CROSSTALK BASED SIDE CHANNEL ATTACKS IN FPGAs

Chethan Ramesh

Follow this and additional works at: https://scholarworks.umass.edu/masters_theses_2

Part of the Electrical and Computer Engineering Commons

Recommended Citation
Ramesh, Chethan, "CROSSTALK BASED SIDE CHANNEL ATTACKS IN FPGAs" (2020). Masters Theses. 886.
https://scholarworks.umass.edu/masters_theses_2/886

This Open Access Thesis is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Masters Theses by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
CROSSTALK BASED SIDE CHANNEL ATTACKS IN FPGAs

A Thesis Presented

by

CHETHAN RAMESH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

February 2020

Electrical and Computer Engineering
CROSSTALK BASED SIDE CHANNEL ATTACKS IN FPGAs

A Thesis Presented
by
CHETHAN RAMESH

Approved as to style and content by:

_________________________________
Russell Tessier, Co-chair

_________________________________
Daniel Holcomb, Co-chair

_________________________________
Jay Taneja, Member

_________________________________
Christopher V. Hollot, Department Chair
Electrical and Computer Engineering
ACKNOWLEDGMENTS

I would like to thank Prof. Russell Tessier and Prof. Daniel Holcomb for their guidance on this thesis. I would like to express my gratitude to Prof. Jay Taneja for his time and for accepting to be on my thesis committee. I would also like to thank Shivukumar Basanagouda Patil, George Provelengios, and Siva Nishok Dhanuskodi for their help while working on the project.
ABSTRACT

CROSSTALK BASED SIDE CHANNEL ATTACKS IN FPGAs

FEBRUARY 2020

CHETHAN RAMESH
B.E., R V COLLEGE OF ENGINEERING, BANGALORE
M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Russell Tessier and Professor Daniel Holcomb

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 makes two contributions that 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 50MHz. Finally, we present results from the AES attack performed using a cloud FPGA in a Microsoft Project Catapult cluster. These experiments show the effect can be used to attack a remotely-accessed cloud FPGA.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS ........................................ iv</td>
</tr>
<tr>
<td>ABSTRACT ........................................ v</td>
</tr>
<tr>
<td>LIST OF TABLES ....................................... x</td>
</tr>
<tr>
<td>LIST OF FIGURES ..................................... xi</td>
</tr>
</tbody>
</table>

CHAPTER

1. INTRODUCTION ......................................... 1
   1.1 FPGA Security ...................................... 1
   1.2 Thesis Outline .................................... 3

2. BACKGROUND .......................................... 5
   2.1 Multi-Tenant FPGAs ................................ 5
       2.1.1 Long Wire Attacks in FPGAs .................. 6
       2.1.2 Relationship to Previous FPGA Attack Approaches .... 7

3. EXPERIMENTAL APPROACH .............................. 9
   3.1 Basic Channel-Based Routing Architecture of Intel Cyclone IV FPGAs ................. 9
   3.2 CAD Flow .......................................... 10
   3.3 Decoding the Adjacency Routing Graph of Intel FPGAs ............................... 12
   3.4 Experimental Methodology ............................ 13
   3.5 Description of AES attack .......................... 15
       3.5.1 Extracting the AES key ....................... 16
       3.5.2 8-bit AES Implementation .................... 20
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 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 $\Delta RC$ values of the other configurations are evaluated.</td>
<td>26</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A multi-tenant FPGA attack scenario [6]. A single FPGA is shared by multiple users and an attacker can snoop information by using an interconnect wire that is adjacent to another user’s (User Design’s) wire.</td>
<td>2</td>
</tr>
<tr>
<td>3.1</td>
<td>Routing architecture model of an SRAM-based Intel FPGA [17, 16]. Intra-LAB routing is used to connect logic elements within a LAB.</td>
<td>10</td>
</tr>
<tr>
<td>3.2</td>
<td>Intel FPGA CAD Tool Flow.</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>Experimental setup for information leakage characterization.</td>
<td>13</td>
</tr>
<tr>
<td>3.4</td>
<td>Transmitter (victim) and receiver implemented in a Cyclone IV FPGA.</td>
<td>14</td>
</tr>
<tr>
<td>3.5</td>
<td>Generic AES round [19].</td>
<td>16</td>
</tr>
<tr>
<td>3.6</td>
<td>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 [19].</td>
<td>17</td>
</tr>
<tr>
<td>3.7</td>
<td>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 [19].</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Relative ring oscillator count difference due to the value of the adjacent transmitter (see Eq. 3.1). 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).</td>
<td>23</td>
</tr>
</tbody>
</table>
4.2 Repeating the experiment from Fig. 4.1 on the Cyclone IV E devices from three identical DE2-115 boards produces comparable results.

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

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

5.1 RO counter values at the transmitter operating frequency of 50MHz. The RO count values for logic '1' and logic '0' cannot be distinguished. This experiment is done on a Cyclone IV device and uses two C4 vertical routing elements as a transmitter.

5.2 Implementation of a basic delay-line based time-to-digital converter.

5.3 Operating principle of a time-to-digital converter. Delayed versions of the start signal are sampled on the rising edge of the stop signal, resulting in a thermometer code at the outputs of the sampling elements (flip-flops or comparators) [8].

5.4 160-bit delay line TDC constructed using buffers and D-flip flops.

