Hardware-Entangled Inherently Secure Field Programmable Gate Arrays

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

in

Electrical and Computer Engineering

BURAK ERBAĞCI

B.S., ELECTRONICS ENGINEERING, SABANCI UNIVERSITY

M.S., ELECTRICAL AND ELECTRONICS ENGINEERING, EPFL

M.S., ELECTRICAL AND COMPUTER ENGINEERING, CARNEGIE MELLON UNIVERSITY

CARNEGIE MELLON UNIVERSITY

PITTSBURGH, PA

DECEMBER, 2018

Abstract

Field programmable gate arrays (FPGAs) are one of the most attractive programmable platforms because they combine the flexibility of software programmed CPUs and the performance and efficiency of custom ASICs. FPGAs enable rapid prototyping and development of complex ASICs and facilitate deployment of embedded systems with performance nearing ASICs. Consequently, FPGAs are now the workhorses behind a broad variety of applications including aerospace, supercomputing, high-speed signal processing, and cryptography. Additionally, FPGAs are a highly attractive implementation platform for secure systems due their lack of application design information at manufacture-time, which mitigates the risk if the supply chain is compromised. Cryptographic applications and security protocols can be efficiently implemented on FPGAs and they can easily be modified in the field unlike their ASIC counterparts.

The configuration bitstream is a persistent source of security vulnerability in FPGA designs. The possible compromise of configuration data by the attacker poses significant threats for deployed systems in the field. These threats include cloning the FPGA configuration for use in counterfeit/unauthorized systems, modifying the FPGA configuration to increase side-channel emissions, and adding malicious Trojan hardware into the compromised design. To ensure design protection, FPGA manufacturers have implemented bitstream encryption and authentication, flash FPGAs, and active defense mechanisms for FPGA test and support circuits. However, these security measures can be circumvented in a number of ways, which includes direct probing the key storage, side-channel attacks on the bitstream decryption logic, and attacks on the test and verification support circuits. Furthermore, as is common with hardware implementations, cryptographic systems implemented on FPGAs leak inadvertent side-channel information (i.e., power, timing, electromagnetic emissions) that can be exploited by an attacker to bypass the security of the algorithms. One highly effective and easy-to-mount side-channel attack is power analysis, which exploits the data-dependent power consumption in the hardware. Hiding and masking countermeasures seek to achieve resistance against power side-channel attacks at the expense of significant timing, area, and energy overheads; however, these techniques are vulnerable against a number of successful side-channel attacks.

In this thesis, we present a secure hardware-entangled FPGA design in order to address these FPGA security concerns. The proposed FPGA design never stores the configuration data in the clear, even at the lowest level of the hardware. We deeply hardware-entangle both the reconfigurable logic and interconnect by one-time pad encrypting the bitstream using a secret die-specific response. Physical unclonable functions (PUFs) are used in the implementation as a mechanism to generate this secret response. By leveraging our recent work in efficient PUF design with both low VLSI overheads (i.e., area, power, delay) and strong security metrics (i.e., randomness, uniqueness, reliability), we tightly integrate a PUF bit with every configuration bit. This has significant security benefits that include high resistance to probing attacks and unique per-die configuration bitstreams. Also, PUFs can be fired on-the-fly during FPGA operation for enhanced security against probing attacks. In addition to bitstream protection, the proposed FPGA fabric has resistance to power analysis attacks embedded within the reconfigurable fabric that enables side-channel secure operation. The fabric uses post-charged dynamic logic with self-timed discharge operation to ensure secure operation of user designs. Hardware-entangled secure FPGAs are a promising alternative to layering countermeasures on top of insecure conventional-off-the-shelf (COTS) FPGAs.

Acknowledgments

During my long years at CMU I had the opportunity to work and collaborate with many wonderful people and without their support and help this journey would not have been possible. I would like to express my deepest gratitude to my advisor and committee chair Prof. Ken Mai for his guidance and support from the very first moment I started my CMU journey. Not only his vast technical knowledge, but also his approachability, insightfulness, and enthusiasm make him a unique mentor and a role model who I have looked up to since I met him. My time as a graduate student at CMU has been an amazing experience and I owe it to his unique ability to inspire his students with a sense of hope and purpose that kept me motivated. When I look back, I feel accomplished and confident about my technical skills and I am extremely grateful for everything he taught me during my time as a graduate student.

I am also very grateful to my committee members Prof. James Hoe, Prof. Yusuf Leblebici, Prof. Patrick Schaumont and Dr. Mudit Bhargava for reviewing my proposal and dissertation and providing their valuable feedback. Their time and flexibility made this thesis possible. I also acknowledge the funding from DARPA (Contract #FA8650-18-1-7814) that supported this work.

I would like to specially thank to Prof. Yusuf Leblebici. I consider myself extremely fortunate to have known him since my undergraduate years. His mentorship and support have always inspired me to learn more and make me a better researcher.

Special thanks to Dr. Mudit Bhargava for his technical insight and advice that proved invaluable to my research and understanding. I was very fortunate to work with him in our research group. He is also an exceptional friend and I learned a lot from him both professionally and personally.

I have been very fortunate to learn a lot from many amazing professors at CMU. I would like to specially thank to Prof. Larry Pileggi and Prof. Andrzej Strojwas for many interesting lectures and their support.

The staff at CMU are some of the kindest and most professional people I have ever met. I am profoundly appreciative of the efforts of Judy Bandola, Nathan Snizaski and all of the administrative staff in the ECE department for handling so many concerns and making life run more smoothly for all graduate students.

I had a great pleasure working with many wonderful and smart people in our research group. I'm very grateful for their input and company that improved the quality of my work and my life outside of work. Thank you so much, Dr. Mudit Bhargava, Dr. Mark McCartney, Dr. Yu Cai, Dr. Cagla Cakir, Craig Teegarden, Nail Etkin Can Akkaya, Raymond Carley, Onur Kibar, Prashant Mohan, Rachel Dondero.

I would like to specially thank to Nail Etkin Can Akkaya for his selfless efforts in the custom design of my *flip-flops*. He helped me countless times and all the pieces he designed in all my testchips worked well (thank God). He has also been an exceptional friend who I have relied on so many times. He was generous enough to have me as a guest at his place during the last couple of months of my time in Pittsburgh. I am extremely grateful to him for everyhing he has done for me.

During my time at CMU, I have been lucky enough to have a great circle of friends. Each and every one of them made my CMU journey priceless.Yunus Emre Kesim, Dr. Engin Burak Anil, Omer Ozdemir, Ece Isenbike Ozalp, Dr. Batuhan Arisoy, Dr. Abhishek Sharma, Dr. Mohammed Omar Darwish, Sungho Kim, Dr. Mazen Soliman, Xiaoliang Li, Dr. Yigit Isbiliroglu, Meric Isgenc, Dr. Emrullah Korkmaz, Dr. Bekir Bediz, Dr. Recep Onler, among others too numerous to list here. Thank you all for your encouragement; I relished our time together and look forward to continued friendship and correspondence.

I would like to specially thank to Yunus Emre Kesim. He is a true friend who has inspired and supported me in numerous ways both professionally and personally. I had many challenges along the way and he helped me selflessly countless times. I am extremely grateful to him for everything he taught me.

During my time in Pittsburgh, I've had the fortune of crossing paths with a number of people who enriched my life: Busra Tugce Susam, Peren & Mert Hasanreisoglu, Dr. Tekin Kose, Yassin Khalifa, Firat Alemdar, Mustafa Kemal Ozkan, Cengiz Erdogan, Cody Martin. Thank you all for your friendship. I truly enjoyed our time together and look forward to continued friendship.

Special thanks to Busra Tugce Susam for being a caring and supportive friend. She also happens to be an excellent cook who fed me countless times when I was too busy and forgot to eat. I enjoyed her company and friendship so very much.

I would like to specially thank to Peren and Mert. I consider myself blessed to have known them. They both are incredible friends who have always supported me. I cannot thank them enough.

I would like to specially thank to Erdem family: Oznur, Bilge, Selcuk, and Ceyhun Erdem. They have always been my second family here in Pittsburgh and supported me the entire time. I am very grateful and very lucky to have them in my life.

I would like to specially thank to Dr. Ercument Cicek. Although our paths crossed in undergradute years, we became close friends following his move to Pittsburgh. Then he moved back to Turkey but that did not prevent us from being very good friends. Despite the time difference and thousands of miles between us, he has always been there for me both professionally and personally. I do not think I can ever thank him enough for everything he has done for me. He is a true friend and an exceptional person.

I would like to express my gratitude to Dr. Tugce Yazicigil and Dr. Eymen Kurdoglu for their support, friendship, and inspiration. I would also like to thank my close friends: Dr. Gozen Koklu, Dr. Ali Galip Bayrak, Pinar Kaynak, Emrah & Dr. Damla Tas, Dr. Anil Yuce, H. Can Yener, T. Zafer Ozcan. Thousands of miles between us did not prevent them from supporting me. I am extremely grateful.

I would like to express my deepest gratitude to my girlfriend, Dr. Kelly Dickson, for her patience and endless support. She has been the most caring and supportive girlfriend I could ever ask for. She kept me going and motivated especially when things were not going great with the testchip. I am extremely grateful to her for everthing she has done for me: sitting in the lab with me, helping with soldering, listening, motivating me, and unconditionally supporting me. I sincerely do not think I can ever thank her enough for all these.

Last and foremost, I would like to express my deepest gratitude and my endless love to my family. My parents, Derya and Abdulkadir, and my brother, Cagri, provided me with everything I needed to get to where I am today and encouraged me to chase my dreams. I owe everything I accomplished to them. Their incredible support, unconditional love, and interest in my academic career made all these possible. I cannot thank them enough for always being there for me. I wish my grandmother was still alive to see this moment. I humbly dedicate my work to my beloved family and my grandparents.

Contents

List of Figures

List of Tables

“Nihil actu secure.”

Chapter 1

Introduction

1.1 Motivation

As process technologies scale down to nanometer-regime, the long development time and high non-recurring engineering costs associated with modern integrated circuit design are becoming increasingly infeasible for all but the highest-volume parts. These factors have pushed many system designers to use programmable ICs, rather than costly, complex, custom ASICs. Field programmable gate arrays (FPGAs) are one of the most attractive programmable platforms, because they occupy middle ground between CPUs and custom ASICs, offering much of the flexibility of software programmed CPUs, but still achieving much of the performance and efficiency of custom ASICs. FPGAs enable rapid prototyping and development of complex ASICs and facilitate deployment of embedded systems with performance nearing ASICs. Consequently, FPGAs are widely used in a broad variety of applications including aerospace, supercomputing, high-speed signal processing, and cryptography [1].

The reconfigurable structure of FPGAs makes them a highly attractive implementation platform for secure systems due to their lack of application design information at manufacture-time, which mitigates risk in the event of compromise of the supply chain. Cryptographic applications and security protocols are efficiently implemented on FPGAs and are easily modified in the field unlike their ASIC counterparts. While this reconfigurability is useful for designers, it creates a new set of vulnerabilities for FPGA designs. A persistent source of security vulnerability in FPGA designs is the configuration bitstream. Although the actual design content is hidden until programming, the bitstream is required to configure an FPGA and needs to be protected against reverse-engineering. Possible compromise of configuration data by the attacker poses significant threats for deployed systems in the field. These threats include cloning the FPGA configuration for use in counterfeit/unauthorized systems, modification of the FPGA configuration to increase side-channel emissions, and adding malicious Trojan hardware into the compromised design [1, 2].

Another important FPGA security concern is the secure operation of FPGA designs running in the fabric. As is common with any other hardware implementations, cryptographic systems on FPGAs leak inadvertent side-channel information (i.e., power, timing, electromagnetic emissions etc.) that are exploited by the attacker to bypass the security of the algorithms [3, 4, 5]. One highly effective and easy-to-mount side-channel analysis (SCA) attack is power analysis, which exploits the data-dependent power consumption in the hardware [4, 6, 7].

1.2 Secured vs. Secure FPGA

FPGA security concerns are addressed at different levels of abstraction that includes software/algorithm, register-transfer level (RTL), and hardware level depending on the countermeasure. In order to ensure bitstream protection, FPGA manufacturers have implemented various countermeasures within the standard FPGA design flow at both software/algorithm and hardware levels that includes bitstream encryption [1, 2, 8, 9]. However, these security measures can be circumvented in a number of ways that includes side-channel analysis (SCA) attacks on the bitstream decryption logic [1, 2, 10]. The job of the attacker is made simpler by the fact that the encryption key consists of relatively small number of bits (e.g., 128b keys in Microsemi FPGAs [9], 256b keys and 57b device ID in Xilinx FPGAs [8]) and that the bitstream decryptor is a single, identifiable point of attack. The fundamental problem is that the design configuration

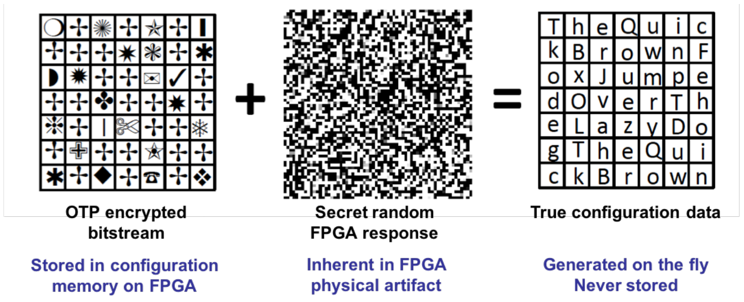


Figure 1.1: FPGA hardware entanglement concept using one-time pad encryption of the configuration bitstream using a secret die-specific response as the codebook.

data is stored in the clear at some level of hardware.

Secure operation is mainly addressed at the register-transfer-level (RTL) by layering countermeasures on an unsecured conventional-off-the-shelf (COTS) FPGA [6, 11, 12, 13, 14, 15, 16] instead of at the hardware level, since doing so requires modification on the FPGA fabric. These countermeasures achieve resistance against power SCA attacks at the expense of significant timing, area, and enery overheads [6, 11, 12, 13, 14, 15, 16]. Additionally, these techniques often require significant rework in the standard tool/programming flow since they involve low-level manipulation and relocation of logic on the FPGA fabric.

In this thesis, we present a *secure* FPGA design, wherein the hardware security features are implemented in the FPGA itself, rather than at the tool/application level. We alter the FPGA fabric itself in order to build security features. By this method, the FPGA is hardened in a virtually user-transparent manner. More specifically, we protect bitstream by hardware-entangling the configuration deep in the hardware with a secret die-specific response. This is equivalent to a one-time pad encryption of the true configuration data using the secret die-specific response as the codebook (Figure1.1). Hence, the proposed FPGA design never stores the configuration data in the clear, even at the lowest level of the hardware. Furthermore, we also address one of the most common side-channel vulnerabilities, power side-channel, by using a side-channel resistant logic embedded in the fabric. This is a circuit-level hiding countermeasure employed in the fabric to reduce side-channel information leakage. We show that the hardware-entangled secure FPGA design is more efficient for iso-security against the existing countermeasures.

1.3 Dissertation Organization

In Chapter 2, we provide background on the fundamentals of FPGA. We review the internal structure and security features of FPGAs with special attention to the major security concerns of bitstream protection and secure operation.

In Chapter 3, we discuss a novel hardware-entanglement method for FPGA bitstream protection against reverse-engineering attacks. We quantitatively evaluate area, performance, and power overheads of the hardware-entangled scheme, as well as evaluate the security benefits.

In Chapter 4, we introduce a secure FPGA fabric that employs a novel post-charged differential dynamic logic style with self-timed discharge (PCDL-STD) to achieve side-channel analysis (SCA) resistant secure operation. We compare VLSI metrics of the secure FPGA tile (i.e., area, delay, energy) against a reference unsecured static FPGA tile and the D-WDDL countermeasure layered on this unsecured FPGA. We also evaluate power SCA resistance of the secure PGA tile using the metric of normalized energy deviation (NED).

Finally, in Chapter 5, we present the implementation of the prototype testchip implementation for a proof-of-concept for our secure FPGA design. We describe the testchip details and present the measured silicon results, which includes benchmark implementation, PUF reliability analysis, power-SCA resistance evaluation.

Chapter 2

Background

In the introduction, we provided motivation and context for the thesis. In this chapter, we pave the way to FPGA security. First, we discuss the fundamentals of FPGA, cover the basic structure, and describe the programmable logic and interconnect. Then, we shift our focus to security by reviewing reverse-engineering attacks and their countermeasures. Next, we cover side-channel analysis (SCA) attacks and their countermeasures. Then, we discuss the security implications of reverse-engineering and SCA attacks to FPGA. Finally, we review the major FPGA security concerns of bitstream protection and side-channel resistant secure operation.

2.1 The Internals of an FPGA

Figure 2.1 shows the basic architecture of a typical island-style FPGA that consists of identical tiles. Each tile consists of a Configurable Logic Block (CLB) and Switch Box (SB). Each CLB contains a look-up table (LUT) that is configured to implement a logic function. The interconnect is programmed to connect CLBs together in order to implement more complicated circuits on an FPGA.

The stream of data that specifies how the LUTs and the interconnect should be programmed in order to implement a benchmark circuit is referred to as a configuration bitstream. The FPGA configuration bitstream can be stored in a number of ways. FPGAs are usually categorized with respect to their configuration storage options (i.e., SRAM-based, flash, anti-fuse.). In the following, we describe different FPGA configuration options and how an FPGA is programmed by the user.

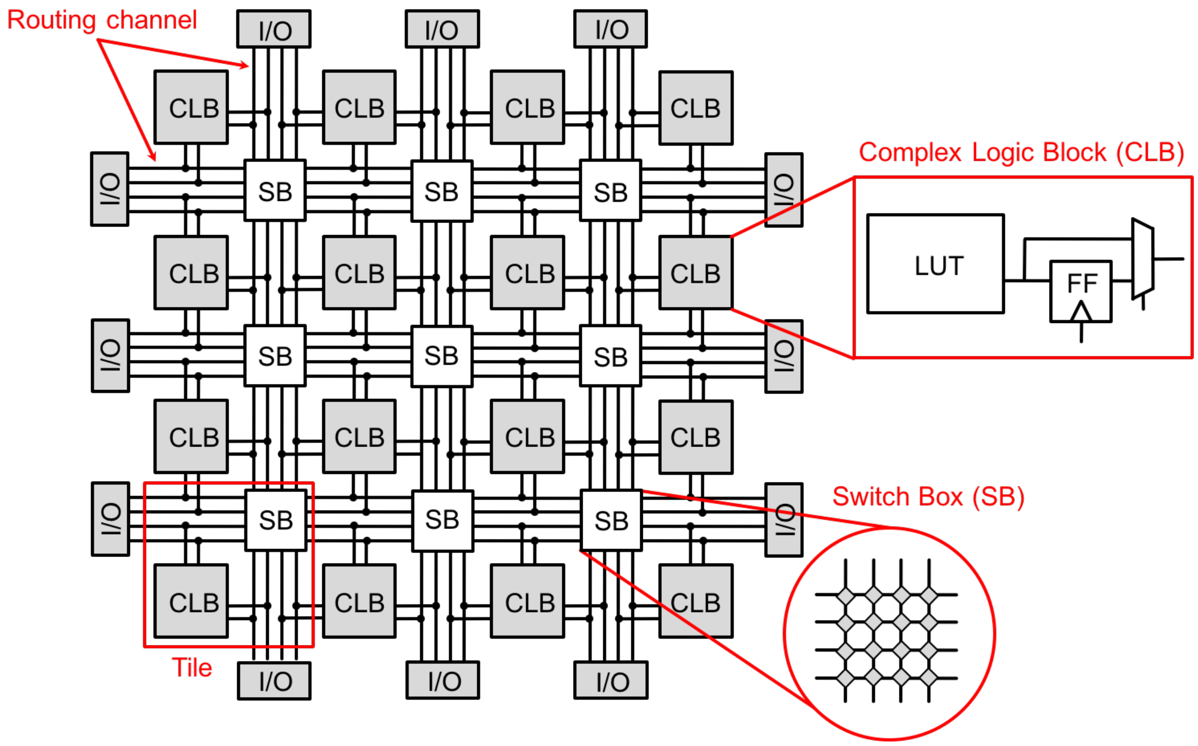


Figure 2.1: Architecture of a typical island-style FPGA. Configurable logic blocks (CLBs) are islands surrounded by routing channels. Each CLB contains a look-up table (LUT) that is configured to implement a logic function. The interconnect is also reconfigurable and it connects CLBs together to implement more complicated circuits on FPGA.

2.1.1 FPGA Configuration Storage

Table 2.1: FPGA configuration storage options

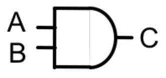
Storage Option	Re-configurable	Non-volatile	No extra process steps	No external NVM
FLASH	✓	✓	✗	✓
Antifuse	✗	✓	✗	✓
SRAM	✓	✗	✓	✗

Both the reconfigurable logic and interconnect are programmed by the user based on their desired application via the configuration bitstream. The user writes an RTL description of the application that is converted into the configuration bitstream by FPGA-vendor-supplied CAD tools, which have detailed information regarding the FPGA internal structure embedded into them. The user can also input some low-level information regarding desired performance, power, and block placement to guide the final configuration.

The configuration bitstream of the FPGA can be stored in a number ways. Table 2.1 compares FLASH, antifuse, and SRAM configuration storage options. For example, FLASH FPGAs use a non-volatile memory (NVM) to store the configuration bitstream on the FPGA die. Antifuse FPGAs are also non-volatile but they use fuses that can be set one-time, hence they cannot be reconfigured. On the other hand, most sytems use volatile SRAM cells as the configuration storage due to high costs for an integrated NVM/logic fabrication process. These SRAM-based FPGAs require a separate NVM chip on the board, typically FLASH memory, as the FPGA die itself does not contain any NVM. At system power-up, the configuration bitstream is transferred to SRAM configuration storage bits on the FPGA from the NVM to configure the reconfigurable logic and interconnect. For the remainder of the thesis, we will focus on SRAM-based FPGAs.

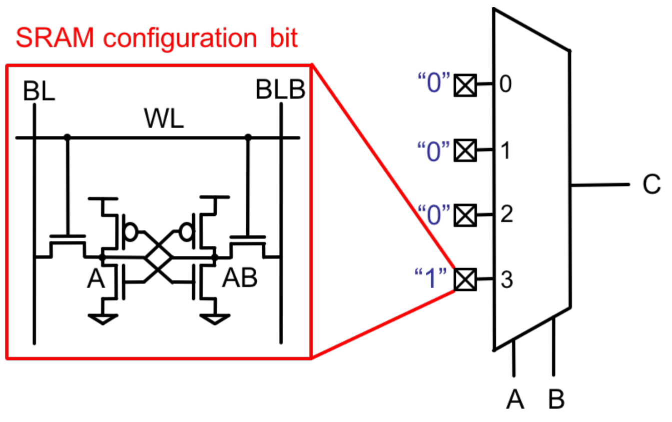
2.1.2 Reconfigurable Logic

FPGA can be configured to look like any arbitrary set of interconnected logic gates. These logic gates are implemented by using look-up tables (LUTs) on the FPGA fabric. Each LUT is very generic because it can implement any k-input function (2^{2^k} possible functions). A k-input LUT consists of an N:1 multiplexer (MUX), where N=2^k, and N configuration bits. Each configuration bit corresponds to a different output row in the truth table.



A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

(a)



(b)

Figure 2.2: (a) 2-input AND gate and its truth table. (b) 2-input LUT that is configured to implement an AND gate. 6T-SRAM cells are used to store LUT configuration bits.

Any logic gate with a corresponding truth table can be mapped into an FPGA by programming the LUTs with the appropriate truth table. For example, a two-input LUT that is configured to implement an AND gate is shown in Figure 2.2b. An AND gate outputs logic-1 only when both the inputs are logic-1 (i.e., A = 1 and B = 1), otherwise the output is logic-0 (Figure 2.2a). Therefore, the logic-1 that is stored in the configuration bit 3 is selected at the LUT output when both A and B are logic-1. For all other input combinations, the LUT output is logic-0. Any other logic function can be programmed in similar fashion.

2.1.3 Reconfigurable Interconnect

Each CLB contains a LUT that can be configured to implement any logic function. More complex circuits are implemented by connecting multiple CLBs via the programmable interconnect. An example connection between two CLBs via two programmed SBs is illustrated in Figure 2.3. Only length-1 (i.e., wires that only connect neighboring SBs) wires are shown for simplicity, but commercial FPGAs typically have routing tracks of varying lengths that connect distant SBs.

Called segmented architecture, these tracks are shown to lead to a higher density, performance, and energy efficiency [17, 18, 19, 20].

An SB is formed when a horizontal and vertical routing channel intersect as shown in Figure 2.1. It contains a matrix of switches that interconnect wires. This structure allows nets to turn corners or extend farther down the routing channel [17]. A full crossbar implementation in a switch matrix (SM) requires N^2 connections [1, 20, 21]. Despite the higher flexibility/routability, full crossbar SM increases both the SB area and the number of configuration bits to program the SB [17, 18, 22]. Sparse SMs, on the other hand, are smaller at the cost of reduced flexibility/routability. There is a trade-off between the flexibility/routability and area efficiency of SM implementation.

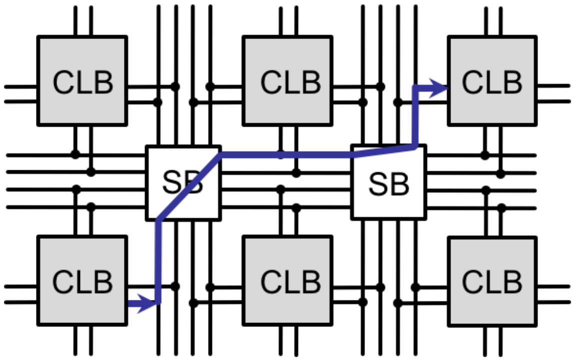
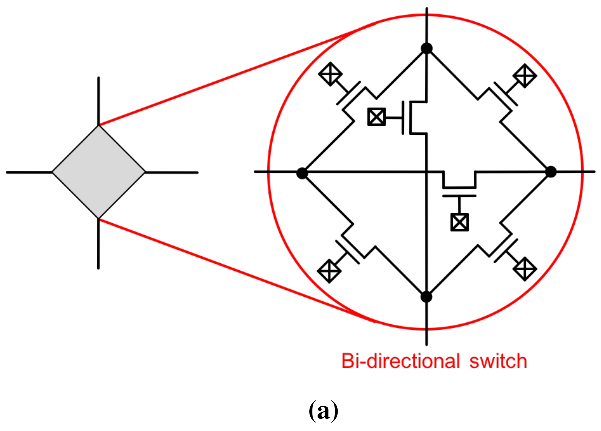


Figure 2.3: FPGA interconnect architecture. Each CLB implements a logic gate. More complex circuits are implemented by connecting multiple CLBs via the programmable interconnect. An example connection between two CLBs via two programmed SBs is highlighted in blue.

Switch design is very critical for modern FPGAs because the area is dominated by the interconnect (i.e., 80-90% FPGA area in the interconnect); the interconnect accounts for most of the FPGA's delay and energy [20, 21]. Designers explored different SM designs to have a high flexibility/routability while maximizing the area efficiency [18, 22, 23]. Switches, which influence the flexibility/routability, can either be bi-directional or uni-directional. Figure 2.4a shows a bi-directional switch that consists of tri-state drivers (i.e., pass transistors). Channel wires can be driven from both sides of the pass transistors. In contrast, Figure 2.4b shows a uni-directional switch that replaces tri-state drivers with regular drivers and the channel wires are driven in one direction. Uni-directional switches are shown to be more efficient in terms of area, performance, and energy [17].



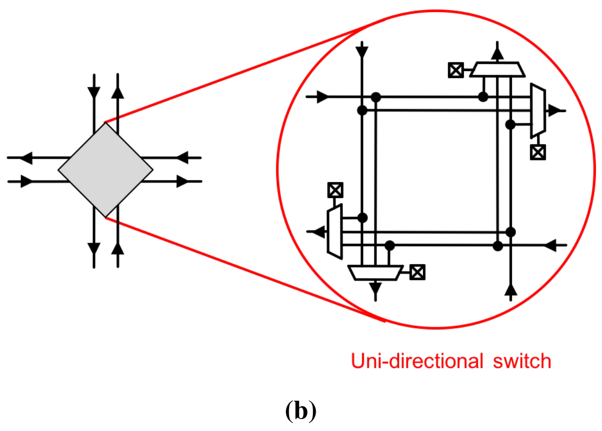


Figure 2.4: (a) A bi-directional switch that consists of tri-state drivers (i.e., pass transistors). (b) a uni-directional switch in which tri-state drivers are replaced with regular drivers. Note that two sets of wires for are shown in each direction: (1) incoming and (2) outgoing. Uni-directional switches are shown to be more efficient in terms of area, performance, and energy [17].

2.2 Reverse-Engineering Attacks and Countermeasures

2.2.1 Reverse-Engineering Attacks

A myriad of security vulnerabilites can be exposed via the reverse-engineering of the integrated circuits (ICs) contained in electronics systems. The goal of IC reverse-engineering is to uncover the functionality and internal structure (e.g., gate netlist, circuit schematic, layout, manufacturing process details) of the chip via techniques such as depackaging/delayering, high-resolution imaging, probing, and side-channel examination. With this knowledge, an attacker can mount various attacks more efficiently (e.g., fault injection, side-channel), clone/counterfeit the design possibly with hardware Trojans inserted, and discover trade secrets (e.g., proprietary algorithms, hard-coded keys and instruction sequences) [24, 25, 26, 27, 28]. A number of commercial entities, such as ChipWorks [29] and TAEUS [30], routinely reverse-engineer chips on even the most advanced process technologies.

2.2.2 Countermeasures

To combat reverse-engineering, researchers have proposed a number of countermeasures that include gate camouflaging wherein an attacker cannot discern the functionality of a particular logic gate based solely on its observable physical characteristics. One proposed camouflaging technique by Rajendran et al. uses a mix of dummy and real vias/contacts to obscure the gate function [31]. Another technique from SypherMedia [32] uses look-alike gates made with special structures requiring special layers, DRC waivers, and other low-level process manipulations. The security of these camouflaging techniques relies on the limited ability of the attacker to distinguish between real and dummy structures. However, delayering and imaging capabilities of state-of-the-art reverse-engineering are likely advanced enough to not be fooled by such structures [29, 30]. Alternatively, Erbagci et al. proposed using threshold voltage (V_{TH}) defined (TVD) camouflaged gates [33] that use different V_{TH} transistors, but with identical layouts, to determine the logic gate function. Every TVD logic gate has the same physical layout and is one-time mask programmed with different threshold implants for different boolean functions. Akkaya et. al. proposed a post-manufacturing programmable version of TVD logic (PMP-TVD) to obscure the design IP from an untrusted foundry and to combat reverse-engineering [34]. Intentional hot-carrier injection (HCI) is used to program PMP-TVD gates with different threshold voltages [34].

Other than obscuring the logical functionality at the layout level, researchers have proposed transformations on the register transfer level (RTL) hardware description language (HDL) (e.g., Verilog, VHDL) to make reverse-engineering more challenging [35, 36]. As these synthesized designs often have significant whitespace, researchers have also proposed adding unused dummy gates and interconnections [37]. These countermeasures are in an arms race against reverse-engineering tools that offer significant capabilities to descramble obfuscated designs [29, 38, 39, 40, 41, 42, 43].

Adding gates with some form of reconfigurability has also been offered as a way to conceal design intent from reverse-engineer threats. These methods typically require a memory element in order to store the configuration in working designs in the field [38, 44, 45]. Thus, even if the basic design database does not contain the design information, the deployed systems do and thus are vulnerable to reverse-engineering.

2.3 Side-Channel Analysis Attacks and Countermeasures

2.3.1 Side-Channel Analysis Attacks

Side-channel analysis (SCA) attacks target inadvertent information leakage in physical implementations rather than the weaknesses of cryptographic algorithms (Figure 2.7). One highly effective and easy-to-mount SCA attack is power analysis, which exploits the data dependent power consumption in the hardware [4, 6, 7]. Types of power analysis include simple power analysis (SPA), differential power analysis (DPA), correlation power analysis (CPA), and leakage power analysis (LPA). All of these attacks have different requirements and limitations.

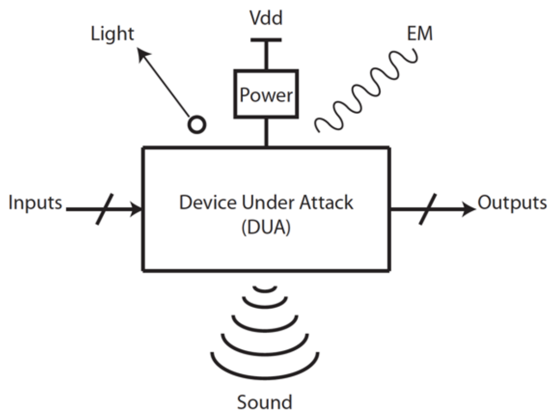


Figure 2.5: Side-channel emissions from device under attack (DUA): optical, power, electromagnetic (EM), and acoustic [46].

SPA attacks focus on the use of visual inspection techniques to identify relevant power fluctuations during cryptographic operations [4]. Hence, SPA attacks often require detailed knowledge about the implementation of the cryptographic algorithm that is executed by the device-under-attack (DUT). Alternatively, DPA attacks do not require detailed knowledge about the attacked device. However, more power traces are required by DPA attacks because they use statistical analysis and error correction techniques to extract information correlated to secret keys [4]. CPA attacks involve the evaluation of the degree of correlation between variations within the set of measurements [4]. LPA attacks exploit the dependence of leakage current of the circuits on their inputs, rather than the dynamic power [47]. Among the types of power analysis, DPA is the most prevalent and effective according to research on various crypto-systems in [4, 16, 48, 49].

2.3.2 Countermeasures

Hiding and masking countermeasures seek to achieve resistance against power SCAs at the expense of significant timing, area, and energy overheads [6, 11, 12, 13, 14, 15, 50, 51]. As illustrated in Figure 2.6, masking logic seeks to remove the correlation between the input data and side-channel emissions from intermediate values by using a random mask at the selected nodes in a specific design. However, the success of masked logic is limited because it is vulnerable against a number of successful higher order power SCAs [52, 53, 54]. Hiding technique decreases the signal-to-noise ratio (SNR) by either increasing the noise or reducing the signal. One method to reduce SNR is to make the side-channel emissions independent of the data being processed.

Various logic families achieve low side-channel leakage. One common use is dynamic differential pre-charge logic (DDPL), wherein a single switching event per clock cycle is guaranteed independent of input data [14, 50, 51, 55, 56]. For instance, wave dynamic differential logic (WDDL) uses traditional static CMOS gates to produce complementary versions of every output, and each output is pre-charged to ensure every output transitions exactly once in each clock cycle. In WDDL, a design is duplicated into a true and a complementary part by restricting the set of logic for each part to AND- and OR-gates to avoid glitches. These two parts are dual and one can be derived from the other by inverting the inputs and changing AND-gates to OR-gates, vice versa. The precharge phase of WDLL is unique such that a block in WDDL precharges without disturbing a global precharge signal to each individual gate as is normally done in DDPL [15]. Since WDDL consists of AND- and OR-gates, whenever the inputs of these gates are logic-0,

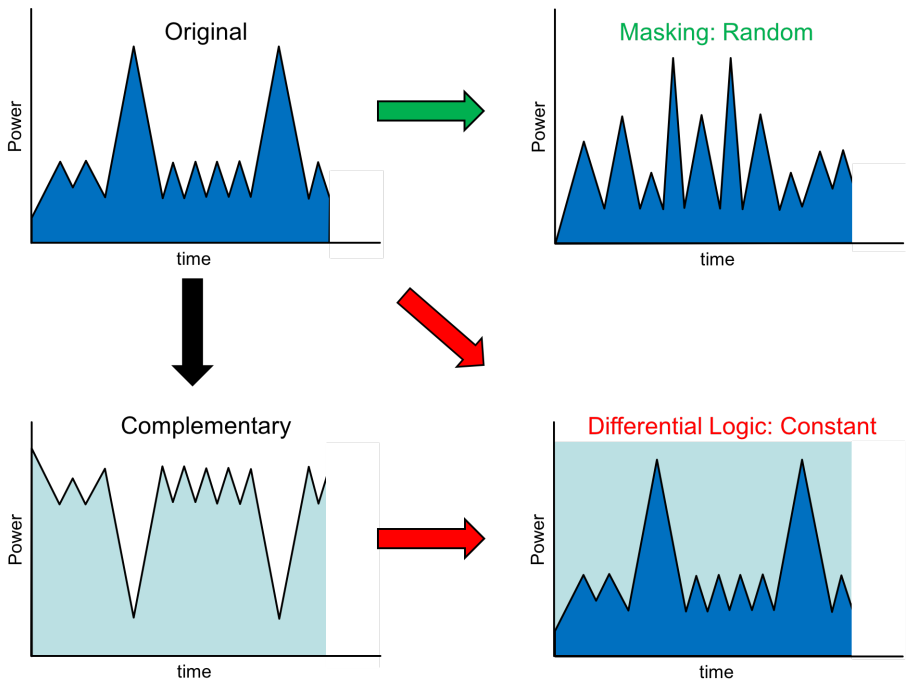


Figure 2.6: Masked logic vs. differential logic to achive power side-channel analysis resistance [16]. Masked logic seeks to remove the correlation between the input data and side-channel emissions from intermediate values by using a random mask at the selected nodes in a design. Differential logic, on the other hand, seeks to to make the side-channel emissions independent of the data being processed.

the output will be logic-0 as well. During the precharge phase, all of the inputs of a WDDL design are set to logic-0 and each individual gate will eventually have all its inputs and outputs at logic-0. This allows the precharge wave propagation throughout the combinational gates. At the same time, the data rate is halved since every evaluation cycle is followed by a logic-0 propagation. WDDL has significant overheads (i.e., up to 4x, 3x, 3.5x in area, delay, energy, respectively [14]). Furthermore, differences in output loading between complementary outputs and internal switching differences can cause data-dependent power consumption, which can be exploited by power SCAs [16, 57].

