModuloNET bitstream file implemented on Artix-7 FPGA. In doing so, for the die temperature 70 °C, the XPE simulates that the ambient temperature is 61.5 °C.

Detecting and leveraging the leakage. For these experiments, we use the design with HG off. The experiments are conducted at 61.5 °C, as discussed above. Figure 14(a)-(b) depict the maximum t-score over $6M$ traces. Similar to what has been presented in [DCEM18], these figures demonstrate how increasing the temperature can facilitate leakage by comparing the number of traces where the threshold is passed. Note that random peaks in the maximum t-score, as shown in Figure 14(a), should not be confused with the leakage that can be exploited. Figures 14(c)-(d) show the results for the first-order DPA with $500K$ traces against Modulonet hidden and output layer, respectively. In Figure 14(c-bottom) and (d-bottom), the correlation exceeds the threshold around $540\mu s$ and $875\mu s$ for the hidden and output layers, respectively. Moreover, as it is shown in Figures 14(c-top) and (d-top), the correlation exceeds the threshold at $70K$ and $110K$ traces for the hidden and output layers, respectively. Compared to the results obtained by employing the BRAM-based HG, the number of traces required to launch the first-order DPA at the hidden and output layers is reduced. Additionally, the correlation is more pronounced when the temperature chamber is used. Yet, using internal HGs could be a valid option to reduce the cost and complexity of the attack.

An observation made in our analysis is that among the two attack vectors used in our experiments (as defined in Section 6.6), the leakage from the output layer was more noticeable when the internal HG was enabled. Interestingly, when an external temperature chamber was used, both attack vectors demonstrated comparable effectiveness. This might be related to the fact that an external temperature chamber can heat up the die more uniformly, in contrast to a (relatively) more local impact of an internal HG. Investigating this can be suggested for future research directions.

7 Discussion

Can we stop generating heat? Reducing the operating temperature of NNs goes hand in hand with designing energy-efficient NNs. In addition to methods devised to predict the energy consumption of a NN and enhance the structure of that accordingly [LWL+20, CJSM17], another line of research has been pursued to allocate memory more carefully to reduce the power consumption in general rather than solely in NNs [GBS+19, TBNG06, TBN+07, KRN+18]. Garcia et al. [GBS+19] have particularly focused on memory-constrained FPGAs in the context of image processing.