5.5 Cyclone IV TDC implementation - Quartus Chip-planner view.

5.6 TDC delay line output captured with a DFF. The TDC output signal is captured in a flip flop when the count enable signal is deactivated. $T_d$ represents the delay due to a buffer element.

5.7 RO counter and TDC circuit based experimental setup.

5.8 TDC post-processing example. Note that the TDC input is shown on the top right. The transition from 0 to 1 appears in the second position from the right.

5.9 Distribution of measurements combining TDC and RO counter circuits. This experiment was conducted with a 50MHz transmitter clock frequency using one C4 long wires and two neighbors.
6.1 Successful attack on a key byte using RO sensor, 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. ................. 38

6.2 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. .................................................. 40

6.3 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. ............. 41

6.4 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. .................................................. 42

6.5 Experimental setup of 160-bit AES with RO counter and TDC measurement circuits. .................................................. 43

6.6 Average count 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. Design clocked at 50MHz, receiver wire is one C4 long with two neighboring transmitters of a replicated AES wire. ................................. 44

6.7 Successful attack on a key byte at 50MHz with RO counter and TDC circuit. .................................................. 45

7.1 Mt Granite FPGA board block diagram [14]. ......................... 48

7.2 High-level block diagram of the Mt. Granite shell [14]. ............... 49

7.3 Stratix V routing architecture [17] ................................. 51

7.4 Catapult crosstalk experimental setup. ................................. 52

7.5 Stratix V FPGA with the crosstalk design partition highlighted. The rest of the region belongs to the Shell. ................................. 53
7.6 Crosstalk attack result using a Microsoft Catapult node on TACC.
    RO counts were sampled from a wire adjacent to a victim wire. The difference in the RO count between logic '0' and logic '1' is 212.35, using 10 C4 long wires for one transmitter and one receiver.

7.7 Successful AES attack on Catapult FPGA node up to a maximum frequency of 4MHz.
CHAPTER 1
INTRODUCTION

1.1 FPGA Security

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 [20] and Amazon EC2 F1 [3] 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 [6], [7]. 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, before our work it was previously confined to Xilinx devices and was not used to perform an attack on a multi-tenant user design.

Figure 1.1 illustrates one of the attack scenarios when an attacker and victim share routing resources. The attacker can use crosstalk between wires to retrieve information carried on a victim (user design) wire. This vulnerability of multi-tenant FPGAs raises security concerns.

The work performed for this thesis project greatly expands the initial Xilinx findings in multiple dimensions. The thesis contributions are as follows:

- Verify the existence of the covert channel in Intel FPGA Cyclone IV and Stratix V families and characterize the information leakage across multiple FPGA devices and boards (Chapter 3 and Chapter 4).

- Demonstrate the use of the covert channel to extract an encryption key from an AES-128 encryption core which is auto placed and routed by the Intel Quartus FPGA design mapping tool (Chapter 6).
• Implement Time-to-Digital Converter (TDC) circuits to perform crosstalk attacks for AES circuits operating at frequencies approaching 50MHz (Chapter 5 and Chapter 6).

• Conduct covert side channel attacks on Microsoft Catapult cloud FPGA instances that are part of the Texas Advanced Computing Center (TACC) (Chapter 7).

The research work and results presented in this thesis have been published in the following conference proceedings:


1.2 Thesis Outline

The rest of this document is organized as follows: Chapter 2 describes previous related research and provides background information. Chapter 3 describes the experimental setup and tool flow of the crosstalk experiment. Chapter 4 analyzes characterization results from Cyclone-IV FPGAs. Chapter 5 describes a time-to-digital converter (TDC) circuit that can be used to detect covert values of signals transferred at frequencies approaching 50 MHz. Chapter 6 documents a side-channel attack on an AES circuit using a ring oscillator-based circuit and details the use of the new TDC measurement circuit in attacking AES circuits. Chapter 7 provides an introduction to the Microsoft Catapult Project. An FPGA-based instance of Catapult is located at the Texas Advanced Computing Center (TACC). The chapter describes
a AES crosstalk attack on one of the FPGAs in the cluster. Chapter 8 reviews the thesis work and offers directions for future work.
CHAPTER 2
BACKGROUND

2.1 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 [20] 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 [1].

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 [3]: “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.
2.1.1 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. [6] 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 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

---

1The term “crosstalk” often refers specifically to capacitive coupling between wires. In this work we adopt the terminology of Giechaskiel et al. [6] and use the word crosstalk in a more general sense to describe the unspecified interaction between neighboring wires.
be effective if a single wire of the core is routed on a vertical C4 long-line that spans four logic array blocks (LABs).

2.1.2 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 [12]. Power in side channel attacks is typically measured through a sense resistor external to the chip [12], or through electromagnetic emanations [2] 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 [5], [22].

It is possible to implement a temperature-to-frequency transducer suitable for thermal monitoring on FPGAs using a ring oscillator [4]. 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 [9] 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 [13]. 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. The root cause of information leakage due to crosstalk is not clear and the specific physical phenomena that causes it has not been publicly identified.

The work in this thesis examines an attacker’s ability to use adjacent long wire crosstalk to obtain information from a victim. We consider a range of attack implementation and test options including the use of ring oscillators attacking circuits operating at up to 10Mhz [19] and time-to-digital converters attacking circuits operating at higher frequencies. We perform attacks using ring oscillators on circuits implemented in TACC Stratix V FPGAs located thousands of miles away from the malicious FPGA user.
CHAPTER 3

EXPERIMENTAL APPROACH

This chapter describes a basic model of the routing architecture of Intel Cyclone IV and Stratix V FPGAs followed by a description of the experimental approach used to perform crosstalk-based attacks using ring oscillators. The routing architecture of the FPGA provides important details to aid understanding of the crosstalk-based attacks.