In order to remove the requirement for equal loading, an unconditional discharge phase on differential outputs is introduced. For example, Three-phase Dual-Rail Pre-charge Logic (TDPL) has a three-phase operation, in which (1) precharge and (2) evaluation phases are followed by (3) an unconditional discharge phase [51]. However, this requires complex three-phase clock generation and distribution that makes TDPL vulnerable to attacks on the clocking infrastructure. An attacker can potentially alter the clock generation to separate the discharge phase from the evaluate phase (or even completely eliminate it), which bypasses the countermeasure [55]. Hence, a self-timed version of TDPL (ST-TDPL) has been introduced [55]. ST-TDPL uses a single clock, versus three in conventional TDPL, thus eliminating the need for complicated three-phase clock generation and distribution. In a chain of ST-TDPL gates, each gate controls the discharge phase of the previous stage. Each stage discharges once the following stage completes the evaluation. However, with the increased gate fan-out, this hand-shake mechanism between the stages complicates the design of ST-TDPL gates.

Switch-Capacitor Logic (SCL) uses a different hiding approach. In this scheme, the internal power supply is isolated with a large capacitor rather than modifying each logic gate to decrease data-dependent power consumption [58]. It has three phases of operation: (1) precharge, (2) evaluation, and (3) discharge. During precharge, the capacitor is charged to VDD from the global power supply of the chip. During evalution, the capacitor disconnects from the power supply and the circuit draws current from the capacitor. During discharge, the capacitor is shunted to a consistent reference voltage. Hence, the SCL circuit uses a constant amount of power even with the use of standard CMOS logic gates [58]. However, if the attacker surpasses the protection, the sensitive circuit will be unprotected. SCL also incurs significant area, delay, and energy overheads [58].

2.4 FPGA Security

2.4.1 Design Protection against Reverse-Engineering

The underlying platform for FPGA does not reveal information about the actual design until programming. Hence, the physical attacks discussed in Section 2.2 are irrelevant at the time of manufacturing. However, the configuration bitstream has the sensitive design information and needs to be protected against reverse-engineering during all stages from bitstream generation to device programming.

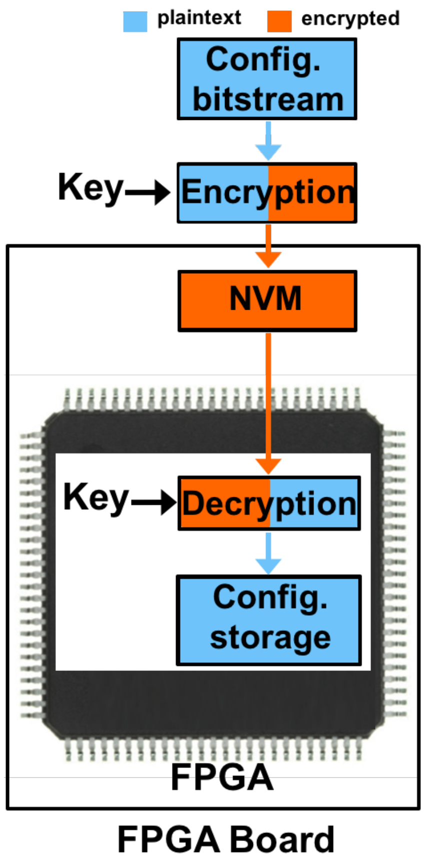


Figure 2.7: Bitstream encryption scheme for protection against reverse-engineering attacks. At the time of bitstream generation in the CAD tool, the user inputs an encryption key that is used to encrypt the configuration data and the encrypted bitstream is stored in NVM on the FPGA board. The same key is used for decryption during the programming [2]. The plaintext and the encrypted bitstream are shown in blue and orange, respectively.

FPGA manufacturers have implemented various countermeasures in the standard FPGA design flow [1]. Among these are bitstream encryption, bitstream authentication/validation, active defense mechanisms on FPGA test and support circuits, and flash FPGAs. At the time of bitstream generation in the CAD tool, the user inputs an encryption key that is used to encrypt the configuration data [2]. The same encryption key is stored on the FPGA in a small amount of non-volatile memory (e.g., using e-fuses) or a battery-backed volatile memory [8] for decryption.

However, the decryptor block remains vulnerable to both direct probing of key storage and SCA attacks, in which the encryption key can be successfully extracted [1, 2, 10, 59, 60, 61]. Furthermore, despite the bitstream encryption, the plaintext configuration bitstream is stored in the lowest level of hardware (i.e., configuration storage on FPGA die) that can be potentially extracted with low-level probing techniques [10, 62, 63, 64, 65].

A unique device ID can be used to validate the bitstream on the FPGA (e.g., a 57b device ID along with 64-byte factory flash ID used in Xilinx FPGAs [8]). Bitstream encryption is used in conjunction with bitstream encryption to prevent any unauthorized access to FPGA. Further, active defense mechanisms can monitor and protect the testing and programming ports [8]. In FLASH FPGAs, there is no optical difference after configuration, which makes direct probing attacks highly complex [2].

2.4.2 Side-Channel Analysis Resistant Secure Operation

The majority of power SCA resistant logic families discussed earlier cannot be ported directly to FPGA due to problems with flow and integration. Hence, the options to implement SCA-resistant circuits on FPGA are very limited. WDDL is a representative example of such circuits. Tiri et. al. proposes a synthesis flow to implement WDDL circuits on FPGA [15]. This technique combines standard FPGA building blocks to make secure compound standard cells with low power side-channel leakage.

Any imbalance or asymmetry between the routings of complementary parts can be exploited to decrease power side-channel resistance of WDDL [16, 57]. However, balancing differential nets is not a trivial task; it requires precise control over the placement and routing of the complementary parts on the FPGA. The techniques to address routing problems include double-WDDL (D-WDDL) [16], seperated-DDL [66], and divided backend multiplication [67]. These techniques involve low-level manipulation and relocation of logic on the FPGA fabric, hence require a rework in the tool/programming flow.

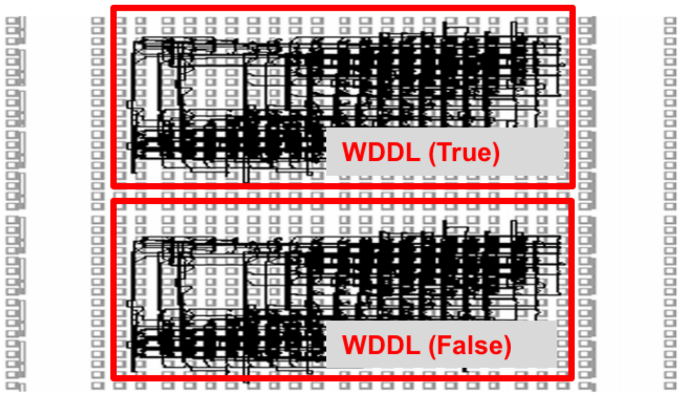


Figure 2.8: Double Wave Dynamic Differential Logic (D-WDDL) route preserving design duplication. Dual WDDL modules (i.e., true and false) with precisely matched routing are generated for power SCA resistance [16].

For instance, D-WDDL generates two complementary WDDL instances (i.e., true and false) by following a WDDL conversion step. Then, as shown in Figure 2.8, a symmetrical routing technique is employed to precisely match these dual modules. Further details can be found in the original D-WDDL paper [16]. D-WDDL has been shown to withstand power SCA attacks at the expense of significantly increased area and energy overhead over WDDL [16] (i.e., twice the size of WDDL).

2.5 Summary

In this chapter, we discussed the basic FPGA internal structure and the two major aspects of FPGA security: (1) design protection against reverse-engineering and (2) side-channel analysis resistant secure operation. In next chapters, we will address these security concerns. In Chapter 3, we explain a novel hardware-entanglement technique to ensure design protection. In Chapter 4, we discuss secure operation with an inherently side-channel resistant fabric.

Chapter 3

Hardware-Entanglement for FPGA Design Protection

In this chapter, we address one of the major FPGA security concerns: design protection. We present a novel hardware-entanglement method for bitstream protection on FPGA against reverse-engineering attacks. In this method, the configuration data is never stored in the clear, even at the lowest level of the hardware. This is possible through deep hardware-entanglement of the reconfigurable logic and interconnect through one-time pad encryption of the bitstream using a secret die-specific response that is generated by physical unclonable functions (PUFs). In this chapter, we introduce the concept and then discuss the design and details, which includes an evaluation of area, performance, and power overheads, and security benefits of the hardware-entanglement.

3.1 Attacker Model

The primary goal of the attacker is to reverse-engineer the FPGA configuration bitstream. To do so, the attacker must know both the value of the plaintext configuration bits and how these bits correspond to the programmable structures on the FPGA. Within our work, we focus on the attacks mounted against deployed systems in the field by a capable outside attacker. By compromising the configuration bitstream, the attacker can:

- clone the FPGA configuration for use in counterfeit/unauthorized systems
- modify the FPGA configuration to introduce faults
- increase side-channel emissions
- add malicious Trojan hardware
- understand the internal organization of the FPGA to clone the base FPGA design

We assume that the attacker has direct physical access to multiple (but limited) device under attack (DUA) samples, a reasonable amount of time to mount the attack, and state-of-the-art integrated circuit reverse-engineering capabilities. These capabilities include decapsulation and delayering of ICs, advanced probing/imaging technologies (e.g., pico-probe, photonic detection, scanning electron, atomic-force microscopy), and focused ion beam (FIB) IC alteration and probing capability. We also assume that the FPGA design, manufacturing, and application programming are secure. While in reality this is not necessarily the case, other researchers have explored protection schemes for these vulnerabilities [68, 69, 70, 71], and thus these scenarios are beyond the scope of this thesis.

3.2 Concept

In conventional FPGAs, SRAM cells are used to store the configuration bits for the reconfigurable logic and interconnect on the FPGA die. These SRAM cells store the configuration in the clear, as the bitstream has already passed through the decryption block if bitstream encryption is being used. Rather than using a centralized decryption block with a single short key, we propose to deeply hardware entangle the FPGA configuration bits by pairing each SRAM configuration bit with a secret response distributed across die [72]. The SRAM bits will store a one-time pad (OTP) encrypted version of the configuration bitstream, with the secret response serving as the encryption codebook. Only when the SRAM bit is XORed with the corresponding secret bit will the true plaintext configuration bit be generated. Hardware-entanglement of a 3-input LUT block is illustrated in Figure 3.1. Any configuration bit can be hardware-entangled in the similar manner.

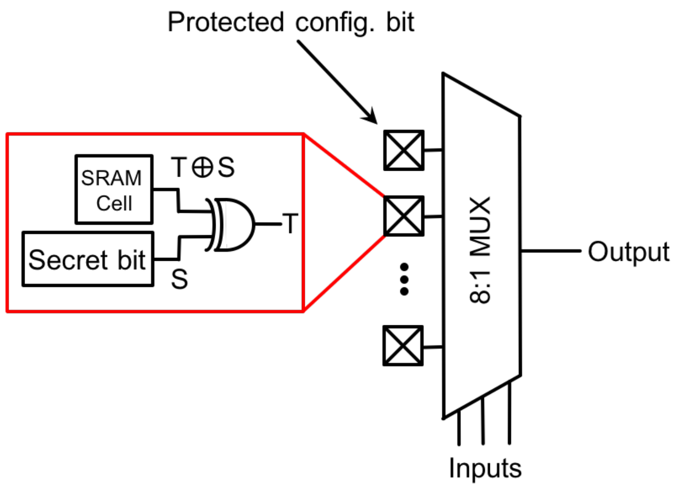


Figure 3.1: Hardware-entanglement of a 3-input LUT block. Each SRAM bit is paired with a random secret bit by an XOR gate and stores an OTP encrypted version of the configuration bitstream. Hence, the configuration bit is never stored in the clear. The SRAM bits will store an encrypted version of the configuration bitstream, with the secret bit serving as the encryption codebook. Only when the SRAM bit is XORed with the secret bit will the true plaintext configuration bit be generated. S and T denote the corresponding random secret bit and true configuration bits, respectively.

Hardware-entanglement technique requires a secret response that is distributed across the FPGA die. This secret response can be either stored on die in a non-volatile memory (NVM) [46, 73, 74] or generated on die [75, 76, 77, 78]. Emerging embedded NVM technologies have very compact bitcells, long endurance, and fast write/speeds; however, they face challenges in terms of process compatibility, manufacturing yield, performance variability, and reliability [79]. Alternatively, physical unclonable functions (PUFs) are increasingly used for secret key generation on die. PUFs provide an attractive alternative to storage of random secret bits in NVM memory, which are vulnerable to attacks [46]; PUFs generate these bits during each instance that the PUF is evaluated [80]. Hence, we use PUFs as a mechanism to generate the secret response to implement our hardware-entanglement concept. Notably, the concept is independent from the mechanism used to generate the secret response during implementation. For the remainder of thesis, we use the term *PUF* instead of *secret response*.

3.3 System Design and Use

The steps of system design and use for hardware-entanglement scheme are summarized in Figure 3.2a. At manufacture-time, the PUF response for each FPGA instance is read out via a secure scan chain. After this initial read of the PUF response, the scan chain should be permanently disabled. The security of PUF response scan chain backdoor is critical and can be achieved through various designs [81, 82, 83]. A simple method to secure the scan chain is to use multiple e-fuses along the scan path, which makes FIB repair of the scan chain challenging as there are multiple points spread across the die that must be repaired. Alternatively, we explored the reconfigurability feature of our PUF design in order to potentially eliminate the requirement of reading-out the PUF responses. The results are discussed in Chapter 5.

When working with the hardware-entanglement scheme, the user writes the application RTL description as usual; however, at the time of bitstream generation, the CAD tool one-time pad encrypts the plaintext bitstream with the PUF response and generates a die-unique configuration bitstream. If portions of the FPGA are not protected using our OTP scheme due to PUF overhead issues, the bitstream can also be conventionally encrypted (e.g., using AES-128) via bitstream protection technique. The hardware-entanglement scheme is orthogonal and complimentary to conventional protection measures. Per standard methods, the bitstream is stored in NVM on the FPGA board.

In the field, the configuration data is transferred from the NVM to the FPGA SRAM configuration storage at the time of system power-up. The data transferred from the NVM is encrypted, thus an attacker reading out the bits from the NVM or snooping on the configuration transfer bus cannot capture useful information. At the time of FPGA power-up, the PUF response bits are evaluated and XORed with the incoming configuration bits from the NVM, which calculate the true plaintext configuration bits. These bits are used throughout the design for to configure logic and interconnect. Potentially, the output node of the combining XOR gate is vulnerable to low level probing techniques. However, the attacker needs to probe hundreds of thousands to millions of nodes in order to begin to uncover the plaintext configuration bitstream. Additionally,

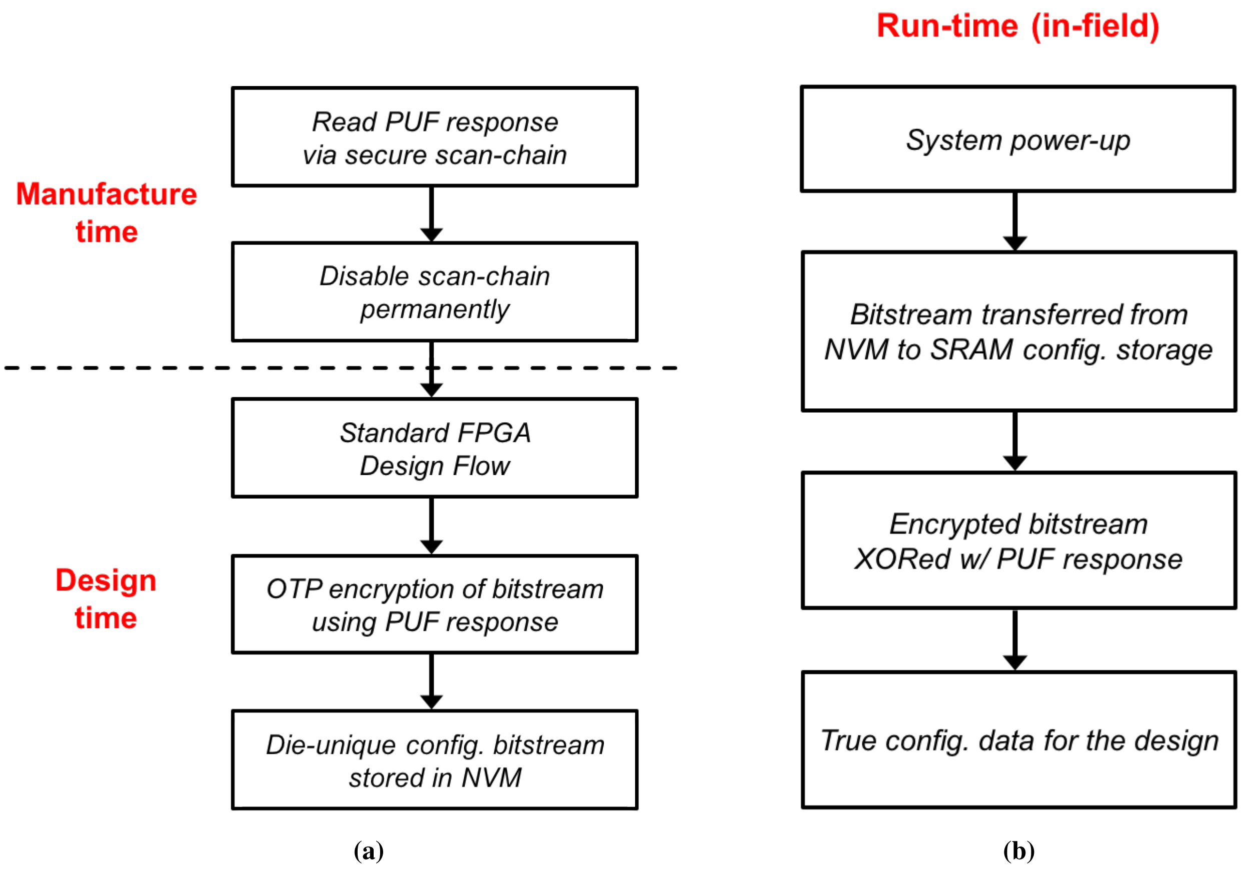

Run-time (in-field)
Manufacture time
Read PUF response via secure scan-chain
Disable scan-chain permanently
Design time
Standard FPGA Design Flow
OTP encryption of bitstream using PUF response
Die-unique config. bitstream stored in NVM
System power-up
Bitstream transferred from NVM to SRAM config. storage
Encrypted bitstream XORed w/ PUF response
True config. data for the design
(a)
(b)

Figure 3.2: System design and use of hardware-entanglement scheme (a) during manufacturing and design times (b) run-time (i.e., in the field).

the PUF design that we are proposing for the hardware-entanglement has a small delay and can fire within only 1 clock cycle. Hence, the PUFs could be fired on-the-fly, rather than only at power-up. The strength of the on-the-fly firing scheme is enhanced security and the downside is the addition of PUF delay to the forward logic.

With the core FPGA design remaining the same, changes are only required in the implementation of the configuration bits. Hardware-entanglement scheme requires the addition of a PUF bit per SRAM configuration bit, the combining XOR gate, and a secure PUF scan chain for manufacture-time PUF response read out. Due to the fact that large FPGA has millions of configuration bits, we require a high density, high performance, low power, and high reliability PUF design to ensure feasible design overheads. In addition, hardware-entanglement protection scheme requires full entropy for maximum security. As a result, any systematic bias within the PUF design that might reduce entropy is minimized. Hence, we also require a PUF design with high randomness.

3.4 Implementation

3.4.1 Physical Unclonable Functions

A PUF is a die-specific random function or a silicon biometric that is unique for every instance of the die [80]. PUFs derive their randomness from uncontrolled, and usually undesirable, random variations in the IC manufacturing process to create practically unclonable functions even if the original design files are compromised.

PUF designs are generally classified as either delay-based or bi-stable depending on the phenomenon of response generation. Delay-based PUFs (i.e., arbiter, ring oscillator) compare nominally identical delay paths, whereas bi-stable PUFs (i.e., SRAM, sense amplifier) use the activation state of a nominally balanced bi-stable element to determine a response bit [84]. Both delay-based and bi-stable PUFs offer good randomness and uniqueness properties [84]. However, the reliability of the raw PUF response bits has been found to be insufficient for direct use in applications that require high reliability [84]. For this reason, PUFs usually require techniques like error control coding (ECC) or fuzzy extraction to increase reliability at the expense of significantly increased area, delay, and power overheads [84].

Recent work in bi-stable PUF design, specifically sense amplifier (SA) based PUFs using hot-carrier injection (HCI) response reinforcement, shows that costly ECC techniques are mitigated by using HCI aging to improve reliability [80, 85]. HCI SA-PUF achieves high reliability that is targeted by many ECC implementations. This avoids the need for ECC and its corresponding overheads, which results in a very compact, fast, and low power PUF design. Hence, we use HCI SA-PUF in the proposed hardware-entanglement scheme. Further details about the original design can be found in the original HCI-SA PUF work [80, 85].

Despite having very high reliability without ECC, HCI-SA PUF is approximately 20x the size of a 6T SRAM cell. While this is very compact compared to other PUF implementations, we need a more compact PUF design since we alter the configuration storage to add a PUF bit. In the following section, we explain the HCI phenomenon and discuss our improved HCI-SA PUF design.

3.4.2 Hot-Carrier Injection

HCI is a phenomenon by which the threshold voltage (V_{TH}) of a transistor may be permanently altered post-manufacturing when high energy carriers become trapped in the gate oxide [85]. The increase in V_{TH} due to HCI stress makes the transistors slower. Figure 3.3 provides an overview of the HCI phenomenon for the NMOS. Figure 3.3a shows the NMOS transistor under normal biasing. Figure 3.3b shows the NMOS under HCI stress. With increasing V_{DS} voltage and current, electrons at saturation velocity continue to acquire kinetic energy. These high energy electrons are called hot carriers. A small fraction of these hot carriers acquire enough energy to overcome the silicon-oxide barrier energy and inject into the gate oxide (i.e., the brown square in Figure 3.3b). Transistors with carriers trapped in the oxide require a higher V_{GS} for inversion, which effectively increases their V_{TH} [85]. As shown in 3.3c, V_{TH} of the device has increased.

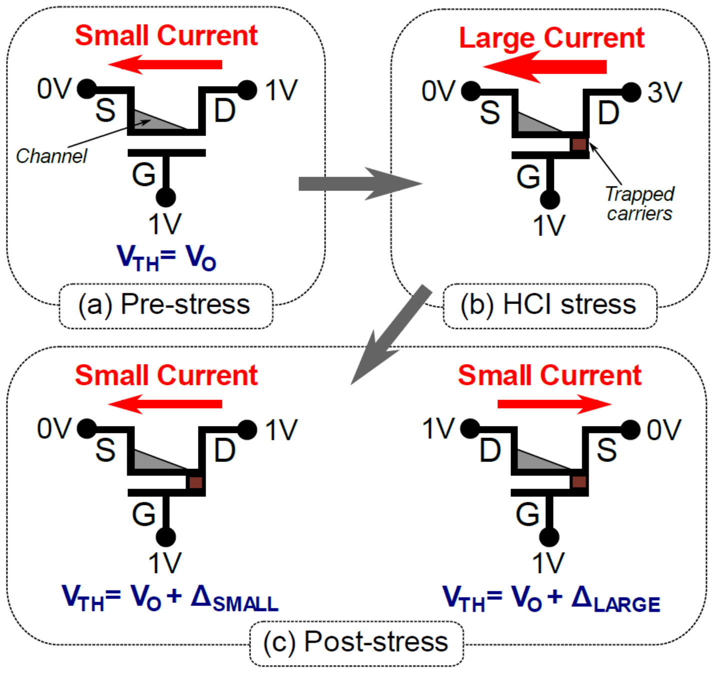


Figure 3.3: (a) Pre-stress NMOS transistor with normal biasing. (b) NMOS transistor under HCI stress conditions. A high V_{DS} generates a large current resulting in some hot electrons getting injected deep into the gate oxide (shown as the brown square). (c) After HCI stress, when the NMOS transistor is biased normally, it sees an increased threshold voltage (V_{TH}). The increase is significant ($> 100mV$) when current is in the opposite direction as during the stress conditions. The increase in V_{TH}, however, is small when current flows is the same direction as during the stress conditions [85].

Notably, the resulting increase in V_{TH} from HCI is higher if the normal operation current of the MOS is in the opposite direction as during the stress conditions.

3.4.3 Improved HCI-SA PUF Design

For the PUF core design, we use a latch-style SA instead of a StrongArm style (Figure 3.4a). This improved design uses HCI response reinforcement to improve reliability. It is designed to have the similar current density and voltage drop across HCI-stressed devices as in the original HCI-SA PUF [85]. Also, the latch-style PUF core has a smaller number of devices compared to the StrongArm PUF core. We also remove the storage unit that is used for self-reinforcement in HCI SA-PUF [85]. Storage of golden PUF responses occurs in already available FPGA configuration SRAM cells during the offset polarity measurement step. Then, the stored values are used to stress highlighted transistors (i.e., M1 or M2 depending on the golden PUF response). Figure 3.4b shows the layout capture of this improved PUF design.

Assume initially that $V_{TH,M2} > V_{TH,M1}$, hence A goes low and AB remains high when the PUF fires. This response is stored in SRAM configuration bit (highlighted in Figure 3.4a) by asserting a high pulse on STORE. Then, HCI_EN goes high and HCI_ENBH goes low. This will cause a high current (i.e., around 200uA) flowing from VDDH to the ground through M2 (shown in red in Figure 3.4a). Note the use of thick gate PMOS transistor connected to VDDH (3V). At this time, AB node is low (i.e., few tens of mVs) and M1 is off. After tens of seconds of HCI stress, HCI_EN goes low and HCI_ENBH goes high to put the PUF back in the normal operation. However, $V_{TH,M2}$ is increased due to injected hot carriers deep in the gate oxide of M2. Hence, the difference between $V_{TH,M2}$ and $V_{TH,M1}$ is increased, which improves the reliability of the PUF.

HCI circuitry constitutes around 40% of total PUF area. This area penalty can be amortized by sharing the HCI circuitry between multiple PUFs. Therefore, HCI circuitry is shared between every two PUF bits to decrease the area overhead. While this doubles the total stress time, HCI reinforcement is one-time and on the order of tens of seconds. Hence, doubling of HCI-stress

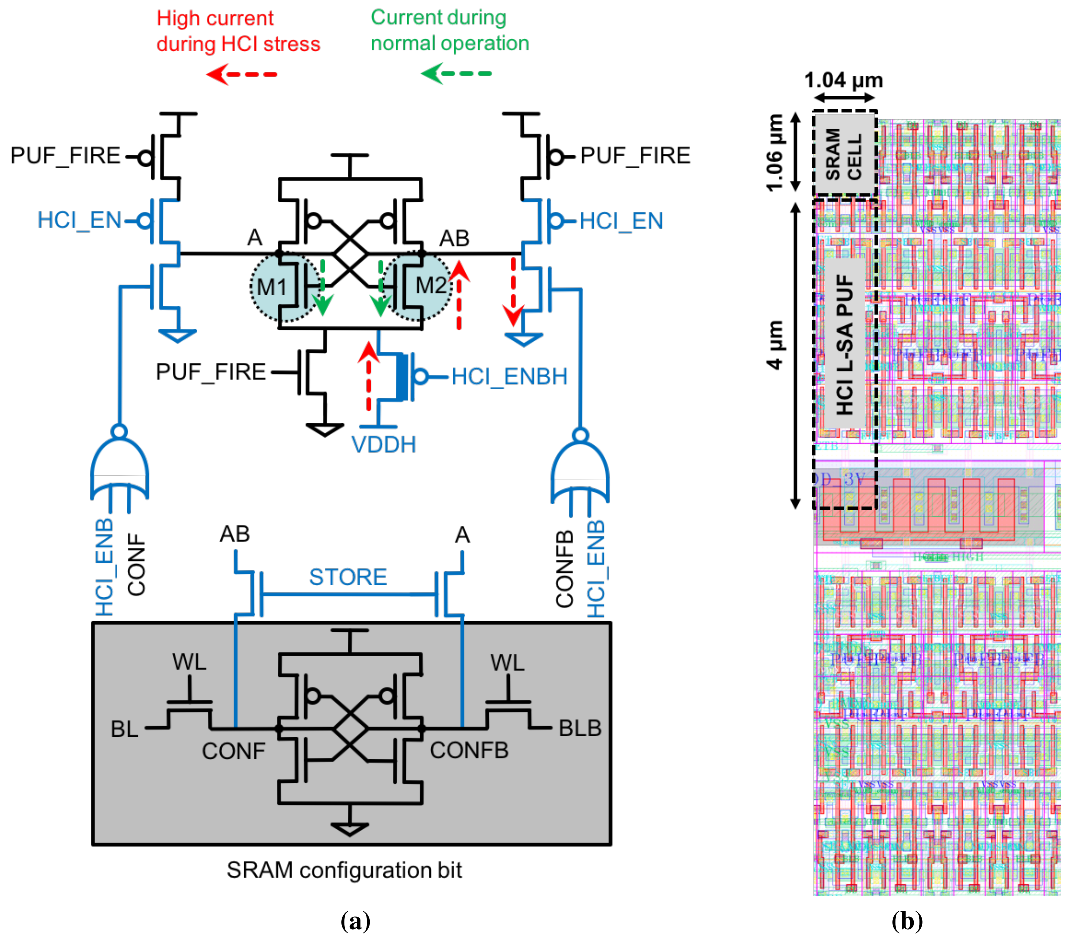


Figure 3.4: (a) Schematic of improved HCI SA PUF. HCI circuitry is highlighted in blue. The initial PUF responses are stored in already available FPGA configuration SRAM cells. Then, the stored values are used to stress M1 or M2. The current directions in normal operation and during HCI stress are shown in green and red, respectively. (b) Layout capture of an array of 2x4 improved HCI SA PUFs. HCI circuitry is shared between every two PUF bits.

time is tolerable. The improved HCI SA PUF design achieves a much smaller area with the layout 3.75x the size of a 6T SRAM cell. While this this appears to indicate an important area overhead for the proposed scheme, FPGA area is actually dominated by the interconnect (i.e., 80-90% FPGA area in the interconnect) with the majority of the needed logic devices fitting under the wiring. Thus, much of the overhead is hidden. Further, we can choose to protect only select portions of the bitstream using OTP encryption (e.g., OTP encrypting the LUT configuration, but not the interconnect configuration). As further described in Section 3.6, we have layed out the necessary LUT and interconnect structures (including HCI SA-PUF bits) for this scheme in an industrial 65nm bulk CMOS process in order to quantitatively evaluate the area, power, and performance overheads.

When the FPGA is powered-down, the PUF response data is not stored in any kind of memory cell. Rather, it is a function of the inherent (and artificially enhanced via HCI) V_{TH} variations within the devices in the sense amplifier differential pair. Thus, the response is not probe-able using conventional methods. Even if a method was devised to determine individual device V_{TH} to mV level accuracy, millions of devices would need to be probed to decrypt the bitstream. Additionally, any probing would need to be non-destructive (i.e., delayering would not be permissible), as the configuration bitstreams are FPGA die-unique. The application of standard anti-tamper techniques could further hinder direct probing [86].

3.5 Experimental Methodology

3.5.1 FPGA Architecture

Our baseline FPGA architecture derives from the widely used VTR [87] and COFFE [88] FPGA compilation tools. We use their suggested base FPGA structure, which is designed for a performance and power balanced design [89]. The logic cluster (LC) in our base architecture consists of N = 10 basic logic elements (BLEs) and each BLE is fitted with a single K = 6-input look-up-table (LUT). The BLE structure consists of a K-input LUT and a Flip-Flop (FF) that can be used together to provide additional computation and routing flexibility. Additionally, 2-input muxes are added into the design, which can potentially improve both speed and density of the BLE [89]. To further improve speed and density, local interconnect and feedback are utilized, which means that the BLE outputs can be fed back to the current LC and to global interconnect structure [88]. The number of cluster inputs (I = 40) is determined by the equation I = K(N + 1)/2 in order to maximize the LC utilization [90].

The structure is a combination of switch boxes (SB), connection blocks (CB), and wires (length-L) [87, 90]. The width of channel (W) is set to 120, which is a 20% increase over the suggested minimum width necessary to route our benchmark circuits [20, 87]. All wires span 4 LCs, L = 4, which is the recommended length for area and speed utilization [89]. Figure 3.5 shows the island style FPGA structure that employs both vertical and horizontal routing channels encircling each LC. Connection block flexibility, F_{cin} and F_{cout} that controls the number of input and output connections between the LC and the global routing structure is taken as default from the VTR initial setup [87] (i.e., Fcin = 0.15*W and Fcout = 0.10*W). The flexibility of the switch boxes, which determines factor how many directions an incoming signal can be redirected, is set as Fs = 3, which is standard in most modern FPGAs [89].

3.5.2 VTR and COFFE Tool Flow

We use *Verilog-to-Routing* (VTR) [87] to evaluate the overhead of hardware-entangling reconfigurable logic in terms of VLSI metrics of delay, area, and power. VTR flow has been widely used for exploring architectures for FPGAs and implementing benchmark circuits on a proposed FPGA fabric. We use the recent *Circuit Optimization For FPGA Exploration* (COFFE) [88] tool in conjunction with VTR in order to calculate transistor sizes in an FPGA tile. COFFE is a fully-automated transistor sizing tool for FPGAs that enables more accurate architecture evaluation by providing area, delay and power estimates of properly sized FPGA circuitry.

We use a modified version the COFFE tool to describe our new hardware-entangled logic architecture. This incorporates PUF bits and XOR gates into baseline FPGA architecture as

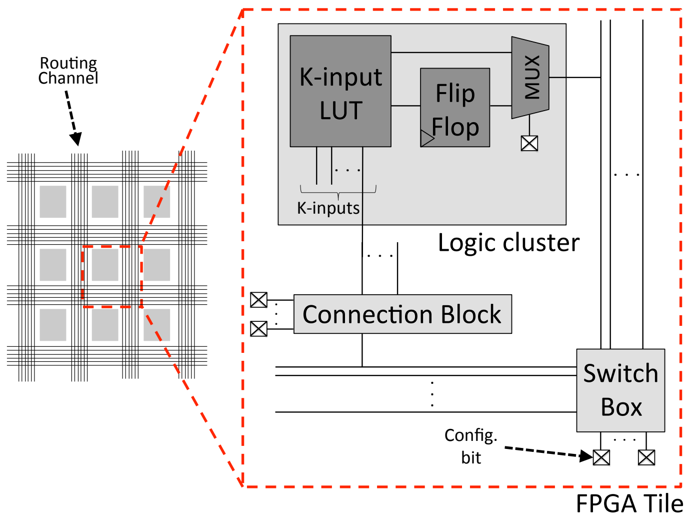


Figure 3.5: Island-style FPGA architecture and detailed view of a tile. Each tile consists of K-input LUTs and FFs, a connection block to select the inputs from the routing channel that connect to LUTs, and a switch box to select the outputs from LUTs that connect to routing channels.

defined earlier. In addition, already available 65nm PTM [91] models are used with VTR and COFFE tools during the evaluations since PUF blocks are implemented in an industrial 65nm bulk CMOS process.

3.5.3 Architectures

We evaluate four FPGA architectures: (1) baseline, (2) LUT-entangled, (3) interconnect-entangled, and (4) fully-entangled. The fully-entangled architecture pairs a PUF bit with each configuration bit in LC, CB, and SB. Interconnect-entangled architecture secures only the interconnect configuration that leaves LUTs unprotected. LUT-entangled architecture secures LUTs with PUF bits. While both the interconnect- and LUT- entangled architectures provide less security than a fully-entangled design, the PUF overheads are reduced and prior art in design obfuscation indicates that securing only the logic gates or the interconnect provide significant protection. Interconnect-entanglement is similar to split-manufacturing [92, 93, 94], wherein the layout of the design is split into the front-end-of-line (FEOL) and back-end-of-line (BEOL) layers. FEOL layers consist of transistors and other lower metal layers (i.e., $\leq$metal-4), while BEOL layers consist of the top metal layers (i.e., >metal-4) [92, 93, 94, 95]. These layers are fabricated separately in different foundries. FEOL layers in advanced technology nodes are often fabricated in high-end untrusted foundries; BEOL layers are fabricated in trusted low-end foundries. This prevents a malicious party FEOL foundry from having all the information about the designs that are fabricated there (i.e., not all of the interconnects are known). LUT-entanglement, on the other hand, is similar to logic obfuscation by camouflaging gates [31, 33]. Therefore, interconnect- and LUT- entangled architectures provide significant security with reduced overhead compared to fully-entangled architecture.