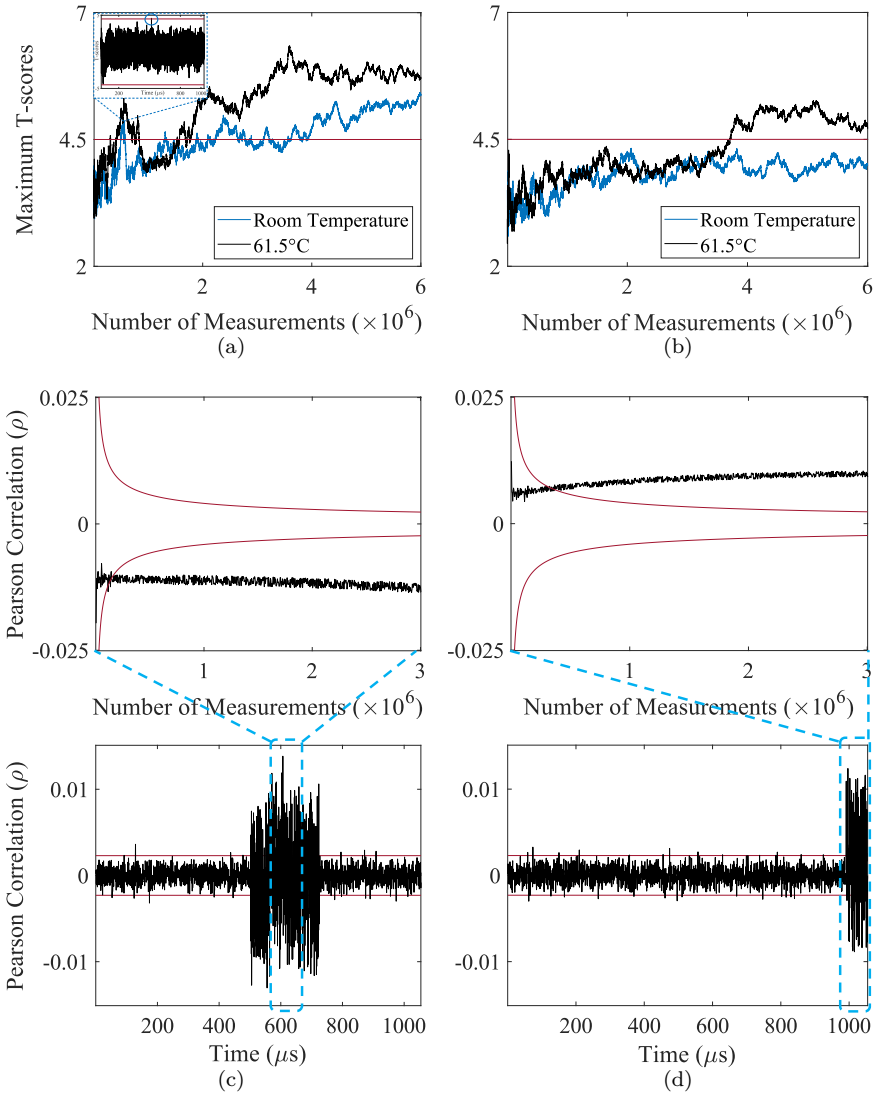

Figure 14: (a) and (b) the maximum t-score vs. the number of traces. Using the temperature chamber, traces are collected, and the t-score is computed as before. Here, for the sake of demonstration, the maximum t-score obtained for (a) the hidden layer and (b) the output layer is depicted. Note that 61.5 °C ambient temperature is equivalent to 70 °C die temperature (see Section 6.3). It is also verified that the random peaks in (a) cannot be exploited to launch an attack (see the t-score vs. time sub-figure in (a)). (c) and (d) results for first-order DPA for $500K$ traces against the ModuloNET's hidden and output layers, with the internal HG off and the PRNG on. In both sub-figures, the results in the bottom image show the correlation vs. time, whereas the top image indicates how many traces are required to launch the attack in a time window around the peak in the bottom image (see dotted line rectangles).

They have proposed a partitioning method backed by theoretical analysis of its effects on resource usage and power consumption. In this way, instead of allocating either some blocks of BRAM with a considerable size or slicing data to smaller groups before memory allocation on a trial-and-error basis, their proposed procedure helps the designer select the

Table 3: Hardware resource allocation in our implementation of ModuloNET used to perform experiments on FF-based HG.

Resource	Used	Utilization (%)	Available
LUT	24072	37.96	63400
FF	**24102**	**19**	126800
BRAM	71	52.59	135

proper memory configuration and optimize on-chip memory usage. Consequently, their approach can decrease the operating temperature. Nevertheless, the effect of Garcia's partitioning method [GBS+19] should be studied more carefully when applied against masked implementation. At the moment, it is unknown how it would counter the risk of unexpected leakage or make heat-induced leakage disappear.

Can off-chip memory provisioning be helpful? It is valid that one could store the *inputs* off-chip and then schedule how the NN should be provided with the inputs efficiently to account for the heat-induced leakage. This could prevent the adversary from generating heat on-chip and protect the design. Yet, this approach causes the bandwidth bottleneck and reduces the efficiency of the accelerator (see Section 4). In some cases, even if the problem with bandwidth would be resolved, off-chip resources for buffering the inputs might not be available due to device constraints. This introduces a trade-off between security and resource requirements. It might be a designer's choice as to whether to go for higher throughput by using on-chip memory and sacrificing security (refer to Section 5) or sacrificing performance for security.

What other heat generators are possible? Besides writing alternating patterns into BRAMs, several other circuit components could generate heat, including the LUT pipeline, shift right logical (SRL) pipeline, FF pipeline, LUT-FF pipeline, digital signal processor (DSP) pipeline, BRAM pipeline, LUT oscillator, and SRL-FF pipeline [HHAP12]. Among all these, based on the results presented in [HHAP12, AHH+14], the BRAM and LUT-FF pipeline could generate the most heat on FPGAs. The first option is similar to our heat generator (ours is not in a pipeline form), where having access to the design input is sufficient to increase the temperature. However, if BRAMs only store intermediate computations (e.g., in BoMoNet [DCA20a], AF BRAMs), generating heat is not under the control of the adversary, although changes in the values stored in those BRAMs might increase the temperature.

FFs. One interesting research direction is whether writing the inputs into FFs also increases the temperature. An observation is that FFs, instead of BRAMs, can be used to store the inputs. Therefore, we examine whether, in both cases of utilizing FFs for storing NN inputs, the heat can be generated by flipping the value of inputs (e.g., image pixels) frequently. Similar to the emulation results presented in Section 6.3, Figure 15 shows the operation temperature of ModuloNET without any HG (black line) and with FF-based HG (black line). Table 3r presents the list of resources used by the design when performing the experiments with FF-based HG. When feeding the flipping images to the design, i.e., writing/reading to/from FFs, the die temperature reaches the maximum of 83.9 °C, which is equivalent to the ambient temperature 73.6 °C. At this ambient temperature, we repeat the test on the delay of FPGA components as demonstrated in Section 6.4, where we observe the FF's delay is increased to 807ps and the delay of the LUT chain raises to 109ps. These changes in the delay can account for generating heat.

