FPGA Side Channel Attacks without Physical Access

Chethan Ramesh*, Shivukumar B. Patil*, Siva Nishok Dhanuskodi*, George Provelengios*,
Sébastien Pillement†, Daniel Holcomb*, and Russell Tessier*
*Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, USA
†IETR, University of Nantes, France

***Abstract*—As FPGA use becomes more diverse, the shared use of these devices becomes a security concern. Multi-tenant FPGAs that contain circuits from multiple independent sources or users will soon be prevalent in cloud and embedded computing environments. The recent discovery of a new attack vector using neighboring long wires in Xilinx SRAM FPGAs presents the possibility of covert information leakage from an unsuspecting user's circuit. The work described in this paper makes two contributions that dramatically extend this finding. First, we rigorously evaluate several Intel SRAM FPGAs and confirm that long wire information leakage is also prevalent in these devices. Second, we present the first successful attack on an unsuspecting circuit in an FPGA using information passively obtained from neighboring long-lines. Information obtained from a single AES S-box input wire combined with analysis of encrypted output is used to rapidly expose an AES key. This attack is performed remotely without modifying the victim circuit, using electromagnetic probes or power measurements, or modifying the FPGA in any way. We show that our approach is effective for three different FPGA devices. Our results demonstrate that the attack can recover encryption keys from AES circuits running at 10MHz, and has the capability to scale to much higher frequencies.**

I. Introduction

FPGAs are quickly growing in importance in a variety of computing spaces including cloud computing and embedded platforms (automotive, military, and aerospace). As FPGAs grow in size and complexity, it is apparent that numerous applications from independent users may simultaneously reside in a single FPGA device. This use of *multi-tenant* FPGAs opens the door to numerous potential attack vectors on unsuspecting co-located FPGA circuits. Although FPGA devices in cloud computing environments such as Microsoft Catapult [1] and Amazon EC2 F1 [2] are currently dedicated to a specific application, the growing capabilities of FPGAs makes it easy to envision single-FPGA platforms containing multiple independent applications created by completely separate entities.

The identification of a new covert channel in FPGAs based on measurable crosstalk between long wires has opened up a new attack vector for multi-tenant FPGAs. A study of several Xilinx FPGAs showed that neighboring long wires in an interconnect routing channel can be used as a transmitter-receiver pair [3], [4]. The receiver is part of a ring oscillator while the transmitter is part of a user design. The ring oscillator frequency was shown to be directly related to the logic value present on the transmitter. The effect was shown to be robust across a variety of transmitter clock frequencies and device locations. This covert communication channel opens up the possibility of on-chip data spying by an adversary with no physical access to the FPGA device. Although this prior work provides value in identifying a new FPGA covert communication channel, it was confined to Xilinx devices and was not used to perform an attack on a multi-tenant user design.

In this paper, we verify that the covert channel exists on long wires in Intel Cyclone IV and Stratix V devices using a diverse set of experiments on multiple FPGA boards. More importantly, we demonstrate that it is possible to covertly extract the encryption key from an AES-128 encryption core that has been automatically placed and routed by Quartus design tools by simply using side channel information leaked from long wires. Our AES experiments are performed multiple times for a variety of design clock frequencies and long wire signal lengths to verify correctness.

The remainder of this paper is structured as follows. Section II provides background on multi-tenant FPGAs and the risks posed by crosstalk information leakage. We present our experiments and characterization results on long wire information leakage for a family of Intel SRAM FPGAs in Section III. In Section IV we describe the details of our AES implementation and the attack approach used to identify the encryption key. The experimental results regarding our attacks on AES are presented in Section V. Section VI provides conclusions and offers directions for future work.

II. Background and Related Work

A. Multi-Tenant FPGAs

The concept of multi-tenant FPGA use by independent applications is perhaps best illustrated in the context of FPGA-based cloud computing. In 2014, the Microsoft Catapult project [1] introduced the scalable use of FPGAs within Microsoft data centers with a goal of accelerating the Bing search engine. This effort has grown to include FPGA-based hardware for many if not all of Microsoft's data center installations [5]. In late 2016, Amazon introduced the EC2 F1 that leverages its AWS cloud infrastructure. To date, both Microsoft and Amazon only allow one user access to an FPGA resource at a time. To do otherwise presents a security threat as evidenced by this comment on the Amazon F1 web site [2]: *"Each F1 instance includes up to eight FPGAs that are dedicated to the instance. They are not shared between instances, users,*

or accounts. This ensures that the full power of the FPGA is dedicated to the instance, and improves security through user and account isolation." However, given the size and cost of FPGAs it is likely that these resources will be shared in the future in much the same way that cloud microprocessors are shared across multiple virtual machines. Additionally, given the distributed interconnect in FPGAs, even if logic for different subcircuits are isolated, their routing resources in channels may be in close proximity.

B. Long Wire Attacks in FPGAs

The discovery of a covert communication channel between neighboring FPGA long wires (also called "long lines") has the potential to dramatically change the threat level of multi-tenant FPGAs. In a comprehensive set of experiments, Giechaskiel *et al.* [3] showed that the logic value carried on a long wire influences the delay of both its immediate neighbor and a long wire in the same channel two wires away. When a logic 1 value is carried on a wire (the transmitter), the delay in the neighboring wire (the receiver) is reduced relative to when a logic 0 is transmitted. This result was shown to be unaffected by the signal switching rate of the transmitter, the long wire location on the FPGA, and the direction of signal transmission for the transmitter and receiver in FPGA channels. In addition to verifying the robust presence of this covert communication channel for multiple generations and instances of Xilinx SRAM FPGAs, the authors characterize the achievable communication bandwidth, investigate several simple countermeasures, and offer directions for possibly using the phenomenon in a data snooping attack. Several hypotheses are provided for the source of the phenomenon, although no definitive cause is provided.