3.5.4 Benchmark Applications

Based on circuit-level simulations, COFFE calculates the area, delay, and power estimates of our baseline FPGA architecture. COFFE is modified to describe LUT-, interconnect-, and fully-entangled architectures to extract the corresponding area, delay, and power estimates. We use some common cryptographic primitives as benchmarks, namely *AES-128*, *DES*, *SHA-160* and *SHA-256*. Each benchmark is mapped onto baseline, LUT-, interconnect-, and fully-entangled architectures in VTR. The corresponding critical path, LC utilization, and total power are calculated.

3.6 Experimental Results

3.6.1 Area Results

The detailed area breakdown for all architectures is shown in Table 3.1. Tile area includes LC and interconnect. Local routing area, which is occupied by local multiplexers and buffers within LC, is considered as interconnect area in this analysis. The number of occupied LC blocks are 647, 126, 228, and 334 for *AES-128*, *DES*, *SHA-160*, and *SHA-256*, respectively. The LC utilization for each benchmark circuit is the same across all architectures. The tile area overhead is 1.4X, 1.98X, and 2.47X for LUT-, interconnect-, and fully-entangled architectures, respectively.

Table 3.1: Tile area breakdown

	Baseline		**LUT-entangled**		**Interconnect-entangled**		**Fully-entangled**	
Block	**Area (μm^2)**	**%**	**Area (μm^2)**	**%**	**Area (μm^2)**	**%**	**Area (μm^2)**	**%**
LUT	2733	35	5832	53	3137	20	6305	33
Local mux & buffers	1965	25	2000	18	4800	31	5210	27
Connection block	1166	15	1184	11	2948	19	3065	16
Switch box	1934	25	2049	19	4247	27	4751	25
Tile	7798	100	11029	100	15486	100	19332	100

The area penalty is further decreased with a more compact PUF block. One method to make PUF blocks smaller is to share more HCI circuitry with multiple PUF bits. However, the initial HCI stress time for PUF blocks increases proportionally with the amount of sharing. Different PUF topologies with smaller areas can further decrease the area overhead. In addition, PUF bits can be shared between multiple configuration bits to decrease the area overhead at the cost of decreased security (i.e., a compromise of a PUF bit would reveal information about multiple true configuration bits).

Table 3.2: Benchmark clock cycle results (ns)

	Baseline	**LUT-entangled**	**Interconnect-entangled**	**Fully-entangled**	**PUFs fired on-the-fly**
AES-128	5.34	5.45 (+2%)	6.05 (+13%)	6.2 (+16%)	+1.49%
DES	3.55	4.38 (+23%)	4.4 (+23%)	4.45 (+25%)	+2.25%
SHA-160	9.77	9.80 (+0.3%)	10.4 (+6.4%)	10.6 (+8.5%)	+0.8%
SHA-256	10.47	10.95 (+4.5%)	11.2 (+7%)	11.8 (+12.7%)	+0.76%
Average	7.28	7.64 (+5%)	8.01 (+12.6%)	8.26 (+13.4%)	+1.1%

3.6.2 Timing Results

Table 3.2 shows the critical path delay of each of the benchmarks for all architectures. Overhead percentage with respect to the baseline architecture is reported next to each timing value. The timing penalty varies with the number of LCs and interconnect distance traversed. Therefore, the actual penalty depends on the circuit that is programmed on the FPGA fabric. The average timing overhead are 5%, 12.6%, and 13.4% for LUT-, interconnect-, and fully-entangled architectures, respectively.

While the PUF evaluation delay is quite fast (i.e., post-layout extracted delay of 80ps), it does not contribute to the forward delay of the logic or interconnect. All PUF blocks evaluate in parallel immediately after the FPGA powers on. As a result of the addition of PUF bits, both the interconnect and logic cluster area are increased. This causes interconnect loading and distance to also increase, which adversely impacts delay. Alternatively, PUFs could also be fired on-the-fly, rather than only at power-up, for enhanced security. In this case, PUF delay adds to the forward logic delay. However, the additional timing overhead as shown in Table 3.2 is low since all PUFs fire in parallel at the beginning of every clock cycle with a small evaluation delay (i.e., 2.25% for the fastest benchmark).

3.6.3 Power Results

The average power overhead across all benchmarks are 18.6%, 47%, and 62.9% for LUT-, interconnect-, and fully-entangled architectures, respectively. All power values are summarized in Table 3.3. Overhead percentage with respect to the baseline architecture is reported next to each power value.

Table 3.3: Benchmark power results (mW)

	Baseline	**LUT-entangled**	**Interconnect-entangled**	**Fully-entangled**
AES-128	32.63	39 (+19.5%)	49 (+50%)	53.7 (+64%)
DES	2.92	3.12 (+6%)	3.98 (+36%)	4.2 (+42%)
SHA-160	2.71	3 (+9%)	4.2 (+55%)	4.45 (+64%)
SHA-256	4.07	5.2 (+27%)	6 (+47%)	6.7 (+65%)
Average	10.52	12.5 (+18.6%)	15.8 (+47%)	17.24 (+62.9%)

3.6.4 Overheads

The timing overhead of the entanglement is relatively modest at below 13.4% on average and below 25% even for full entanglement and the worst benchmark (i.e., SHA-256). The power overhead, while not as small as the timing overhead, is reasonable, especially given the security gains. Although the more pronounced area overhead may seem significant (40% for LUT-entanglement), it is important to note that this is less than the area gains from one technology generation step. Designers may also choose to hardware-entangle only a portion of the FPGA design, thus reducing the hardware overheads. Further, the proposed hardware-entanglement scheme has significant security benefits that offset the hardware overheads. We will discuss these security benefits next.

3.7 Security Evaluation

3.7.1 Security Metric

There are different metrics to evaluate the security of hardware obfuscation schemes. For instance, the hardness of reverse-engineering camouflaged gates is measured by the correlation of the inputs and the outputs in presence of the camouflaged gates [31]. Chakraborty et. al. quantifies the security of obfuscation-based designs by the amount of verification mismatch between the original and the obfuscated designs, which is reported by a formal-verification-based equivalence checker tool. The higher level of obfuscation implies better security [36].

Alternatively, Rajendran et. al. describes that the security of obfuscation-based designs can be measured by the number of test patterns required by an attacker to determine the keys inserted in original design [96]. The number of patterns needed to decipher the obfuscated design increases exponentially with the number of total key bits inserted in the design. We adopt a similar metric and our security evaluation is based on the total number of guesses (2^N for an N-bit bitstream) required to decipher the configuration bits secured with PUF bits. The security offered by all entanglement options increases exponentially with the number of PUF bits.

The total number of PUF bits required is calculated by multiplying the number of occupied tiles and the number of configuration bits. Each tile has a total of 2020 configuration bits (i.e., 640 bits for LUTs and the rest for the interconnect). Even for the smallest benchmark circuit (i.e., DES, 126 LCs) with LUT-entangled scheme, the number of guesses is $2^{126*640}$, which makes it impractical to reverse-engineer the configuration bitstream. Hence, LUT-entangled logic provides significant security with the least overhead compared to interconnect- and fully-entangled versions.

3.7.2 IC De-camouflaging Attacks

From the attacker's perspective, the implementation method for obfuscation (i.e., hardware-entanglement) is irrelevant. The designer hides some information from the attacker somehow and the attacker only cares about the designer's decision about what to hide (i.e., logic, interconnect, or both). For instance, fully-entangled architecture obfuscates both the logic (i.e., LUT) and the interconnect hence, it is not giving the attacker any useful information to work with. Alternatively, LUT-entanglement, obfuscates only the logic and gives away the interconnect information. As a result, the attacker has the information of how the LUTs are connected. This is the FPGA equivalent of logic obfuscation by gate camouflaging techniques proposed for ASICs [33, 96] wherein the attacker cannot discern the functionality of a particular logic gate based solely on its observable physical characteristics. In order to decipher an obfuscated design, the attacker has to assign the correct functionality to each camouflaged gate. To do so, a brute force attack has to search all the camouflaged gates' possible functionalities provided that candidate gates are selected to be inferred and unresolvable [96]. The search space grows exponentially with the number of camouflaged gates in the design, as well as the number of possible functionalities for each camouflaged gate.

Notably, the search space is narrowed down significantly with IC de-camouflaging attacks. Instead of applying randomly generated input patterns to iterate all possible functionalities assigned to camouflaged gates, these attacks use a much smaller subset of input vectors to reverse-engineer a camouflaged netlist. For instance, the proposed attack by Massad et. al. [38] applies only the input patterns that eliminate the possibilities of gate identities among all possible assignments to camouflaged gates. Such input patterns are defined as discriminating input patterns

(DI) and computed by using a SAT solver [38]. By identifying and applying DIs iteratively, the attacker can eliminate all incorrect assignments and obtain the correct assignment to each camouflaged gate. Although the attack complexity grows only linearly with the number of camouflaged gates, the complexity increases exponentially with the number of possible identities for camouflaged gates. The incremental SAT-based attack by Cunxi et. al. [97] uses an incremental SAT solver to decrease the computational effort (i.e., CPU runtime) for the attack. Despite the significant performance improvement (i.e., average speed-up of 6.5x), this incremental attack has been shown to be ineffective when all the gates are camouflaged (i.e., full chip camouflaging). The testing-based attack by Yasin et. al. [98], on the other hand, uses the input patterns generated by conducting ATPG on the original netlist and uses the fault coverage (FC) to guide the attack. The attack has been demonstrated to be ineffective beyond 30% camouflage ratio [98]. These attacks have been proven to be effective against small-scale camouflaging. However, their success rate reduces drastically for large-scale camouflaging and with increased possible gate identities [38, 96, 97, 98].

A circuit mapped to a LUT-entangled FPGA is essentially a fully camouflaged circuit that consists of obfuscated n-input LUTs with possible identities up to 2^{2^n}. Unlike the general assumption in de-camouflaging attacks, not all the obfuscated LUTs are identical in the sense that not all n-inputs are used for each of them. Some of LUTs use fewer inputs, m<n, hence possible identities are reduced. However, LUTs can still assume the maximum number of possible identities for any number of m (i.e., 2^{2^m}). Therefore, LUT-entanglement provides significant security against the state-of-the-art SAT-based and test-based attacks.

3.7.3 Physical Attacks

Table 3.4 compares the proposed hardware-entanglement technique against other common FPGA protection schemes regarding the physical attack resistance. Due to the large number of PUF bits used in our design, it is resistant to side-channel attacks (as there is no single point of attack) and direct probing attacks (as there are thousands or millions of bits that would need to be non-

Table 3.4: Security comparison summary of various countermeasures

	Attacks				Unique
Countermeasures	**Read-back**	**Rev. eng. /cloning**	**Side-channel**	**Direct probing**	**conf. per die**
Bitstream enc. [2][8][9]	✗	✓	✗	✗	✗
Bitstream auth. [8]	✗	✓	✗	✗	✗
Active defense [8]	✓	✓	✗	✗	✗
Flash FPGAs [2] [1]	✗	✓	✗	✓	✗
This work	✓	✓	✓	✓	✓

destructively probed).

There are several low level probing techniques [10, 62, 63, 64, 65]. One technique is to capture the photonic emissions of transistors to extract the contents of the small SRAM instances [10]. The photon generation rate is proportional to transistor size, supply voltage, and the transistor switching frequency. Photon emission only occurs in saturation region for high drain-source voltages [10]. However, transistors operate in saturation mode for a very short period of time during the transition between logic states. Therefore, the photonic emissions of transistors are measured and integrated over time. The integration time ranges from a few tens of seconds to minutes and even hours depending on the technique used for probing [10, 62, 63, 64, 65]. However, this type of attack requires significant time and resources even for a small SRAM instance (i.e. several kilobits in size), thus making it more difficult to mount on our proposed hardware-entanglement technique. Moreover, PUF responses are available for a few hundred of ps when fired on-the-fly. Even then, the same PUFs may not fire again depending on the inputs at the corresponding LUT. Even if we assume that they do fire consecutively, the sheer number of required measurements makes these probing attacks very challenging. For instance, a successful probing attack by Helfmeir et. al. was demonstrated on a 2 kb SRAM array with an SRAM feature size of 600nm [10]. This attack uses a 300s integration time that translates into more than 10^{12} measurements with on-the-fly PUF firing scheme.

PUF response data is not stored in any kind of memory cell, rather it is a function of the inherent V_{th} variations in the devices in the sense amplifier differential pair. While transistor threshold voltages are not directly discernable from delayering and imaging/probing the IC, there are various methods for measuring the channel doping, such as spreading resistance profiling, scanning capacitance microscopy, and electron holography [99, 100, 101]. However, these techniques have limitations in both spatial resolution and accuracy that make them unsuitable for large-scale reverse-engineering of a modern IC [102, 103, 104, 105]. Even if the available techniques could provide the needed resolution and accuracy, the sheer number of devices that would need to be probed would present an additional barrier to the attacker.

Additionally, the PUF response is read once at manufacture-time via a secure scan-chain. Hence, it is resistant to read-back attacks that target the test and verification support circuits. Finally, the proposed deep hardware-entanglement makes each configuration bitstream unique for each FPGA die. Thus, the compromise of one configuration bit does not pose a cloning or tampering threat to other configured FPGAs.

3.7.4 Tool Chain Security

As noted in the attacker model, we assume that the application programming tool chain is secure. For example, a PUF based software protection and authentication scheme described in [70] can be used to secure the programming tool chain. Without a secure application programming tool chain, the attacker could derive a configuration bitstream for a particular FPGA instance from the configuration bitstream for another FPGA instance.

The scenario is as follows: Assume the attacker gained unauthorized access to application programming flow and is capable of synthesizing two arbitrary designs: *D1* and *D2*. Let *PA* and *PB* be the PUF responses for FPGA instances *A* and *B*, respectively. Hence, the configurations for *D1* and *D2* on *A* (*A1* and *A2*) are:

$$A1 = D1 \oplus PA \tag{3.1}$$

$$A2 = D2 \oplus PA \tag{3.2}$$

Similarly, the configuration for *D1* on *B* (*B1*) will be:

$$B1 = D1 \oplus PB \tag{3.3}$$

The following operation:

$$A1 \oplus A2 \oplus B1 = D2 \oplus PB \tag{3.4}$$

gives the configuration for *D2* on *B* (*B2*). If the attacker can configure one design on an FPGA instance, he/she can run the same design on different FPGA instances. As there are published methods for securing the tool chain [70], we leave this type of vulnerability outside of the scope of this work, and hence the attacker model assumes a secure tool chain.

Furthermore, the attacker could potentially recover parts of the PUF response if he/she has knowledge of the bitstream configuration file and the FPGA internal architecture. The scenario is as follows: Assume the attacker can inject a small design (e.g., with at most 40 bits) to the system by XOR-ing it to the encrypted bitstream. The attacker knows the exact behavior of the design and he/she can brute-force the 40 bits by adding all 2^{40} possible patterns to the bitstream at the location where the design should be on FPGA, provided that the design he/she injects has one input pin and one output pin. For every possible bit pattern, the attacker checks for the expected behavior. In this way, the attacker could achieve oracle access to the 40 bits of the PUF response and eventually recover more of the PUF responses by using a divide-and-conquer approach. However, the described attack not only requires significant access to the tool chain but also knowledge of the bitstream configuration file format, FPGA internal architecture, and the ability to either enforce a design at particular location or deduce the exact location of the injected design on the FPGA fabric.

While this attack is theoretically possible, it is difficult in practice because FPGA manufacturers do not make the details of their FPGA architecture and bitstream configuration format public. Although the latter can be compromised by reverse-engineering the tool chain [59, 60], it is not straightforward for the attacker to deduce the mapping of the bits in the bitstream to the

FPGA fabric (i.e., whether a particular bit in the bitstream is used to program a piece of LUT or the interconnect and the exact mapping of the bits) without knowledge of the details of the FPGA internal architecture even if he/she gains unauthorized access to application programming flow. On the other hand, side-channel attacks on the bitstream encryption scheme only require feeding the encrypted bitstream to the decryptor block. That significant level of knowledge on the FPGA internal architecture including the ability to enforce a design at a particular location on the FPGA fabric is not required since the attacker does not attack the entire FPGA fabric. He/she rather targets the decryptor block on the FPGA. Therefore, this proposed scheme provides even more barrier to the attacks compared to bitstream encryption.

3.8 Summary

In this chapter, we presented a novel hardware-entanglement technique to protect the configuration bitstream. The technique only needs modification to FPGA configuration storage; no modification is required to the remainder of the FPGA fabric. We described the implementation of a very compact HCI-based SA design to keep VLSI overheads (i.e., area, performance, power) feasible for this protection method. In summary, hardware-entanglement offers significant security benefits including high resistance to probing and reverse-engineering/cloning attacks, and unique configuration per FPGA die that offset the hardware overheads.

Chapter 4

Power Side-Channel Attack Resistant FPGA Fabric

In the previous chapter, we presented a novel hardware-entanglement technique to protect configuration bitstream against reverse-engineering attacks. In this chapter, we focus on the secure operation of user designs running on the FPGA against side-channel analysis (SCA) attacks, specifically power SCA attacks. We first introduce a novel post-charged differential dynamic logic style with self-timed discharge (PCDL-STD) to achieve resistance against these attacks. PCDL-STD style is clock-less (i.e., asynchronous) and does not require a hand-shake of control signals unlike the existing countermeasures (i.e., TDPL and ST-TDPL), hence mitigates the issues described in Chapter 2.3.2. Our design goal is to achieve efficiency in area, performance, and energy for iso-security with the existing countermeasures. In the following, we first set PCDL-STD in context of previous work in asynchronous logic. Next, we describe PCDL-STD operation in detail and discuss the cavaets with this logic style. Then, we detail our secure FPGA tile that employs PCDL-STD. Finally, we compare VLSI metrics of the secure FPGA tile (i.e., area, delay, and energy) against a reference unsecured static FPGA tile and the D-WDDL countermeasure layered on this unsecured FPGA. We also evaluate power SCA resistance of the secure FPGA tile using the metric of Normalized Energy Deviation (NED).

4.1 Post-Charged Dynamic Logic with Self-timed Discharge

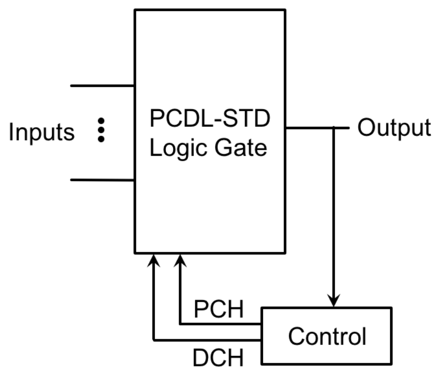


Figure 4.1: PCDL-STD logic and control signals. This logic style does not require a global clock and operates on differential pulsed input signals. PCH and DCH signals are generated asynchronously within the gate.

PCDL-STD does not require a global clock (i.e., clockless) and operates on differential pulsed input signals. It has three-phases of operation: (1) evaluate, (2) self-timed discharge, and (3) post-charge. Unlike the ST-TDPL gates, which use explicit clock phases to evaluate and precharge, and a self-timed signal to discharge, the three-phase operation in PCDL-STD is solely controlled by a control logic as illustrated in Figure 4.1. There are two self-timed control signals: (1) post-charge (PCH) and (2) discharge (DCH), which are generated in the logic structure itself, similar to how asynchronous logic gates generate their own clocking/synchronization signals [106, 107, 108] as illustrated in Figure 4.1. However, PCDL-STD control logic is much simpler because there is no hand-shake between the adjacent logic stages as the typical asynchronous logic gates do (i.e., a completion detector (CD) that consists of NOR gate or NOR gates a Muller C-element [108] when required). In other words, the control signals for a particular PCDL-STD gate are generated entirely based on the state of the true and complementary outputs.

Table 4.1 shows the control signals of PCH and DCH with the corresponding state information that they encode. The control logic for PCDL-STD needs to sequence three phases (i.e., evaluate, discharge, and post-charge) instead of two (i.e., pre-charge and evaluate) to accomodate the unconditional discharge operation. In the following, we discuss PCDL-gate structure and operation in more detail.

Table 4.1: State encoding for PCDL-STD control logic

Gate output	PCH	DCH	State	To-do
00	1	0	Post-charged	-
01/10	1	0	Eval (Prev: post-charged)	Activate DCH
11	1	1	Discharged (Prev: Eval)	Activate $\overline{\text{PCH}}$ / Deactivate DCH
00	0	0	Post-charged (Prev: Discharged)	Deactivate $\overline{\text{PCH}}$

4.1.1 PCDL-STD Gate Operation

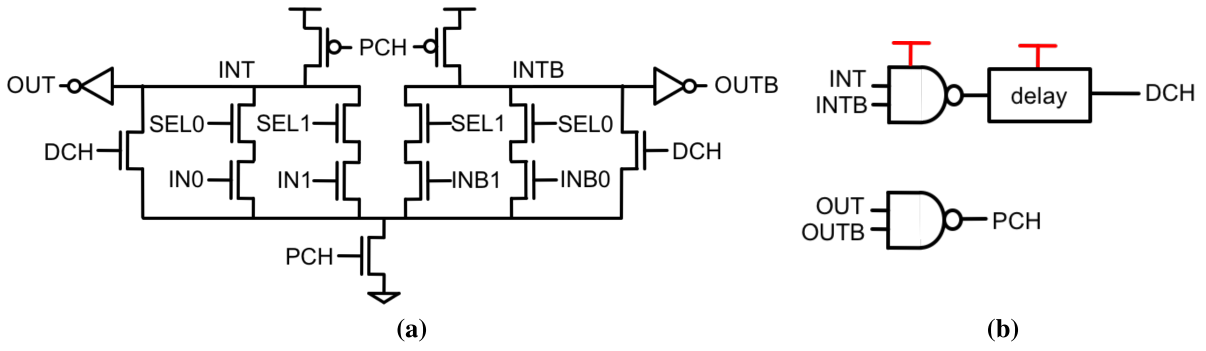


Figure 4.2: (a) A PCDL-STD 2to1 MUX schematic. (b) Control logic to generate PCDL-STD control signals. VDD of DCH signal generation logic is separate from the rest of the gate for some external controllability on the output pulse width.

Figure 4.2a shows a simple PCDL-STD 2:1 multiplexer (MUX). Figure 4.2b illustrates asynchronous control signals that are generated by the control block. Figure 4.3 is the timing diagram of PCDL-STD 2:1 MUX for an input pulse. Initially, both OUT and OUTB are low so the PCH signal is high. The gate is ready to fire and it evaluates as soon as an input pulse arrives. Assume that SEL0 is high and SEL1 is low, the first MUX input will be selected at the output. Hence, when IN0 goes high, INT goes low and OUT will go high. Following the evaluation phase, DCH goes high, forcing INTB to undergo an unconditional discharge. This discharge phase is self-timed, meaning that DCH signal goes low after a predetermined time within the control logic, set by the NAND gate and the delay line. After the self-timed discharge phase, OUTB goes high, causing PCH signal to go low, resetting the gate. Following the gate reset, PCH goes high again and completes one fire-cycle.

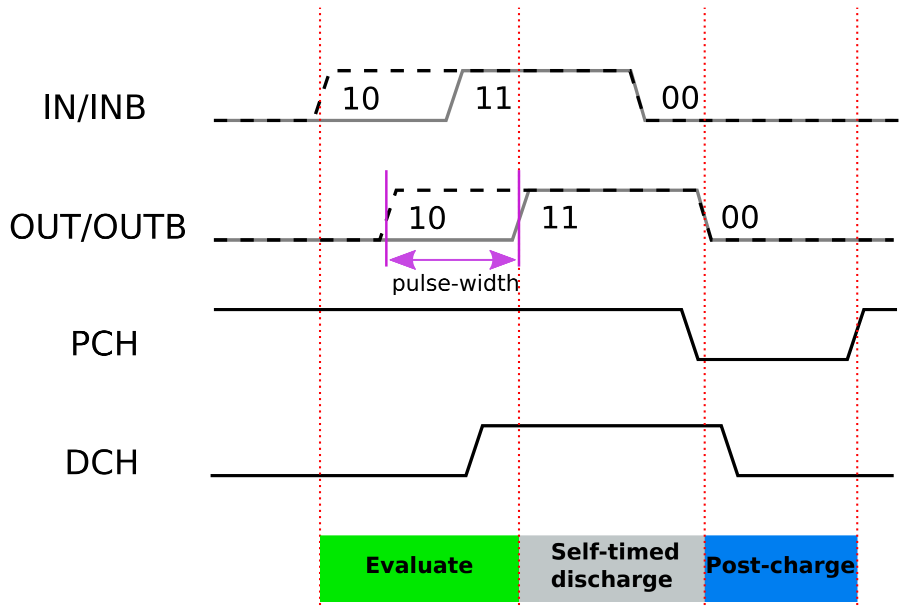


Figure 4.3: Timing diagram of a PCDL-STD 2:1 MUX. SEL0 is 1 and SEL1 is 0. Hence, the first MUX input is selected at the output. The gate evaluates as soon as an input pulse arrives. Following the evaluation, self-timed unconditional discharge ensures that both differential outputs switch. The gate resets after the discharge operation. True and complementary signals are shown in in dashed and straight lines, respectively. Pulse width (t$_{PW}$) is shown in purple.

4.1.2 Chaining PCDL-STD Gates

In a chain of PCDL-STD gates, each stage resets itself following the evaluation and the unconditional discharge within each gate. An input pulse propogates in a wave like fashion through the PCDL-STD stages. Figure 4.4 illustrates a chain of two PCDL-STD gates. Any number of PCDL-STD gates can be chained in similar manner.

Gate-1 evaluates as soon as the input pulse from the flip-flop arrives. Note that we also implement a PCDL-STD flop, which will be described in detail later in this chapter. Following the evaluation, Gate-1 undergoes the self-timed discharge operation and then the post-charge operation. In order for Gate-2 to evaluate correctly, the evaluation cycle for Gate-1 (i.e., t$_{PW}$) should be long enough and the post-charge operation for Gate-1 should finish before the Gate-2 resets. Fortunately, the regularity of FPGA makes these criteria easier to meet. Thus, PCDL-STD logic is very suitable for FPGA implementation.

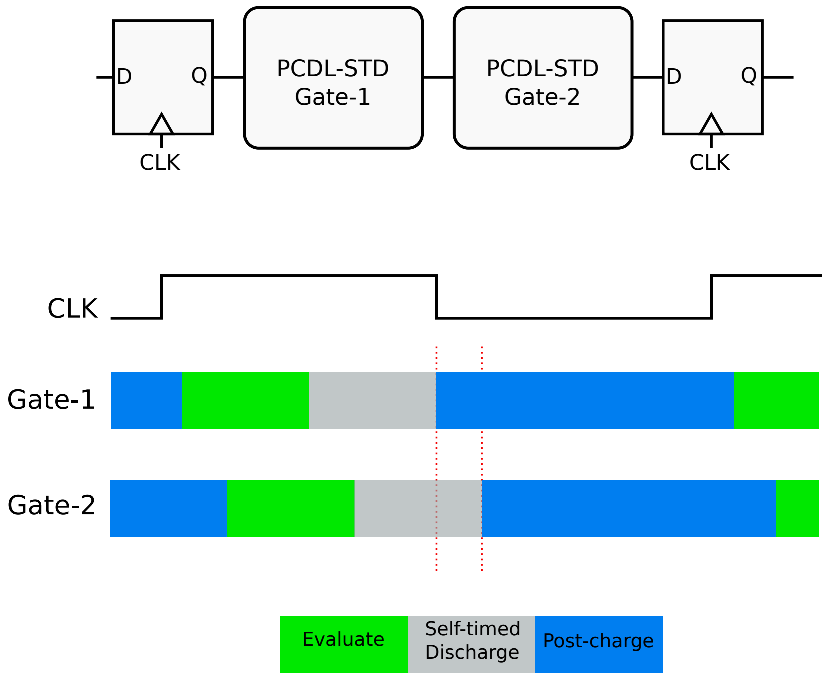


Figure 4.4: Simplified timing diagram of a chain of two PCDL-STD gates. An input pulse propogates in a wave like fashion through the PCDL-STD stages. Each stage evaluates correctly provided that the input pulse is large enough and the corresponding post-charge operation finishes before the next one as highlighted with dashed lines.

4.1.3 The Caveat: Avoiding Potential PCDL-STD Timing Issues

Output pulse-width (t_{PW}) is defined as the time difference between the beginning of evaluation (i.e., OUT goes high) and unconditional discharge (i.e., OUTB goes high) phases. OUT and OUTB are valid within this window. t_{PW} is set by the control logic that consists of a static 2-input NAND gate and a delay path. It should be large enough so that the following stage has enough time interval for evaluation. In advanced process nodes, maintaning the minimum t_{PW} for proper circuit operation can be problematic due to more pronounced variability. Therefore, PCDL-STD gates are designed with some margin on t_{PW}. We conservatively set $t_{PW} > 8$*FO_4 delay for PCDL-STD gates but they can be designed with larger (or smaller) margins depending on the target technology and corresponding variability.

Although DCH pulse control is significant for the correct operation of PDCL-STD gates, DCH signal is self-timed and generated asynchronously within the gate, not controlled externally. Despite the conservative margin imposed on t_{PW}, VDD of DCH signal generation is also separated from the rest of the gate to add some external controllability on t_{PW} (Figure 4.2b). For instance, now t_{PW} can be tuned within $\pm$ 25% of the nominal value by modifying VDD in $\pm$ 20% range for the same core VDD, according to post-layout simulations.

In addition, inputs to a PCDL-STD gate should go low before the gate completes the post-charge operation. If any (or both) of the inputs is high when PCH goes back to high, then, the gate can misfire. To avoid this, PCDL-STD gates are designed in such a way that there is enough separation between the end of input and output post-charge cycles.

4.2 Secure FPGA Tile

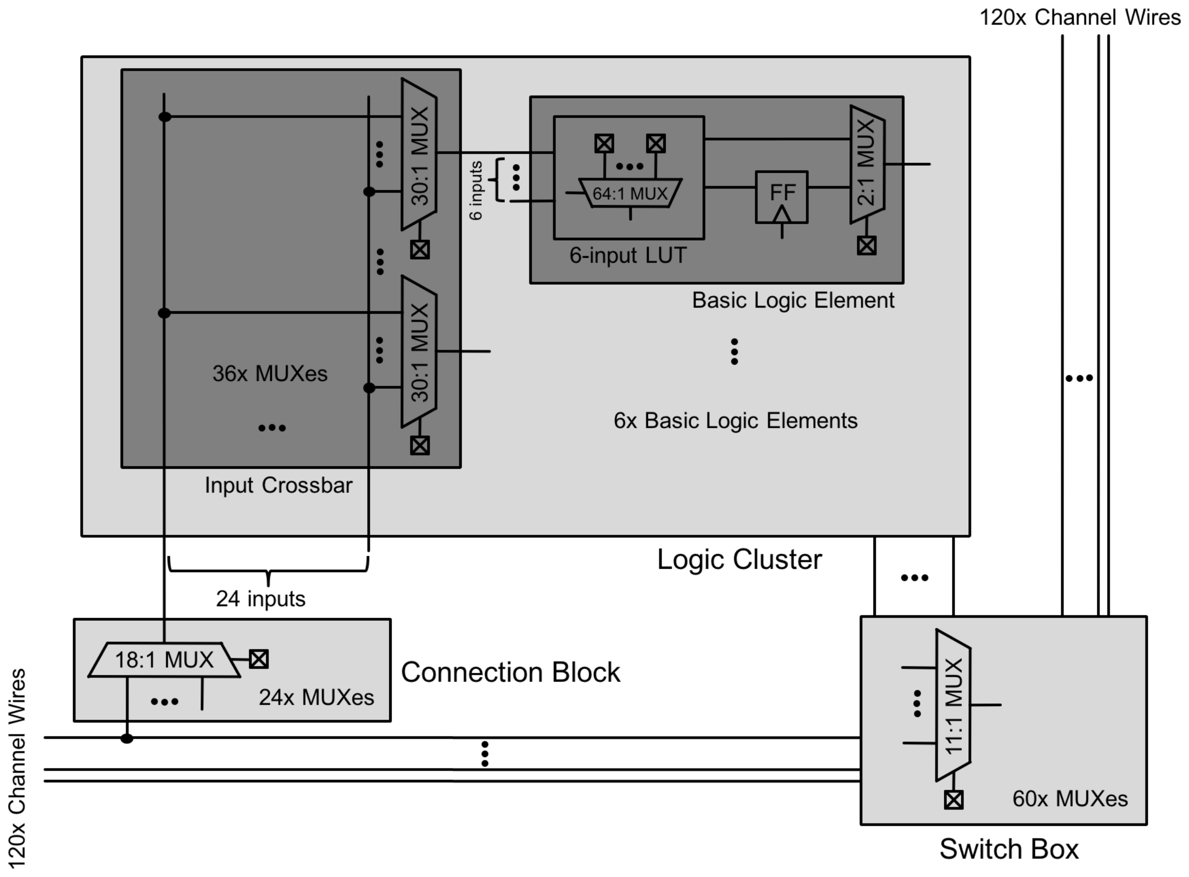


Figure 4.5: Block diagram of the FPGA tile that consists of three sub-blocks: Logic Cluster (LC), Connection Block (CB), and Switch Box (SB). LC includes the programmable logic and a crossbar to route LC inputs to LUT inputs. CB selects and routes the corresponding channel wires to LC inputs and SB is the programmable interconnect between the tiles.

An FPGA tile consists of three sub-blocks: Logic Cluster (LC), Connection Block (CB), and Switch Box (SB), as shown in Figure 4.5. LC includes the programmable logic and a crossbar to route LC inputs to LUT inputs. CB selects and routes the corresponding channel wires to LC inputs and SB is the programmable interconnect between the tiles. Each sub-block is described in detail in the following section.

A MUX is the basic building block of an FPGA tile. Each tile consists of clusters of different sized MUXes, whose sizes and numbers are dictated by the FPGA architecture parameters (Table

4.2). The corresponding MUX sizes and their counts are summarized in Table 4.3.

Table 4.2: FPGA architecture parameters

Parameter	Value
Number of LUT inputs	6
Number of BLEs	6
Number of LC inputs	24
Number of channel wires	120
Input connection flexibility	0.15
Output connection flexibility	0.16

Table 4.3: FPGA multiplexers

MUX type	MUX size	MUX count
Input crossbar	30:1	36
LUT	64:1	6
BLE output	2:1	12
FF input	2:1	6
Connection block	18:1	24
Switch box	11:1	60

Two-stage MUX architecture minimizes the number of configuration bits and is also shown to have the optimum area*delay [109]. Hence, all MUXes, except LUT, are implemented in two stages and have one-hot encoded select bits. LUT, on the other hand, has a different structure, since LUT select bits are not constant unlike the other FPGA MUXes. Therefore, LUT is implemented in three-stages as described in more detail later.

PCDL-STD N:1 Multiplexer

The schematic and the control unit of PCDL-STD N:1 MUX are shown in Figure 4.6 and Figure 4.7, respectively. Control logic is modified to support different operating modes that are selected by PC_MODE signal. If PC_MODE is 0, then post-charge mode is turned off and the gate operates as a conventional domino logic gate with a global clock signal (CLK). In this mode, DCH_EN=1 (0) enables (disables) discharge operation. Domino mode is also used for the reset at the initial power-up before switching to PCDL-STD mode enabled by PC_MODE=1. As discussed before, PCDL-STD logic is designed with some margins to ensure correct functionality. Fortunately, an FPGA consists of identical tiles and the regularity of FPGA fabric makes these margins easier to meet. Hence, PCDL-STD logic is very suitable for an FPGA implementation.

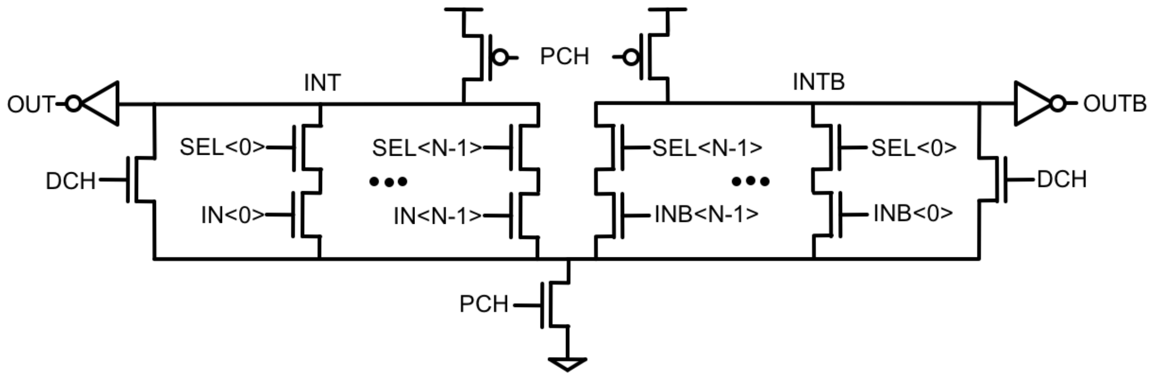


Figure 4.6: Schematic of a PCDL-STD N:1 MUX. There are N parallel 2-stack NMOS transistors, N one-hot-encoded select bits select one out of N inputs. Keepers on dynamic INT and INTB nodes are not shown for simplicity.