In a nutshell, writing/reading to/from a significant number of BRAMs/FFs could generate sufficient heat that leads to detectable leakage; we suggest that designs with these components could be assessed to avoid unexpected leakage.

Moreover, other circuit components could cause high temperatures and are generally used in NNs, although, to the best of our knowledge, they have not been used in masked ones. In this regard, DSPs have been used in the NN design, which is a good source

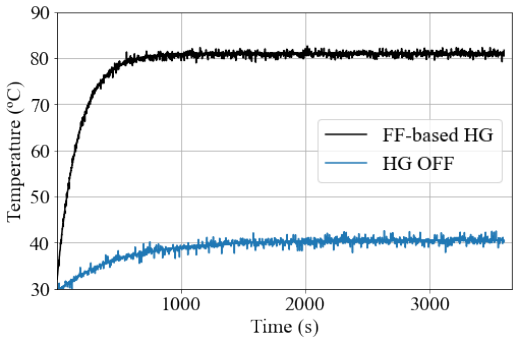

Figure 15: Temperature testing of ModuloNET when giving a normal (not flipping) image and flipping inputs that are written into FFs (FF-based HG).

of heat generation as they operate in a combinational manner under a high clock frequency [HHAP12, AHH+14]. Hence, we highlight the possibility of heat-induced leakage if such a structure is used in masked NNs. As a recommendation, we refer to the results by Happe et al. [HHAP12] indicating that minimizing the interconnection between these components inside the core through pipelining can generate heat on an FPGA.

Leakage verification of B2A and A2B conversion algorithms. As discussed before, we have observed leakage if the B2A module is employed with respect to instructions given in [DAP+22] and as depicted in their Figure 8. For this purpose, we used VERICA [RBFSG22], which discovered the leakage caused by writing/reading the single-bit Boolean shares into/from BRAMs. The issue with verifying the leakage from verified A2B/B2A conversion algorithms has recently been addressed in [GPM22]. More specifically, their findings indicate glitch-based issues for hardware A2B as presented in [CGV14] and transition-based leakage in Goubin's schemes in software [Gou01]. They have, in particular, pointed out several registers overwrite leaks in Goubin's A2B/B2A algorithm implemented in software. Our results complement theirs by demonstrating that Goubin's B2A algorithm should be carefully implemented in hardware, in particular, when dealing with single-bit Boolean shares. We believe that this might have been addressed in Dubey's design as well, irrespective of the design shown in their Figure 8 [DAP+22]), although we could not verify it since their code was unavailable.

Limitations of proposed attack. Regardless of mounting the attack remotely or via direct access to the device, common aspects of the attack include crafting inputs with alternating patterns and collecting and processing the traces. While the former is straightforward to do, the latter two processes might be challenging in a remote attack. Collecting traces, in particular, requires sensors that remain calibrated despite the significant temperature changes. As a prime example, it has been demonstrated that FPGA time-to-digital converter (TDC) sensors would not fit this purpose [GBPS23], but delay-line-based sensors [UJS+22] and routing delay sensors [SGS23] could be exploited in our attack.

Other activities on the chip might affect the signal-to-noise ratio (SNR) of collected traces. Similar to other side-channel attacks, this should be resolved by post-processing the traces before mounting the attack. Additionally and in general, in both remote attacks and ones with physical access, increasing the device's temperature causes changes in the physical properties of its components, such as timing, as described earlier. These changes would indirectly lead to an increase in the thermal noise level in the power traces. This inherent effect of temperature can be reduced using filters, as shown by Moos et al. [MMR19]. They also demonstrated the effectiveness of utilizing simple moving average filters for post-processing trace data. Moreover, physical filters have proven to be efficacious. For instance, a low-pass filter has been strategically integrated with an SLP-30+ probe. We

emphasize that this work aims to demonstrate the possibility of inducing leakage at the first stage. Hence, future investigations could explore various post-filtering techniques in conjunction with SNR analysis, facilitating a comprehensive understanding of their combined effects.