Although interesting, this earlier work leaves several unanswered questions. Specifically, since the source of the crosstalk[1] effects between long wires is unclear it is unknown whether the same effect can be observed and measured in SRAM FPGAs from Intel. In this work, we confirm that the effect is indeed present. Perhaps more importantly, we demonstrate that the encryption key for an AES-128 circuit can be successfully obtained by adding a snooping (receiver) circuit to a design that is automatically placed and routed by FPGA physical design tools. The attack is shown to be effective if a single wire of the core is routed on a vertical C4 long-line that spans four logic array blocks (LABs).

C. Relationship to Previous FPGA Attack Approaches

Attempts to extract information from FPGAs via physical attacks have mostly focused on power or thermal analysis. Power side channel attacks apply statistical processing to steal encryption keys based on data-dependent differences in the power consumption of block ciphers [6]. Power in side channel attacks is typically measured through a sense resistor external to the chip [6], or through electromagnetic

[1]The term "crosstalk" often refers specifically to capacitive coupling between wires. In this paper we adopt the terminology of Giechaskiel et al. [3] and use the word crosstalk in a more general sense to describe the unspecified interaction between neighboring wires.

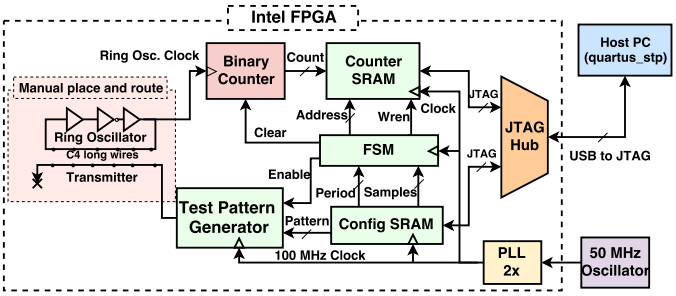


Fig. 1: Experimental framework for evaluating long wire delay effects on Intel SRAM FPGAs.

emanations [7] which can provide more localized information about consumption. When data-dependent power consumption causes small local fluctuations in supply voltage, these same power analysis techniques may allow supply voltage sensing circuits, such as oscillators or tuned path delays inside the FPGA, to detect the local fluctuations and steal data. Recent work has explored power side channel attacks inside FPGAs [8], [9].

It is possible to implement a temperature-to-frequency transducer suitable for thermal monitoring on FPGAs using a ring oscillator [10]. The dependence between delay and temperature can be used to measure temperature in the FPGA. Further, multiple such modules can be realized to measure temperature in different parts of the FPGA. One sender circuit co-located on the same FPGA with a receiver circuit (but which do not have a direct communication path) can leak secrets. The sender can heat up the FPGA fabric and the receiver can read the increased temperature. A temperature-based covert communication channel has been shown to be possible in stand-alone FPGAs [11] under tight restrictions. However, in these cases temperature information transmission is deliberate. In contrast, thermal leakage is of limited use for monitoring data from unsuspecting victims.

It is well known that interconnect crosstalk inside an FPGA can change signal values and cause critical signal delays [12]. Crosstalk is potentially a much larger threat in security than in reliability. An attacker can leverage a wide array of detectable couplings via trial and error (e.g. via dynamic FPGA reconfiguration) while reliability is only compromised by a relatively large coupling. FPGA physical design tools for individual designs explicitly check and avoid crosstalk conditions to prevent on-chip interference. In the multi-tenant scenario where sub-designs are created separately, it is more difficult to prevent crosstalk.

III. Long Wire Attack Verification for Intel FPGAs

In an initial set of experiments, we carefully examine the potential leakage of information across near-neighbor long wires in a vertical interconnect channel.

A. Experimental Approach

To evaluate long wire effects we perform experiments on three Altera DE2-115 (EP4CE115F29) boards, one Cyclone

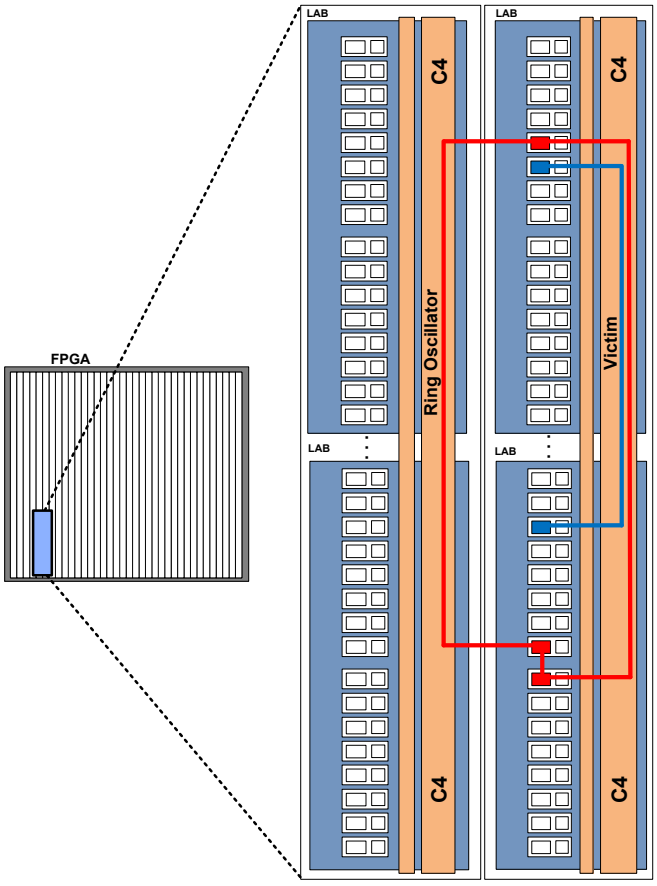


Fig. 2: Transmitter (victim) and receiver implemented in a Cyclone IV FPGA.