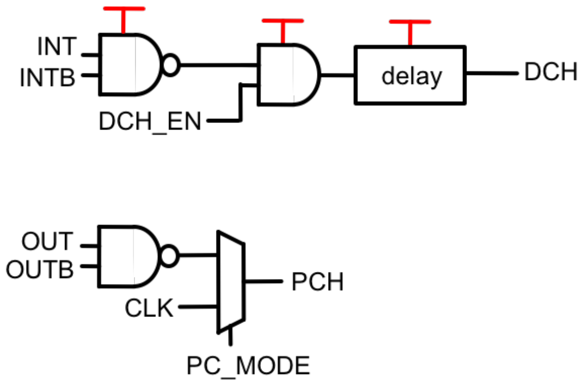


Figure 4.7: PCDL-STD control logic with different operating modes. If PC_MODE is 0, then post-charge mode is turned off and the gate operates as a conventional domino logic gate. A global CLK signal is used. In this mode, DCH_EN signal is used to enable/disable discharge operation. Domino mode is also used for the reset at the initial power-up before switching to asynchronous PCDL-STD mode. PC_MODE=1 enables PCDL-STD logic mode.

4.2.1 Logic Cluster

LC consists of an XBAR and 6 BLEs. Each BLE has a 6-input LUT and a flip-flop (Figure 4.5). Each LC has 24 inputs and 6 outputs (i.e., one for every LUT). Layout capture of a BLE and LUT XBAR MUXes are shown in Figure 4.8. In the following, we describe each sub-block in detail.

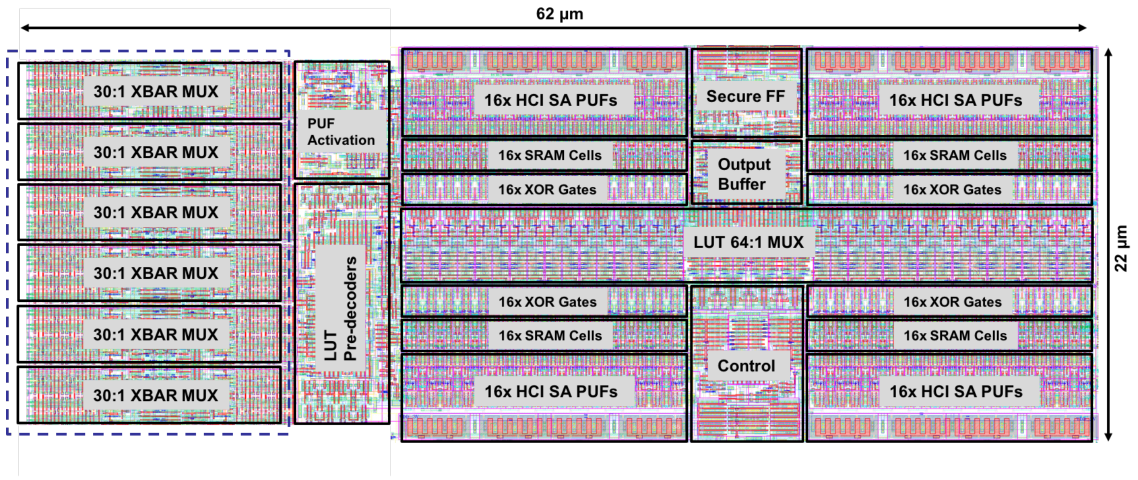


Figure 4.8: Layout capture of BLE and LUT input XBAR MUXes. Each sub-block is highlighted. LUT 64:1 MUX is implemented in three-stages. Hence, 3x 2:4 pre-decoders are used for generating 4x one-hot encoded select signals at each MUX stage. A common control circuitry generates asynchronous PCH and DCH signals for both LUT and XBAR MUXes. PUF activation circuitry controls on-the-fly firing of PUFs for improved security.

LUT

LUT consists of 64x hardware-entangled configuration bits (i.e., each SRAM configuration bit is paired with a PUF bit, as described in the previous chapter) and a 64:1 MUX that is implemented in three-stages (i.e., a total of 16x 4:1 MUXes) as shown in Figure 4.9 . 4x one-hot encoded select signals required for each MUX stage are generated by 3x 2:4 pre-decoders (Figure 4.10a). Each 2:4 pre-decoder consists of 4x 2-input PCDL-STD gates as shown in Figure 4.10b. The loading on inputs A and B are balanced by using two NMOS stacks (i.e., A-B and B-A). The schematic of a 4:1 MUX used in LUT MUX tree is shown in Figure 4.6 (N=4). The common control logic in BLE generates the control signals for both LUT and XBAR MUXes (Figure 4.7).

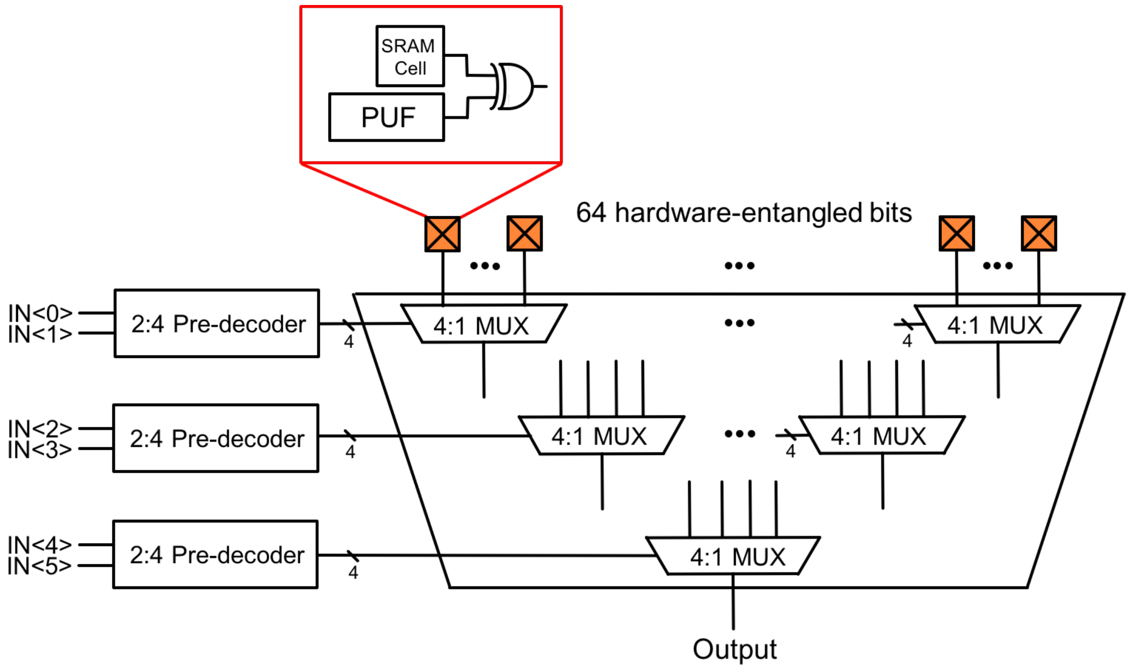
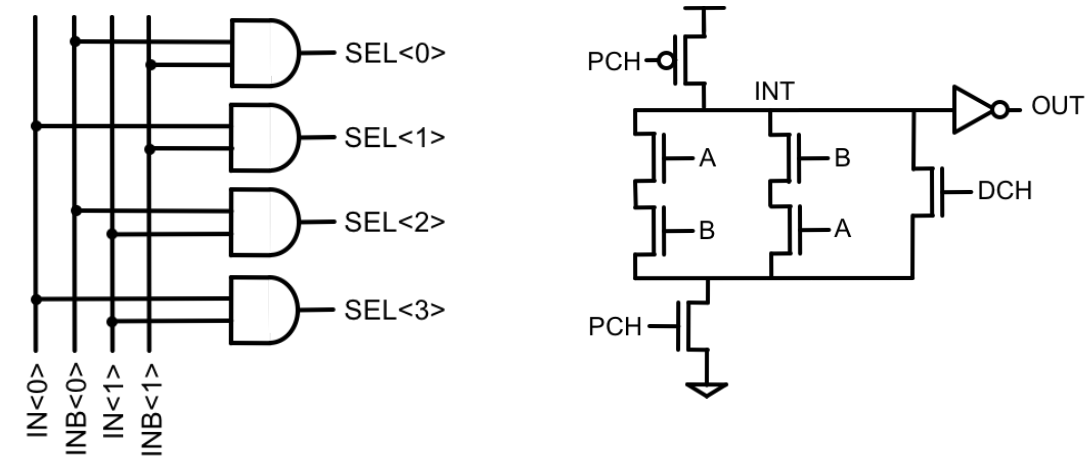


Figure 4.9: Block diagram of 6-input LUT. 64:1 LUT MUX is implemented in three-stages. Hence, 3x 2:4 pre-decoders are used for generating 4x one-hot encoded select signals at each MUX stage. Also note the hardware-entanglement of 64 LUT configuration bits (shown in orange). Each SRAM bit is paired with a PUF bit.



(a) **(b)**

Figure 4.10: (a) 2:4 pre-decoder structure with 2-input PCDL-STD AND gates are used to generate 4x one-hot encoded select signals. (b) PCDL-STD 2-input AND gate schematic. Note, the loading on inputs A and B are balanced by using two NMOS stacks (i.e., A-B and B-A). Keeper on dynamic INT node is not shown for simplicity.

Input Crossbar

Input XBAR is fully populated and can route any 24 LC inputs to any LUT input. Therefore, each input crossbar (XBAR) MUX has 30 inputs: 24 tile inputs from the channel and 6 from the LUTs in basic logic elements (BLEs) (one for each LUT). There are 36 such MUXes (i.e., 6x 6-input LUTs, one for every LUT input). Each XBAR 30:1 MUX is implemented in two-stages: the first stage has 2x 7:1 MUXes and 2x 8:1 MUXes, the second stage has a 4:1 MUX (Figure 4.11). The layout capture including BLE and the corresponding input XBAR MUXes is shown in Figure 4.8.

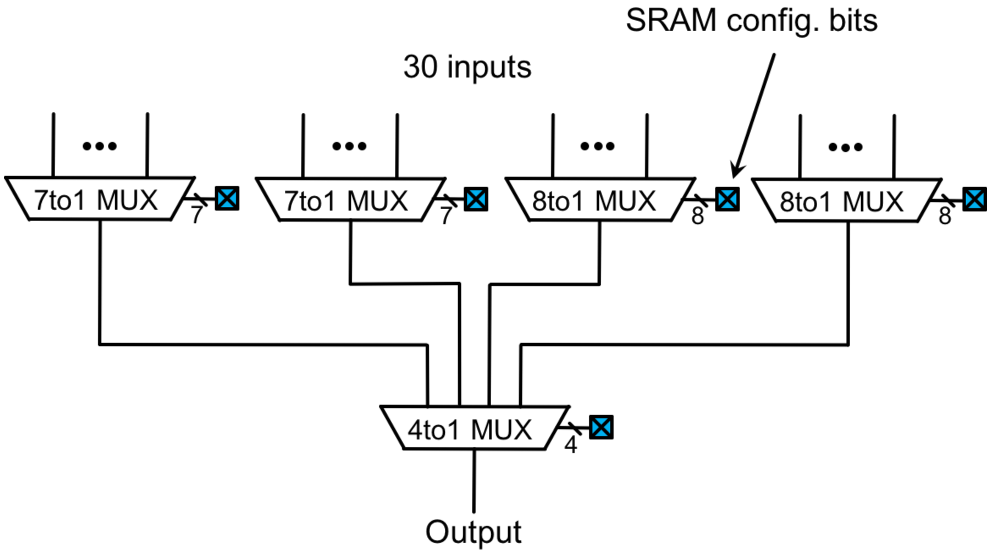


Figure 4.11: Block diagram of an XBAR 30:1 MUX that is implemented in two-stages. The first stage has 2x 7:1 MUXes and 2x 8:1 MUXes, the second stage has a 4:1 MUX. There are 12 SRAM configuration bits (8 bits and 4 bits for the first and second stages, respectively) shown in blue.

Pulsed inputs are available for a certain amount of time, as defined by t_{PW}. However, not all LUT inputs arrive at the same time. In fact, there is an unbounded time difference between the arrival times of different LUT inputs. We have observed that this time difference can range from a few tens of ps to a few ns, depending on the benchmark circuits and the constraints. Therefore, LUT inputs need to be latched until LUT evaluation is completed. Instead of using explicit latches, input XBAR MUX is designed with inherent latching capability. This requires adding full keepers to internal nodes in the first stage MUX circuits that were shown in Figure 4.6. The schematic of second stage 4:1 MUX is shown in Figure 4.12 with N=4. Once the gate evaluates, the cross-coupled inverters will be locked. Thus, the output will be latched and immune to any input change.

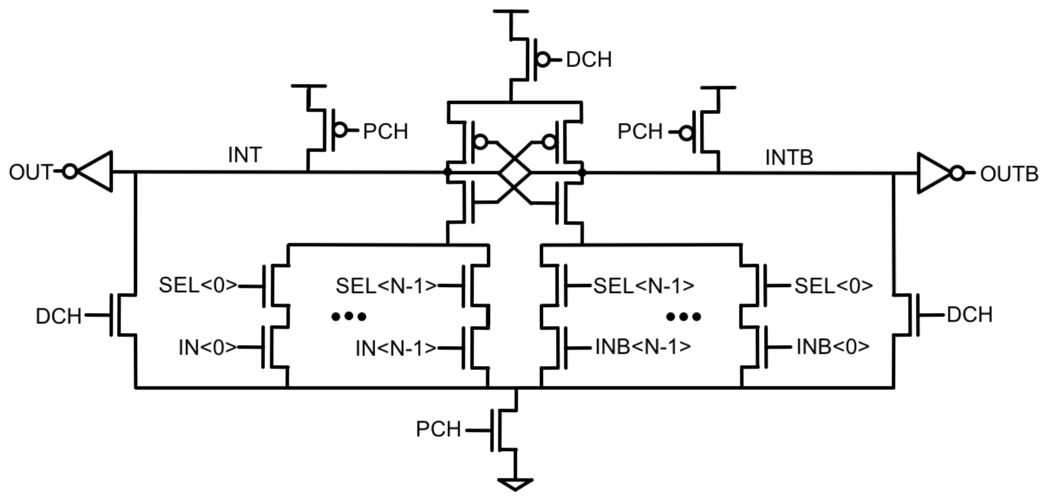


Figure 4.12: Schematic of PCDL-STD N:1 MUX with inherent latching capability. Pulsed inputs are available for a certain amount of time, as defined by t_{PW}. Hence, inherent latching is important for proper LUT operation due to unbounded time difference between the arrival time of different LUT inputs. Once the gate evaluates, the output will be latched and immune to any input change. Keepers on dynamic INT and INTB nodes are not shown for simplicity.

PUF Activation

PUFs can be fired on-the-fly for improved security. Each XBAR MUX output has an associated READY signal. This signal is generated by a static 2-input balanced NAND gate (Figure 4.13a). When, either one of the internal nodes at the second stage XBAR 4:1 MUX (i.e., INT or INTB) goes low in evaluation, the corresponding READY signal goes high and PUF activation circuit collects READY signals from all LUT inputs. If PUF-one-time-fire signal (POTF) is low, PUFs are fired on-the-fly that is when all LUT inputs are ready. After LUT completes the discharge phase, all READY signals will go low. Hence, PUFs will be reset until new inputs arrive. On the other hand, if POTF is high, then PUFs are fired one-time. PUF activation circuit is shown in Figure 4.13b.

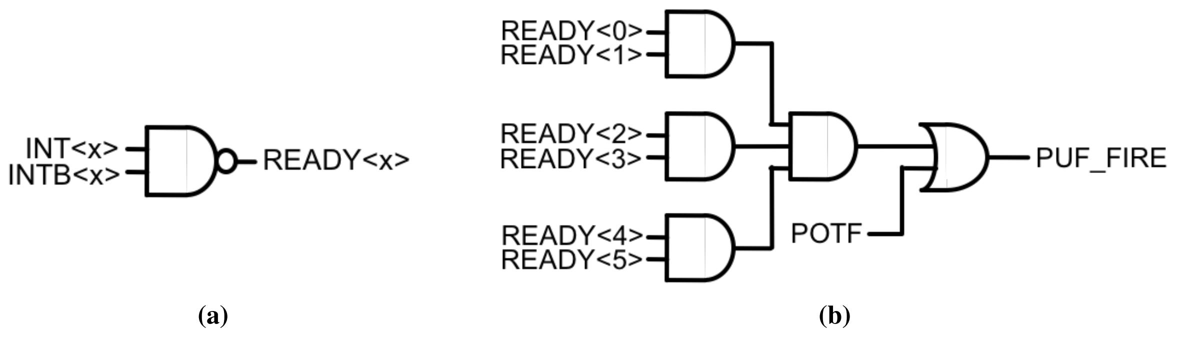


Figure 4.13: (a) READY signal generation per LUT input. (b) PUF activation circuit schematic. It controls on-the-fly firing of PUFs for improved security. If PUF-one-time-fire signal (POTF) is low, PUFs are fired when all LUT inputs are ready (i.e., all READY signals are high). Otherwise, PUFs are fired one-time.

Secure PCDL-STD Flip-Flop

Secure PCDL-STD flip-flop (FF) consists of two parts: (1) a latch and (2) a shaper (Figure 4.14). The latch becomes opaque as soon as an input pulse arrives. It becomes transparent again once the shaper evaluates in rising clock edge. The shaper generates the pulsed signal at the output and has a similar structure with the rest of PCDL-STD MUXes in FPGA fabric.

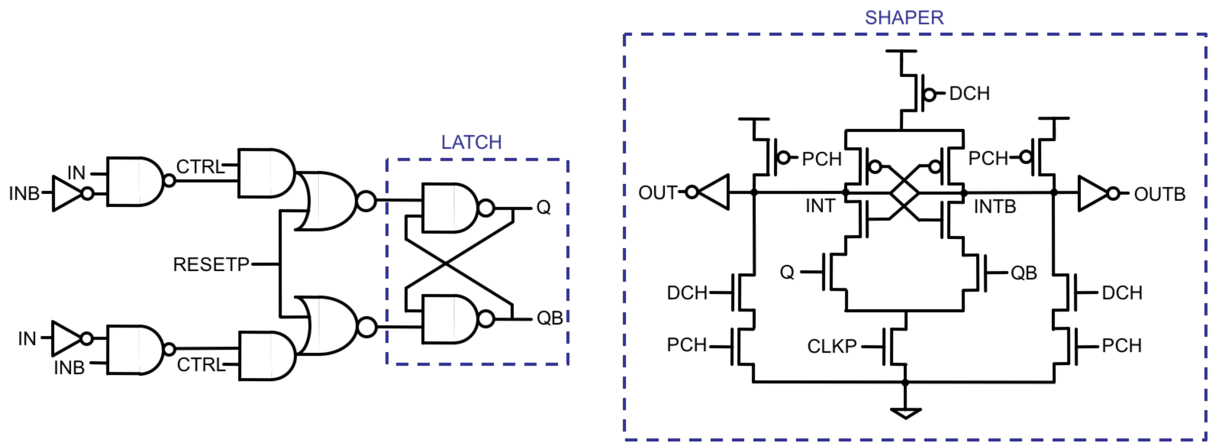


Figure 4.14: Schematic of a secure PCDL-STD FF. It consists of two parts: (1) a latch and (2) a shaper. The latch becomes opaque as soon as an input pulse arrives. It becomes transparent again once the shaper evaluates in rising clock edge. The shaper generates the pulsed signal at the output and has a similar structure with the rest of PCDL-STD MUXes in FPGA fabric.

Secure PCDL-STD FF control logic is shown in Figure 4.15. It generates two pulses: CLKP and RESETP. Since the captured data is stored in the latch, the shaper evaluates with CLKP so that multiple firing is prevented. RESETP clears the latch and it becomes transparent again following the evaluation. Once the latch is opaque, it cannot capture a new input pulse. A pulsed reset signal is used for a fast release of latch after the evaluation. CTRL is the active high enable signal for the latch. Similar to the rest of fabric, PC_MODE and DCH_EN enable/disable the post-charge mode and unconditional discharge, respectively.

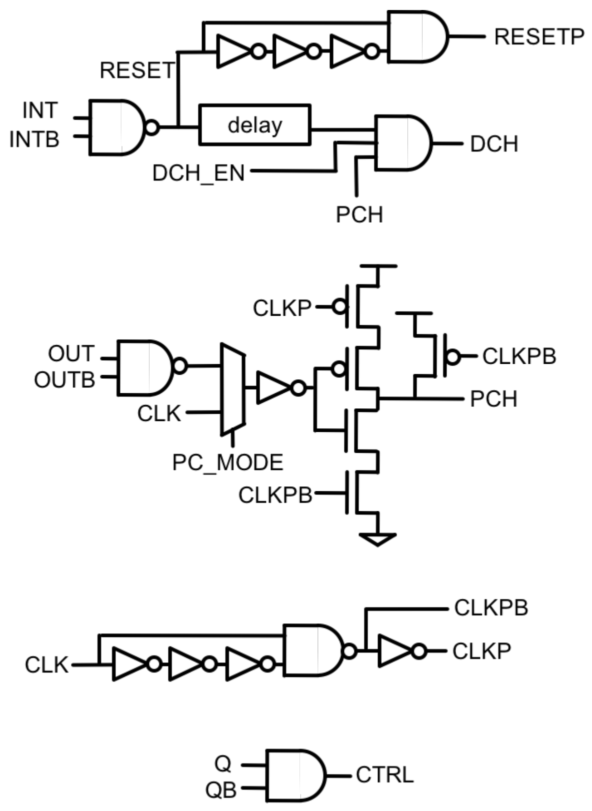


Figure 4.15: Secure PCDL-STD FF control logic. It generates two pulses: CLKP and RESETP. Since the captured data is stored in the latch, the shaper evaluates with CLKP so that multiple firing is prevented. RESETP clears the latch and it becomes transparent again following the evaluation. Once the latch is opaque, it cannot capture a new input pulse. A pulsed reset signal is used for a fast release of latch after the evaluation. CTRL is the active high enable signal for the latch. Similar to the rest of fabric, PC_MODE and DCH_EN enable/disable the post-charge mode and unconditional discharge, respectively.

Consider the timing diagram shown in Figure 4.16. Initially the latch is in reset state (both Q and QB are high), waiting for input. Assume that an input pulse (IN=1) arrives before the rising edge of CLK. Hence, QB goes low, capturing the data. However, the pulsed inputs are available for a certain amount of time. The captured data will be corrupted as soon as the input pulse undergoes the unconditional discharge. Therefore, the latch is locked with CTRL going low and no new data will be captured until CTRL goes high again. When the CLK goes high, CLKP will be generated and shaper evaluates, OUT goes high. Following the evaluation phase, DCH goes high, forcing OUTB to high as a result of the unconditional discharge phase. This will also cause RESETP to go high a brief amount of time, clearing the latch. QB goes back high and CTRL goes high again. This will make latch transparent again so that it can capture new data. After discharge, PCH goes low, resetting the shaper. Following the reset, PCH goes high again and completes one fire-cycle for the shaper.

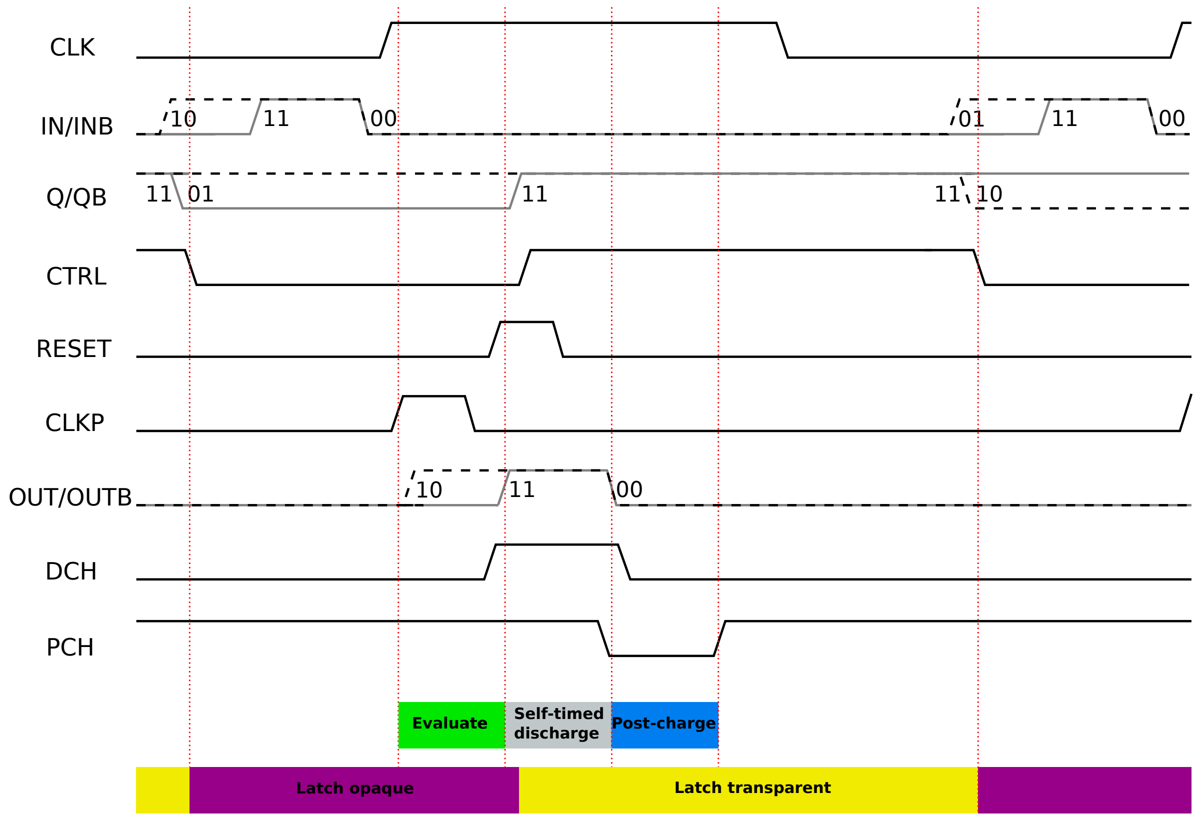


Figure 4.16: Timing diagram of a secure PCDL-STD flop. Assume that an input pulse arrives before the rising edge of CLK, captured by the latch. It becomes opaque until the shaper evaluates with CLKP generated from the rising edge of CLK. Then, the latch is cleared and becomes ready for a new input pulse.

4.2.2 Connection Block

CB is used to select 24x LC inputs from the routing channel. Each LC input can select 18 out of 120 channel wires. Hence, CB consists of 24x 18:1 MUXes. Depending on the tile, different sets of 18 channel wires can be tapped and there are 4 unique sets. Each set repeats in every 4-tiles due to length-4 wires (i.e., they span 4 tiles before being driven again). Further details on CB input patterns can be found in the original architecture paper [87].

Each CB 18:1 MUX is implemented in two-stages: the first stage has 3x 6:1 MUXes and the second stage has a 3:1 MUX (Figure 4.17). The schematic of first stage 6:1 MUX and the second stage 3:1 MUX is shown in Figure 4.6 with N=6 and N=3, respectively. The schematic of control logic is shown in Figure 4.7. The layout capture of a CB MUX is shown in Figure 4.18.

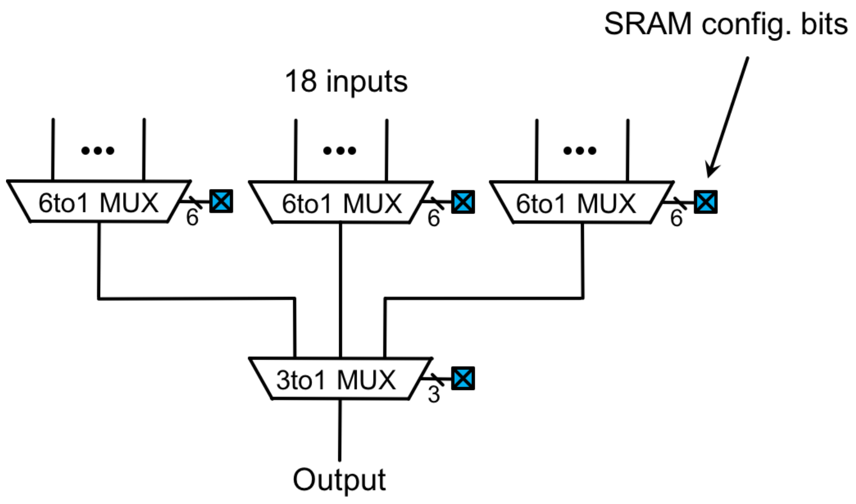


Figure 4.17: Block diagram of a CB 18:1 MUX that is implemented in two-stages. The first stage has 3x 6:1 MUXes and the second stage has a 3:1 MUX. There are 9 SRAM configuration bits (6 bits and 3 bits for the first and second stages, respectively) shown in blue.

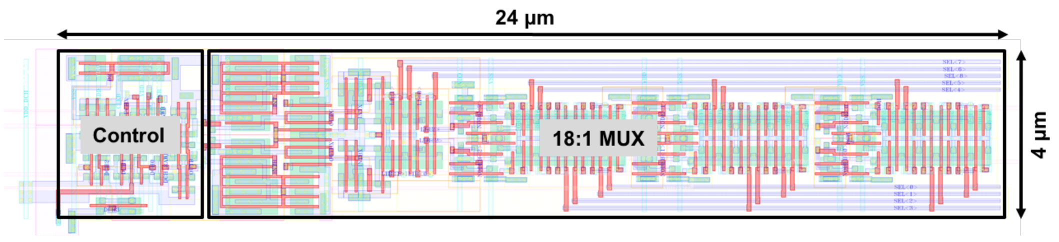


Figure 4.18: Layout capture of a CB 18:1 MUX. It occupies 24 μm x 4 μm. The control block is shown on the left.

4.2.3 Switch Box

SB is a programmable interconnect between the routing channels. Each routing channel has 120 directional, length-4 wires and each group of 8-wires constitutes a switch point (SP). There are 15 SPs in SB and each of them has 4 MUXes to drive wires in every direction (i.e., NORTH, SOUTH, WEST, and EAST). Each SP MUX has 11 inputs: incoming direction, 8 perpendicular directions (i.e., 4 NORTH (EAST) + 4 SOUTH (WEST) for EAST/WEST (NORTH/SOUTH) direction, and 2 from different BLEs depending on the SP location [87] (Figure 4.19). Therefore, there are 60x 11:1 SB MUXes in each tile.

Only 2 out of 8 wires in each direction are driven by a 11:1 SB MUX in a particular SP. For example, in Figure 4.19, wires with the index $<$0:1$>$ (i.e., WEST$<$0$>$, EAST$<$1$>$, NORTH$<$0$>$, and SOUTH$<$1$>$) are driven. Hence, the notation SP$<$0:1$>$ is used to describe the type of SP (i.e., SP pattern). There are 4 different SP patterns in the programmable interconnect (i.e., SP$<$0:1$>$, SP$<$2:3$>$, SP$<$4:5$>$, and SP$<$6:7$>$). Each pattern drives wires with different indices and is repeated in every 4-tiles in both vertical and horizontal directions due to length-4 wires as shown in Figure 4.20. Further details on SPs and their patterns can be found in the work by Lemieux et. al. [17, 87].

Each SB 11:1 MUX is implemented in two-stages: the first stage has a 5:1 MUX and a 6:1 MUX, the second stage has a 2:1 MUX (Figure 4.21). The schematics of first stage 5:1 MUX and 6:1 MUX are shown in Figure 4.6 with N=5 and N=6, respectively. On the other hand, the second stage 2:1 MUX (Figure 4.12 with N=2) benefits from the inherent latching capability to set t_{PW} individually at each SB 11:1 MUX. By this way, if t_{PW} of a particular pulsed signal is disturbed in a channel wire (i.e., narrower or larger as a result of noise or crosstalk between consecutive channel wires), the receiving SB 11:1 MUX will reinforce it, following the evaluation. The layout capture of a SB MUX is shown in Figure 4.22.

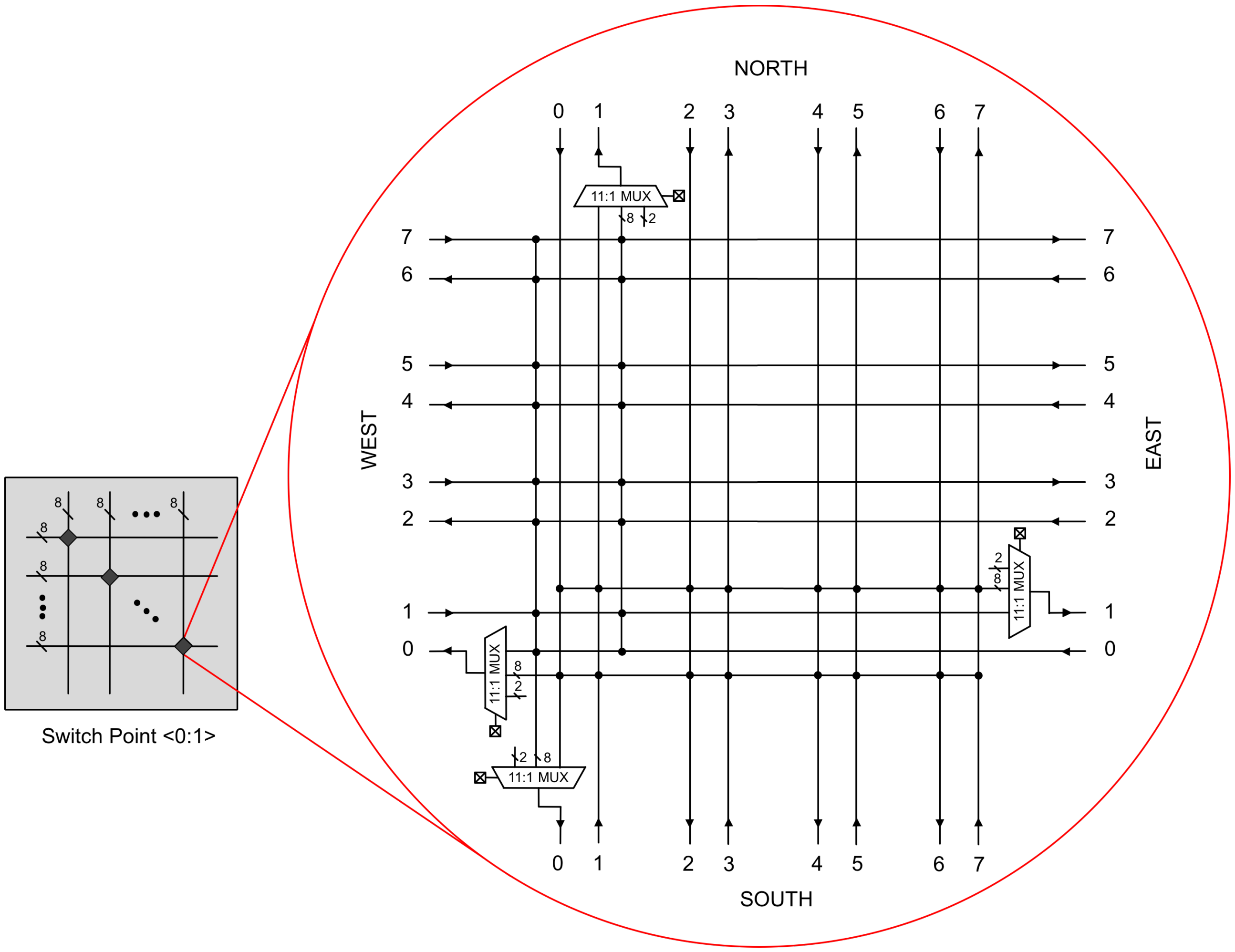


Figure 4.19: Each channel has 120 directional length-4 wires and each group of 8-wires constitutes a switch point (SP). There are 15 SPs and each of them has 4 MUXes to drive wires in every direction (i.e., NORTH, SOUTH, WEST, and EAST). Only 2 out of 8 wires in each direction are driven by a 11:1 SB MUX. Here wires with the index <0:1> (i.e., WEST<0>, EAST<1>, NORTH<0>, and SOUTH<1>) are shown to be driven. Hence, the notation SP<0:1> is used to describe the type of SP.

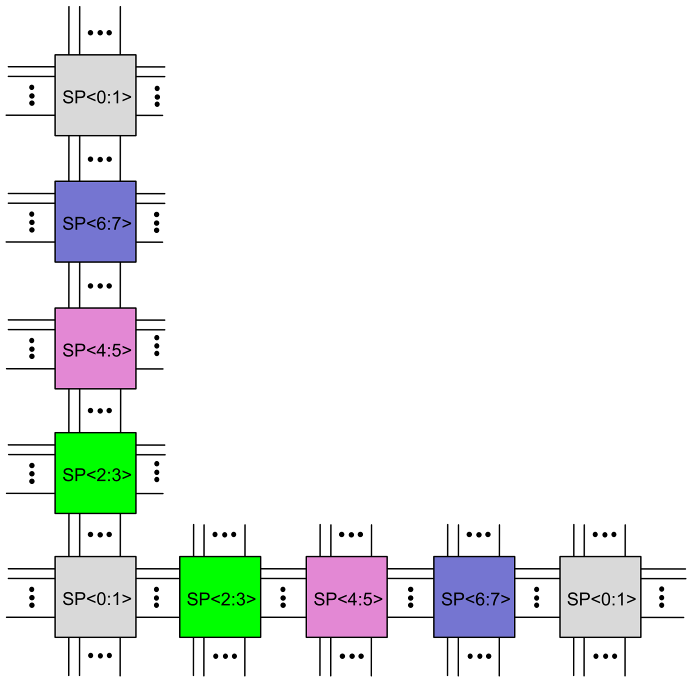


Figure 4.20: There are 4 different SP patterns in the programmable interconnect (i.e., SP<0:1>, SP<2:3>, SP<4:5>, and SP<6:7>). Each pattern drives wires with different indices in the corresponding SP. Each switch pattern is repeated in every 4-tiles in both vertical and horizontal directions due to length-4 wires.