8 Conclusion

Many approaches have been proposed using FPGAs to support a large number of calculations in NN accelerators and achieve the maximum benefit of parallel calculations. In order to protect such accelerators, among various proposed techniques, masking has received a great deal of attention in the recent work [DCA20a, DCA20b, DAP+22]. Our paper introduces a methodology for inducing first-order leakage in FPGA-based accelerators that offer first-order side-channel resilience through masking. Starting from the observation made in [DCEM18], our technique attempts to make the power consumption of different shares dependent on each other by increasing the temperature. For this purpose, in contrast to [DCEM18], we apply novel internal heat generators composed of the NN's components, making our heat generators cheap and an inseparable part of the design. To verify the effectiveness of our method experimentally, we consider ModuloNET [DAP+22] as one of the most recent examples of masked NNs. We observe that by writing alternating patterns into BRAMs, ModuloNET shows unexpected first-order leakage. We emphasize that our paper aims to highlight the possibility of unexpected leakage in correctly implemented masked NNs by means of t-test leakage assessment and DPA. We further note that the modules in our design are tested via VERICA [RBFSG22], where a new vulnerability in the hardware implementation of Goubin's B2A algorithm is identified and resolved in our design. Finally, we discuss possible countermeasure and their associated challenges. As for future directions, security-aware memory allocation for masked NNs is suggested. We also attempted to explain why increasing the temperature can result in leakage. Further investigation of the theory behind this phenomenon can be within the scope of future work. Moreover, it is essential to reiterate that our primary goal was to underscore the potential for inducing first-order leakage in first-order protected NNs with high utilization of BRAMs. The idea of inducing leakage by increasing the temperatures in other masked schemes has been left for future exploration.

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 number 2138420. David S. Koblah is supported by the Department of Defense through the Science, Mathematics, and Research for Transformation (SMART) Scholarship-for-Service Program.

References

[Adv24] Advanced Micro Devices (AMD), Inc. Xilinx power estimator (xpe). [Online]https://www.xilinx.com/products/technology/power/xpe.html [Accessed: Mar.29, 2024], 2024.

[AHH+14] Andreas Agne, Hendrik Hangmann, Markus Happe, Marco Platzner, and Christian Plessl. Seven recipes for setting your fpga on fire–a cookbook on heat generators. *Microprocessors and Microsystems*, 38(8):911–919, 2014.

[ATF16] Md Mahbub Alam, Mark Tehranipoor, and Domenic Forte. Recycled fpga detection using exhaustive lut path delay characterization. In *2016 IEEE Intrl. test Conf. (ITC)*, pages 1–10. IEEE, 2016.

[ATG+19] Md Mahbub Alam, Shahin Tajik, Fatemeh Ganji, Mark Tehranipoor, and Domenic Forte. Ram-jam: Remote temperature and voltage fault attack on fpgas using memory collisions. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 48–55, 2019.

[BBB+22] Anubhab Baksi, Shivam Bhasin, Jakub Breier, Dirmanto Jap, and Dhiman Saha. A survey on fault attacks on symmetric key cryptosystems. *ACM Computing Surveys*, 55(4):1–34, 2022.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 116–129, 2016.

[BBJP19] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symp. (USENIX Security 19)*, pages 515–532, 2019.

[BCD+13] George Becker, Jim Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, Timofei Kouzminov, Andrew Leiserson, Mark Marson, Pankaj Rohatgi, et al. Test vector leakage assessment (tvla) methodology in practice. In *International Cryptographic Module Conference*, volume 1001, page 13. sn, 2013.

[BDK+09] Julien Brouchier, Nora Dabbous, Tom Kean, Carol Marsh, and David Naccache. Thermocommunication. *Cryptology ePrint Archive*, 2009.

[BH22] Jakub Breier and Xiaolu Hou. How practical are fault injection attacks, really? *IEEE Access*, 10:113122–113130, 2022.

[BJP22] Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. On (in) security of edge-based machine learning against electromagnetic side-channels. In *2022 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI)*, pages 262–267. IEEE, 2022.

[BKMN09] Julien Brouchier, Tom Kean, Carol Marsh, and David Naccache. Temperature attacks. *IEEE Security & Privacy*, 7(2):79–82, 2009.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck families of lightweight block ciphers. *cryptology eprint archive*, 2013.

[CGTV15] Jean-Sébastien Coron, Johann Großschädl, Mehdi Tibouchi, and Praveen Kumar Vadnala. Conversion from arithmetic to boolean masking with logarithmic complexity. In *International Workshop on Fast Software Encryption*, pages 130–149. Springer, 2015.

[CGV14] Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala. Secure conversion between boolean and arithmetic masking of any order. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 188–205. Springer, 2014.

[CHS+16] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[CJSM17] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, and Diana Marculescu. Neuralpower: Predict and deploy energy-efficient convolutional neural networks. In *Asian Conference on Machine Learning*, pages 622–637. PMLR, 2017.

[CLL+14] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao: A machine-learning supercomputer. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622. IEEE, 2014.

[Cor17] Jean-Sébastien Coron. High-order conversion from boolean to arithmetic masking. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 93–114. Springer, 2017.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Transactions on Information Forensics and Security*, 15:2542–2555, 2020.