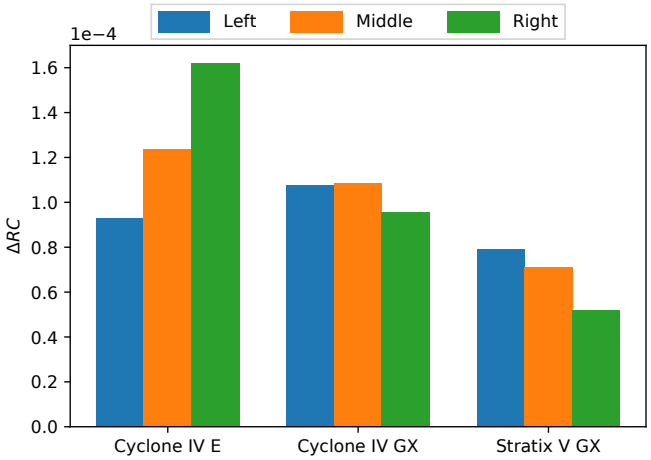


Fig. 3: Relative ring oscillator count difference due to the value of the adjacent transmitter (see Eq. 1). For this experiment, both transmitter and receiver include two vertically-connected C4 wires. Across locations and FPGAs, driving a 1 onto the transmitter causes a receiver speed up on the order of 0.01% (1 part per 10,000).

IV GX (EP4CGX150DF31) FPGA Development Kit, and one Stratix V (5SGXEA7K2F40C2N) GX Development Kit. Fig. 1 shows the block diagram of the test setup used to assess the long wire covert channels in these system types. The *transmitter* is implemented in the FPGA in one or more vertical C4 long wires. The test pattern generator (Fig. 1) assigns either a logic 1 or a logic 0 to the transmitter. Similar to [3], the receiver is implemented as a three-stage ring oscillator (RO) with one inverter and two buffers. One of the wires of the RO is located on one or more connected long wires adjacent to the transmitter. For this experimentation, the ring oscillator, transmitter and receiver are placed and routed using place and route constraints. A view of a transmitter and a receiver (RO) using LABs and C4 wires is shown in Fig. 2.

A binary counter measures the RO's frequency by incrementing a 32-bit count value at every positive edge of the ring oscillator clock for a fixed time duration (measurement period). After the measurement period, the count values are sampled and stored into an on-chip SRAM. A SignalTap II JTAG interface is used to read the stored count values. The design has a JTAG-accessible configuration memory and a finite-state machine (FSM) which coordinate test pattern generation, counting, and sampling of counter values. Unless otherwise noted, all circuitry, except the transmitter and receiver, are auto-placed and routed by Quartus Prime v17.0. Each experiment is performed five times on each device and a total of 4,096 samples are collected for each measurement period. Unless otherwise noted, each test uses a 21 ms measurement period derived from a 100 MHz system clock generated by an on-chip PLL. After each test, the binary counter is reset.

We evaluate the difference in RO frequency for two trials by using a relative count metric [3] determined over two measurement periods. For example, the count difference ΔRC of the receiver for trials when the transmitter is first 0 (first trial) and then 1 (second trial) can be represented as:

$$\Delta RC = \frac{C^1 - C^0}{C^1} \quad (1)$$

where C^1 and C^0 are the measured counts for transmitted logic 1 and 0, respectively.

B. Characterization Results

In this section, we verify the existence of the covert communication channel between neighboring long wires (C4) in three Intel SRAM FPGAs. Unless otherwise noted, all experiments use a single transmitter with one or more vertical C4 wires. The transmitter is driven with either a static logic 0 or a static logic 1 during a measurement period. In an initial experiment, transmitter-receiver pairs that consist of two consecutive vertical C4 wires were implemented in three distinct locations (left, middle, and right) on the Intel FPGAs described in Section III. As seen in Fig. 3, relative counts in all three cases clearly differentiate the transmitted logic 0 from a logic 1. The value of ΔRC varies across the chip location and the FPGA model, but in all cases it is observed that the state of the neighboring wire impacts the ring oscillator frequency by an amount that is on the order of 0.01% (one part per ten-thousand), which we will show later in the paper is sufficient

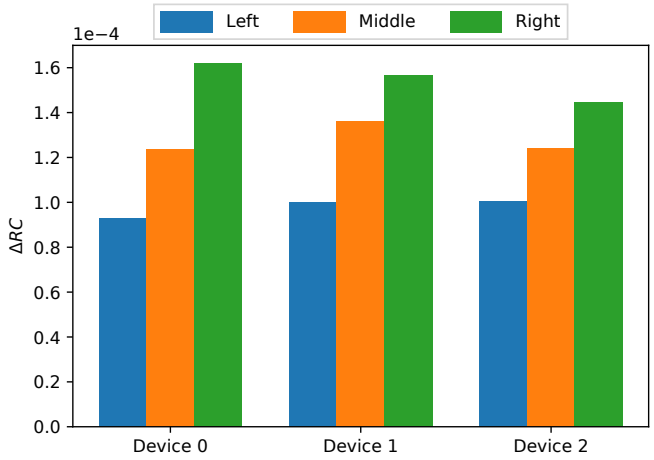


Fig. 4: Repeating the experiment from Fig. 3 on the Cyclone IV E devices from three identical DE2-115 boards produces comparable results.