3.1 Basic Channel-Based Routing Architecture of Intel Cyclone IV FPGAs

Most logic functions in FPGAs are implemented using lookup tables (LUTs). In Intel SRAM FPGAs, a LUT is combined with a flip flop to form a logic element (LE). LEs are grouped into a cluster to form a Logic Array Block (LAB). The LEs are connected using routing wires within the LAB [16]. Connections between LABs are made using wires that span multiple LABs either horizontally (rows) or vertically (columns).

The basic routing architecture of an SRAM-based Intel FPGA, such as a Cyclone IV or Stratix V, is shown in Figure 3.1. Vertical routing elements (wires) are used to connect to LABs in the same column. Horizontal routing elements are used to connect to LABs in different columns. The number of LABs spanned by a routing element can vary within an FPGA. For a Cyclone IV FPGA, vertical routing elements span four LABs (C4 elements) and 16 LABs (C16 elements) in same column. Horizontal routing is made of R4 and R24 elements that cross 4 and 24 vertical LAB columns,
Intra-LAB routing is used to connect logic elements within a LAB.

respectively. As an example, a Cyclone IV FPGA has 96 C4 elements and 16 C16 elements per column. The R4 and C4 routing elements can be directly driven by logic elements, while the longer R24 and C16 routing elements can be only driven by R4 routing elements.

3.2 CAD Flow

The first step in our work was to better understand the detailed FPGA routing structure of a Cyclone IV device. Intel Quartus Prime Standard Edition version 17.0 software was used to compile designs for a Cyclone IV device (EP4CE115F29C7). Our initial experiments included placement and routing constraints on the attacker’s (receiver) circuit and the victim’s (transmitter) circuit. The design was specified in Verilog and SystemVerilog RTL (register transfer level) code. RAM modules were
generated using the Quartus IP generator tool (qsys) and integrated with the design RTL. Quartus starts (quartus_syn) by performing RTL analysis and synthesis on input RTL files. A Quartus Setting File (QSF) file was used to set placement constraints on the design and a Routing Constraints File (RCF) was used to define the routing constraints for the transmitter and receiver wires. The placement and routing tools in Quartus (quartus_fit) used these files to generate a design with defined placement and routing. To check if the routing constraints were satisfied, the routing results were back-annotated using the Quartus database tool (quartus_cdb) and compared with the original constraints. The Quartus Assembler tool (quartus_asm)
was used to generate an FPGA bitstream. The CAD tool flow for experiments is shown in Figure 3.2.

3.3 Decoding the Adjacency Routing Graph of Intel FPGAs

An attacker needs precise information about the routing architecture in a horizontal or vertical channel to perform a crosstalk attack involving neighboring wires. The Altera device handbook [16] provides a high-level picture of the routing elements and architecture in a Cyclone IV FPGA. A TCL package available in the Quartus tools provides additional details on the routing elements available in a LAB. However, Intel does not publish the complete channel routing architecture of its SRAM-based FPGAs. By performing routing experiments and reverse engineering the results, it was possible to reconstruct sufficient information about the routing graphs to perform experimentation. The RCF is used to pass routing constraints to the Quartus software in the place and route stage (quartus_fit).

To find the routing graph, a sample design connecting a source logic cell (LCELL), representing a LE, to a destination LCELL is used. The LCELL is manually placed using the constraints specified in a Quartus Prime Settings file (QSF). All routing elements adjacent to the LCELL are examined to find the legal fan-outs from the source LCELL. For each routing element, the RCF specifies the fan-out connections. Experiments were conducted using all routing elements available to a LAB to determine connectivity of all the routing elements. After each experiment, the results from the Quartus fit tool were analyzed and if the routing constraints were satisfied, the legal routing constraints were added to a database. If the experiment failed, the routing constraints were changed and the experiment was rerun. Since the routing elements available for each LAB are constant, all the legal constraints can be determined in a short duration of time. As a LAB’s architecture is replicated for a given FPGA
family, the routing connections found for one LAB can be used for other LABs in the FPGA.

3.4 Experimental Methodology

Figure 3.3 shows the experimental setup for our crosstalk attack experiments using a ring oscillator. The transmitter is the ‘victim’ wire which carries the information the attacker intends to steal. The receiver is made of a ring oscillator (RO) which has one NOT gate and two buffers. Both circuits are routed using one or more C4 elements that are placed next to each other. A view of a transmitter and a receiver (RO) using LABs and C4 wires is shown in Figure 3.4. A test pattern generator drives a logic ’0’ or logic ’1’ pattern onto the transmitter wire.

The 32-bit binary counter measures the RO’s frequency at the positive edges of the RO for a fixed time duration (test measurement period) set at the beginning of each experiment. The time duration of each experiment can be set to any value, in our case it varies from 10ns to 21ms. Most of our experiments are conducted for a fixed time duration of 21ms. At the end of each trial the value of the counter is stored in the counter SRAM. The FSM controls the entire operation of enabling
the transmitter for specified test duration and storing the value on the SRAM. Each experiment has a fixed sample size which defines how many trials of static logic '0' and '1' are transmitted on the victim wire. For an experiment with a sample size of 1,024, there are 1,024 trials of '0' and '1' transmitted on the transmitter with half logic '0' and other half logic '1' (512 - '0' transmitted and 512-'1's transmitted). The sample size can be set to 1,024, 2,048 and 4,096.