[CSJC10] Srimat Chakradhar, Murugan Sankaradas, Venkata Jakkula, and Srihari Cadambi. A dynamically configurable coprocessor for convolutional neural networks. In *Proceedings of the 37th annual international symposium on Computer architecture*, pages 247–257, 2010.

[DAP+22] Anuj Dubey, Afzal Ahmad, Muhammad Adeel Pasha, Rosario Cammarota, and Aydin Aysu. Modulonet: Neural networks meet modular arithmetic for efficient hardware masking. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, pages 506–556, 2022.

[DCA20a] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. Bomanet: Boolean masking of an entire neural network. In *2020 IEEE/ACM Intrl. Conf. On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2020.

[DCA20b] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. Maskednet: The first hardware inference engine aiming power side-channel protection. In *2020 IEEE Intrl. Symp. on Hardware Oriented Security and Trust (HOST)*, pages 197–208. IEEE, 2020.

[DCEM18] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware masking, revisited. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, pages 123–148, 2018.

[DFC+15] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. Shidiannao: Shifting vision processing closer to the sensor. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 92–104, 2015.

[DFS19] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. *Journal of Cryptology*, 32(4):1263–1297, 2019.

[DKAA22] Anuj Dubey, Emre Karabulut, Amro Awad, and Aydin Aysu. High-fidelity model extraction attacks via remote power monitors. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 328–331. IEEE, 2022.

[DMRB18] Lauren De Meyer, Oscar Reparaz, and Begül Bilgin. Multiplicative masking for aes in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 431–468, 2018.

[DN20] Siemen Dhooghe and Svetla Nikova. My gadget just cares for me-how nina can prove security against combined attacks. In *Cryptographers' Track at the RSA Conference*, pages 35–55. Springer, 2020.

[FA01] IM Filanovsky and Ahmed Allam. Mutual compensation of mobility and threshold voltage temperature effects with applications in cmos circuits. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(7):876–884, 2001.

[FGDP+18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 89–120, 2018.

[GA03] S. Govindavajhala and A.W. Appel. Using memory errors to attack a virtual machine. In *2003 Symposium on Security and Privacy, 2003.*, pages 154–165, 2003.

[GBPS23] Ognjen Glamočanin, Hajira Bazaz, Mathias Payer, and Mirjana Stojilović. Temperature impact on remote power side-channel attacks on shared fpgas. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.

[GBS+19] Paulo Garcia, Deepayan Bhowmik, Robert Stewart, Greg Michaelson, and Andrew Wallace. Optimized memory allocation and power minimization for fpga-based image processing. *Journal of Imaging*, 5(1):7, 2019.

[GGJR+11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[GM17] Hannes Groß and Stefan Mangard. Reconciling $d + 1$ masking in hardware and software. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 115–136. Springer, 2017.

[GMK16] Hannes Gross, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, TIS '16, page 3, New York, NY, USA, 2016. Association for Computing Machinery.

[GMK17] Hannes Groß, Stefan Mangard, and Thomas Korak. An efficient side-channel protected aes implementation with arbitrary protection order. In *Cryptographers' Track at the RSA Conference*, pages 95–112. Springer, 2017.

[GOKT16] Dennis RE Gnad, Fabian Oboril, Saman Kiamehr, and Mehdi B Tahoori. Analysis of transient voltage fluctuations in fpgas. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 12–19. IEEE, 2016.

[Gol07] Jovan Dj Golic. Techniques for random masking in hardware. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(2):291–300, 2007.

[Gou01] Louis Goubin. A sound method for switching between boolean and arithmetic masking. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 3–15. Springer, 2001.

[GPM22] Barbara Gigerl, Robert Primas, and Stefan Mangard. Formal verification of arithmetic masking in hardware and software. *Cryptology ePrint Archive*, 2022.

[GWWT12] Martin Gag, Tim Wegner, Ansgar Waschki, and Dirk Timmermann. Temperature and on-chip crosstalk measurement using ring oscillators in fpga. In *2012 IEEE 15th Intrl. Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 201–204. IEEE, 2012.

[GYSC17] Yijin Guan, Zhihang Yuan, Guangyu Sun, and Jason Cong. FPGA-based accelerator for long short-term memory recurrent neural networks. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 629–634. IEEE, 2017.

[HAM+86] Hoji Hanamura, Masaaki Aoki, Toshiaki Masuhara, Osamu Minato, Yoshio Sakai, and Testsuya Hayashida. Operation of bulk cmos devices at very low temperatures. *IEEE journal of solid-state circuits*, 21(3):484–490, 1986.

[HHAP12] Markus Happe, Hendrik Hangmann, Andreas Agne, and Christian Plessl. Eight ways to put your fpga on fire—a systematic study of heat generators. In *2012 International Conference on Reconfigurable Computing and FPGAs*, pages 1–6. IEEE, 2012.