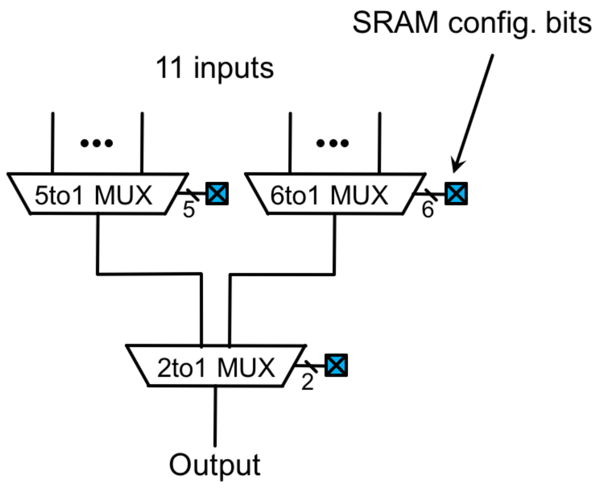


Figure 4.21: SB 11:1 MUX is implemented in two-stages: the first stage has a 5:1 MUX and a 6:1 MUX, the second stage has a 2:1 MUX.

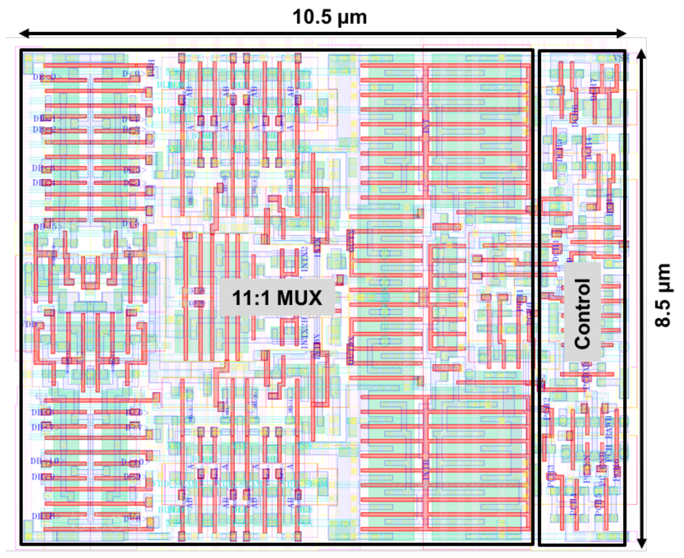


Figure 4.22: Layout capture of a SB 11:1 MUX. The control block is shown on the right.

Wire-energy-save (WES) mode

Interconnect wires are heavily loaded since they span 4x FPGA tiles. Therefore, significant energy is consumed during the unconditional discharge of interconnect wires. While setting DCH_EN=0 can disable the discharge operation, this is done for the entire fabric. Hence, control logic for SB MUX is modified to add a wire-energy-save (WES) operation mode. When set to 1, WES signal disables unconditional discharge phase only for interconnect wires. Instead, post-charge is directly activated after a certain delay to set the output pulse-width. WES mode offers significant energy savings as will be shown in the results section.

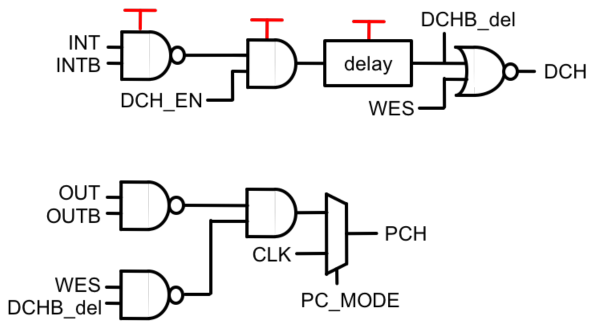


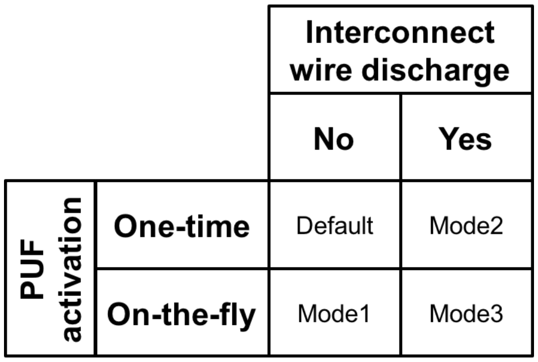
Figure 4.23: Modified control logic used in SB MUX. Wire-energy-save (WES) signal disables unconditional discharge phase for interconnect wires, instead post-charge is directly activated after a certain delay to set the output pulse-width. This saves significant energy since interconnect wires span four adjacent FPGA tiles in every direction and they are heavily loaded.

4.3 Experimental Results

4.3.1 VLSI Overhead

Experimental Methodology

We implemented our secure FPGA tile in an industrial 65nm bulk CMOS process. The nominal supply voltage for the technology is 1V. The sub-blocks of a static unsecured tile (i.e., LUT, XBAR, CB and SB MUXes) were designed and layed out in the same technology for comparison. We compared our secure FPGA against D-WDDL countermeasure and the static unsecured FPGA. The parasitic RC extraction for a large FPGA fabric that can accommodate all the security benchmark circuits described in the previous section is not feasible. Instead, we extracted FPGA sub-blocks and used the post-layout extracted simulation results in VTR flow. This simulation strategy enabled us to evaluate the overhead of our secure FPGA fabric in terms of VLSI metrics of timing, area, and energy across multiple security-focused benchmarks. Furthermore, our secure FPGA has different operating modes (Figure 4.24) that affect timing and energy of benchmarks. Hence, the results are presented across different operating modes that are controlled by PUF activation (i.e., one-time or on-the-fly) and WES (i.e., with or without unconditional wire discharge) knobs.



		Interconnect wire discharge	
		No	Yes
PUF activation	One-time	Default	Mode2
	On-the-fly	Mode1	Mode3

Figure 4.24: Operating modes of secure FPGA are controlled by PUF activation (i.e., one-time or on-the-fly) and WES (i.e., with or without unconditional wire discharge) knobs.

For D-WDDL results, first, WDDL netlist conversion is done for all benchmarks by using the methodology described in [15]. The combinatorial logic is synthesized with a limited standard cell library, which contains AND, OR and INV gates. INV gates are stripped off from this intermediate design. The input of each INV gate now becomes a global output and the output of each INV becomes a global input. The resulting design is used in VTR flow to get WDDL results. Conversion from WDDL to D-WDDL using a symmetrical routing technique is described in [16, 110] in detail. This procedure is FPGA architecture-specific and is described for Xilinx Spartan 3E. For instance, an already available asynchronously-cleared transparent latch is used as the memory element to implement pre-charge logic that is required by WDDL. D-WDDL is shown to double the slice utilization in [16, 110]. Therefore, we doubled WDDL results for area and energy results. For D-WDDL timing results, WDDL results are used. In fact, this approach underestimates D-WDDL overheads since the area increase due to the implementation of pre-charge logic is not considered here. The results are normalized against the static unsecured tile.

Area

Figure 4.25 shows the normalized area comparison. Area is calculated by multiplying the number of occupied slices and layout area. The results are benchmark average values. Secure FPGA has around 2x overhead while D-WDDL has around 4x overhead over unsecured FPGA.

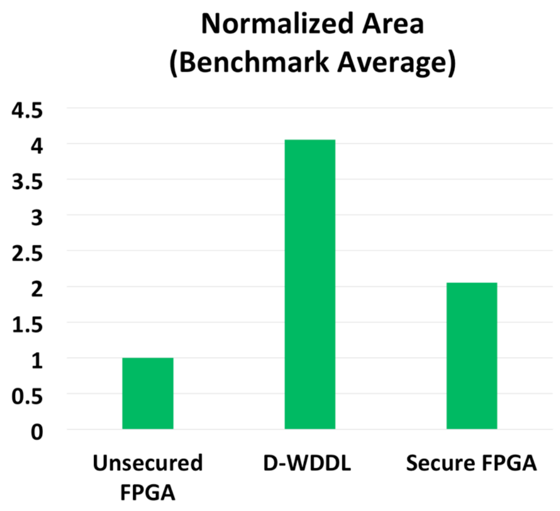


Figure 4.25: Normalized area comparison results for unsecured FPGA, D-WDDL and secure FPGA.

Timing

Figure 4.26 shows the normalized timing comparison. D-WDDL has around 2x timing overhead. There is no timing overhead in our secure FPGA. In fact, it is faster in all different operating modes. The actual speed-up depends on the PUF activation mode. For instance, secure FPGA is 1.25x faster when PUFs are fired one-time. The speed-up is 1.1x when PUFs are fired on-the-fly. WES mode does not have any affect on the timing.

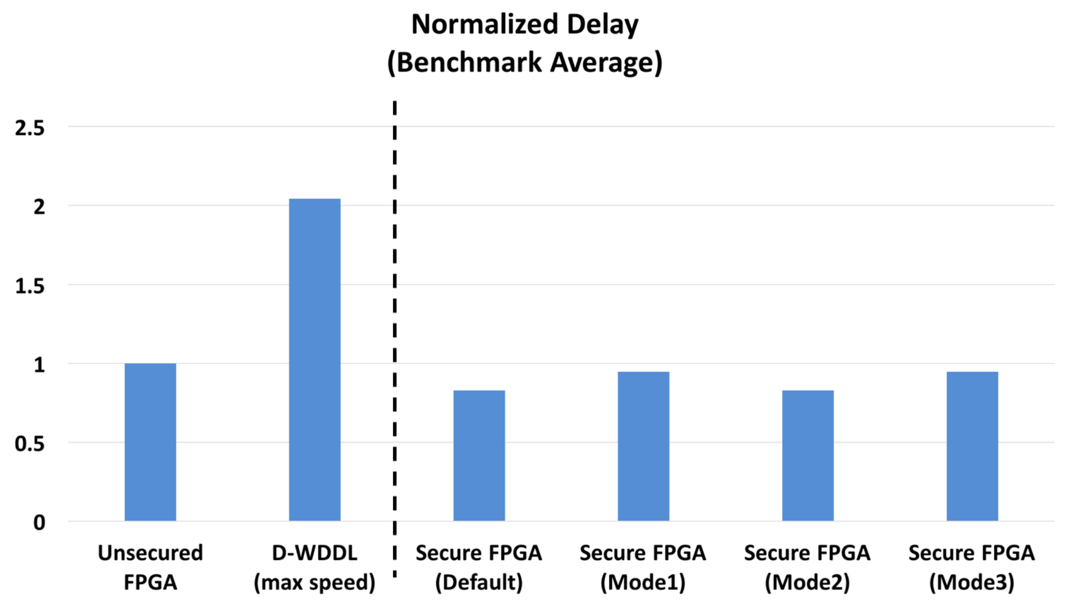


Figure 4.26: Normalized timing comparison results for unsecured FPGA, D-WDDL and secure FPGA. Timing speed-up in secure FPGA depends on different operating modes.

Energy

Figure 4.27 shows the normalized energy comparison. D-WDDL has around 7.7x energy overhead. On the other hand, the energy overhead in secure FPGA depends on the mode of operation. For instance, in default mode (i.e., one-time PUF fire and no wire discharge), the energy overhead is around 3x. It increases to 5x when PUFs are fired on-the-fly and interconnect wires are discharged unconditionally.

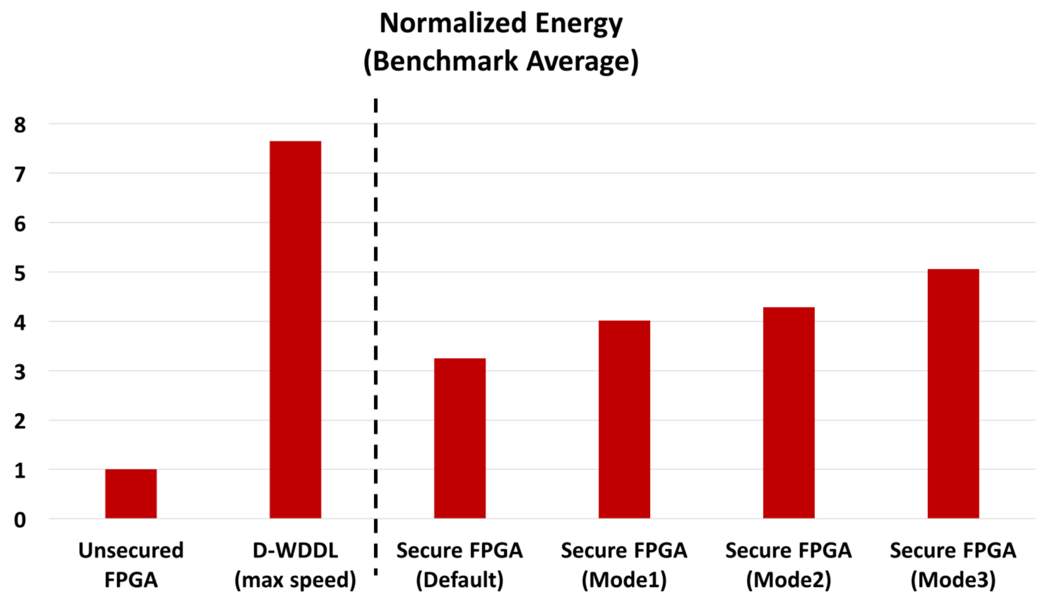


Figure 4.27: Normalized energy comparison results for unsecured FPGA, D-WDDL and secure FPGA. The energy overhead in secure FPGA depends on the operating mode.

Overhead Discussion

The area overhead of secure FPGA is around 2x. While this may seem significant, it is still half the overhead of D-WDDL countermeasure. The energy overhead, on the other hand, is more pronounced. Depending on the mode of operation, it can range from 3x to 5x. However, secure FPGA has the speed advantage over unsecured FPGA and D-WDDL. Hence, this performance gain can be traded off for energy savings depending on the application. For instance, secure FPGA consumes the highest energy when PUFs are fired on-the-fly and interconnect wires are discharged unconditionally (i.e., Mode3).

Average delay and energy results in this mode across the supply voltage range of 0.6-1V are shown in Figure 4.28. In this mode, secure FPGA is 3x more energy efficient compared to D-WDDL countermeasure at iso-performance (i.e., when the supply voltage is lowered to 0.7V). In order to find the optimum operating conditions, we also evaluated energy-delay product. Figure 4.29 illustrates energy-delay product scaling with voltage. The optimal range for energy-delay product is around 0.8V.

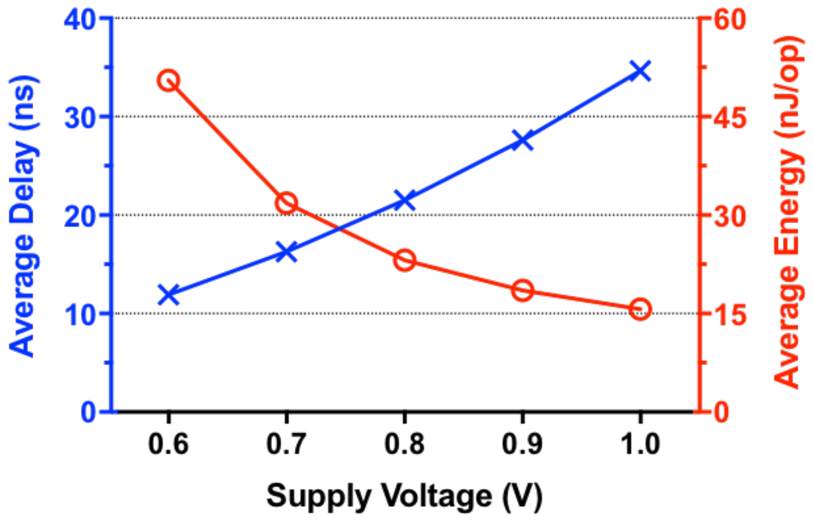


Figure 4.28: Benchmark average (a) delay and (b) energy across a supply voltage range of 0.6-1V. In this mode of operation, PUFs are fired on-the-fly and interconnect wires are discharged unconditionally.

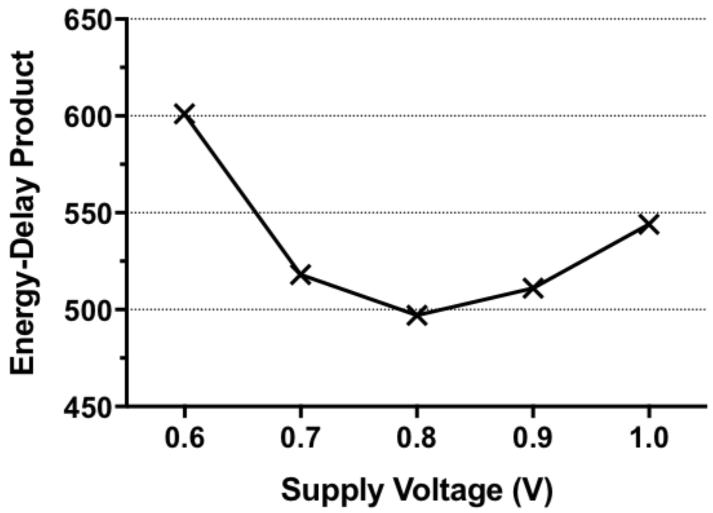


Figure 4.29: Energy-delay product scaling with supply voltage. In this mode of operation, PUFs are fired on-the-fly and interconnect wires are discharged unconditionally.

In addition, we have not included any hard macros (i.e., block RAMs, arithmetic units etc.) in our secure FPGA tile. However, commercial FPGA designs have these hard macros in conjunction with the programmable logic and the interconnect. Therefore, the actual area overhead will be much less pronounced for secure FPGA fabric.

Furthermore, the secure fabric can be implemented in a dedicated region on FPGA die and this secure section can be allocated for security-critical applications. All other applications will be mapped to the conventional unsecured fabric. Alternatively, 2.5D or 3D integration technologies [111] can be used to integrate a dedicated secure FPGA die with larger conventional unsecured FPGA dies. For example, a silicon interposer with through-silicon vias (TSVs) can be used similar to Xilinx Virtex-7 heterogeneous 2.5D FPGAs [112]. This class of FPGAs use multiple dies on the same package rather than using a single large monolithic FPGA die to achieve better yield, higher capacity and bandwidth, and better power efficiency [113]. We can use the same rationale to implement heterogeneous secure FPGAs to significantly reduce the overall area, delay, and power overheads.

4.3.2 Power Side-Channel Analysis Evaluation

One method of evaluating the power SCA resistance of a logic family is to simulate the gate under all input transition conditions and measure the normalized energy deviation (NED) [50]. To calculate NED, the energy consumption for every possible input transitions is measured in simulation. The energy per cycle is:

$$E = V_{DD} \cdot \int_0^T I_{DD}(t)dt \tag{4.1}$$

The NED is the maximum difference in energy normalized to the maximum energy consumed:

$$NED = \frac{max(E) - min(E)}{max(E)} \tag{4.2}$$

A design's resistance against an attack is determined by its weakest point [3]. Therefore, our approach is to focus on the building blocks of the secure FPGA tile. We calculate NED values for (1) the LUT, (2) the secure FF, (3) the XBAR 30:1 MUX, (4) the CB 18:1 MUX, and (5) SB 11:1 MUX. Note that PUF bits are irrelevant in this evaluation regardless of the PUF firing mode (i.e., one-time or on-the-fly) because PUF bits always consume the same amount of energy when they are fired.

As discussed before, the LUT consists of identical 4:1 MUXes and is implemented in three stages. All the other MUX blocks are implemented in two-stages and have one-hot encoded select bits. The MUX architecture narrows down the total number of inputs, which we need to calculate NED for. For instance, a CB 18:1 MUX has 3x identical 6:1 MUXes and a 3:1 MUX in the first and second stages, respectively. Regardless of the selected MUX input (i.e., any one of 18 inputs), only one out of 6 inputs will be activated in all the first-stage MUXes. In the second stage, only one out of 3 inputs will be activated (i.e., the one that corresponds to the selected input. From the power consumption perspective, only 3 MUX inputs will be relevant (i.e., consumes dynamic power). Therefore, we need measure the energy consumption for only 8 (i.e., 2^3) input transitions, as opposed to 2^{18} (i.e. for all input transitions). By the same logic, the number of relevant input transitions is reduced significantly, including the LUT. For the SB 11:1 MUX, NED values are calculated for both WES=0 and WES=1 (i.e., interconnect discharge is on and off, respectively).

Table 4.4: NED values for different FPGA building blocks

Block	NED
LUT	0.28%
Secure FF	0.51%
XBAR 30:1 MUX	0.38%
CB 18:1 MUX	0.75%
SB 11:1 MUX (WES=0)	0.36%
SB 11:1 MUX (WES=1)	0.76%

As seen in Table 4.4, the NED values for the LUT, the secure FF, the XBAR 30:1 MUX, and the CB 18:1 MUX are 0.28%, 0.51%, 0.38%, and 0.75%, respectively. Turning off the interconnect discharge (WES=1) increases the NED value of SB 11:1 MUX from 0.36% to 0.76%. Each block has a NED value smaller than 1% threshold for a sufficient power analysis resistance [58].

4.4 Summary

In this chapter, we presented a power SCA resistant secure FPGA tile implemented using a novel PCDL-STD logic style. The secure tile has also hardware-entangled LUT configuration bits for bitstream protection as decribed in the previous chapter. The secure FPGA offers significantly reduced area, delay, and energy overheads compared to D-WDDL countermeasure layered on the reference unsecured FPGA. Next chapter, will detail the prototype secure FPGA testchip that verifies the feasability of the proposed design.

Chapter 5

Secure FPGA Testchip

In the previous chapters, we introduced two security features for FPGA. We presented hardware-entanglement for bitstream protection against reverse-engineering attacks and a built-in power SCA attack resistant FPGA fabric for secure operation. Our design goal was to achieve efficiency in area, performance, and energy for iso-security with the existing countermeasures on FPGA against these security concerns. We implemented and taped-out a prototype hardware-entangled secure FPGA testchip as a proof-of-concept for our design. The results from our testchip demonstrate that hardware-entangled inherently secure FPGAs are a promising alternative to layering countermeasures on top of insecure conventional-off-the-shelf (COTS) FPGAs.

5.1 Test Chip Overview

We implemented the secure FPGA testchip in an industrial 65nm CMOS process with 9-metal layers. Nominal supply voltage is 1V for standard devices and 2.5V for thick gate devices. Measured fan-out-of-four (FO4) delay is 35ps at TTLH corner. Testchip dieshot is shown in Figure 5.1. The die area is 3.14 mm x 2.47 mm and it has 170 I/O pads along the periphery. Table 5.1 summarizes the technology and testchip features.

There are eight different voltage domains in the testchip and each has its own power grid. Among these, VDDE and VDDHE (i.e., 3V) power the I/O ring, while VDDH (i.e., 3V) is used

Table 5.1: Technology and testchip features

Technology	65nm CMOS, 9-metal Cu
FO4	35ps (TTLH)
Chip area	7.75 mm^2 (3.14 mm x 2.47 mm)
Number of I/O pads	170
FPGA core area	3.45 mm^2 (2.04 mm x 1.69 mm)
FPGA core interface	60 inputs - 60 outputs
Number of tiles	10x10
Number of LUTs	600
Number of PUF instances	38,400

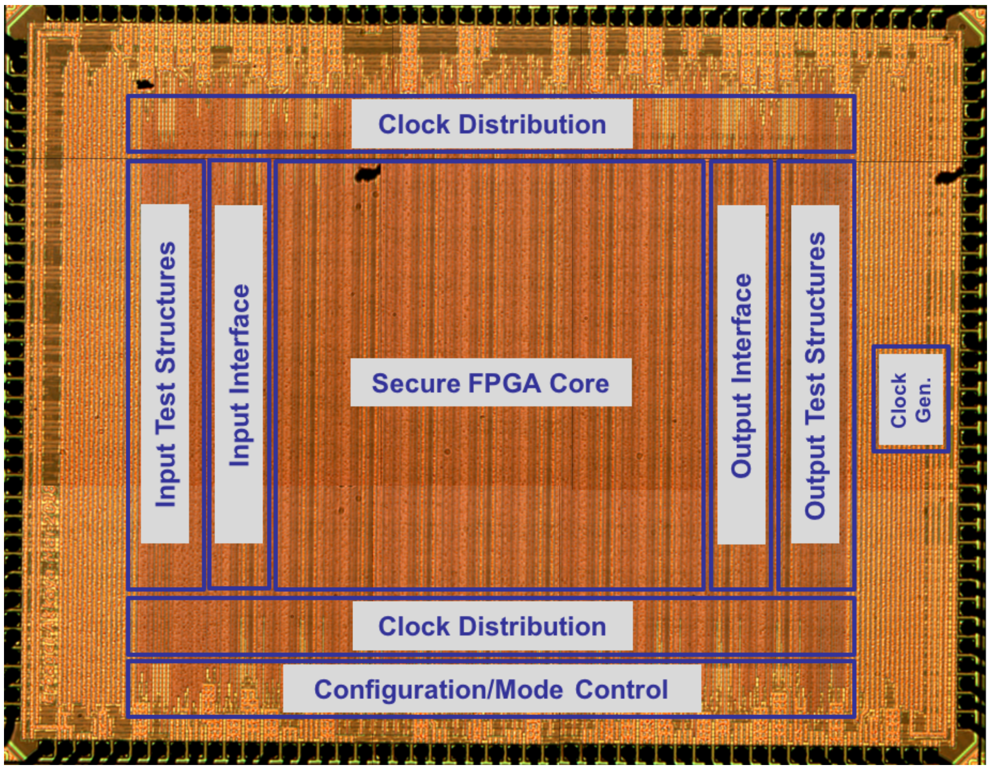


Figure 5.1: Secure FPGA prototype testchip dieshot. The chip is implemented in an industrial 65nm bulk CMOS process. It occupies 7.75 mm^2 and has 170 I/O pads along the periphery. Secure FPGA core occupies 3.45 mm^2 with 60 inputs and 60 outputs. It consists of 10x10 tiles with 600 LUTs and 38,400 HCI SA PUF instances.

for HCI stressing the PUF bits. The supply voltage of the secure FPGA core (i.e., VDD_CORE) is seperate from the test infrastructures (i.e., VDD_TEST) to enable measurements at different supply voltage levels. The supply voltage of PUF instances (i.e., VDD_PUF), discharge (i.e., VDD_DCH) and postcharge (i.e., VDD_PCH) pulse generator circuits within each PCDL-STD logic gate, and SRAM configuration bits (i.e., VDD_SRAM) are also separate from VDD_CORE for external tunability/controllability.

As shown in Figure 5.2, the testchip consists of a 10x10 secure FPGA core (2.04 mm x 1.69 mm), input shift registers and interface, output interface and shift registers, ring-oscillator (RO)-based programmable clock generator, and controller circuitry for both FPGA configuration and operation. In the previous chapter, we described the implementation details of a secure FPGA tile. The remainder of chapter is dedicated to configuration circuitry, test infrastructure, silicon results, and security analysis of the secure FPGA testchip.

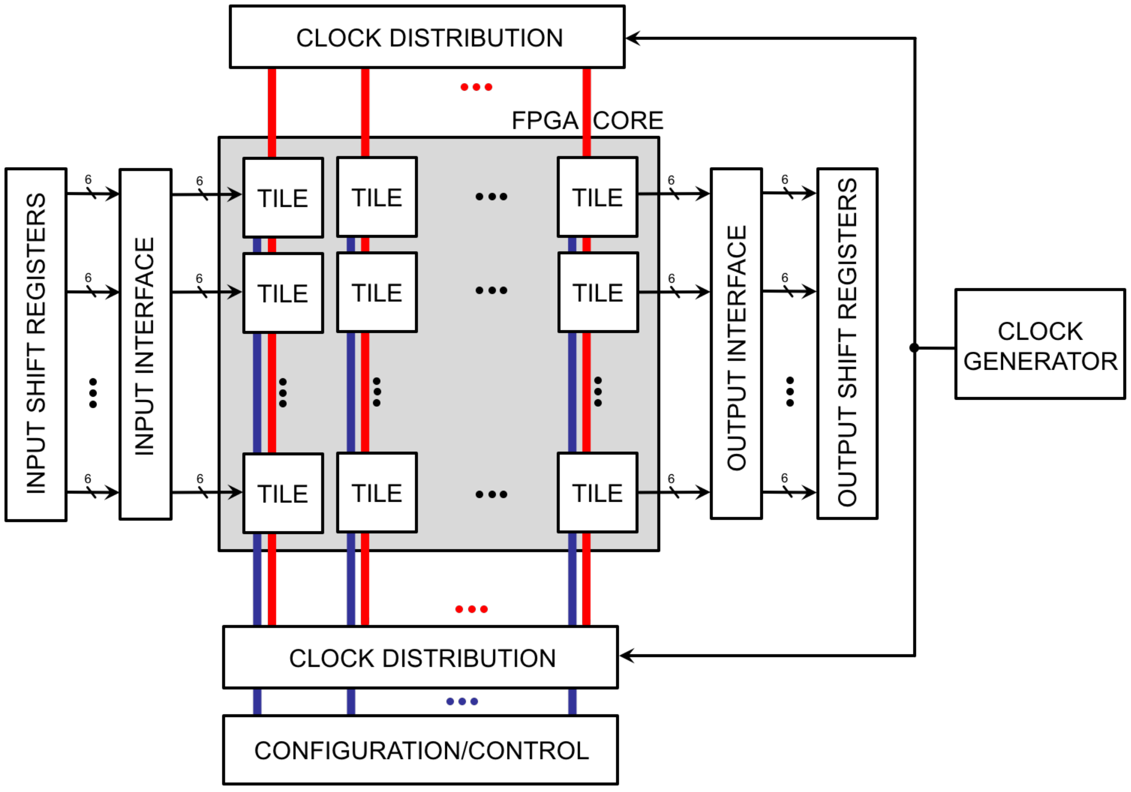


Figure 5.2: Top level block diagram of the testchip. The testchip consists of a 10x10 secure FPGA core, input shift registers and interface, output interface and shift registers, ring-oscillator (RO)-based programmable clock generator, and controller circuitry for both FPGA configuration and operation. Clock lines and configuration/control signals are highlighted in red and blue, respectively.

5.2 FPGA Configuration Circuitry

There are a total of 150,000 configuration bits in the secure FPGA (i.e., 1500 bits per tile). The number of configuration bits increases with the number of tiles that are present; a typical large FPGA has hundreds of thousands of tiles. Consequently, the configuration circuitry can become significantly large and complicated. Thus, we used a configuration circuitry [114] in which the configuration cells are arranged in an array and are loaded with bitstream configuration data. The circuitry is modified to support not only writing into configuration cells, but also reading out their contents since the initial PUF responses are read via storing the responses in LUT configuration cells.

In Figure 5.3, the overall structure of programming circuitry is described for MxN configuration cells. All configuration bits are arranged in arrays of N-rows and M-columns similar to an SRAM macro with N wordlines (WLs) and M bitlines (BLs). In this array structure, N+M flops are required for NxM configuration bits. WL scan-chain activates the corresponding WL (highlighted in red) and data scan-chain is used to load the corresponding data for the accessed row (highlighted in blue). WL and data scan-chains consist of simple master-slave D flip-flops (MS-DFF) and scannable MS-DFFs, respectively. The configuration data is scanned-in through the data scan-chain under control of DATA_SCLK. Each row is addressed by shifting a logic-1 token through WL scan-chain under control of WL_CLK. Each time a frame of configuration data is loaded and written into, the logic-1 token is shifted to the next flop in the chain to access the next row.

Since secure FPGA has 10x10 tiles, there are 10 configuration columns in secure FPGAs (M=10) and each column has 10 WLs (i.e., 10 rows). WL generation for configuration cells are 2-input AND gates. Figure 5.4 displays the details of read/write circuit. The cell BLs feed into the read/write circuit that contains the read and write support circuits. During a write operation, BLs are conditioned according to DATA_IN value. A skewed inverter is used for single-ended reads. BL reset transistors are also included in read/write circuit. BL half-keepers are not shown for simplicity.

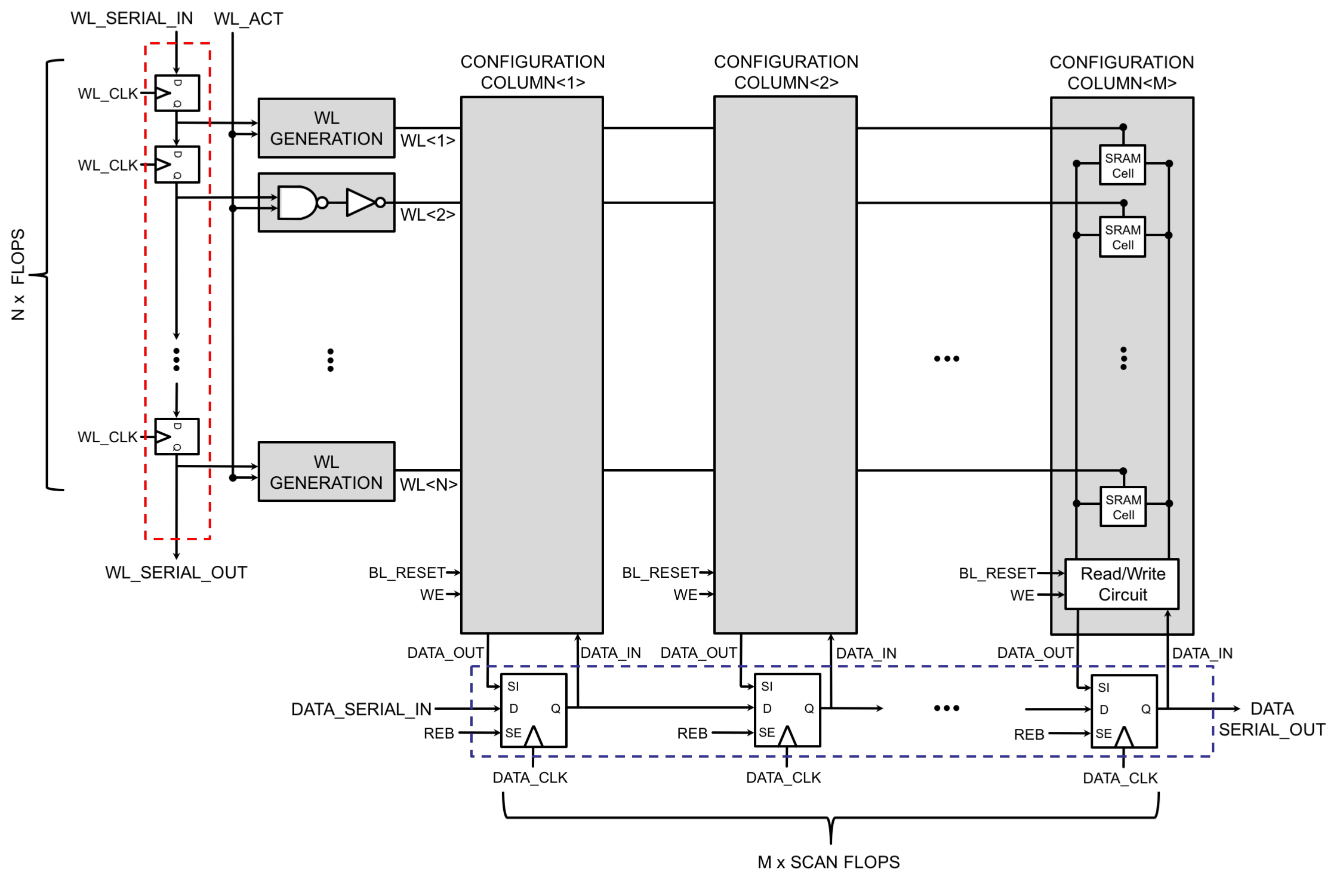


Figure 5.3: Overal structure of FPGA programming circuitry for MxN configuration cells. All configuration bits are arranged in arrays of N-rows and M-columns similar to an SRAM macro with N wordlines (WLs) and M bitlines (BLs). In this array structure, N+M flops are required for NxM configuration bits. WL scan-chain activates the corresponding WL (highlighted in red) and data scan-chain is used to load the corresponding data for the accessed row (highlighted in blue). The configuration data is scanned-in through the data scan-chain under control of DATA_SCLK. Each row is addressed by shifting a logic-1 token through WL scan-chain under control of WL_CLK. Each time a frame of configuration data is loaded and written into, the logic-1 token is shifted to the next flop in the chain to access the next row.