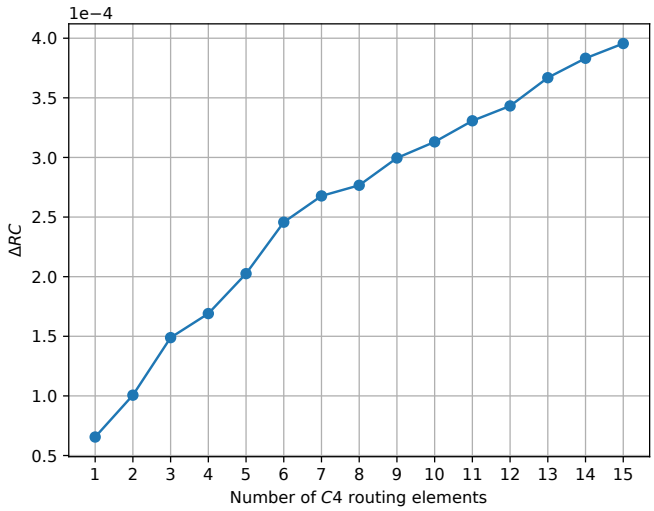


Fig. 5: Relative count differences increase with the transmitter/receiver lengths (in terms of number of C4 long wires). The experiment uses a transmitter/receiver pair at the bottom left of the Cyclone IV (EP4CE115F29) FPGA.

for conducting side channel attacks. Consistent count results were achieved for all five trials on the three distinct boards.

In a second experiment, the same transmitter-receiver experiment as described above is applied to identical Cyclone IV FPGAs on three DE2-115 boards. The results shown in Fig. 4 indicate that the results from Fig. 3 are repeatable and consistent across FPGA chip instances.

The covert channel between the transmitter and receiver becomes stronger as the length of the neighboring wires increases. As seen in Fig. 5, relative count differences increase linearly as the extent of the pair increase. However, it is noteworthy that the effect can be seen for pairs that are only one or two C4 wires long. As shown in Section V, a successful key extraction attack on AES can be performed

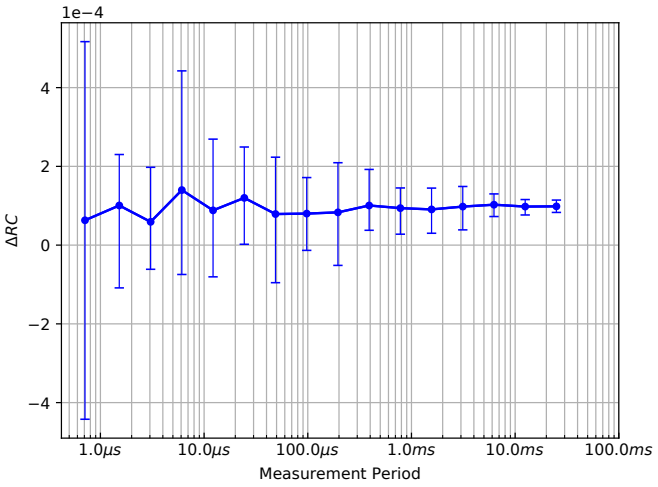


Fig. 6: Relative frequency count with respect to measurement period. Both the receiver and transmitter use two C4 wires.

Configuration	ΔRC (1e-4)
0 0 1 RO 0 0 0	0.965
0 0 0 RO 1 0 0	1.002
0 0 1 RO 1 0 0	2.000
1 1 0 RO 0 1 1	-0.041
0 0 0 RO 0 0 0	–

TABLE I: Relative count results for different transmitter configurations. The transmitter and receivers have a length of two C4 wires. The final row is the baseline configuration against which the ΔRC values of the other configurations are evaluated.

using a transmitter that is a single C4 wire.

Fig. 6 indicates that, for short measurement periods, relative oscillator counts are noisy, due to the use of a small number of samples (an effect also seen for Xilinx devices [3]). As the measurement period extends towards 21 ms (the period used for other experiments in this section), the results become more stable due to an averaging of noise across many oscillations and because the integer counts collected from the oscillator become relatively less granular when the count values are higher.

Finally, we consider relative count differences when multiple transmitters are used with a single receiver. The transmission configurations and resulting relative counts are shown in Table I. It can be seen in the first two rows that the impact to ΔRC is roughly the same for either neighbor. The third row shows that using both neighbors as logic 1 transmitters roughly doubles the impact to ΔRC. Non-immediate neighbors appear to have little effect, as shown by the fourth row of the table.

The final row of the table is the baseline configuration against which all the other configurations are evaluated.

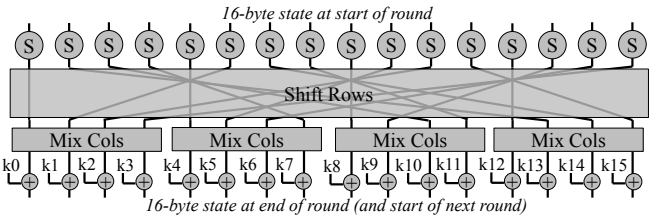
IV. Description of AES Attack

AES is a symmetric-key encryption algorithm that is widely used in electronic circuits. The algorithm uses a number of iterated rounds to transform blocks of input plaintext into blocks of output ciphertext based on the encryption key. The algorithm has variants for 128, 192, and 256-bit key lengths, which all make use of the same basic round function but differ in the number of iterations through the round. We will focus in this work on AES-128 in particular, which uses 10 rounds. Each round uses in its transformation a 128-bit round key; the 10 different 128-bit round keys are computed from the encryption key using an iterated key scheduling computation that runs alongside the iterations of the round.