At the end of each experiment, 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 and
each test uses a measurement period derived from a 100 MHz system clock generated
by an on-chip phase-locked loop (PLL). After each test, the binary counter is reset.

The difference in RO frequency for two trials is evaluated by using a relative
count metric [6] determined over two measurement periods. For example, the count
difference \( \Delta RC \) of the receiver for trials when the transmitter is first 0 (first trial)
and then 1 (second trial) can be represented as:

\[
\Delta RC = \frac{C^1 - C^0}{C^1}
\]

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

### 3.5 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 make use of
the same basic round function but differ in the number of iterations through the
round. This work is focused on AES-128 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 operations that reorder the bytes,
Mix Columns operations that perform a modular multiplication with an irreducible
polynomial, and the addition of the round keys using bitwise XOR (Figure 3.5).
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 preprocessing step before the first round.

3.5.1 Extracting the AES key

The side channel analysis that we use to attack AES is inspired by Differential Power Analysis [12]. 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 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
Figure 3.6: 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 [19].

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 3.6. 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 Figure 3.6, 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 \( (r_0, ct_0), (r_1, ct_1), \ldots, (r_{n-1}, ct_{n-1}). \)
Among the 256 possible key byte values, the attacker correctly identifies the key byte used in the circuit based on side channel measurements as follows. 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. 3.2 to invert the circuit’s round key addition and S-Box computation.\(^1\)

$$b_{i,j} = S^{-1}(ct_i \oplus k_j) \quad (3.2)$$

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 $k_j$.

\(^1\)Note that shift rows is not considered in Eq. 3.2 when inverting the round function as it only reorders the bytes.
For key guess $k_j$, the computed values at the S-Box input in the $n$ encryptions would be denoted $b_{0,j}, b_{1,j}, \ldots, b_{n,j}$. 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 $k_j$. 
Figure 3.7 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 Figure 3.7. 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 Figure 3.7 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 performs the analysis as described above on each key byte independently. In a circuit that implements a full AES round combinatorially 3.6, 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.

3.5.2 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 3.6 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.
CHAPTER 4
CHARACTERIZATION RESULTS

4.1 Characterization results

In this chapter, 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 3.4.

As seen in Figure 4.1, 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 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 Figure 4.2 indicate that the results from Figure 4.1 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 Figure 4.3, relative count differ-
Figure 4.1: Relative ring oscillator count difference due to the value of the adjacent transmitter (see Eq. 3.1). 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).

...ences 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 6, a successful key extraction attack on AES can be performed using a transmitter that is a single C4 wire.
Figure 4.2: Repeating the experiment from Fig. 4.1 on the Cyclone IV E devices from three identical DE2-115 boards produces comparable results.

Figure 4.3: Relative count differences increase with the transmitter/receiver lengths (in terms of number of C4 long wires). Experiment uses a transmitter/receiver pair at the bottom left of the Cyclone IV (EP4CE115F29) FPGA.
Figure 4.4 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 [6]). 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.

![Graph showing relative frequency count with respect to measurement period](image)

**Figure 4.4**: Relative frequency count with respect to measurement period. Both the receiver and transmitter use two C4 wires.

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 4.1. It can be seen in the first two rows that the impact to $\Delta RC$ is roughly the same for either neighbor. The third row shows that using both neighbors
as logic 1 transmitters roughly doubles the impact to $\Delta RC$. 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.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>$\Delta RC$ (1e-4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 RO 0 0 0</td>
<td>0.965</td>
</tr>
<tr>
<td>0 0 0 RO 1 0 0</td>
<td>1.002</td>
</tr>
<tr>
<td>0 0 1 RO 1 0 0</td>
<td>2.000</td>
</tr>
<tr>
<td>1 1 0 RO 0 1 1</td>
<td>-0.041</td>
</tr>
<tr>
<td>0 0 0 RO 0 0 0</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 4.1: 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 $\Delta RC$ values of the other configurations are evaluated.

### 4.2 Conclusion

In this chapter, characterization results on long wire information leakage for a family of Intel SRAM FPGAs are presented. From the results we conclude that the crosstalk effect is present and detectable in Intel Cyclone IV and Stratix-V FPGA families. The experiments show that information leakage increases with an increase in the number of overlapping C4 routing elements. The ability to differentiate transmitted 0 and 1 values is more reliable at lower design frequencies.
CHAPTER 5
TIME-TO-DIGITAL CONVERTER BASED MEASUREMENT CIRCUIT

5.1 Enhancing the Measurement Circuitry

The experimental setup described in Section 3.4 includes a 32-bit binary counter connected to the ring oscillator clock. The binary counter increments at each rising edge of a signal taken from the ring oscillator. At the end of a sample period, the count is stored in SRAM memory. This setup works well when the measurement time period is greater than 1 µs, which allows enough time to capture sufficient oscillations for differentiating between transmitted zeros and ones. When the measurement time period is less than 1 µs (frequency 1 MHz or higher), quantization from only counting entire oscillations becomes increasingly problematic. For high frequencies, only a few oscillations can be counted. Ring-oscillator (RO) counts in these cases are indistinguishable for both logic '0' and logic '1' as shown in Figure 5.1. This experiment was performed using a transmitter design operating at a frequency of 50 MHz using two vertical C4 wires. This lack of resolution must be addressed to determine if the crosstalk attack can be successful for design clock frequencies of greater than 10 MHz, the apparent RO-based detection limit [19].