[Hor14] Mark Horowitz. Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[HS13] Michael Hutter and Jörn-Marc Schmidt. The temperature side-channel and heating fault attacks. volume 8419, 11 2013.

[HT19] Michael Hutter and Michael Tunstall. Constant-time higher-order boolean-to-arithmetic masking. *Journal of Cryptographic Engineering*, 9(2):173–184, 2019.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Annual International Cryptology Conference*, pages 463–481. Springer, 2003.

[Kal13] Shruti Kalra. Effect of temperature dependence on performance of digital cmos circuit technologies. In *2013 Intrl. Conf. on Signal Processing and Communication (ICSC)*, pages 392–395. IEEE, 2013.

[KGT22] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. Remote fault attacks in multitenant cloud fpgas. *IEEE Design & Test*, 39(4):33–40, 2022.

[KHEB14] Thomas Korak, Michael Hutter, Baris Ege, and Lejla Batina. Clock glitch attacks in the presence of heating. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 104–114, 2014.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[KRN+18] Inderpreet Kaur, Lakshay Rohilla, Alisha Nagpal, Bishwajeet Pandey, and Sanchit Sharma. Different configuration of low-power memory design using capacitance scaling on 28-nm field-programmable gate array. In *System and Architecture*, pages 151–161. Springer, 2018.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. Silver–statistical independence and leakage verification. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 787–816. Springer, 2020.

[LeC19] Yann LeCun. 1.1 deep learning hardware: Past, present, and future. In *2019 IEEE Intrl. Solid-State Circuits Conf.-(ISSCC)*, pages 12–19. IEEE, 2019.

[LFJ+16] Huimin Li, Xitian Fan, Li Jiao, Wei Cao, Xuegong Zhou, and Lingli Wang. A high performance fpga-based accelerator for large-scale convolutional neural networks. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–9. IEEE, 2016.

[LGFX21] Yukui Luo, Cheng Gongye, Yunsi Fei, and Xiaolin Xu. Deepstrike: Remotely-guided fault injection attacks on dnn accelerator in cloud-fpga. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 295–300, 2021.

[LS94] Kenneth R Laker and Willy MC Sansen. *Design of analog integrated circuits and systems*, volume 1. McGraw-Hill New York, 1994.

[LWL+20] Shengwen Liang, Ying Wang, Cheng Liu, Lei He, LI Huawei, Dawen Xu, and Xiaowei Li. Engn: A high-throughput and energy-efficient accelerator for large graph neural networks. *IEEE Transactions on Computers*, 70(9):1511–1525, 2020.

[MBS+24] Zitouni Messai, Abdelhalim Brahimi, Okba Saidani, Nacerdine Bourouba, and Abderrahim Yousfi. Investigation of temperature and channel dimension effects on cmos circuit performance. *East European Journal of Physics*, (1):417–425, 2024.

[Mes00] Thomas S Messerges. Using second-order power analysis to attack dpa resistant software. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 238–251. Springer, 2000.

[MHS08] A Muthuramalingam, S Himavathi, and E Srinivasan. Neural network implementation using fpga: issues and application. *International Journal of Electrical and Computer Engineering*, 2(12):2802–2808, 2008.

[MLS22] Dina G Mahmoud, Vincent Lenders, and Mirjana Stojilović. Electrical-level attacks on cpus, fpgas, and gpus: Survey and implications in the heterogeneous era. *ACM Computing Surveys (CSUR)*, 55(3):1–40, 2022.

[MMR19] Thorben Moos, Amir Moradi, and Bastian Richter. Static power side-channel analysis—an investigation of measurement factors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(2):376–389, 2019.

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M Gammel. Side-channel leakage of masked cmos gates. In *Cryptographers' Track at the RSA Conference*, pages 351–365. Springer, 2005.

[MS19] Dina Mahmoud and Mirjana Stojilović. Timing violation induced faults in multi-tenant fpgas. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1745–1750, 2019.

[MTH+21] Shayan Moini, Shanquan Tian, Daniel Holcomb, Jakub Szefer, and Russell Tessier. Remote Power Side-Channel Attacks on BNN Accelerators in FPGAs. *Proceedings -Design, Automation and Test in Europe, DATE*, 2021-February(2):1639–1644, 2021.

[MTMM07] Robert McEvoy, Michael Tunstall, Colin C Murphy, and William P Marnane. Differential power analysis of hmac based on sha-2, and countermeasures. In *International Workshop on Information Security Applications*, pages 317–332. Springer, 2007.

[MVZ+21] Jian Meng, Shreyas Kolala Venkataramanaiah, Chuteng Zhou, Patrick Hansen, Paul Whatmough, and Jae-sun Seo. FixyFPGA: Efficient fpga accelerator for deep neural networks with high element-wise sparsity and without external memory access. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pages 9–16. IEEE, 2021.