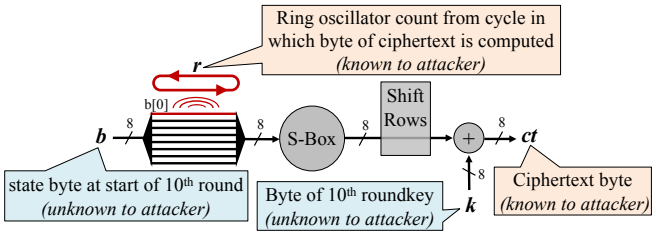
The data transformation operations performed in each round of AES are bytewise substitutions on each byte according to a known substitution table (S-Box), Shift Rows operation that reorders the bytes, Mix Columns operation that performs a modular multiplication with an irreducible polynomial, and addition of the round keys using bitwise XOR (see Subfig. 7a). All 10 rounds of AES-128 are identical, except that the final round omits the Mix Columns operation. There is also an additional key addition that is performed as a pre-processing step before the first round.

A. Extracting AES Key

The side channel analysis that we use to attack AES is inspired by Differential Power Analysis [6], [13]. In DPA, and our own technique, an attacker measures a side channel, and uses the side channel measurements to confirm or refute specific guesses on the value of key bytes. The attack is



(a) Generic AES round



(b) Portion of last round used in attacking one key byte

Fig. 7: (a) shows a single round of AES. (b) Bottom shows the portion of final round circuit that is used to attack one key byte. The final round of AES omits mix columns and has no interaction between the 16 different bytes being processed.

powerful because it can, given enough data, extract keys from extremely small correlations that exist between the side channel measurements and data values in the algorithm which depend on the secret key. To give a sense of the power of DPA, in classical DPA, where the side channel measurement is the power consumption of the entire chip, the data dependency exploited in the attack can be as small as the key-dependent charging or discharging of a single logic node in the computation. In comparison to DPA, the frequency effects we exploit in our attacks are rather large.

The attack as described here extracts a single byte of the round key in the final round of 128-bit AES by using a ring oscillator for which the measured counts are correlated to the value of a specific wire in the design. The ring oscillator serves as the receiver of the leakage, and the wire that is the source of the leakage is denoted here as the victim wire. All bytes of the round key are recovered in the same way. Once all bytes of the final round key are recovered through the side channel attack, then the original AES encryption key can be calculated from the round key by inverting the key schedule.

The relevant portion of the final round circuit for attacking a key byte, using information leaked from a single wire, is shown in Subfig. 7b. Recall that the final round of the AES algorithm performs bytewise substitution (S-Box), shift rows, and key addition using XOR, but it omits the mix columns operation. The output of the final round is the ciphertext, which is public information. To set up the attack scenario for recovering a key byte, the attacker chooses as the victim any bit of S-Box input that is routed on a long wire (C4); in Subfig. 7b, bit 0 of the S-Box input is chosen as the victim. The ring oscillator is then routed next to this signal so that its oscillation count in each clock cycle will depend slightly on the value of the S-Box input bit.

Using the ring oscillator as a sensor, the attacker monitors many encryptions to collect information for the side channel attack. For each of n encryptions performed, the attacker records the ciphertext byte and the ring oscillator count during the cycle the ciphertext byte was produced; we denote these two quantities as ct_i and r_i respectively for the i^{th} encryption. After n encryptions, the attacker has a collection of measured oscillator count and ciphertext pairings $(r_0, ct_0), (r_1, ct_1), \ldots, (r_{n-1}, ct_{n-1})$. The attacker will consider all 256 possible values for each key byte and use the collection of measurements to confirm one of the 256 guesses as being the correct key byte value which is used in the circuit.

To confirm one of the key byte guesses as correct, the attacker considers all 256 possible values to find the one that is consistent with the side channel measurements. For each key guess k_j (i.e. $k_0 \ldots k_{255}$), the attacker computes an S-Box input value $b_{i,j}$ for each of the $i \in [0, n-1]$ measurements using Eq. 2 to invert the circuit's round key addition and S-Box computation. [2]

$$b_{i,j} = S^{-1}\left(ct_i \oplus k_j\right) \tag{2}$$

[2]Note that shift rows is not considered in Eq. 2 when inverting the round function as it only reorders the bytes.