Since most circuits operate above 10 MHz, it is important to assess whether the attack can succeed at higher design frequencies. In this chapter, we describe how the crosstalk attack can be enhanced to work on designs with higher clock frequencies by employing a TDC-based circuit as the receiver.
Figure 5.1: RO counter values at the transmitter operating frequency of 50MHz. The RO count values for logic ‘1’ and logic ‘0’ cannot be distinguished. This experiment is done on a Cyclone IV device and uses two C4 vertical routing elements as a transmitter.

5.2 Time-to-Digital Converters

Time-to-digital converters are circuits that can be used to measure the time between two events. TDCs are widely used in systems such as medical imaging equipment, laser rangefinders, and nuclear science detectors to precisely measure delays on the order of picoseconds. A TDC circuit converts the time interval between two clock edges into a digital value by sampling the output of all delay elements at the same time. TDC subdivides the clock period into smaller intervals, improving the resolution of the measurement [8].
A delay-line based TDC can be implemented using inverters or buffers with D flip-flops (DFF) as shown in the Figure 5.2. The inverters and buffers act as delay elements. The resolution of the TDC circuit depends on the delay of each element in
the chain. Figure 5.3 shows the operation of the TDC [8]. The start signal is delayed when it propagates through the delay line which is made of buffers. When the stop signal generated by the control logic arrives, the delayed versions of the start signal are sampled in parallel using D-flip-flops. The result is a series of digital ones and zeros captured in the flip-flops which can be analyzed as a thermometer code [11]. In Figure 5.2, if the start signal propagates through a delay element before the stop signal rises, the result in the attached flip-flop is high, and if not, the flip-flop value is low. The position of the low-high transition in the DFF chain reveals how far the start signal has propagated by the arrival of the stop signal.

5.2.1 Implementation of TDC on Cyclone-IV

Figure 5.4 illustrates the 160-bit TDC used to improve the precision of our crosstalk measuring circuit. The delay elements are implemented using LE primitives which are found in the LABs of the Cyclone IV FPGA. The propagation delay through the LE in the Cyclone IV FPGA are found to vary between 66ps to 274ps based on information from the Quartus Timing Analyzer [18] using slow corner (Slow 850mv 100C model). The TDC should be wide enough that it captures at least one edge of the propagating signal. In our setup, the TDC width is fixed to 160.

An LE can operate in either normal or arithmetic mode [16]. For TDC implementation we use LE arithmetic mode to achieve precise uniform delay between the sampling DFF elements. An LE in arithmetic mode implements a 2-bit full adder and carry circuit. Figure 5.5a shows an LE implemented in arithmetic mode. The “COMBOUT” signal of the LE is connected to the input of the adjacent DFF in the LAB. The LE output is also driven to “COUT” which becomes the “CIN” of the next LE in the carry-chain. The LEs are the delay elements of the TDC and FFs capture the samples as shown in Figure 5.5b.
The RO binary counter and TDC operate together, with the counter recording the number of oscillations and TDC capturing the fractional part. The ring oscillator’s output that drives the counter also drives the delay-line input of the TDC. The oscillator signal is delayed as it passes through an LE. The stop signal for the TDC is the negative edge of the “count_enable” signal generated from an FSM. The RO counter starts when “count_enable” goes high. When the “count_enable” transitions to low, the RO-based counter stops. On the negative edge of “count_enable”, the TDC buffer states are clocked into the flip-flops, as shown in Figure 5.6. The TDC and RO binary counter modules are placed and routed manually, The ring oscillator signal arrives at both modules at about the same time.

TDC outputs are stored in SRAM and transferred to the host PC using the JTAG hub for post-processing. Figure 5.7 shows the experimental setup of the new measurement circuit. Effectively, the TDC provides additional inter-cycle precision to the ring oscillator count. Fractions of cycles can now be considered based on the delay through the 160 buffer chain.
(a) Implementation of a TDC buffer and flip-flop using an LE in arithmetic mode. “COMBOUT” drives the D input of the DFF, “COUT” drives the next LE element. The “CLK” signal is connected to stop signal.

(b) Physical implementation of the TDC carry-chain. The blue arrow indicates the carry-chain propagation.

Figure 5.5: Cyclone IV TDC implementation - Quartus Chip-planner view.
Figure 5.6: TDC delay line output captured with a DFF. The TDC output signal is captured in a flip flop when the count enable signal is deactivated. $T_d$ represents the delay due to a buffer element.

Figure 5.7: RO counter and TDC circuit based experimental setup.
Figure 5.8: TDC post-processing example. Note that the TDC input is shown on the top right. The transition from 0 to 1 appears in the second position from the right.