[NHCB02] Anshuman Nayak, Malay Haldar, Alok Choudhary, and Prithviraj Banerjee. Accurate area and delay estimators for fpgas. In *Proc. 2002 Design, Automation and Test in Europe Conf. and Exhibition*, pages 862–869. IEEE, 2002.

[NSS+16] Eriko Nurvitadhi, David Sheffield, Jaewoong Sim, Asit Mishra, Ganesh Venkatesh, and Debbie Marr. Accelerating binarized neural networks: Comparison of fpga, cpu, gpu, and asic. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 77–84. IEEE, 2016.

[OH11] Takaaki Okumura and Masanori Hashimoto. Setup time, hold time and clock-to-q delay computation under dynamic supply noise. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 94(10):1948–1953, 2011.

[OMHT06] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical second-order dpa attacks for masked smart card implementations of block ciphers. In *Cryptographers' Track at the RSA Conference*, pages 192–207. Springer, 2006.

[PBR17] Roberta Piscitelli, Shivam Bhasin, and Francesco Regazzoni. Fault attacks, injection techniques and tools for simulation. In *Hardware security and trust*, pages 27–47. Springer, 2017.

[RBFSG22] Jan Richter-Brockmann, Jakob Feldtkeller, Pascal Sasdrich, and Tim Güneysu. VERICA - verification of combined attacks: Automated formal verification of security against simultaneous information leakage and tampering. Cryptology ePrint Archive, Paper 2022/484, 2022.

[RBK19] Siavash Rezaei, Eli Bozorgzadeh, and Kanghee Kim. Ultrashare: Fpga-based dynamic accelerator sharing and allocation. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–5. IEEE, 2019.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *Annual Cryptology Conference*, pages 764–783. Springer, 2015.

[RBSG22] Jan Richter-Brockmann, Pascal Sasdrich, and Tim Guneysu. Revisiting fault adversary models–hardware faults in theory and practice. *IEEE Transactions on Computers*, 2022.

[RBSS+21] Jan Richter-Brockmann, Aein Rezaei Shahmirzadi, Pascal Sasdrich, Amir Moradi, and Tim Güneysu. Fiver–robust verification of countermeasures against fault injections. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 447–473, 2021.

[RHCM+16] Siavash Rezaei, Cesar-Alejandro Hernandez-Calderon, Saeed Mirzamohammadi, Eli Bozorgzadeh, Alexander Veidenbaum, Alex Nicolau, and Michael J Prather. Data-rate-aware fpga-based acceleration framework for streaming applications. In *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–6. IEEE, 2016.

[RHL+18] Zhenyuan Ruan, Tong He, Bojie Li, Peipei Zhou, and Jason Cong. St-accel: A high-level programming platform for streaming applications on fpga. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 9–16. IEEE, 2018.

[RPD+18] Chethan Ramesh, Shivukumar B Patil, Siva Nishok Dhanuskodi, George Provelengios, Sébastien Pillement, Daniel Holcomb, and Russell Tessier. Fpga side channel attacks without physical access. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 45–52. IEEE, 2018.

[SC06] Pete Sedcole and Peter YK Cheung. Within-die delay variability in 90nm fpgas and beyond. In *2006 IEEE Intrl. Conf. on Field Programmable Technology*, pages 97–104. IEEE, 2006.

[SFM17] Yongming Shen, Michael Ferdman, and Peter Milder. Maximizing cnn accelerator efficiency through resource partitioning. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 535–547. IEEE, 2017.

[SGMT18] Falk Schellenberg, Dennis RE Gnad, Amir Moradi, and Mehdi B Tahoori. An inside job: Remote power analysis attacks on fpgas. In *2018 Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, pages 1111–1116. IEEE, 2018.

[SGS10] John E Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, 12(3):66, 2010.

[SGS23] David Spielmann, Ognjen Glamočanin, and Mirjana Stojilović. Rds: Fpga routing delay sensors for effective remote power analysis attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(2):543–567, 2023.

[Sko09] Sergei Skorobogatov. Local heating attacks on flash memory devices. In *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 1–6, 2009.

[SLL+08] Rudolf Schlangen, Rainer Leihkauf, Ted Lundquist, Peter Egger, Uwe Kerst, and Christian Boit. Trimming of ic timing and delay by backside fib processing-comparison of conventional and strained technologies. In *2008 IEEE Intrl. Electron Devices Meeting*, pages 1–4. IEEE, 2008.

[SSEM18] Ahmad Shawahna, Sadiq M Sait, and Aiman El-Maleh. Fpga-based accelerators of deep learning networks for learning and classification: A review. *ieee Access*, 7:7823–7859, 2018.