By inverting the S-Box function under key guess k_j, the attacker now knows what S-Box input value would have induced ciphertext ct_i if the key byte was in fact k_j. For key guess k_j, the computed values at the S-Box input in the n encryptions would be denoted $b_{0,j}, b_{1,j}, \ldots, b_{n-1,j}$. The predicted S-Box inputs each contain a specific prediction on the value of the victim wire (bit 0 of the S-Box input), and we check for its effect on the oscillator counts to know whether k_j is the correct key byte value. The attacker next partitions the n measurements into two subsets according to whether the victim wire would have a 0 or 1 value under the key guess k_j – one subset contains all the measured RO counts (r_i) for encryptions when the victim would have a 1 value, and the other subset contains all the measured RO counts when the victim would have a 0 value. The attacker then uses the average RO counts of the two subsets to confirm or refute his guess that k_j is the key byte value as follows:

- If the key byte is in reality k_j, then partitioning according to key guess k_j is accurately partitioning the data based on whether the victim is 0 or 1. The average RO count will tend to be higher in the subset of encryptions that predict a 1-value for the victim wire, and lower in the subset of encryptions that predict a 0-value. Observing a sufficient difference between the average RO counts in the two subsets confirms that the partition is meaningful, and thus supports the hypothesis that the correct key byte value is k_j.
- If the key byte is not in reality k_j, then partitioning according to key guess k_j is arbitrary and not correlated to the computation of the circuit. Because the partition is arbitrary, each subset will contain a similar proportion of RO counts taken when the victim wire is 0 and 1. In this case, the average RO count from each subset will be similar, and the difference between the average RO counts of the two sets will approach 0 with enough data. Observing no difference between the average RO counts of the two subsets therefore serves to refute the hypothesis that the key byte value is k_j.

Fig. 8 shows graphically how a collection of RO counts can confirm or refute a key guess. The attacker in this case collects 500 RO counts and corresponding ciphertexts; the RO counts for the 500 measurements are shown in the top plot of Fig. 8. The middle plot shows which of the counts are predicted, according to the correct key guess, to occur when the victim wire is 1 and 0. We can see that, in measurements when the key guess predicts the victim wire to have a 1 value, the RO counts tend to be higher. The significant difference in average RO counts gives an attacker confidence that the key guess is correct. The lower plot of Fig. 8 uses an incorrect key guess to predict the 1 and 0 values of the victim wire. Using this key guess there is no difference between the average RO counts, indicating to an attacker that the key guess is not the correct one. Using this approach, with enough side channel data, the attacker will be able to identify the correct key byte guesses, even when the difference between the average RO counts is quite small.

To extract all 16 bytes of the final round key, the attacker

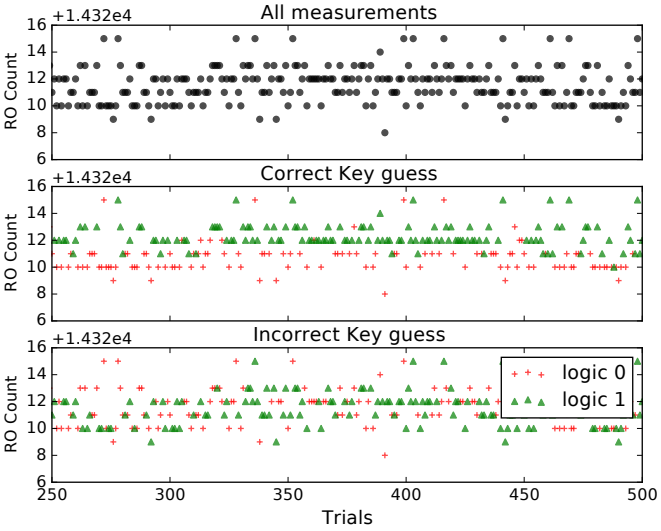


Fig. 8: RO count values partitioned into two classes based on predicted value of victim wire (10KHz clock, 2C4 long wire). Average count difference between classes is higher in the case of a correct key guess.

performs the analysis as described above on each key byte independently. In a circuit that implements a full AES round combinationally (as shown in Subfig. 7a), this requires one ring oscillator and one victim wire for each of the 16 key bytes. Once the attacker has guessed all bytes of the final round key, he can invert the key schedule and compute the encryption key. In the next subsection we show that for compact 8-bit AES datapaths, which use a single S-Box, the entire attack can be performed using a single oscillator and leakage from a single victim wire.

B. 8-bit AES Implementation

Applications that don't require high encryption throughput often implement AES with an 8-bit datapath in order to save area. Each round in an 8-bit AES implementation completes in 16 clock cycles, and all 16 substitution operations of the round (see Subfig. 7a) are computed using a single S-Box circuit operating on different data bytes in each cycle of the round. Serializing the computation to use a single physical S-Box circuit allows all key bytes to be attacked on a single victim wire at the S-Box input. Therefore, instead of measuring counts on 16 different signals (one per key byte) in the final round of encryption, the attacker can measure the oscillation counts of a single wire during the 16 different cycles (one per key byte) of the final round. To restate this for clarity and emphasis, *in the 8-bit architecture, the entire 128-bit key is recovered from a single victim wire using counts from a single well-placed ring oscillator.*

V. AES Attack Results

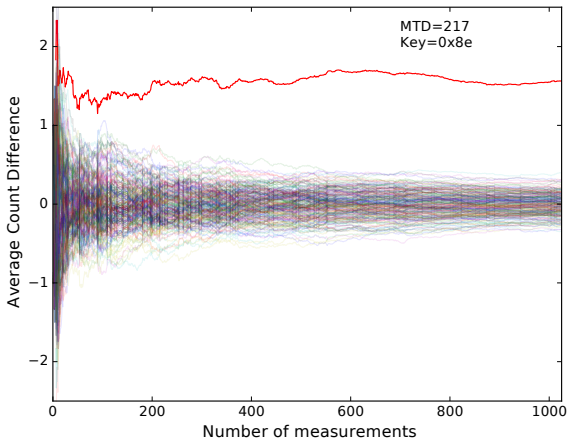
In this section we present results of attacking an 8-bit datapath implementation of AES-128. The RTL is obtained from an online source [14]. We synthesize and implement the design on Cyclone IV E and Cyclone IV GX FPGAs

using Quartus Prime. Initially, we perform a basic attack on a design where a victim wire (2 C4 long) and ring oscillator are manually placed-and-routed. We also perform the attack on an auto placed-and-routed AES design that uses a single C4 wire as a victim. In all results, we successfully extract the correct 128-bit AES encryption key with a ring oscillator that snoops on a single victim wire.