5.3 Processing TDC output

The RO count and TDC samples stored in SRAM are input into a Python script for post-processing. The 160-bit TDC counter captures at least two rising edges of the ring oscillator waveform. A captured TDC output is a series of '0' and '1' values. A transition of '0' to '1' or '1' to '0' represents the posedge and negedge of the signal transiting through the TDC. The time period of the ring oscillator signal by determined by taking the posedge to posedge transition, as shown in Figure 5.8. The distance between two posedges shows how many TDC stages correspond to one full RO period. This time period is used to calculate the phase of the signal by taking the position of the closest rising transition from the start of the TDC and dividing its index by the number of stages in one RO period to obtain the fractional part. This fractional part is then added to the RO count value to obtain a decimal count. This decimal output represents how many oscillations have occurred since “count_enable” has triggered.
5.4 Results from Combining Binary Counter and TDC Measurements

An experiment with the TDC circuit was performed on the Cyclone-IV board. Figure 5.9 shows the distribution of decimal RO measurements when the transmitter switching at 50 MHz has two neighbors with one C4 vertical long wire each. The introduction of the TDC circuit produces a difference of 0.00053 of a RO count between logic ‘0’ and logic ‘1’ based on the metric described in Section 5.3. Figure 5.1 shows the same experiment at 50 MHz without the TDC. Using only an RO results in indistinguishable values for logic ‘0’ and logic ‘1’. In this case, the constant RO counts are partially due to the synchronization of the RO to the start of each measurement period. In the next chapter, we show that the TDC can be applied to the AES design from Section 3.4 to allow for crosstalk attacks at higher design frequencies than an RO-only based measurement circuit.

5.5 Conclusion

In this chapter, an RO measurement circuit that combines a 160-bit delay line time-to-digital converter (TDC) with a binary counter is described and implemented on a Cyclone IV FPGA. The TDC circuit improves the precision of the RO binary counter, allowing a crosstalk attack to be performed at higher frequencies than an RO-only based circuit.
Figure 5.9: Distribution of measurements combining TDC and RO counter circuits. This experiment was conducted with a 50MHz transmitter clock frequency using one C4 long wires and two neighbors.
CHAPTER 6
AES ATTACK RESULTS

6.1 AES side channel attack results

In this chapter we present results of attacking an 8-bit datapath implementation of AES-128. The RTL is obtained from an online source [10]. We synthesize and implement the design on 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. Some experiments are then enhanced using the TDC-based measurement circuit described in the previous chapter.

As explained in the Section 3.5, 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.
Figure 6.1: Successful attack on a key byte using RO sensor, 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.
Consider the plots in Fig. 6.1. 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.

Figures 6.1a and 6.1b show that 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. Figure 6.2 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.

6.1.1 MTD versus Length of Wire

Figure 6.3 shows the variation of the MTD with the increase in the length of the victim wire. As the length of the victim wire increases, so does the coupling effect leading to lower values of MTD. This effect is consistent across cyclone IV FPGAs. 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.

6.1.2 MTD versus Clock Frequency

Figure 4.4 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
Figure 6.2: 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.

diminished at higher operating frequency. One would, therefore, expect larger MTD values at higher clock frequencies. Figure 6.4 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.

6.1.3 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 17.1 software. 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 an adjacent
Figure 6.3: 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.

C4 routing element. The transmitter and receiver have one C4 overlapping. 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 Figure 6.4, the auto-placed design follows a similar trend as the manually-placed design which has a two C4 long victim wire with two transmitter and one receiver. With more measurements, the auto-placed design can be attacked at higher clock speeds.
Figure 6.4: 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.

6.2 Improving the AES attack with TDC

We described an AES attack using a ring oscillator-triggered binary counter in Section 3.5. The results of the AES attack using the RO binary counter were described in Section 6.1. Using the RO-based counter, the AES attack was performed successfully for a maximum design frequency of 10MHz with a measurements-to-disclosure (MTD) value of 10 million samples, as shown in Figure 6.4. Collecting 10 million data samples takes more than 24 hours. In the remainder of this chapter, we present a technique to use an RO-based binary counter along with a TDC circuit to im-
Figure 6.5: Experimental setup of 160-bit AES with RO counter and TDC measurement circuits.

prove measurement precision and reduce the number of samples required to perform a successful AES attack.

6.2.1 Experimental setup for TDC attack on AES

Figure 6.5 shows the experimental setup for attacking an AES encryption core using a 32-bit RO counter and 160-bit TDC circuit. The AES encryption circuit described in Section 3.5.2 is used. The experimental setup is similar to the setup described in Section 3.5 except that it includes 160-bit TDC circuitry. The design was made of one vertical C4 routing element with two transmitters and one receiver. In other words, the single snooped bit of the AES circuit is replicated to amplify the effect on the receiver RO. The experimental setup runs on a 100MHz system clock generated by an on-chip phase-locked loop (PLL) circuit. We synthesized and implemented the design on a Cyclone IV FPGA using Quartus Prime Edition 17.0 software. The victim wire from the 8-bit AES datapath is manually placed and routed
adjacent to the ring oscillator wire. The design includes a number of SRAM memory instances which store experiment data such as plain text, AES key, ciphertext, RO count, and TDC measurement.

6.2.2 Results of the AES attack using TDC circuit

Figure 6.6 shows the count difference for all 256 key guesses for each of the 16 bytes. Using the TDC circuit, a successful AES attack is achieved at a design frequency of 50MHz, with 15 out of 16 key bytes breaking within 20K samples. The failure of key byte 0 needs further investigation. Figure 6.7 shows the average count difference for each of the 256 guesses of a single key byte in the 50MHz attack using an RO counter
Figure 6.7: Successful attack on a key byte at 50MHz with RO counter and TDC circuit.