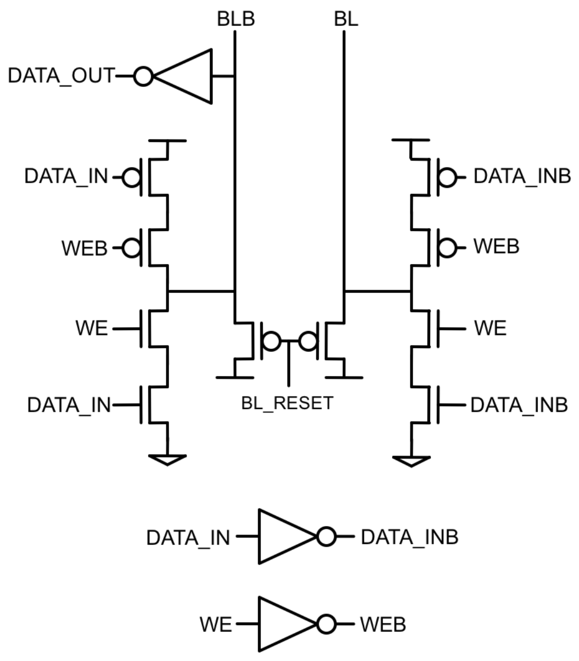


Figure 5.4: The details of read/write circuit used in each configuration column. The cell BLs feed into the read/write circuit that contains the read and write support circuits. During a write operation, BLs are conditioned according to DATA_IN value. A skewed inverter is used for single-ended reads. BL reset transistors are also included in read/write circuit. BL half-keepers are not shown for simplicity.

5.3 Test Infrastructure

32-deep scan-enabled input and output shift registers are used for testing the benchmark circuit implemented on secure FPGA core at speed. Input shift registers can hold 32 inputs and are configured to operate in a circular manner to continuously provide input data, and the output shift registers store the most recently processed 32 outputs. Input and output shift registers are single-ended. Therefore, input and output interface circuits are required to generate (capture) differential inputs (outputs) to (from) secure FPGA core. In PCDL-STD mode, they generate (capture) pulsed inputs (outputs).

5.3.1 Input Test Structures

As shown in Figure 5.5, there are 60 input test structures (i.e., one per FPGA input) and each consist of a 32b scannable register and an input interface circuitry. In scan mode, external data is scanned in one bit at a time. In normal operation mode, scanned-in data is shifted to the right.

For continuous operation and power measurements, the data outputs of the last input FFs are connected as data inputs of the first FFs.

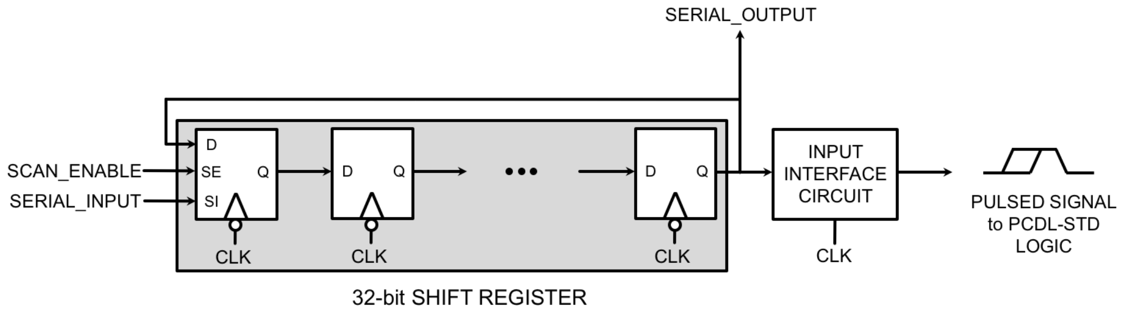


Figure 5.5: Input test structures consist of 32-deep scan-enabled shift registers that are configured to operate in a circular manner to continuously provide input data in normal operation mode. Input registers are single-ended, thus input interface circuits generate differential pulsed-inputs to secure FPGA core. Input scan-registers and interface circuits operate in opposite clock edges.

Input registers are single-ended, therefore an input interface circuit is used for signal conversion between input registers and the secure FPGA core (Figure 5.6). It is essentially a PCDL-STD buffer with a modified control block, which generates a clock pulse (CLKP) in the rising clock edge for evaluation of interface circuits so that the multiple firing is prevented (Figure 5.7). PC_MODE and DCH_EN enable/disable the post-charge mode and unconditional discharge, respectively.

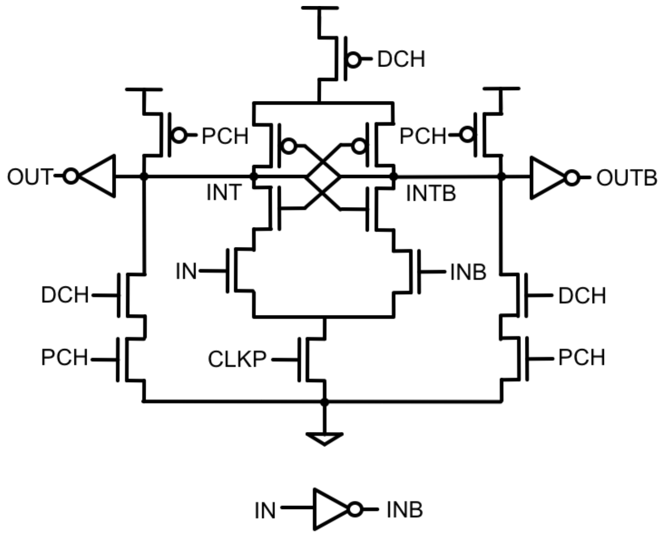


Figure 5.6: Input interface circuits are used for single-ended to differential-pulsed-input conversion between input registers to secure FPGA core. It is essentially a PCDL-STD buffer with a modified control block.

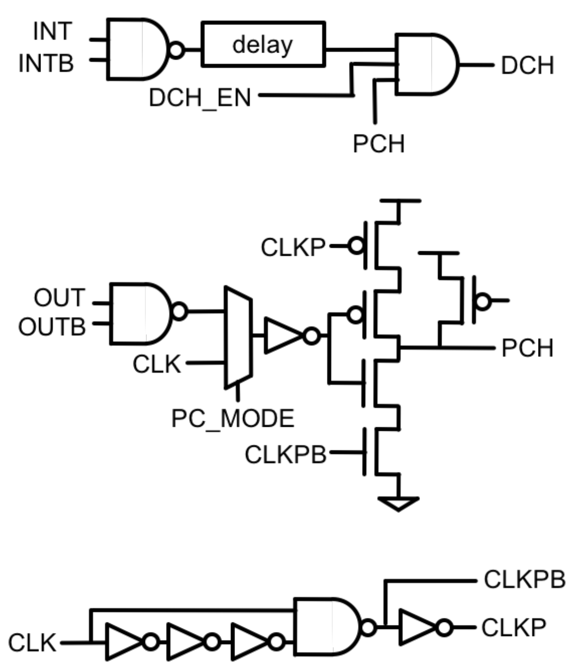


Figure 5.7: Control logic of input interface circuits. It generates a clock pulse (CLKP) in the rising clock edge for evaluation of interface circuits so that the multiple firing is prevented. PC_MODE and DCH_EN enable/disable the post-charge mode and unconditional discharge, respectively.

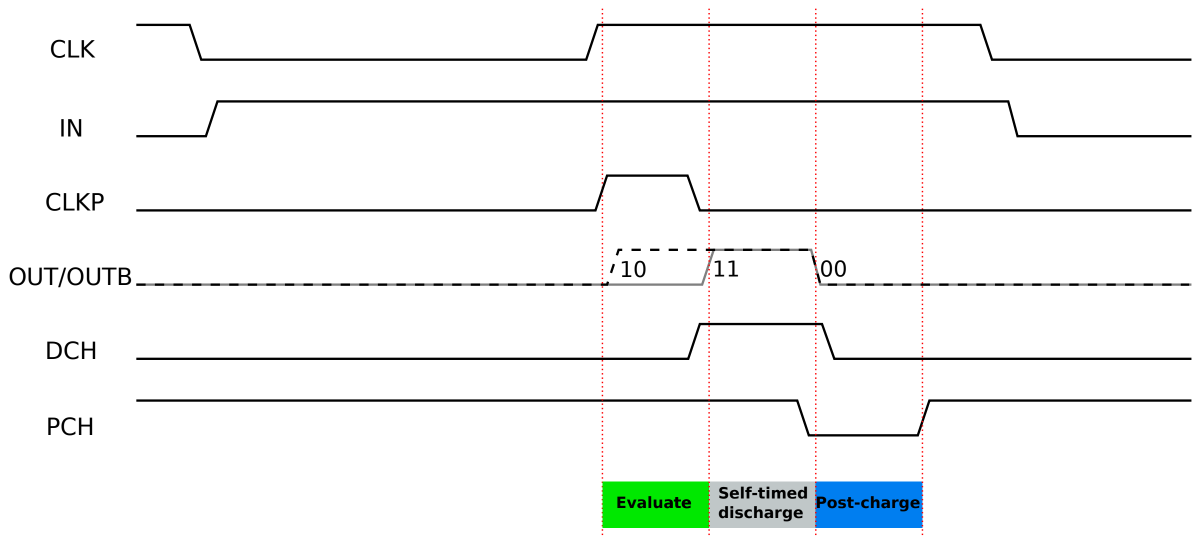


Figure 5.8: Timing diagram of input test structures. Assume that input scan register samples a logic-1 in the falling clock edge. The interface circuit evaluates with CLKP generated from the rising edge of CLK. Evaluation is followed by a self-timed discharge and reset for the next clock cycle.

Consider the timing diagram of an input test structure shown in Figure 5.8. Scan registers and interface circuits operate in opposite clock cycles. Assume that the input scan register samples a logic-1 in the falling clock edge. At this time, both IN and INB are generated. The interface circuit evaluates with CLKP generated from the rising edge of CLK. Evaluation is followed by a self-timed discharge and reset for the next clock cycle.

5.3.2 Output Test Structures

There are 60 output test structures (i.e., one per FPGA output). Each consists of an output interface circuitry and a 32b scannable register as shown in Figure 5.9. Output shift registers are implemented similarly to input shift registers; however, last outputs are not connected back to the first inputs. Last outputs store the most recently processed 32 outputs. In scan mode, captured data from the secure FPGA core is scanned out one bit at a time. In normal operation mode, captured data is shifted to the right to make room for new data.

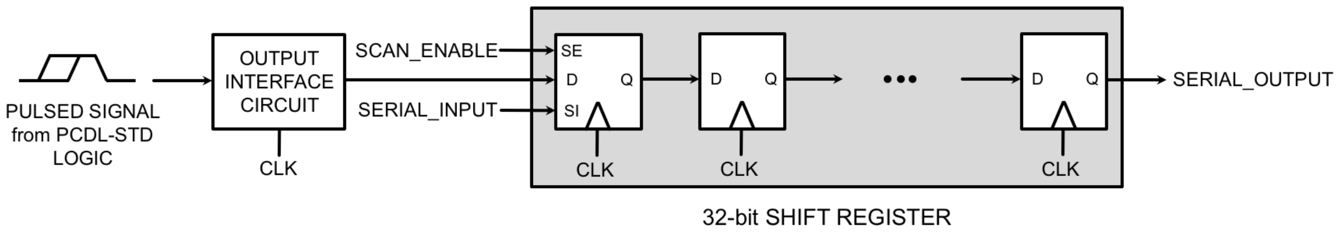


Figure 5.9: Output test structures consist of an output interface circuit and 32-deep scan-enabled shift registers. They store the most recently processed 32 outputs. Output interface circuit captures PCDL-STD signal from the secure FPGA core and convert it to single-ended signal for output registers.

As illustrated in Figure 5.10a, output interface circuit captures PCDL-STD signal from the secure FPGA core and convert it to single-ended signal for output registers. After the data is captured, the latch in the output interface circuitry becomes opaque and cannot capture a new data from the secure FPGA core. In the rising clock edge, the output registers sample and the latch resets to capture a new data. As displayed in Figure 5.10b, control logic orchestrates locking and releasing of the latch with respect to the rising clock edge. A pulsed reset signal is used for a fast release of latch after the output register samples.

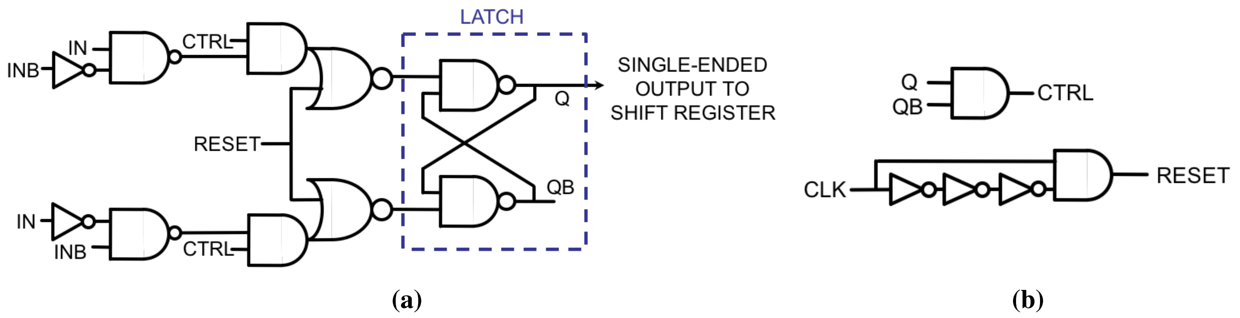


Figure 5.10: (a) Output interface circuits between the secure FPGA and output registers. (b) Control logic orchestrates locking and releasing of the latch with respect to the rising clock edge. A pulsed reset signal is used for a fast release of latch after the output register samples.

Consider the timing diagram of an output test structure as shown in Figure 5.11. Initially, CTRL is high and latch is transparent. Assume that a pulsed input (IN=1) from the secure FPGA core is captured in the latch, and QB goes low. This will also bring CTRL low and lock the latch. In the following rising clock edge, the captured data within the latch is sampled by the output register. This will trigger the RESET pulse generation and reset the latch. The latch becomes transparent again to capture new data from the secure FPGA core.

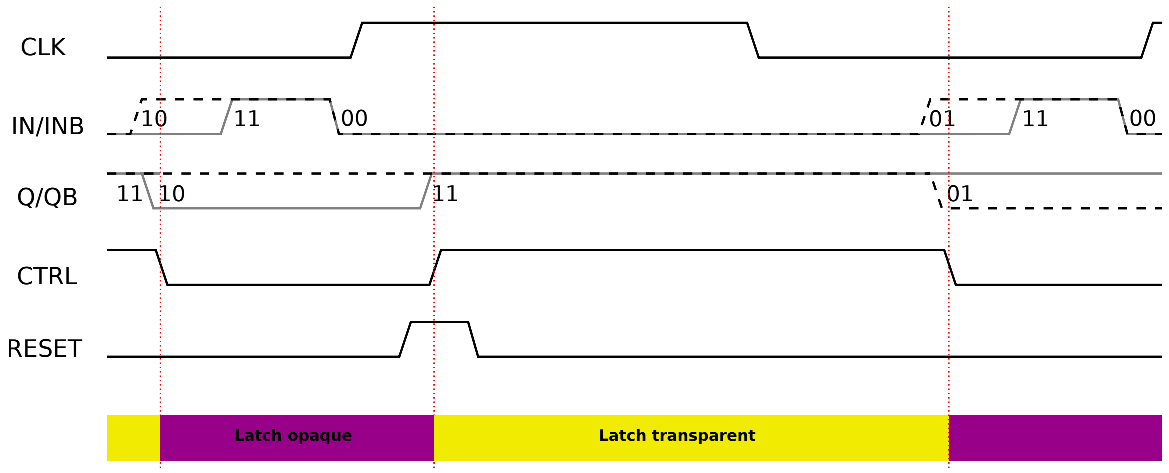


Figure 5.11: Timing diagram of output test structures. The latch in the output interface circuit becomes opaque after capturing the data from the secure FPGA core. In the following rising clock edge, the captured data inside the latch is sampled by the output register, and the latch is released.

5.3.3 Clocking

The chip is clocked by using an on-chip ring-oscillator (RO) based clock-generator, which is shown in Figure 5.12, that can be programmed externally for a wide-range of frequency options (i.e., 500kHz to 5.7GHz). It consists of four ROs to generate the fast base clock (i.e., 2.9GHz, 3.8GHz, 4.6GHz, and 5.7GHz) and a twelve-stage divide-by-2 stages to generate clocks with a wide-range of frequency options. Alternatively, the on-chip clock generator can be bypassed in order to use an external clock supplied from the pads. A fast 14:1 MUX selects the programmed frequency and feeds it to the secure FPGA core via a clock distribution network. Clock is distributed to each tile with a low-skew clock grid that is driven from both the top and bottom of the FPGA core. It is re-buffered within each tile and can be gated for unused tiles to save power.

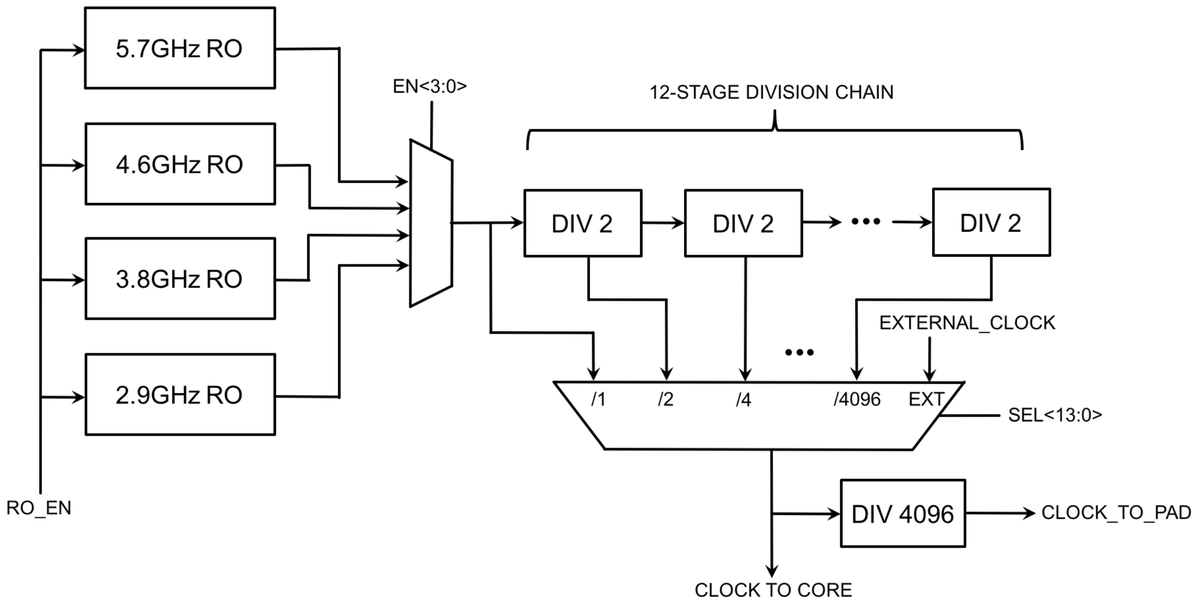


Figure 5.12: Programmable on-chip ring-oscillator based clock generator.

5.4 Measurement Results

We perform two sets of measurements: (1) benchmark measurements to evaluate performance, power and energy consumption, and the energy-efficiency of the secure FPGA core in different operating modes, and (2) PUF measurements to assess the reliability of PUF instances across environmental variations (i.e., voltage and temperature). In the following, we first present the benchmark results and then we discuss the PUF measurement results.

5.4.1 Benchmark Measurements

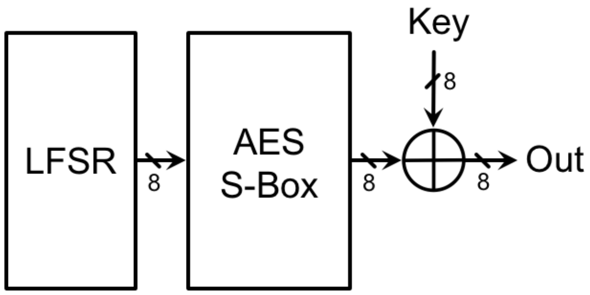
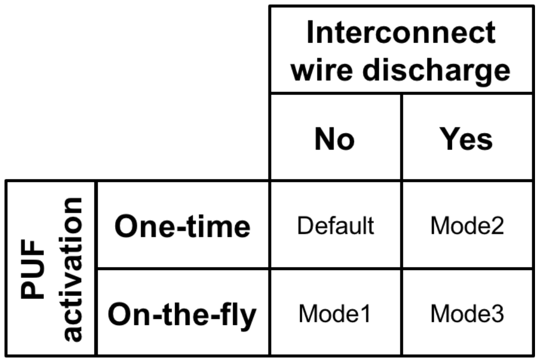


Figure 5.13: Sample security primitive implemented on secure FPGA testchip. It consists of a Linear-Feedback-Shift-Register (LFSR) and an AES S-Box. LFSR generates pseudo-random input patterns and AES S-Box output is XORed with an 8-bit key. A key of 8'hAE (i.e., 174 in decimal) is used in all measurements.

For benchmark measurements, we implement the sample security primitive on secure FPGA testchip. It consists of a Linear-Feedback-Shift-Register (LFSR) and an AES S-Box as shown in Figure 5.13. LFSR generates pseudo-random input patterns and AES S-Box output is XORed with an 8-bit key. A key of 8'hAE (i.e., 174 in decimal) is used in all measurements. We use our VTR-based custom programming tool to map the verilog of the sample security primitive into our secure FPGA. Figure 5.14 displays the operating modes of the secure FPGA testchip.



		Interconnect wire discharge	
		No	Yes
PUF activation	One-time	Default	Mode2
	On-the-fly	Mode1	Mode3

Figure 5.14: Operating modes of secure FPGA testchip. PUF activation can be one-time or on-the-fly and the unconditional wire discharge can be turned off for the interconnect for additional power savings.

Performance

The benchmark circuit runs at 350 MHz at 1V nominal supply voltage and 27°C in one-time PUF firing mode. On-the-fly PUF firing mode trades off performance to increase the security against reverse-engineering attacks on the PUF bits. In this mode, PUFs are fired for a few hundred of ps (i.e., when all LUT inputs are ready) and then reset until the next cycle. Thus, PUF delay adds to the forward logic. The benchmark circuit operates at 290 MHz at 1V nominal supply voltage and 27°C (i.e., 15% performance penalty). Figure 5.15 displays the Shmoo plot across a wide supply range of 0.6-1.3V for both one-time and on-the-fly PUF firing modes. Note that WES mode does not affect the operating frequency (i.e., same performance for both WES=0 and WES=1). The performance is commensurate with commercial FPGAs in the same technology [115, 116].

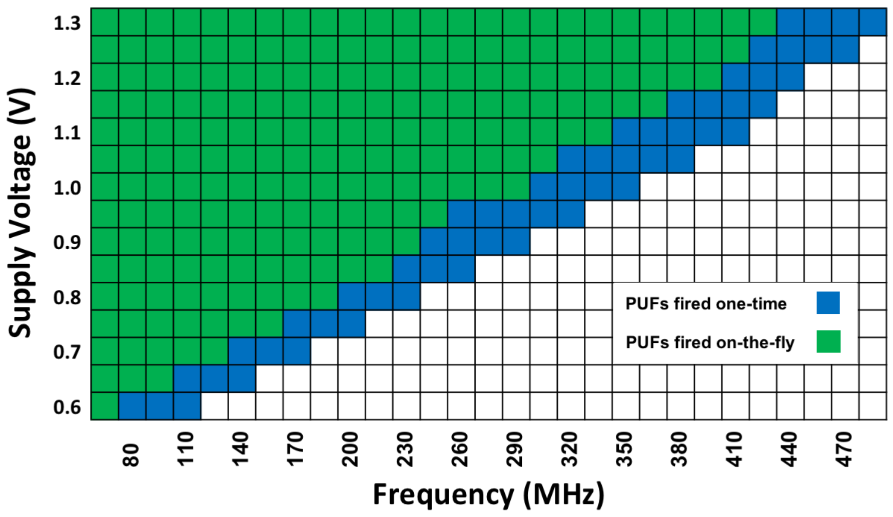


Figure 5.15: Shmoo plot of the benchmark circuit implementation on the secure FPGA. Green and blue areas represent the voltage-frequency points in which the chip is functional when PUFs are fired on-the-fly and one-time, respectively. WES mode does not affect the operating frequency. At 27°C, the benchmark circuit operates at 70-430 MHz and 90-500 MHz in one-time and on-the-fly PUF firing modes, respectively, across a supply range of 0.6-1.3V. At the nominal supply voltage of 1V, the secure FPGA operates at 290 MHz and 350 MHz when PUFs fired on-the-fly and one-time, respectively.

Power and Energy

Measured dynamic power consumption vs. supply voltage at 27°C for all operating modes are shown in Figure 5.16a-5.16b. At 1V nominal supply voltage, the benchmark circuit consumes

574 mW of dynamic power when PUFs are fired one-time and the interconnect discharge is turned on (i.e., Mode2). Turning off the interconnect discharge while firing PUFs one-time (i.e., default mode) saves 11% in dynamic power consumption. In default mode, the benchmark circuit consumes 522 mW of dynamic power.

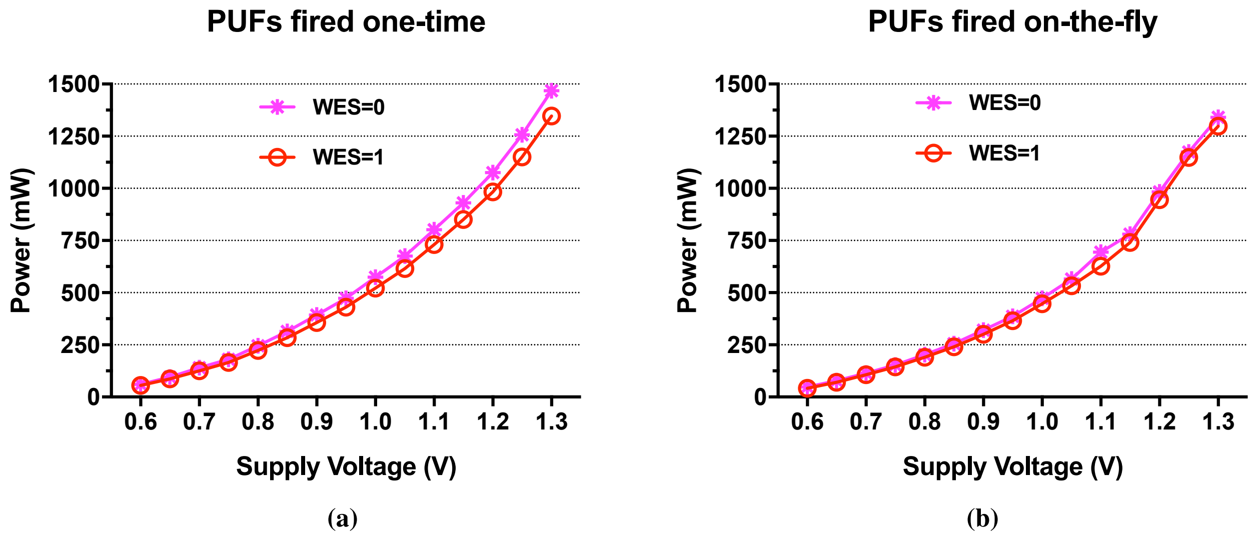


Figure 5.16: Measured dynamic power consumption vs. supply voltage for (a) when PUFs fired are fired one-time and (b) PUFs are fired on-the-fly. WES=0 and WES=1 modes are shown in magenta and red, respectively. Turning off the interconnect discharge saves 11% and 6% in dynamic power consumption when PUFs are fired one-time and on-the-fly, respectively.

Although PUFs consume dynamic power when fired on-the-fly, the benchmark circuit operates at a lower frequency and consumes less power compared to the modes in which PUFs are fired one-time. The benchmark circuit consumes 473 mW of dynamic power when PUFs are fired on-the-fly and interconnect is discharged (i.e., Mode3) at 1V nominal supply voltage. Turning off the interconnect discharge while keeping on-the-fly PUF firing mode (i.e., Mode1) saves 6% in power consumption. In Mode1, the benchmark circuit consumes 447 mW of dynamic power. Clock gating for unused tiles saves in 18% in dynamic power consumption. The leakage power consumption vs. supply voltage is shown in Figure 5.17. At 1V nominal supply voltage, the fabric consumes 70 mW of leakage power.

Figure 5.18a-5.18b shows the average energy consumption for all operating modes. The benchmark circuit consumes 1.5 nJ/cycle of energy in default mode. The energy consumption

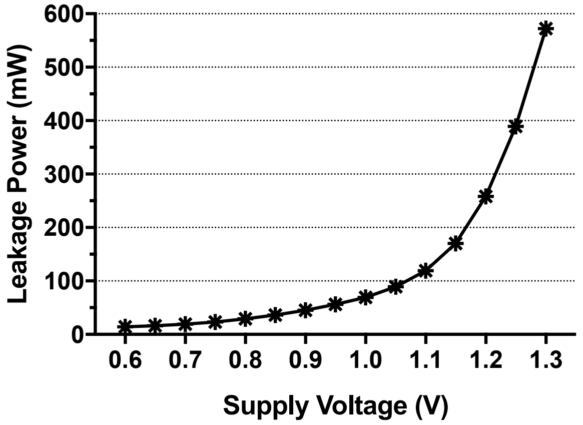


Figure 5.17: Measured leakage power consumption vs. supply voltage at 27°C. At 1V nominal supply voltage, the secure FPGA fabric consumes 70 mW of leakage power.

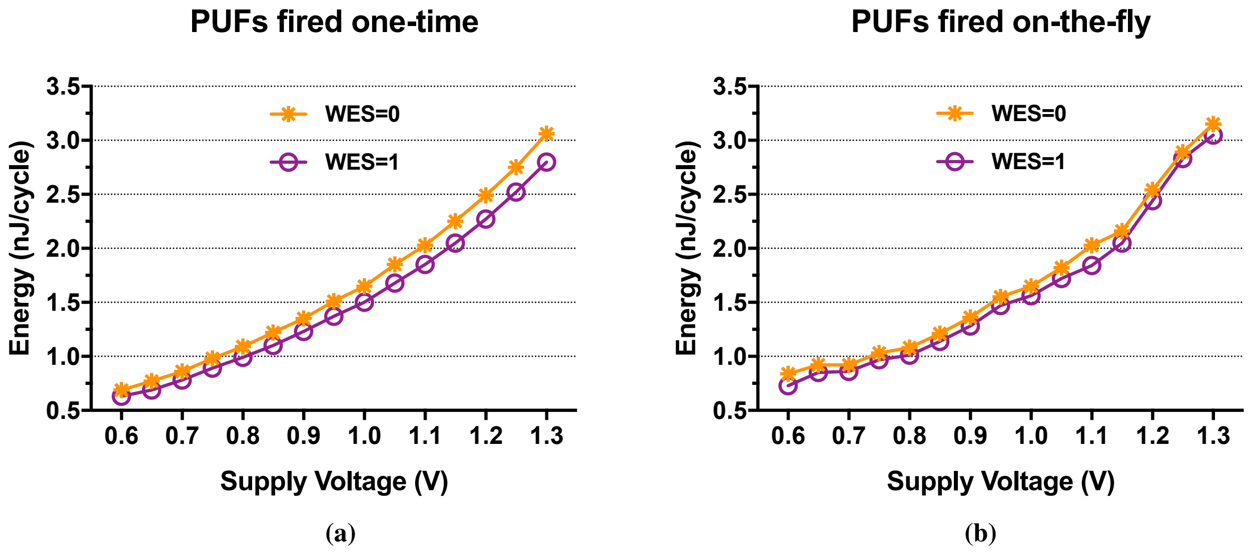


Figure 5.18: Energy consumption per clock cycle for all operating modes. The highest energy (i.e., 1.7 nJ/cycle) is consumed in Mode3, where PUFs are fired on-the-fly and the interconnect is discharged. The secure FPGA consumes the lowest energy in default mode in which PUFs are fired one-time and the interconnect discharge is turned off. WES=0 and WES=1 modes are shown in orange and purple, respectively.

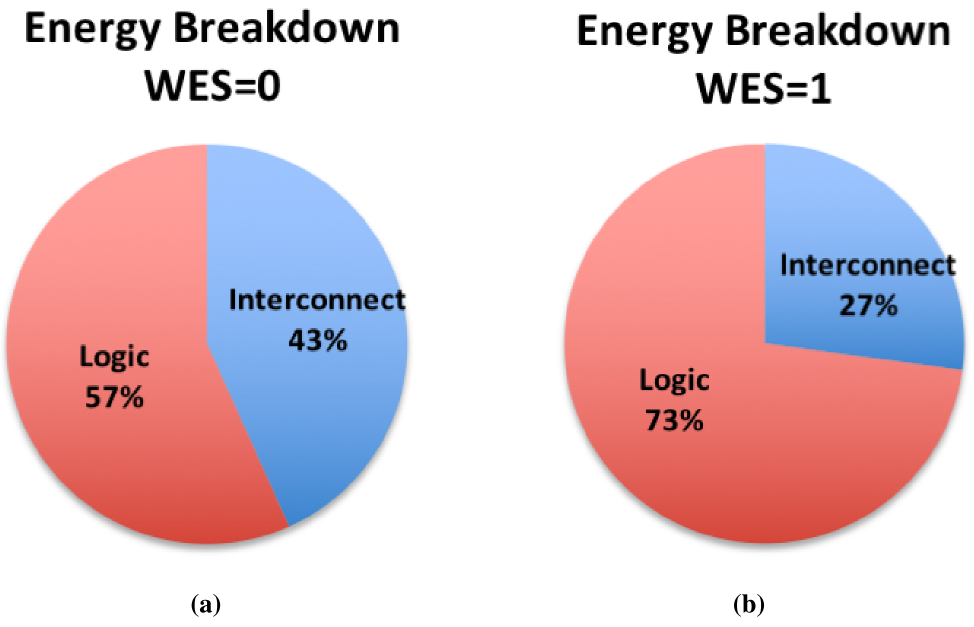


Figure 5.19: Energy consumption breakdown for when PUFs are fired one-time, (a) WES=0 and (b) WES=1. Turning on WES reduces the interconnect energy by 40%, but this only translates to 10% total energy savings due to the LUT energy overheads.

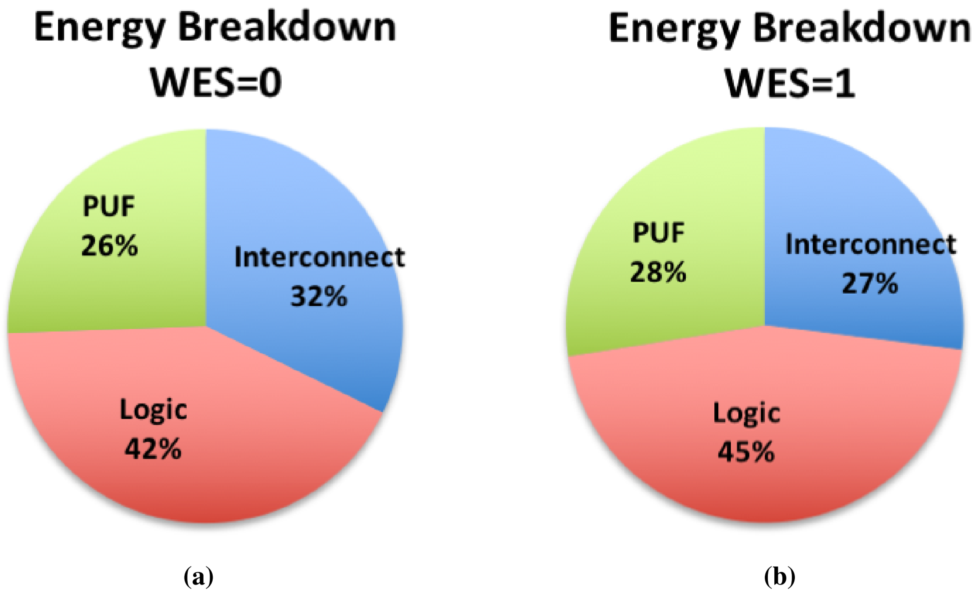
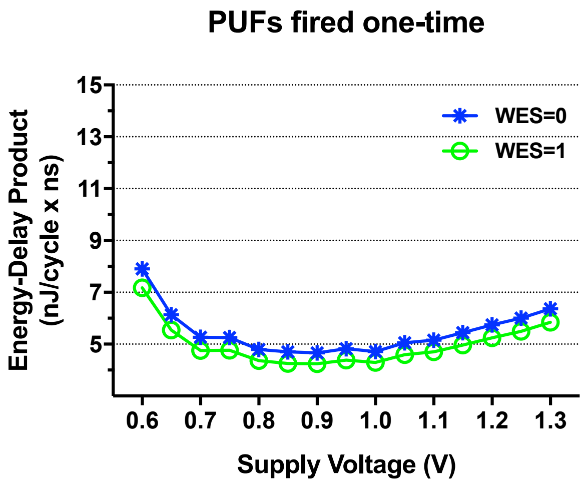


Figure 5.20: Energy consumption breakdown for when PUFs are fired on-the-fly, (a) WES=0 and (b) WES=1. Turning on WES reduces the interconnect energy by 40%, but this only translates to 9.6% total energy savings due to the LUT and PUF energy overheads.