[Syn20] Synopsys. v2020.09-sp4. [Online]https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html [Accessed: Jan.11, 2023], 2020.

[TBN+07] Russell Tessier, Vaughn Betz, David Neto, Aaron Egier, and Thiagaraja Gopalsamy. Power-efficient ram mapping algorithms for fpga embedded memory blocks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(2):278–290, 2007.

[TBNG06] Russell Tessier, Vaughn Betz, David Neto, and Thiagaraja Gopalsamy. Power-aware ram mapping for fpga embedded memory blocks. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, pages 189–198, 2006.

[TG22] Shahin Tajik and Fatemeh Ganji. Artificial neural networks and fault injection attacks. In *Security and Artificial Intelligence*, pages 72–84. Springer, 2022.

[TL23] Inc. Teledyne LeCroy. Wavepro 254hd 2.5 ghz high definition oscilloscope. [Online]https://www.teledynelecroy.com/oscilloscope/wavepro-hd-oscilloscope/wavepro-254hd [Accessed: Sep.14, 2023], 2023.

[TMW+21] Shanquan Tian, Shayan Moini, Adam Wolnikowski, Daniel Holcomb, Russell Tessier, and Jakub Szefer. Remote Power Attacks on the Versatile Tensor Accelerator in Multi-Tenant FPGAs. *Proceedings - 29th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2021*, pages 242–246, 2021.

[Tsi87] Yannis Tsividis. *Operation and modeling of the MOS transistor*. McGraw-Hill, Inc., USA, 1987.

[UJS+22] Brian Udugama, Darshana Jayasinghe, Hassaan Saadat, Aleksandar Ignjatovic, and Sri Parameswaran. Viti: A tiny self-calibrating sensor for power-variation measurement in fpgas. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 657–678, 2022.

[USA22] USA Today. Are us data centers fueling climate change? the best (and worst) regions for clean energy. [Online] https://www.usatoday.com/story/tech/2022/08/24/climate-change-data-center-oil-gas-wind/7875280001/?gnt-cfr=1 [Accessed: Jan.11, 2023], 2022.

[WFKP98] Glen O Workman, Jerry G Fossum, Srinath Krishnan, and MM Pelella. Physical modeling of temperature dependences of soi cmos devices and circuits including self-heating. *IEEE Transactions on Electron Devices*, 45(1):125–133, 1998.

[WGY+16] Chao Wang, Lei Gong, Qi Yu, Xi Li, Yuan Xie, and Xuehai Zhou. Dlau: A scalable deep learning accelerator unit on fpga. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(3):513–517, 2016.

[XAQ21] Qian Xu, Md Tanvir Arafin, and Gang Qu. Security of neural networks from hardware perspective: A survey and beyond. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 449–454. IEEE, 2021.

[XCC+20] Yun Xiang, Zhuangzhi Chen, Zuohui Chen, Zebin Fang, Haiyang Hao, Jinyin Chen, Yi Liu, Zhefu Wu, Qi Xuan, and Xiaoniu Yang. Open dnn box by power side-channel attack. *IEEE Trans. on Circuits and Systems II: Express Briefs*, 67(11):2717–2721, 2020.

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

[YKO+20] Kota Yoshida, Takaya Kubota, Shunsuke Okura, Mitsuru Shiozaki, and Takeshi Fujino. Model reverse-engineering attack using correlation power analysis against systolic array based neural network accelerator. In *2020 IEEE Intrl. Symp. on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2020.

[YMY+20] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. Deepem: Deep neural networks model recovery through em side-channel information leakage. In *2020 IEEE Intrl. Symp. on Hardware Oriented Security and Trust (HOST)*, pages 209–218. IEEE, 2020.

[ZLS+15] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 161–170, 2015.

[ZP17] Chi Zhang and Viktor Prasanna. Frequency domain acceleration of convolutional neural networks on cpu-fpga shared memory system. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 35–44, 2017.

[ZS18] Mark Zhao and G Edward Suh. FPGA-based remote power side-channel attacks. In *2018 IEEE Symp. on Security and Privacy (SP)*, pages 229–244. IEEE, 2018.

[ZSZ+17] Ritchie Zhao, Weinan Song, Wentao Zhang, Tianwei Xing, Jeng-Hau Lin, Mani Srivastava, Rajesh Gupta, and Zhiru Zhang. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 15–24, 2017.

[ZYC+21] Yicheng Zhang, Rozhin Yasaei, Hao Chen, Zhou Li, and Mohammad Abdullah Al Faruque. Stealing neural network structure through remote fpga side-channel analysis. *IEEE Transactions on Information Forensics and Security*, 16:4377–4388, 2021.