... and TDC. For comparison, with just the RO binary counter and no TDC, the key was unbreakable at a design frequency of 50 MHz even when 25 million samples were used. The experimental run time for collecting the 25 million samples was over 48 hours. With the help of the TDC circuit, 15 out of 16 key bytes are broken within 2 hours.

### 6.3 Conclusion

In this chapter, the results of AES attacks using crosstalk were presented. The attack was conducted at several design clock frequencies. It was shown that, at frequencies approaching 50 MHz, significant sample counts are required to extract an AES key using a crosstalk attack using only an RO counter. The maximum design
frequency at which the AES attack could be performed was increased from 10MHz to 50MHz when a TDC was used to enhance the RO. We also showed that the number of measurements required to perform an attack at a frequency of 50MHz is significantly reduced by using a TDC circuit in conjunction with an RO-based counter, versus an RO-only approach.
CHAPTER 7

EXPERIMENTATION WITH MICROSOFT CATAPULT

7.1 Project Catapult

Catapult is a cloud-based compute framework developed by Microsoft [20]. To assist academic researchers, a Catapult cluster located at the Texas Advanced Computing Center [21] in Austin, TX has been made remotely available for cloud-based experimentation. The deployment can be used to investigate the use of FPGAs as data center accelerators to improve performance, reduce power consumption, and open new research avenues of investigation, particularly for machine learning. Catapult is currently deployed in production Microsoft datacenters and has been used to accelerate the Bing search engine [20].

In this chapter, we explore the implementation of a crosstalk attack on a Stratix V based Catapult node. The crosstalk experimental setup is migrated to Catapult hardware to determine if a crosstalk attack can be performed on a remote cloud FPGA.

7.2 Catapult Architecture

Mt. Granite is an FPGA-based card designed for use in Catapult nodes [14]. A Mt. Granite board contains an Intel Xeon-based processor and an Intel Stratix V D5 FPGA (5SGSMD5H2F35). These FPGAs are optimized for DSP-centric applications found in military, broadcast, and high-performance computing markets. This circuitry interfaces to two DDR3 channels, each of 4GB memory, that are accessible through the FPGA (Figure 7.1). An FPGA communicates to its CPU via a PCIe
Gen3 x8 connection, providing 8GB/s maximum bandwidth. Each FPGA can read and write data stored on its host node using this connection. The slot-based architecture allows software running on the host to send and receive data from/to the FPGA at latencies on the order of microseconds. The host machine can be accessed remotely.

The PCIe register space is divided into three categories:

- **Internal PCIe registers**: These registers are used by the low-level FPGA driver stack and are required for FPGA-to-host DMA transfers and synchronization.

- **Shell registers**: These registers are used to communicate status information that is specific to the shell, such as statistics and health monitoring.
Figure 7.2: High-level block diagram of the Mt. Granite shell [14].

- **Soft registers**: The soft registers allow for communication between software and user application logic residing in the FPGA. This interface is typically used during run-time initialization or to query the FPGA for status or statistics.

### 7.3 Catapult Shell

The FPGA fabric on the Mt. Granite Stratix V is divided into two areas, the Shell and the Role (Figure 7.2). The Shell provides a hardware abstraction layer for FPGA access from the server. It isolates application writers from the low-level
details of the platform, board, and components through standardized and easy-to-use interfaces. The Shell is used by the cloud provider to instantiate hardware modules such as PCIe, JTAG and other peripheral IPs and is the top-level module that directly interfaces to the FPGA’s pins [14]. The Role is the FPGA real-estate that is used by the user and is part of the Shell. During compilation, the entire Shell, along with the user design defined in the Role, is compiled. Any changes to any part of the design require the Role to be recompiled. Each Mt Granite board has four bi-directional, point-to-point SerialLite III (SL3) lanes, enabling Mt Granite boards to directly communicate with other Mt Granite boards in a two-dimensional 6x8 torus network [14]. Config-Flash (RSU) is used to program and reconfigure the FPGA. The board includes configuration ROM scrubbing logic, Single Event Upset (SEU) to help mitigate the effects from random particle strikes that might otherwise slowly corrupt the FPGAs programming.

7.4 Crosstalk Attack on Catapult Cloud FPGA

7.4.1 Stratix V routing architecture

The routing architecture of the Stratix V differs from the routing architecture of the Cyclone IV (Section 3.1). A Stratix V device has vertical routing elements (C4 and C14) which span four and 14 LABs, respectively, in the same column (Figure 7.3). Horizontal routing is made of R3, R6, and R24 elements which can cross 3, 6, and 24 vertical LAB columns. A Stratix V FPGA has 144 C4 elements per column. LAB can connect to the vertical routing channels on either side (left and right). In Cyclone IV devices, LABs can only connect to the C4 column on the right. The increased accessibility of a LAB to C4 vertical routing elements on both sides (left and right) increases the complexity of routing graph extraction. The per-channel routing element adjacency leading to a routing graph was determined using the methodology described in Section 3.3.
7.4.2 Crosstalk experiment setup implementation on Catapult

The crosstalk experimental setup described in Section 3.4 was migrated to the Catapult environment with a few modifications. The crosstalk experimental setup is instantiated inside the Role of the Catapult Shell, as shown in Figure 7.4. A soft register (Section 7.2) is instantiated as a part of the Role. The register is used to set design parameters such as the test time duration, the experiment sample size and the transmitter pattern to the configuration SRAM registers. The design clock of 100MHz is derived from the Shell’s core clock of 175MHz that is generated by the Shell core clock generator. The transmitter is manually placed and routed next to
the ring oscillator using place and route constraints. A routing graph extracted using the workflow described in Section 3.4 is used to determine channel wire adjacency.