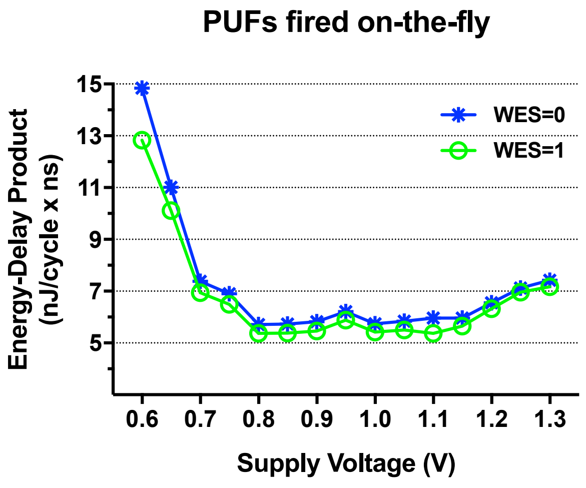
increases to 1.65 nJ/cycle when the interconnect discharge is turned on (i.e., Mode2). Turning off the interconnect discharge (i.e., WES=1) reduces the interconnect energy by 40%, but this only translates to 10% total energy savings due to the LUT energy overheads (Figure 5.19a-5.19b). Alternatively, the benchmark circuit consumes 1.55 nJ/cycle in Mode1, where PUFs are fired on-the-fly and the interconnect discharge is turned off. Finally, the highest energy is consumed when both PUFs are fired on-the-fly and the interconnect is discharged (i.e., 1.70 nJ/cycle energy consumption in Mode3). Turning off the interconnect discharge (i.e., WES=1) reduces the interconnect energy by 40%, but this only translates to an approximately 10% total energy savings due to the LUT and PUF energy overheads (Figure 5.20a-5.20b). While additional energy is consumed to fire PUFs on-the-fly, this mode provides enhanced security against reverse-engineering attacks on the PUF bits.

Energy-Efficiency

In order to find the optimum operating conditions, we evaluate the energy-delay product. Figure 5.21a-5.21b illustrates energy-delay product scaling with supply voltage in different operating modes. The optimal range for energy-delay product is 0.8-1.1V for all operating modes.



(a)



(b)

Figure 5.21: Energy-delay product scaling with supply voltage. The optimal range for energy-delay product is 0.8-1.1V for all operating modes.

5.4.2 PUF Measurements

For measuring the PUF reliability, we perform the characterization as described in the original HCI-SA PUF work [85]. We first perform small-scale measurements to identify the worst-case corner for reliability and then proceed with large-scale measurements on the nominal and the worst case corners.

Small-scale measurements

The following three steps are performed for small-scale measurements of reliability across environmental variations of voltage and temperature as suggested by Bhargava et. al. [80].

1. The first step involves preliminary analysis. We perform 100 PUF evaluations for each possible voltage (i.e., 0.8V, 1V, 1.2V) and temperature (i.e., -20°C, 27°C, 85°C) combination, which results in a sizeable dataset because each PUF evaluation generates 38,400 response bits. We consider the majority vote of 100 responses at the nominal conditions (i.e., 27°C and 1V) as the golden response.
2. The second step includes error testing. We compare the response bits for all other voltage-temperature combinations to those from the golden response. $Error_{i,V_1,T_1}$ is defined as the number of bits out of the 38,400 PUF responses that do not match the golden response for the i^{th} evaluation at voltage=V_1 and temperature=T_1°C.
3. The third step involves determination percent error (i.e., % errors). For all calculations, % errors is defined as the maximum error across 100 evaluations of corresponding variations. In other words, % errors is the largest % of bits that were erroneous for any of the voltage-temperature combinations in any of 100 PUF evaluations performed at that voltage.
   - For a voltage-temperature combination, $Error_{V_1,T_1}$ is defined as the maximum $Error_{i,V_1,T_1}$.
   - For voltage-only variations, $Error_{V_{only}}$ is the maximum error at all voltage variations and at nominal temperature (i.e., the maximum of $Error_{V_{0.8},T_{27}}$, $Error_{V_1,T_{27}}$, and

$Error_{V_{1.2},T_{27}}$).

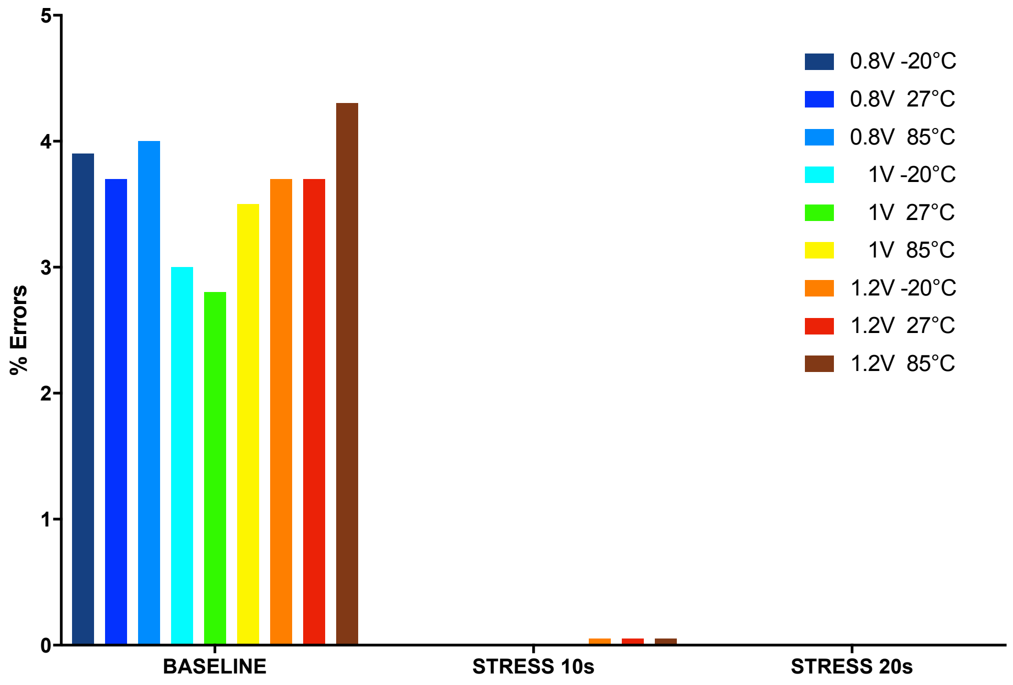
- For temperature-only variations, $Error_{T_{only}}$ is the maximum error at all temperature variations and at nominal voltage (i.e., the maximum of $Error_{V_1,T_{-20}}$, $Error_{V_1,T_{27}}$, and $Error_{V_1,T_{85}}$).
- Overall $Error_{V\&T}$ is the maximum error at all voltage and temperature combinations.

The measurements are performed for the die before and after different stress durations (i.e., 10s intervals). Figure 5.22a shows the improvement in reliability for improved HCI-SA PUFs with different stress durations (i.e., as decrease in the % error). As seen in Figure 5.22b, the overall errors across all voltage-temperature combinations reduce from 4.4% to 0.05% when stressed for 10s. After a total 20s of stress, there are no errors for any of the 38,400 PUF instances across all of 100 evaluations for each voltage-temperature combination.

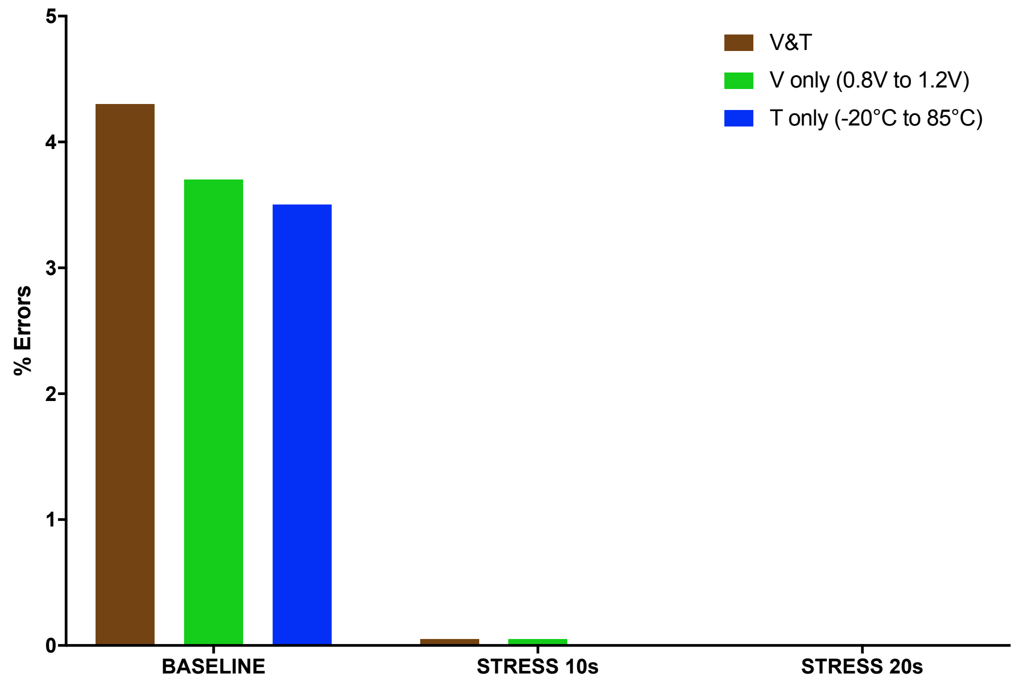
Figure 5.22b shows that variations in voltage have a slightly stronger impact on reliability compared to temperature variations. For voltage-only variations and temperature-only variations, the % errors is reduced from 3.8% to 0.05% and from 3.4% to 0% with a stress of 10s, respectively. After a total stress of 20s, there are no errors. The highest % errors is observed at high-voltage and high-temperature combination (i.e., 1.2V and 85°C).

Large-scale measurements

Following the identification of the worst-case corner for reliability (i.e., 1.2V and 85°C), we conduct large-scale experiments at the worst-case corner to experimentally measure bit error rate (BER). After 872,000 measurements at the worst-case corner and 36 days of continuous testing, no errors were observed. A very conservative assumption that the very next measurement would be an error leads to a bit-error-rate (BER) $< 2.98*10^{-11}$, which is on par with the theoretical BER targeted by the ECC in the commercial PUFs (i.e., Intrinsic ID SRAM PUF BER $< 3.9*10^{-12}$ [117]) and close to the reported BER of SRAMs at this technology node (i.e., BER for ST SRAMs in 65nm node $< 1.66*10^{-12}$ [118]). The total error rate (TER) of an N-bit key can be defined as TER = 1 - (1 - BER)^N [85]. In our secure FPGA testchip, LUT bits are protected with



(a)



(b)

Figure 5.22: (a) Reliability of improved HCI-SA PUFs shown as a percentage of errors (100 - % reliability). Errors shown are the maximum errors across 100 evaluations. (a) Errors across all the environmental conditions. They are measured for voltage varions ±20% from nominal 1V and temperatures of -20°C, 27°C, 85°C. (b) Errors across only voltage, only temperature, and all voltage & temperature variations.

a total of 38,400 LUT bits, hence, the TER for LUT-entanglement is less than 1.15*10^{-6}.

Reliability of LUT-entanglement vs. Commercial FPGAs

In commercial FPGAs, failure rates are typically expressed in failure-in-time (FIT) units. 1 FIT is equivalent to 1 failure in 1 billion or 10^9 device-hours [119, 120]. Devices operate under high stress conditions (i.e., high temperature and high voltage) for failure rate calculations. In addition, the reported failure rates do not distinguish LUT configuration bits from other bits (i.e., interconnect). However, commercial FPGAs typically have a few thousand to hundred thousand LUTs [121, 122]. Therefore, the reported failure rates are conservative in comparison to PUF bits that are used for hardware-entanglement (i.e., LUT-entanglement). Failure rates around 8-10 FIT are reported for 65nm Xilinx and Intel FPGAs [119, 120]. For example, 10 failures expected out of 1 million components operating for 1,000 hours have a failure rate of 10 FIT. In other words, if we had 1 million PUF instances running for 1000 hours under high voltage and high temperature, then a TER of 10*10^{-9} would be expected.

While we do not have 1 million PUF instances implemented on our testchip, we experimentally show that no error is detected after 872,000 measurements (i.e., BER $<$ 2.98*10^{-11}). This BER is on par with the theoretical BER targeted by the ECC in the commercial PUFs and close to the reported BER of SRAMs at this technology node. The source of reliability for HCI-SA PUFs is the V_{TH} shift (i.e., as a result of 20s of HCI stress) on the stressed transistor. A significant percentage of the V_{TH} shift has been shown to be permanent with device aging [34, 85]. Bhargava et. al. [80, 85] reported that a reversal in V_{TH} shift saturates around 30 mV mean value beyond 1.7 years of aging (i.e., 1.5V and 100°C). This level of saturation constitutes a very small portion of the overall V_{TH} shift (i.e., up to 300mV of V_{TH} shift). All of our measurements thus far are in the good direction without any error and we have not seen anything that suggests otherwise.

Uniqueness and Randomness of improved HCI-SA PUF

Uniqueness is a measure of how uncorrelated the bits are across chips. Ideally, the response bits across chips differ with a 0.5 probability. The hamming distance (HD) of a k-bit response from ideally unique chips follows a binomial distribution with parameters N=k, p = 0.5, a mean of k/2 [85]. In the secure FPGA, 64 PUF bits at each LUT are layed out as groups of 16 and physically placed in the corners of the LUT. Therefore, we group PUF response bits as 16-bit words to analyze the uniqueness of HCI-SA PUFs. We create 2,400 16-bit words from the measured outputs of 38,400 PUF bits on 3 chips. These 16-bit words are generated at 27°C and 1V after the HCI-SA PUFs are stressed for 20s. Figure 5.23 shows that the pairwise HDs of response bits from 3 chips are close to ideal (i.e., mean of 8) with means of 7.94, 8.02, and 7.96.

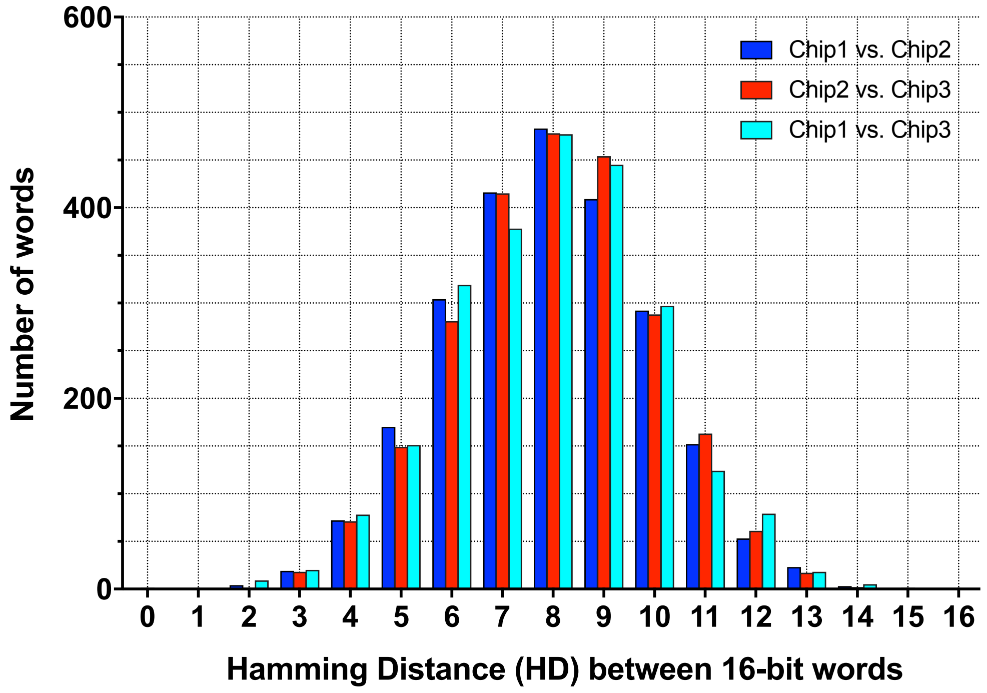


Figure 5.23: Histogram of Hamming distance (HD) of 16-bit HCI-SA PUF response words from 3 chips. For HD calculations, PUF response bits from 38,400 HCI-SA PUFs on each chip are grouped to create 2,400 16-bit words. The pairwise HD of response bits from 3 chips is close to ideal (i.e., mean of 8) with means of 7.94, 8.02, and 7.96.

Randomness is a measure of the unpredictability of the PUF response bits. The %1's and %0's in the response bits are ideally equal. From the measured responses of 38,400 HCI-SA PUF bits across 3 chips after 20s of stress, the %1's are found to be 48.6%, 48.8%, and 49.1%, which are very close to ideal (i.e., 50%). PUF response bits also pass the NIST randomness tests with minimum P value of 0.976.

Programmability of HCI-SA PUFs

Based on the previous data, we can reach a very low BER ($< 2.98*10^{-11}$) after an initial 20s HCI response reinforcement. In the following experiment, we investigate the programmability of HCI-SA PUFs. Instead of the initial 20s stress to increase the reliability, we applied an HCI stress in the opposite direction to flip all PUF response bits (i.e., to write the opposite of the golden response bits into PUFs). We find that 30s of stress is sufficient to flip all PUF response bits (i.e., write the opposite of the golden response bits into PUFs) reliably across voltage (i.e., 0.8V, 1V, 1.2V) and temperature (i.e., -20°C, 27°C, 85°C) variations.

In order to analyze re-programmability of HCI-SA PUFs, we started with initially stressed PUF bits (i.e., initial response reinforcement with a 20s of stress). Then, we applied longer stress (i.e., 20s, 40s, 100s, 300s, 600s) at 27°C and 1V in the direction to flip the PUF outputs (i.e., to write the opposite of the golden response). Figure 5.24 shows that increasing stress time has diminishing returns beyond 100s of stress. In other words, V_{TH} shift will be less with the increasing stress time due to HCI saturation and any further stress to increase V_{TH} of one transistor will not decrease the number of errors after 600s of total stress. Therefore, not every single one of PUF bits can be flipped with high enough reliability (i.e., 100% reliability). There are some outliers that are likely to be saturated in the initial side. For these outliers, 20s of initial stress is too much and a smaller stress (i.e., $<$ 20s) in the response reinforcement step is required.

If we want to re-program all the PUF bits with 100% reliability, the minimum amount of stress necessary to get the target BER should be applied. Any more stress than the required amount will hurt re-programmability. Due to this trade-off between the initial reliability and re-programmability, a more relaxed BER target makes re-programmability easier. Alternatively, instead of applying HCI-stress in a preferred direction to keep increasing V_{TH} of one transistor, techniques like charge de-trapping [123] can be explored to reverse the effect of HCI on the other

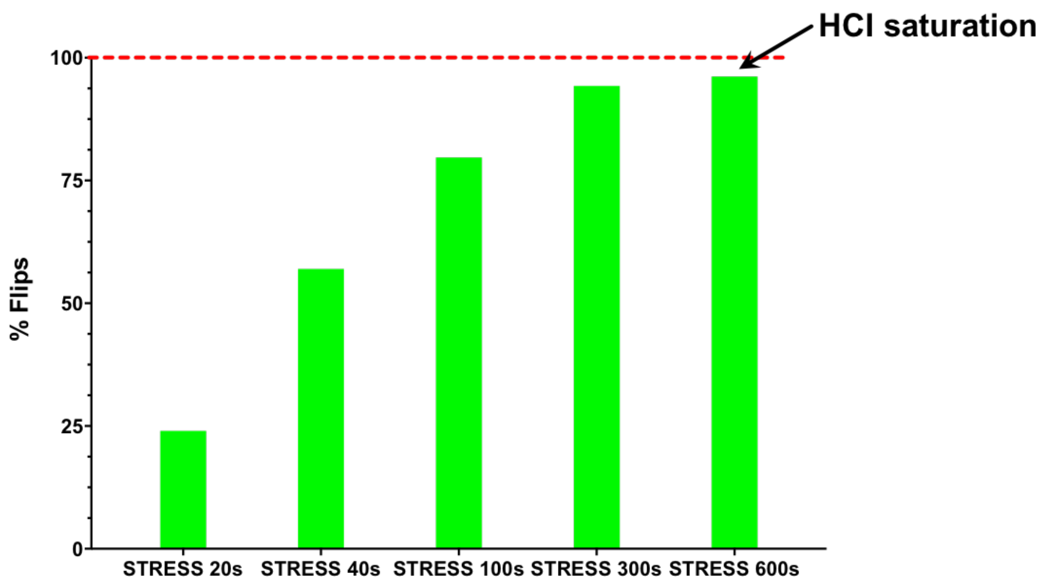


Figure 5.24: Re-programming HCI-SA PUFs (i.e., writing the opposite value of the golden response) with via an HCI stress in the opposite direction. After the 20s HCI response reinforcement at 27°C and 1V (i.e., no errors), PUFs cannot be re-programmed (i.e., their response cannot be flipped) due to HCI saturation. The number of errors cannot be decreased with further stress.

transistor (i.e., cancel out the V_{TH} increase by the previous programming cycles). Consequently, the programmability of HCI-SA PUFs can be improved.

5.5 Power Side-Channel Analysis Security Evaluation

In order to evaluate power side-channel analysis (SCA) resistance of the secure FPGA fabric in different mode operations, we mount a DPA attack on the sample security primitive that we used for the benchmark measurements. This structure is the similar device-under-attack (DUA) that was used in the original D-WDDL work [16, 110]. The DUA is a simplified version of a structure found in many block ciphers, including AES. LFSR generates pseudo-random input patterns and AES S-Box output is XORed with an 8-bit key. A key of 8'hAE (i.e., 174 in decimal) is used in all measurements.

5.5.1 Attack Methodology

The procedure for the DPA attack mounted on D-WDDL countermeasure is described in [16, 110]. We mount an identical DPA attack on our secure FPGA fabric to assess the power SCA resistance. In this attack, the power consumption of DUT is estimated by using a power model that combines the observed output values with an estimate for the key value. Then, the estimated power for each key guess is correlated with the measured power. The correct key will have highest correlation across all key guesses. The power model is the Hamming Weight (HW) of a single bit of the AES S-Box input [16]. Then, the correlation plot for each of the 8-bits is obtained and the maximum in each plot is identified. Majority voting among all bits is used to find the final key.

$$HW(bit(IN,i)) \mid_{i=0\ldots 7} = HW(bit(SBOX^{-1}(OUT \otimes KEY_{guess}), i)) \mid_{i=0\ldots 7} \tag{5.1}$$

Yu et al. obtained 20 sample points per clock cycle and used the 10th sample point for the attack [16, 110]. In order to increase the attack success, we instead use 800 points for each clock cycle (i.e., a total of 256 clock cycles). Then, we measure and save the power traces for DUA for all possible 8-bit input combinations. Due to LFSR, we obtain a complete power trace (i.e., one for each input) in 256 clock cycles. Furthermore, we repeat the measurements 100 times (i.e., collect 100x 256-cycle power traces) to average out the noise element. We also repeat the DPA attack across all different mode of operations in order to demonstrate the power-SCA resistance in each mode. For instance, the operation modes that have the interconnect discharge turned off (i.e., default and Mode1) trade off the security with energy overhead. By mounting the attack in these modes, we evaluate the effectiveness of the discharge operation for the interconnect wires.

5.5.2 Attack Equipment and Setup

We used an Agilent 548559A digital sampling oscilloscope, which has a bandwidth of 6 GHz with a maximum sampling rate of 20 GS/s (i.e., Gigasamples per second). To obtain enough samples, the DUA is clocked at 1 MHz. We bypass the on-chip programmable clock generator and use an external clock from the Ni-DAQ 6259 board. The same external clock as the trigger signal is used for the oscilloscope to register the power traces. Figure 5.25 shows the measurement and test setup for the DPA attack mounted on the secure FPGA fabric.

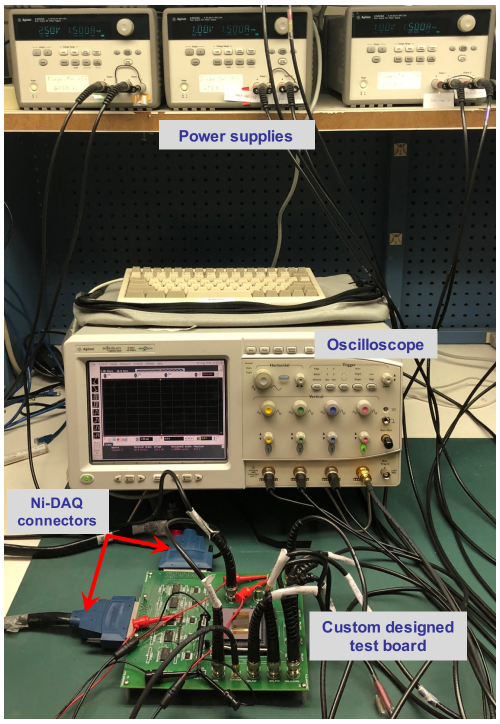


Figure 5.25: Measurement and test setup for the DPA attack mounted on the secure FPGA fabric.

The power traces are measured on a custom designed test board. Note that there are separate voltage domains on the secure FPGA testchip. We only need to measure the power consumption variation for secure FPGA fabric that was powered by VDD_CORE supply. We remove all the decoupling capacitors on our custom test board in order to maximize the success of detecting the variations in the power consumption across different inputs.

We use a small resistor (i.e., $2\,\Omega$) in series with the core power supply for measuring the current drawn by the DUA as shown in Figure 5.26. We carefully pick the value of the resistor because it creates negative feedback that interferes with the secure FPGA operation. When the current (i.e., I) flowing through the resistor increases, the corresponding voltage drop (i.e., ΔV) also increases. Since the power supply is fixed at a certain value (i.e., 1V), the voltage at the secure FPGA fabric also decreases proportionally. Consequently, the current drawn by the fabric will decrease. The bigger resistor is preferable since it causes bigger voltage variations that can produce enough ΔV for the signal to be detected by the oscilloscope. However, a very large ΔV will prevent the correct operation of the secure FPGA. Hence, we use a 2Ω resistor in our measurements. We convert the measured ΔV to the current by dividing by 2. In the following, we discuss the DPA attack results for all operation modes.

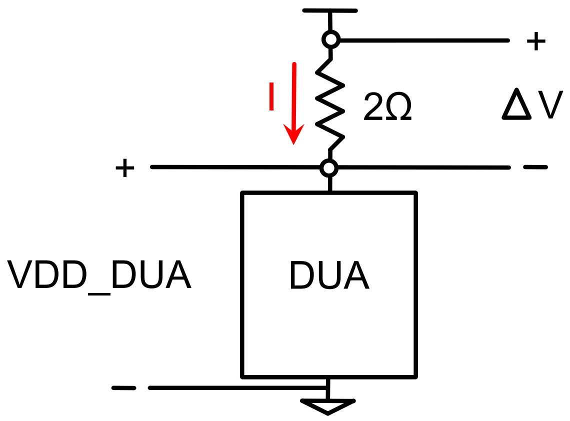


Figure 5.26: Measuring the DUA current with a small resistor (i.e., 2Ω). The measured voltage drop (i.e., ΔV) is converted to the current by dividing by 2.

5.5.3 Attack Results

Figure 5.27 shows the measured power trace for Mode1, where the PUFs are fired on-the-fly and the interconnect discharge is turned off. The power traces for the other operation modes can be found in Appendix A. Note the current peaks (highlighted in blue) and the corresponding voltage dips on the supply of the secure FPGA fabric. The DUA evaluation window is highlighted in light green. Only 4 cycles in a 255-cycle period (i.e., all LFSR inputs) for the 20^{th} power trace measurement is shown for simplicity.

We mount the DPA attack using the first 200 sample points out of 800 points in every clock cycle (i.e., 200 attacks). We repeat the attacks across all modes of operation (i.e., 800 attacks in total). Figure 5.28a-5.28h shows the correlation plots vs. key guesses (in decimal) for the DPA attack mounted in Mode1 on each individual bits (i.e., from Bit-0 to Bit-7). Key guesses with the

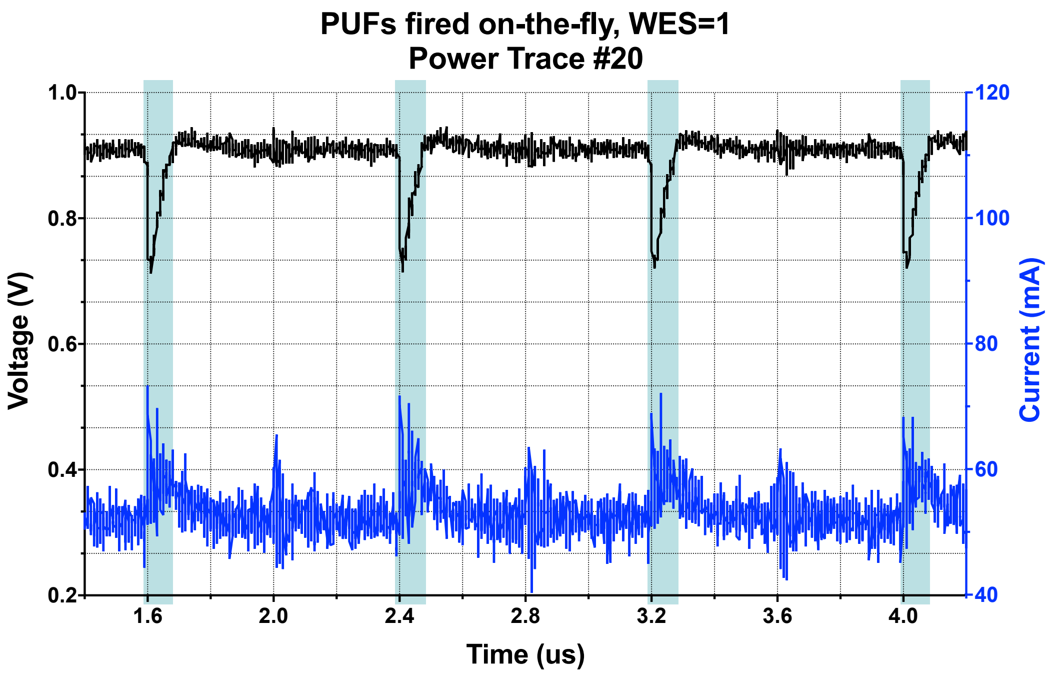
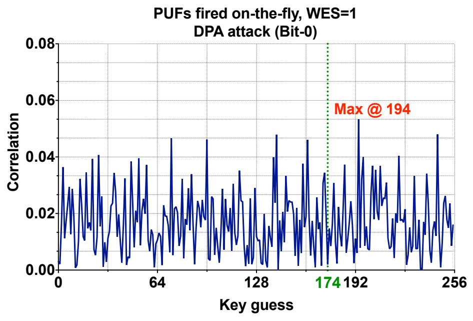
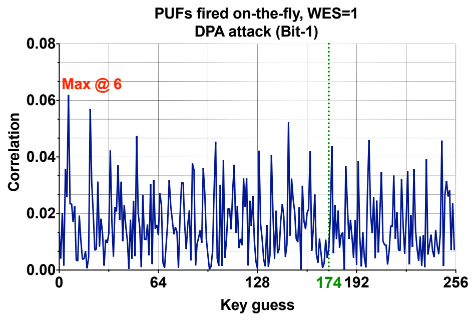


Figure 5.27: Measured power trace for Mode1, where PUFs are fired on-the-fly and the interconnect discharge is turned off. The current (highlighted in blue) that is drawn by the secure fabric is measured through a 2Ω resistor in series with the supply power. 4 cycles in a 255-cycle period (i.e., all LFSR inputs) for the 20^{th} power trace measurement are shown. The DUA evaluation windows are highlighted in light green.

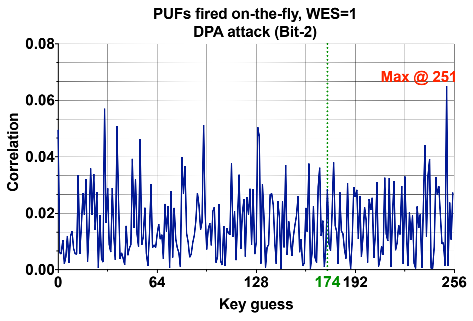
highest correlation are highlighted in red. The key value 174 used in the DUA is highlighted in green. The correlation plots for the other operation modes can be found in Appendix A.



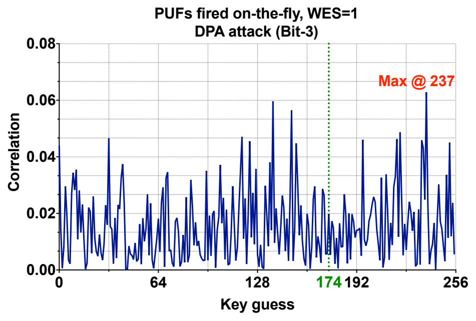
(a)



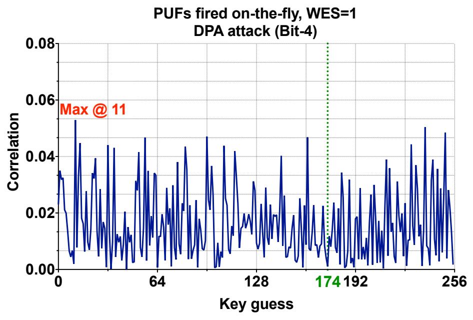
(b)



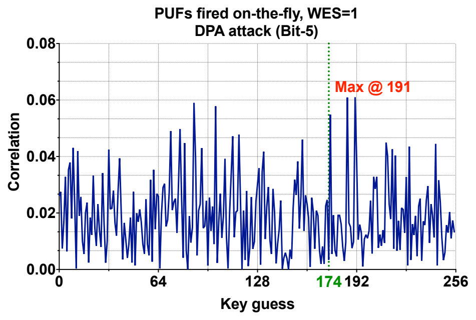
(c)



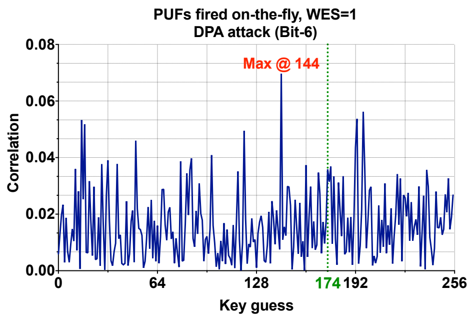
(d)



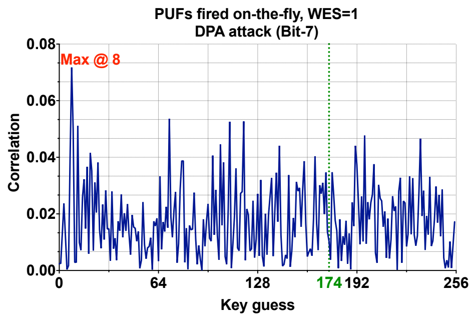
(e)



(f)



(g)



(h)

Figure 5.28: DPA attack in Mode1, where the PUFs are fired on-the-fly and the interconnect discharge is turned off. Correlation plots vs. key guesses (in decimal) for DPA attack on (a) bit-0, (b) bit-1, (c) bit-2, (d) bit-3, (e) bit-4, (f) bit-5, (g) bit-6, (h) bit-7 are shown. Key guesses with the highest correlation are highlighted in red. The key value 174 (highlighted in green) is used in the benchmark circuit.

In each of these DPA attacks, the extracted key values have very small correlations with the measured power variations and are random with no single outstanding value that dominates the other key guesses. For instance, the attack on bit-7 has the maximum correlation of 0.074 at the key value 8. The maximum correlation values for all other bits are smaller and correspond to different key values. Thus, each bit leads to a different key guess.

Table 5.2: Correlation analysis results for the DPA attacks on each bit, where PUFs are fired on-the-fly and WES=1.

Correct key = 174	**Bit-0**	**Bit-1**	**Bit-2**	**Bit-3**	**Bit-4**	**Bit-5**	**Bit-6**	**Bit-7**
Max. correlation	0.0533	0.0619	0.0650	0.0628	0.0529	0.0609	0.0696	0.0717
Min. correlation	0.00032	0.00006	0.00024	0.00011	0.00007	0.00003	0.00010	0.00009
Avg. correlation	0.0168	0.0161	0.0159	0.0158	0.0160	0.0188	0.0161	0.0176
Extracted key	194	6	251	237	11	191	144	8
Correlation of correct key	0.0344	0.0107	0.0009	0.0056	0.0083	0.0225	0.0096	0.0346
Rank of correct key	23	156	244	188	169	88	165	27

Furthermore, the correlation values for the actual key (i.e., 174) are much smaller than the highest correlation values. Compared to key guesses with the highest correlations, the actual key has also much lower ranks (Table 5.2). Thus, none of the key guesses will be the actual key even if the attacker uses the first couple of the highest correlation values.

The extracted key values for each bit value and across all operation modes, which correspond to the attack on the sample point with the highest correlation value, are summarized in Table 5.3. Each entry represents the extracted key for the particular DPA attack (i.e., different bit and mode of operation). In every mode, the attack on each bit leads to a different key in the correlation process and all of these keys are equally likely. Hence, the secure FPGA withstands the DPA attack in each mode of operation.

Table 5.3: DPA attack results for each operating mode. Each entry represents the extracted key for the particular DPA attack. The extracted key values are random with no single outstanding value that dominates the other keys.

Operation Mode	**Bit-0**	**Bit-1**	**Bit-2**	**Bit-3**	**Bit-4**	**Bit-5**	**Bit-6**	**Bit-7**
Default	110	3	2	121	41	4	101	95
Mode1	194	6	251	237	11	191	144	8
Mode2	12	196	2	181	215	170	221	94
Mode3	235	140	19	1	29	144	253	159

5.6 Summary

In this chapter, we discussed the details of our prototype hardware-entangled secure FPGA testchip. We used an AES S-Box security primitive to benchmark the characteristics of the FPGA fabric. We evalulated performance, power and energy consumption, and the energy-efficiency of the secure FPGA core in different operating modes. We mounted a DPA attack to assess the power side-channel resistance of our secure FPGA fabric for all modes of operation. We presented the PUF measurements to assess the reliability of PUF instances across environmental variations (i.e., voltage and temperature).