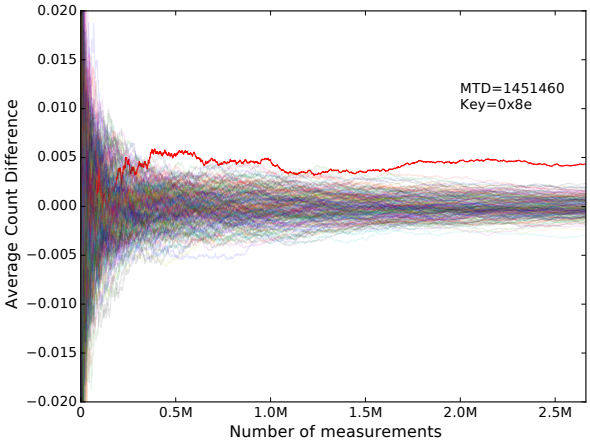
As explained in the previous section, the encryption key is obtained by first guessing all 16 bytes of the final round key, and then inverting the key schedule. When attacking a key byte, the correct guess can be distinguished from incorrect ones because it partitions the side channel measurements into two subsets that are correlated to the value of the victim wire. This results in a non-zero difference between the average ring oscillator counts of the two subsets. Incorrect guesses for the key byte lead to partitions that are uncorrelated to the victim wire value, and therefore the difference in average ring oscillator counts between these subsets approaches 0 with enough data. In many cases, it may be necessary for the attacker to collect a large dataset before the attack succeeds at distinguishing the correct key guess from the incorrect one.

Consider the plots in Fig. 9. The 256 lines in the graph show, for each of the 256 key byte guesses, the difference of the average ring oscillator counts in the two subsets partitioned according to the key guess. Due to measurement noise, it takes some number of encryptions before the correct key guess stands apart from others. We use the metric of "measurements-to-disclosure" (MTD) to quantify the number of encryptions performed before the correct guess can be distinguished. More specifically, we consider one key guess to be distinguished from others when it has the highest average ring oscillator count difference, and remains the highest for 200 encryptions.

From Fig. 9, it takes 217 and 1.5M encryptions, respectively, to extract a key byte at operating frequencies of 10kHz and 4MHz. The higher clock frequency has a smaller side channel signal and requires more encryptions for the correct key guess to stand apart. Similarly, the attack is repeated on other key bytes and the correct key guess is determined by the "peak" in the average count difference trace. Fig. 10 shows the average count difference for each of the 256 guesses for all 16 key bytes after observing 2.66M encryptions at 4MHz clock frequency; for each key byte, the highest count difference coincides with the correct key byte value. We regenerate the 128-bit encryption key from the recovered final round key to verify attack success.



(a) 10kHz



(b) 4MHz

Fig. 9: Successful attack on a key byte, with 2C4 long victim wire at two clock frequencies. Figures show the average count difference for each of the 256 guesses of a single key byte. Once enough measurements are collected, the correct key guess stands apart from all others. The attack is harder at 4MHz and requires more measurements to disclose the key.

A. MTD versus Length of Wire

As the length of the victim wire increases, so does the coupling effect (Fig. 5) leading to a larger side channel signal, making the attack easier as seen in Fig. 11. This effect is consistent across both the Cyclone IV E and Cyclone IV GX boards. As the wire length is increased, the MTD decreases from 328k for a length of one C4 long wire, to 40k for a length of 10 C4 wires.

B. MTD versus Clock Frequency

Fig. 6 shows that, although the frequency change of the ring oscillator doesn't depend on the system clock frequency, the stability of the side channel signal is diminished at higher operating frequency. One would therefore expect larger MTD values at higher clock frequencies. Fig. 12 shows the observed increase in MTD with clock frequency on Cyclone IV E and Cyclone IV GX boards. The attack remains successful up to the highest frequencies we've tested, which is 10MHz. There is no fundamental limit to the clock frequencies that can be attacked, and higher clock frequencies can similarly be attacked given enough measurements or a more sophisticated measurement circuit instead of a simple ring oscillator.

C. Attack on an auto-placed design

Apart from our experiments with a manually-placed victim wire, we also try our attack on the 8-bit AES design automatically placed-and-routed using Quartus Prime. For the attack, we identify a vulnerable signal routed through a C4 element and manually place-and-route a wire in the ring oscillator in

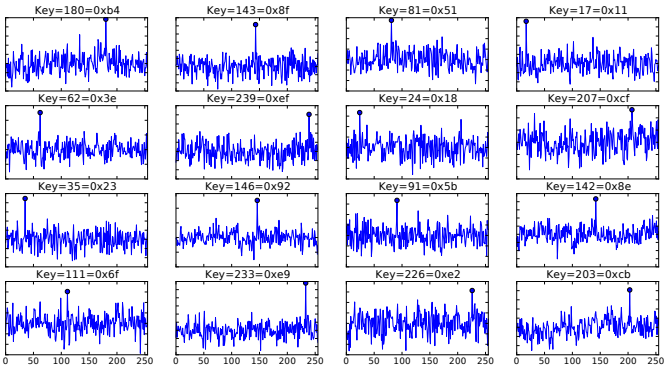


Fig. 10: Trace shows the bin difference for all 256 key guesses for each of the 16 bytes in the final round key. The correct key guess in each byte has the largest average count difference, which allows the attacker to extract the key using the side channel. Design clocked at 4MHz, target wire is two C4 long.