The setup is compiled using Intel Quartus Prime Edition 18.1 software. Every design change requires the re-compilation of the Shell and the Role, increasing test and debug time. On average, a Quartus compile can take up to 40 minutes. This time length impacts the number of builds and the design parameter space that can be explored in a given time.

The crosstalk design region contains a ring oscillator and the victim wire. These components are manually placed and routed using compile constraints to allow for isolation from other portions of the Shell. Otherwise, Shell placement and routing can interfere with the evaluation of the ring oscillator wire and the victim wire causing a routing failure during compilation and the collection of incorrect results at run time.
Figure 7.5 shows that the implementation isolates the crosstalk design from the rest of the system.

At run-time, the PCIe interface is used to write to the configuration SRAM memory through soft registers. The soft registers are accessed through the PCIe bus, as described in the Catapult User Guide [14]. Once the experimental parameters are configured, the experiment is run. The ring counter values corresponding to the transmission of logic '1' and logic '0' on the victim wire are stored in the counter SRAM memory. Upon experiment completion, the counter values are transferred to the host PC through the PCIe interface and JTAG interface for post-processing.
Figure 7.6: Crosstalk attack result using a Microsoft Catapult node on TACC. RO counts were sampled from a wire adjacent to a victim wire. The difference in the RO count between logic ‘0’ and logic ‘1’ is 212.35, using 10 C4 long wires for one transmitter and one receiver.

### 7.5 Crosstalk Attack Results

The experiment described in the Section 7.4.2 was conducted on a Mt. Granite FPGA node located at TACC. The results from the crosstalk experiment are shown in Figure 7.6. The experimental setup was clocked at 100MHz. A total of 1,024 trials of logic ‘1’ and logic ‘0’ were each performed for a time duration of 21ms. As shown in the figure, the average difference in the RO count was 212.35 count values. The experiment included an overlap of 10 C4 long wires for one transmitter and one receiver.

A similar experiment was conducted in the lab on a Stratix V GX FPGA Development Kit (5SGXEA7K2F40C2N). The FPGA speed grade of the in-lab Stratix V FPGA was ‘2’, which is faster than the Catapult’s Stratix-V FPGA which is ‘3’. The crosstalk experiment was conducted and the count difference between logic ‘0’ and logic ‘1’ on this board was found to be 121.03 [15].
7.6 AES Attack Results

The AES encryption circuit described in Section 3.5.2 was integrated with the setup described in the Section 7.4. The experimental setup runs on a 100MHz system clock generated by an on-chip PLL circuit. One of the victim wires from the 8-bit AES datapath was manually placed and routed adjacent to the ring oscillator wire. The two transmitters and one receiver wire are made of 10 vertical C4 routing elements. We synthesized and implemented the design on a Cyclone IV FPGA using Quartus Prime Version 18.1 software. Figure 7.7 compares the MTD at increasing clock frequencies for the implemented AES attack on Catapult FPGA. All key bytes were extracted up to a maximum clock frequency of 4MHz with an MTD of 1.21 million samples.

7.7 Conclusion

In this chapter, we introduced the Microsoft Catapult prototyping environment available to researchers at the Texas Advanced Computing Center (TACC). The architecture of the Catapult Mt. Granite board and the major components of the Catapult system were described. The crosstalk attack experimental setup for Catapult was also described. Our experiment successfully showed that crosstalk could be used to deduce whether a transmitter sends a logic ‘0’ or ‘1’. We also showed that all the keys of the 128-bit AES circuit can be extracted at a frequency of 4MHz on a cloud FPGA setup.
Figure 7.7: Successful AES attack on Catapult FPGA node up to a maximum frequency of 4MHz.
CHAPTER 8
CONCLUSION

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 thesis work, 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 servers. The information leakage was characterized at different locations in the FPGA and across various FPGA families for varying lengths of vertical routing elements and frequencies.

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. To attack the AES circuit at higher frequencies, a delay line time-to-digital converter (TDC) circuit was introduced to address the limitation of the ring oscillator (RO) binary counter design. With a 160-bit TDC augmented with a 32-bit RO counter, we have extracted AES sub-keys for a 128-bit AES circuit at a maximum design frequency of 50 MHz. An AES side channel attack can be performed in a few hours. This attack was also performed remotely using only a ring oscillator on a Microsoft Catapult node with no physical FPGA device access. The attack was successful for design frequencies of up to 4 MHz.

In the future, we plan to determine techniques to locate exposed long wire signals in victim designs. Currently we have focused using this attack to extract key from 128-bit AES encryption core. Going forward, we plan to apply similar methodologies to assess security risks in other encryption core and machine learning circuits. Counter
measures like shielding the victim’s wire from the attacker needs to be developed and incorporated into existing CAD tools to prevent this kind of attack. Additional testing and experimentation with design isolation technique needs to be preformed on the cloud environment to determine and prevent adversarial attacks once the multi-tenant FPGA environment is deployed on the cloud.
BIBLIOGRAPHY


[10] Jin, Chenglu. 8bit datapath hardware implementation of AES. https://github.com/ChengluJin/8bit_datapath_AES.