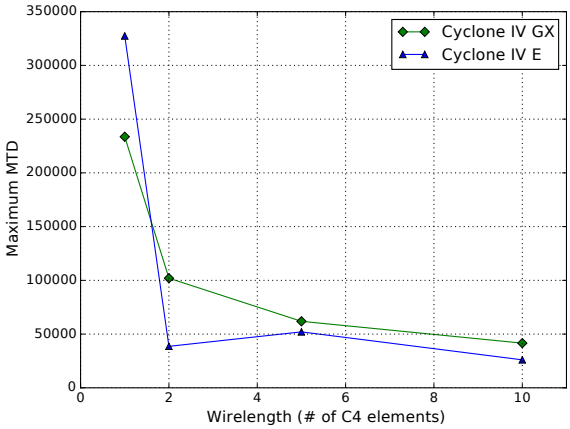


Fig. 11: MTD to break a 1MHz design when the target wire has a length of 1, 2, 5, and 10 C4 long wires. The coupling effect is stronger when the target wire is longer, and this reduces the MTD on both Cyclone IV E and Cyclone IV GX.

an adjacent C4 routing element. After the automatic place and route, no element of the design is modified except for the routing of the ring oscillator that we use to snoop on the victim signal. We are able to successfully perform our attack in this auto-placed design running at 1MHz with an MTD of 233k encryptions on the Cyclone IV GX device. From Fig. 12, the auto-placed design follows a similar trend as the manually-placed design which has a two C4 long victim wire. With more measurements, the auto-placed design can be attacked at higher clock speeds.

VI. Conclusions and Future Work

The recent discovery of a new attack vector using neighboring long wires in Xilinx SRAM FPGAs has exposed a threat to FPGAs that contain subcircuits created and used by different users. In this paper, we show that the long wire covert channel is also present in a collection of Intel SRAM FPGA families, including the Stratix V family used in Microsoft Catapult

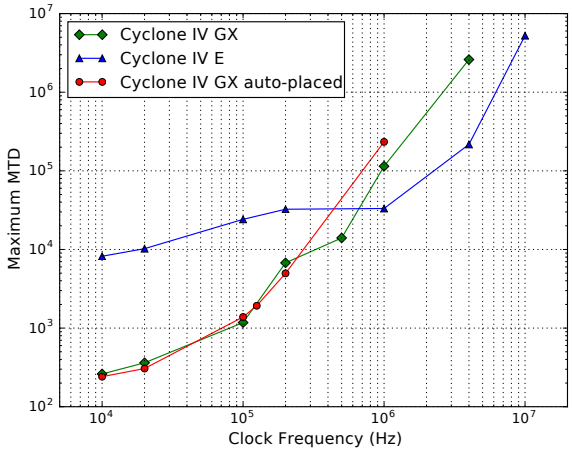


Fig. 12: MTD increases with clock frequency in all designs due to a smaller signal in the side channel. 10MHz was the maximum frequency tested but with more measurements higher frequencies can be attacked.

servers. Information leaked through the channel enables a side channel attack to extract the key from an AES circuit that has been auto-placed and routed in an Intel FPGA. This attack is performed remotely with no need for physical access to the device, a scenario similar to FPGA-based data center use. In the future, we plan to assess attacks on designs that operate at higher clock frequencies, determine techniques to locate exposed long wire signals in victim designs, and develop countermeasures to prevent these attacks.[3]

References

[1] A. Putnam et al., "A reconfigurable fabric for accelerating large-scale datacenter services," in *ISCA*, Jun. 2014, pp. 13–24.
[2] "Amazon F1 web site," https://aws.amazon.com/ec2/instance-types/f1/.
[3] I. Giechaskiel, K. B. Rassmussen, and K. Eguro, "A robust covert channel on FPGAs based on long wire delays," *CoRR*, vol. abs/1611.08882v2, 2017. [Online]. Available: http://arxiv.org/abs/1611.08882v2
[4] ——, "Leaky wires: Information leakage and covert communication between FPGA long wires," in *AsiaCCS*, Jun. 2018.
[5] A. Caulfield et al., "Configurable clouds," *IEEE Micro*, vol. 37, no. 3, pp. 52–61, 2017.
[6] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology, CRYPTO*, Aug. 1999, pp. 789–789.
[7] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *CHES*, Sep. 2003, pp. 29–45.
[8] F. Schellenberg et al., "An inside job: Remote power analysis attacks on FPGAs," in *DATE*, Mar. 2018.
[9] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *IEEE Symp. Security and Privacy*, May 2018, pp. 805–820.
[10] E. Boemo and S. Lopez-Buedo, "Thermal monitoring on FPGAs using ring-oscillators," in *FPL*, Aug. 1997, pp. 69–78.
[11] T. Iakymchuk, M. Nikodem, and K. Kepa, "Temperature-based covert channel in FPGA systems," in *ReCoSoC*, Jun. 2011, pp. 1–7.
[12] S. J. E. Wilton, "A crosstalk-aware timing-driven router for FPGAs," in *FPGA*, Feb. 2001, pp. 21–28.
[13] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 5–27, Apr. 2011.
[14] "8bit datapath hardware implementation of AES," https://github.com/ChengluJin/8bit_datapath_AES.

[3]This research was funded by NSF/SRC grant CNS-1619558 and a grant from Intel's Corporate Research Council.