In summary, we demonstrated a functional hardware-entangled secure FPGA prototype testchip and showed that the secure fabric has significant power SCA resistance without the interconnect discharge due to well-balanced differential wires (i.e., regularity of FPGA). Hence, the discharge operation could be avoided to save energy. We also demonstrated a very compact HCI-SA PUF design that achieved a very high reliability (i.e., bit-error rate $< 2.98*10^{-11}$) without any ECC, which is on par with the theoretical BER targeted by the ECC in the commercial PUFs and close to the reported BER of SRAMs at this technology node. HCI-SA PUFs can be programmed one-time due to HCI saturation. The programmability can be improved by HCI reversal techniques to cancel out V_{TH} increase from the previous programming cycles.

Chapter 6

Conclusion

The current trend in FPGA security is to address the security concerns at different levels of abstraction including software/algorithm, RTL, and hardware. For design protection, FPGA manufacturers implement various countermeasures within the standard FPGA design flow at both software/algorithm and hardware levels (i.e., bitstream encryption). However, these security measures are vulnerable to a variety of attacks. The fundamental problem is that at some level of the hardware, the design configuration data is stored in the clear. Alternatively, secure operation is mainly addressed at the RTL level with layered countermeasures on an unsecured conventional-off-the-shelf (COTS) FPGA. FPGA designers are restricted by flow and integration problems that limit the available countermeasures for secure FPGA operation. Additionally, the countermeasures at their disposal have significant overheads in area, performance, and energy.

In this work, we took a different approach. Instead of mitigating the vulnerabilities of COTS FPGA (i.e., to secure the FPGA), we proposed a secure FPGA design, wherein the hardware security features are implemented within the FPGA itself. In order words, we altered the FPGA fabric itself have built-in security features. We protect the bitstream by hardware-entangling the configuration deep within the hardware with a secret die-specific response. Consequently, the configuration data is stored encrypted at each level of hardware including the configuration storage on the FPGA die (i.e., the lowest level of hardware). The configuration data is decrypted on-the-fly for a very brief moment of time (i.e., a few hundred ps) during the FPGA operation, then stored in an encrypted form until the next time it is used. Hardware-entanglement technique relies on a random secret response in order to encrypt the configuration bitstream. In the current implementation, we used physical unclonable functions (PUF) to generate this random secret response. We essentially used PUFs to generate a very wide encryption key that is distributed across the FPGA die. Hence, the hardware-entanglement technique is valid regardless of the *key* generation method. Furthermore, we addressed the most common SCA vulnerability (i.e., power-SCA) through use of a novel power-SCA resistant logic embedded within the fabric. Thus, the proposed FPGA was found to be inherently secure by-design.

Silicon results from our prototype secure FPGA prove the feasibility of the secure FPGA. We showed the viability of the hardware-entanglement scheme and demonstrated significant improvements over the original HCI-SA PUF design in terms of both reliability and compactness. In addition, we explored the programmability feature of HCI-SA PUF design. As a result, our prototype is the first-ever-reported FPGA that is designed specifically for secure operation. Despite the engineering effort, we show that there are significant security benefits with secure FPGA and the corresponding overheads are much lower compared to existing countermeasures. Thus, secure FPGA is a very promising alternative to layering countermeasures on top of insecure COTS FPGAs.

Future Work

The ideas presented in this thesis are open to further exploration, improvement, and new directions. The reliability is the one the main challenges for PUF designs. We emprically demonstrated a very low bit error rate with HCI response reinforcement in a planar 65nm CMOS process. However, the viability of HCI response reinforcement is yet to be tested in more advanced technology nodes (i.e., sub-28nm) and for different device structures (i.e., FinFET).

Although HCI-PUFs can be initially programmed, their re-programmability is restricted due to HCI saturation. Instead of applying HCI-stress in a preferred direction to write into HCI-SA PUFs, techniques like charge de-trapping [123] can be explored to reverse the effect of HCI from the previous programming cycles. Consequently, the programmability of HCI-SA PUFs can be improved.

Instead of using PUFs to generate random secret response for hardware-entanglement, we can store responses in a non-volatile memory (NVM). The emerging embedded NVM technologies with high density, low energy, and high endurance can yield to more efficient hardware-entangled implementations.

Security comes at a price. In the case of secure FPGAs, the price is relatively low compared to the existing countermeasures. While some users are willing to pay the price for security-critical applications, others have different design goals. For instance, some mobile applications may prioritize area and energy efficiency over the security. In order to decrease the overheads and make secure FPGAs more attractive for further applications, the concept of *heterogeneous* secure FPGAs can be explored. The secure fabric can be implemented in a dedicated region on FPGA die and this secure section can be allocated for security-critical applications. All other applications will be mapped to the conventional unsecured fabric. Alternatively, 2.5D or 3D integration technologies can be used to integrate a dedicated secure FPGA die with larger conventional unsecured FPGA dies.

Appendix A

DPA Attack Results For All Modes of Operation

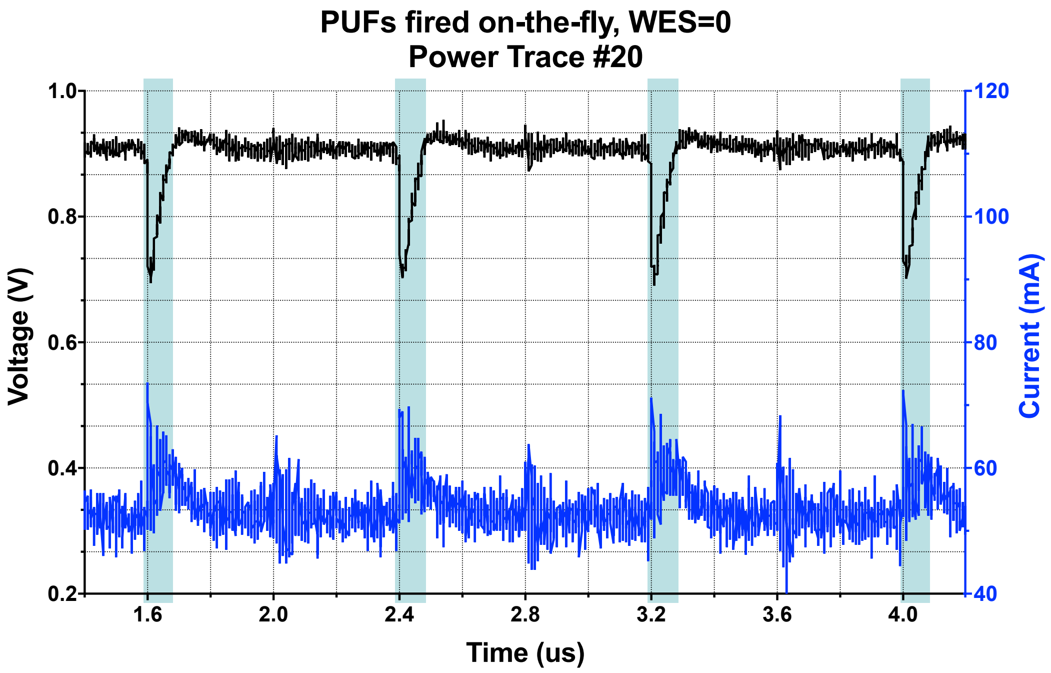


Figure A.1: Measured power trace for Mode3, where PUFs are fired on-the-fly and the interconnect discharges. The current (highlighted in blue) that is drawn by the secure fabric is measured through a $2\,\Omega$ resistor in series with the supply power. 4 cycles in a 255-cycle period (i.e., all LFSR inputs) for the 20^{th} power trace measurement are shown.

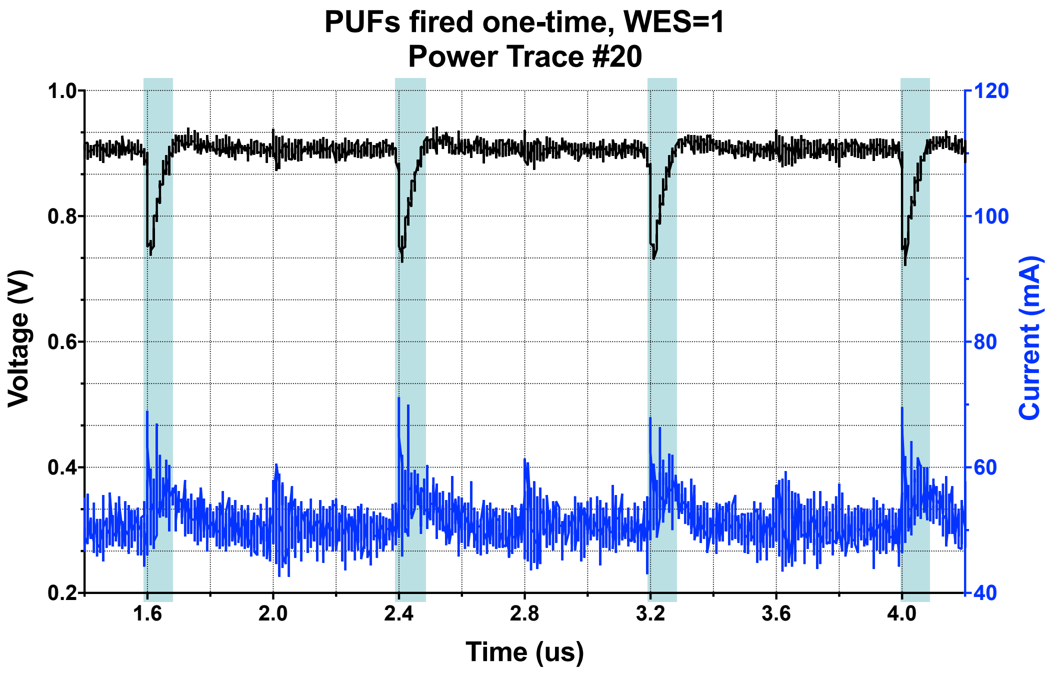


Figure A.2: Measured power trace for the default mode, where PUFs are fired one-time and the interconnect discharge is turned off. The current (highlighted in blue) that is drawn by the secure fabric is measured through a $2\,\Omega$ resistor in series with the supply power. 4 cycles in a 255-cycle period (i.e., all LFSR inputs) for the 20^{th} power trace measurement are shown.

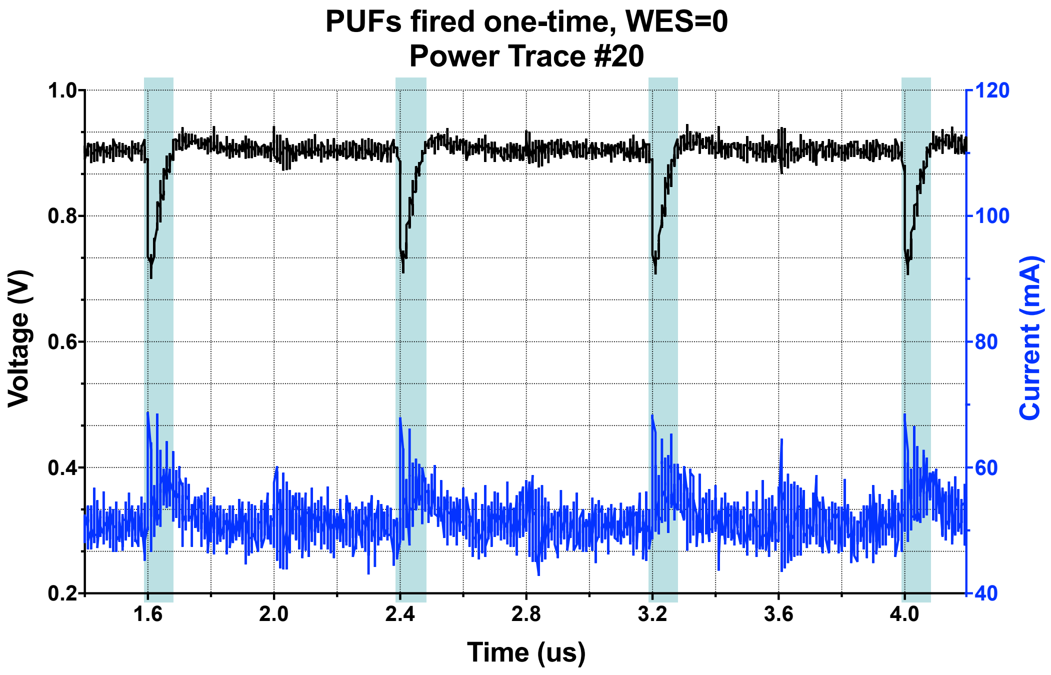
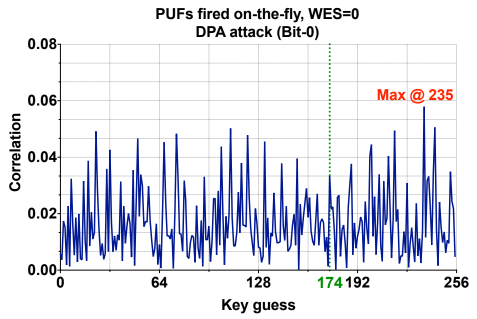
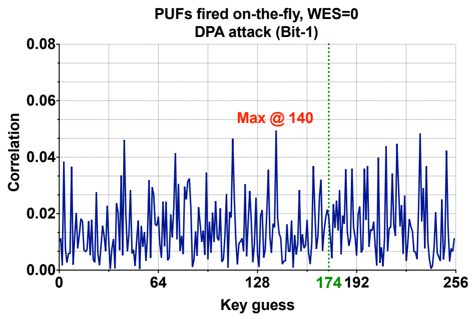


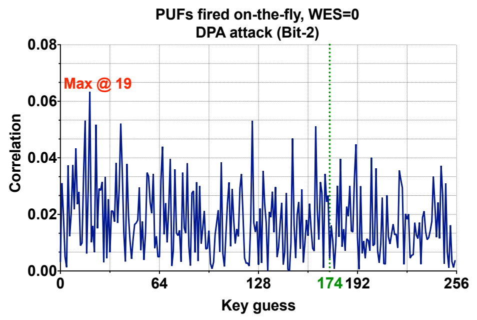
Figure A.3: Measured power trace for Mode2, where PUFs are fired one-time and the interconnect discharges. The current (highlighted in blue) that is drawn by the secure fabric is measured through a $2\,\Omega$ resistor in series with the supply power. 4 cycles in a 255-cycle period (i.e., all LFSR inputs) for the 20^{th} power trace measurement are shown.



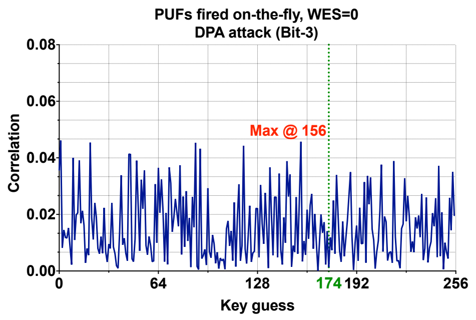
(a)



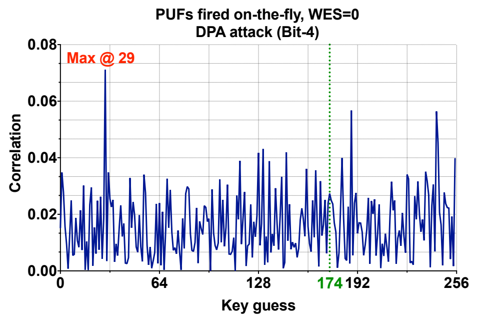
(b)



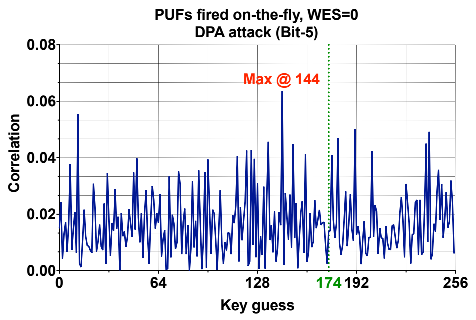
(c)



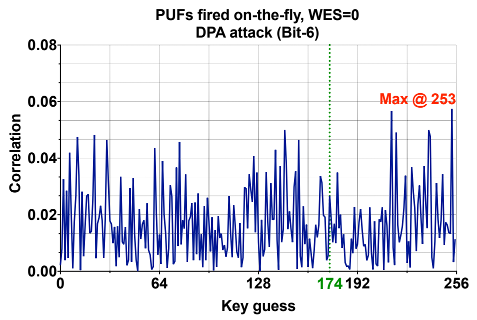
(d)



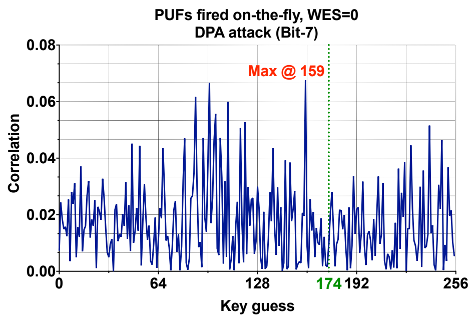
(e)



(f)

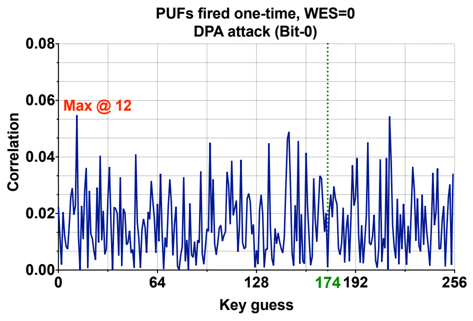


(g)

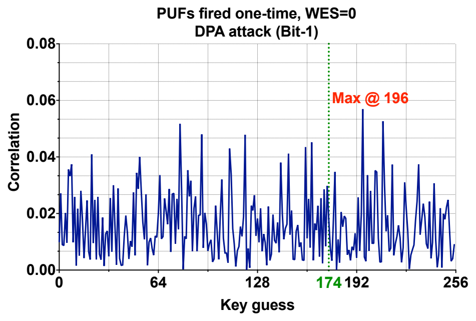


(h)

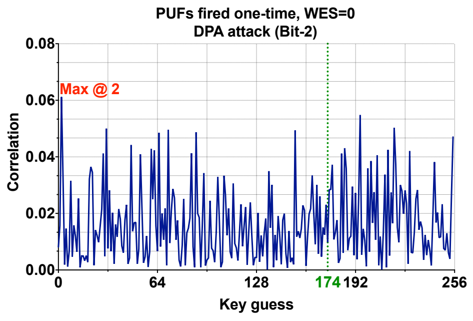
Figure A.4: DPA attack in Mode3, where the PUFs are fired on-the-fly and the interconnect discharges. Correlation plots vs. key guesses (in decimal) for DPA attack on (a) bit-0, (b) bit-1, (c) bit-2, (d) bit-3, (e) bit-4, (f) bit-5, (g) bit-6, (h) bit-7 are shown. Key guesses with the highest correlation are highlighted in red. The key value 174 (highlighted in green) is used in the benchmark circuit.



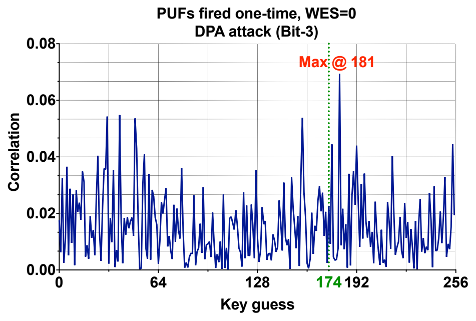
(a)



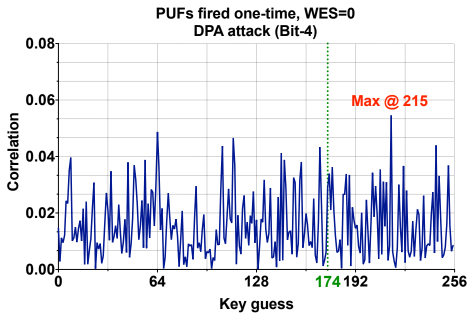
(b)



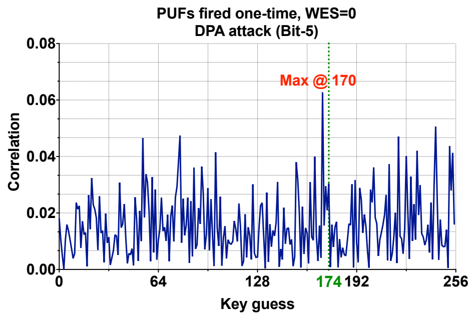
(c)



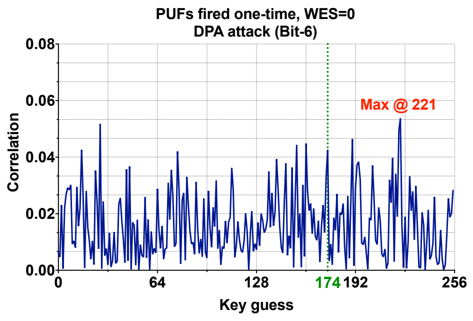
(d)



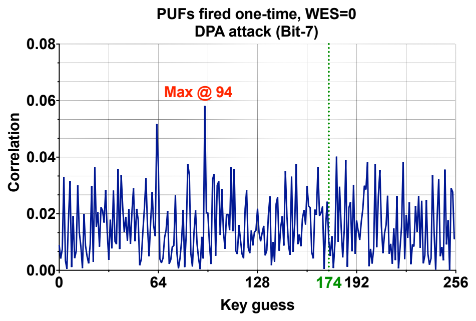
(e)



(f)

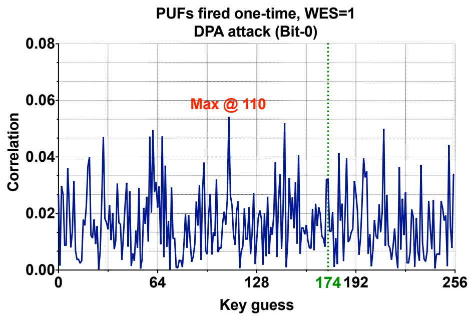


(g)

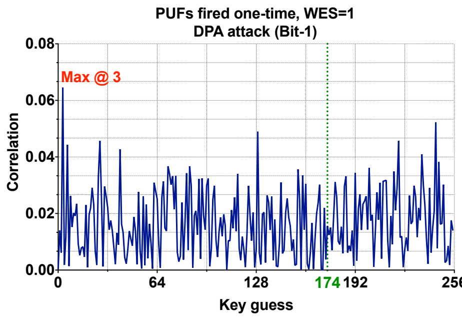


(h)

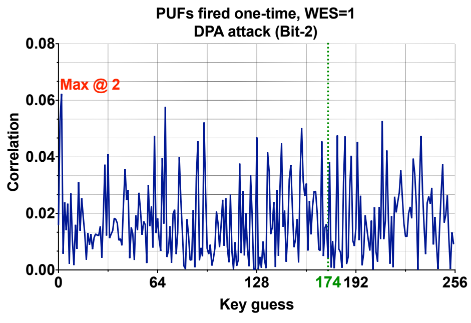
Figure A.5: DPA attack in Mode2, where the PUFs are fired one-time and the interconnect discharges. Correlation plots vs. key guesses (in decimal) for DPA attack on (a) bit-0, (b) bit-1, (c) bit-2, (d) bit-3, (e) bit-4, (f) bit-5, (g) bit-6, (h) bit-7 are shown. Key guesses with the highest correlation are highlighted in red. The key value 174 (highlighted in green) is used in the benchmark circuit.



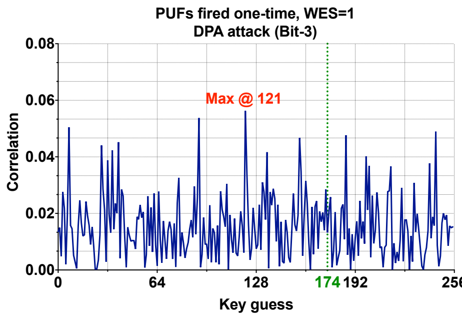
(a)



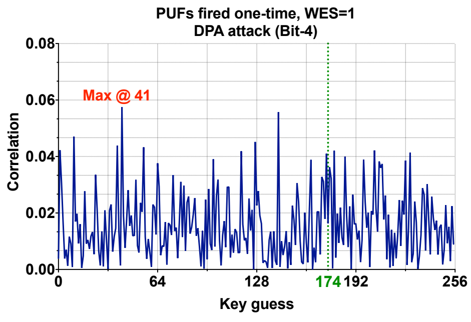
(b)



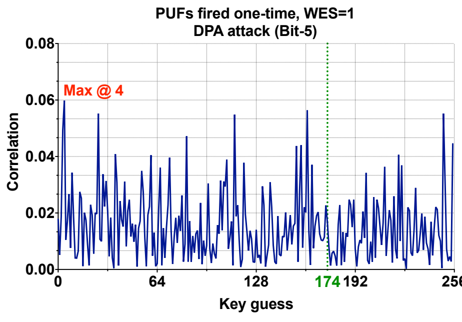
(c)



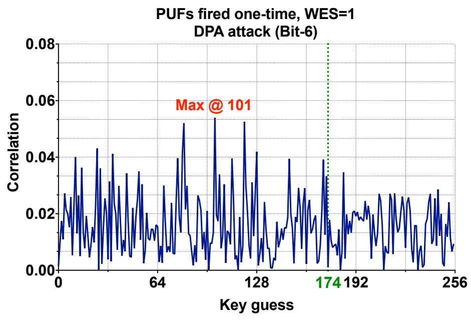
(d)



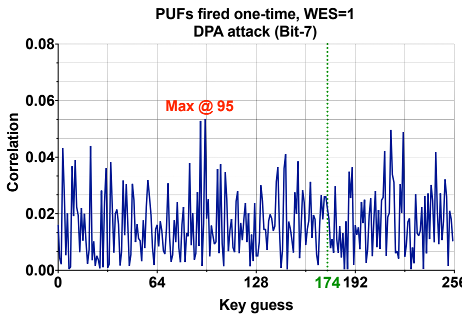
(e)



(f)



(g)



(h)

Figure A.6: DPA attack in the default mode, where the PUFs are fired one-time and the interconnect discharge is turned off. Correlation plots vs. key guesses (in decimal) for DPA attack on (a) bit-0, (b) bit-1, (c) bit-2, (d) bit-3, (e) bit-4, (f) bit-5, (g) bit-6, (h) bit-7 are shown. Key guesses with the highest correlation are highlighted in red. The key value 174 (highlighted in green) is used in the benchmark circuit.

Bibliography

[1] T. Huffmire, C. Irvine, T. D. Nguyen, T. Levin, R. Kastner, and T. Sherwood, editors. *Handbook of FPGA Design Security*. Springer, 2010. ISBN 9789048191567. 1.1, 1.2, 2.1.3, 2.4.1, 3.4

[2] B. Badrignans, J. L. Danger, V. Fischer, G. Gogniat, and L. Torres, editors. *Security Trends for FPGAS*. Springer, 2011. ISBN 9789400713383. 1.1, 1.2, 2.7, 2.4.1, 3.4

[3] I. Verbauwhede, K. Tiri, D. Hwang, and P. Schaumont. Circuits and Design Techniques for Secure ICs Resistant to Side-Channel Attacks. In *2006 IEEE International Conference on IC Design and Technology*, pages 1–4, May 2006. doi: 10.1109/ICICDT.2006.220791. 1.1, 4.3.2

[4] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48405-9. 1.1, 2.3.1, 2.3.1

[5] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology — CRYPTO '96*, pages 104–113, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-68697-2. 1.1

[6] F. Regazzoni and Y. Wang. FPGA implementations of the AES masked against power analysis attacks. In *COSADE*, pages 56–66, 2011. 1.1, 1.2, 2.3.1, 2.3.2

[7] S. Mangard, E. Oswald, and T. Popp, editors. *Power Analysis Attacks*. Springer, 2007. ISBN 9780387381626. 1.1, 2.3.1

[8] M. Smerdon. Security Solutions Using Spartan-3 Generation FPGAs. 2008. URL `https://www.xilinx.com/support/documentation/white_papers/wp266.pdf`. 1.2, 2.4.1, 3.4

[9] ProASIC3 Flash Family FPGAs Datasheet. Technical report, Microsemi, 2013. URL `https://www.microsemi.com/document-portal/doc_download/130704-ds0097-proasic3-flash-family-fpgas-datasheet`. 1.2, 3.4

[10] C. Helfmeier, C. Boit, D. Nedospasov, and J. Seifert. Cloning Physically Unclonable Functions. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 1–6, June 2013. doi: 10.1109/HST.2013.6581556. 1.2, 2.4.1, 3.7.3

[11] E. Trichina, T. Korkishko, and K. H. Lee. Small Size, Low Power, Side Channel-Immune AES Coprocessor: Design and Synthesis Results. In *Advanced Encryption Standard – AES*, pages 113–127, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31840-8. 1.2, 2.3.2

[12] D. Suzuki, M. Saeki, and T. Ichikawa. Random Switching Logic: A Countermeasure against DPA based on Transition Probability. In *IACR ePrint on Transition Probability*, 2004. 1.2, 2.3.2

[13] F. X. Standaert, G. Rouvroy, and J. Quisquater. FPGA Implementations of the DES and Triple-DES Masked Against Power Analysis Attacks. In *2006 International Conference on Field Programmable Logic and Applications*, pages 1–4, Aug 2006. doi: 10.1109/FPL.2006.311315. 1.2, 2.3.2

[14] K. Tiri and I. Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition (DATE)*, volume 1, pages 246–251 Vol.1, Feb 2004. doi: 10.1109/DATE.2004.1268856. 1.2, 2.3.2, 2.3.2

[15] K. Tiri and I. Verbauwhede. Synthesis of secure FPGA implementations. In *IACR Cryptology ePrint Archive*, 2004. 1.2, 2.3.2, 2.3.2, 2.4.2, 4.3.1

[16] P. Yu and P. Schaumont. Secure FPGA circuits using ontrolled placement and routing. In *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 45–50, Sept 2007. 1.2, 2.3.1, 2.6, 2.3.2, 2.4.2, 2.8, 2.4.2, 4.3.1, 5.5, 5.5.1, 5.5.1

[17] G. Lemieux, E. Lee, M. Tom, and A. Yu. Directional and single-driver wires in FPGA interconnect. In *Proceedings. 2004 IEEE International Conference on Field- Programmable Technology (FPT)*, pages 41–48, Dec 2004. doi: 10.1109/FPT.2004.1393249. 2.1.3, 2.1.3, 2.4, 4.2.3

[18] M. I. Masud and S. J. E. Wilton. A New Switch Block for Segmented FPGAs. In *Field Programmable Logic and Applications (FPL)*, pages 274–281, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48302-1. 2.1.3, 2.1.3

[19] I. Kuon, R. Tessier, and J. Rose. FPGA Architecture: Survey and Challenges. In *FTEDA*, 2008. 2.1.3

[20] V. Betz, J. Rose, and A. Marquardt, editors. *Architecture and CAD for Deep-Submicron FPGAs*. Springer, 1999. ISBN 9781461551454. 2.1.3, 2.1.3, 3.5.1

[21] V. Betz and J. Rose. FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density. In *Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays*, FPGA '99, pages 59–68, New York, NY, USA, 1999. ACM. ISBN 1-58113-088-0. doi: 10.1145/296399.296428. URL `http://doi.acm.org/10.1145/296399.296428`. 2.1.3, 2.1.3

[22] S. J. E. Wilton. *Architectures and Algorithms for FPGAs with Embedded Memory*. PhD thesis, University of Toronto, 1997. 2.1.3, 2.1.3

[23] Y. W. Chang, D. F. Wong, and C. K. Wong. Universal Switch Modules for FPGA Design. *ACM Transactions on Design Automation Electronic Systems*, 1(1):80–101, January 1996. ISSN 1084-4309. doi: 10.1145/225871.225886. URL `http://doi.acm.org/10.1145/225871.225886`. 2.1.3

[24] K. Bennett. Newegg selling fake Intel CPUs. URL http://www.hardocp.com/article/2010/03/05/newegg_selling_fake_intel_cpus. 2.2.1

[25] US Senate Armed Service Committee. Inquiry into counterfeit electronic parts in the Department of Defense supply chain. URL https://www.gpo.gov/fdsys/pkg/CRPT-112srpt167/pdf/CRPT-112srpt167.pdf. 2.2.1

[26] US General Accounting Office. DOD supply chain suspect counterfeit electronic parts can be found on internet purchasing platforms. URL http://www.gao.gov/assets/590/588736.pdf. 2.2.1

[27] S. Skorobogatov and C. Woods. Breakthrough Silicon Scanning Discovers Backdoor in Military Chip. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 23–40, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33027-8. 2.2.1

[28] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design Test of Computers*, 27(1):10–25, Jan 2010. ISSN 0740-7475. doi: 10.1109/MDT.2010.7. 2.2.1

[29] Chipworks. Intel's 22-nm Tri-gate Transistors Exposed. URL http://www.chipworks.com/blog/technologyblog/2012/04/23/intels-22-nm-trigate-transistors-exposed/. 2.2.1, 2.2.2

[30] TAEUS. URL http://www.taeus.com/taeus-services/ip-litigation-services/reverse-engineering/. 2.2.1, 2.2.2

[31] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri. Security Analysis of Integrated Circuit Camouflaging. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and ; Communications Security (CCS)*, pages 709–720, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2477-9. doi: 10.1145/2508859.2516656. URL http://doi.acm.org/10.1145/2508859.2516656. 2.2.2, 3.5.3, 3.7.1

[32] SypherMedia. SypherMedia library circuit camouflage technology. URL http://www.chipworks.com/blog/technologyblog/2012/04/23/

intels-22-nm-trigate-transistors-exposed/. 2.2.2

[33] B. Erbagci, C. Erbagci, N. E. C. Akkaya, and K. Mai. A Secure Camouflaged Threshold Voltage Defined Logic Family. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 229–235, May 2016. doi: 10.1109/HST.2016.7495587. 2.2.2, 3.5.3, 3.7.2

[34] N. E. C. Akkaya, B. Erbagci, and K. Mai. A secure camouflaged logic family using post-manufacturing programming with a 3.6GHz adder prototype in 65nm CMOS at 1V nominal VDD. In *IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 128–130, Feb 2018. doi: 10.1109/ISSCC.2018.8310217. 2.2.2, 5.4.2

[35] Y. M. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *USENIX Security Symposium (USENIX Security)*, Boston, MA, 2007. USENIX Association. URL https://www.usenix.org/conference/16th-usenix-security-symposium/active-hardware-metering-intellectual-property-protection. 2.2.2

[36] R. S. Chakraborty and S. Bhunia. HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1493–1502, Oct 2009. ISSN 0278-0070. doi: 10.1109/TCAD.2009.2028166. 2.2.2, 3.7.1

[37] L.W. Chow et al. Camouflaging a standard cell based integrated circuit, April 3 2012. URL http://www.google.com/patents/US8151235. US Patent 8,151,235. 2.2.2

[38] M. Massad, G. Siddharth, and M. V. Tripunitara. Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes . In *NDSS*, 2015. 2.2.2, 3.7.2

[39] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 137–143, May 2015. doi: 10.1109/HST.2015.7140252. 2.2.2

[40] Degate. An open source software for semi-automatic reverse engineering of logic in integrated circuits. URL `http://www.degate.org`. 2.2.2

[41] M. C. Hansen, H. Yalcin, and J. P. Hayes. Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. *IEEE Design Test of Computers*, 16(3):72–80, July 1999. ISSN 0740-7475. doi: 10.1109/54.785838. 2.2.2

[42] W. Li, Z. Wasson, and S. A. Seshia. Reverse engineering circuits using behavioral pattern mining. In *IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 83–88, June 2012. doi: 10.1109/HST.2012.6224325. 2.2.2

[43] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik. Reverse engineering digital circuits using functional analysis. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1277–1280, March 2013. doi: 10.7873/DATE.2013.264. 2.2.2

[44] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Logic encryption: A fault analysis perspective. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 953–958, March 2012. doi: 10.1109/DATE.2012.6176634. 2.2.2

[45] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design Test of Computers*, 27(1):66–75, Jan 2010. ISSN 0740-7475. doi: 10.1109/MDT.2010.24. 2.2.2

[46] K. Mai. Introduction to Hardware Security and Trust. Chapter: Side-Channel Attacks and Countermeasures. Springer NY. 2012. 2.5, 3.2

[47] M. Alioto, L. Giancane, G. Scotti, and A. Trifiletti. Leakage Power Analysis Attacks: A Novel Class of Attacks to Nanometer Cryptographic Circuits. *IEEE Transactions on Circuits and Systems*, 57(2):355–367, Feb 2010. ISSN 1549-8328. doi: 10.1109/TCSI.2009.2019411. 2.3.1

[48] D. D. Hwang, K. Tiri, A. Hodjat, B. . Lai, S. Yang, P. Schaumont, and I. Verbauwhede. AES-Based Security Coprocessor IC in 0.18-um CMOS With Resistance to Differential

Power Analysis Side-Channel Attacks. *IEEE Journal of Solid-State Circuits (JSSC)*, 41 (4):781–792, April 2006. ISSN 0018-9200. doi: 10.1109/JSSC.2006.870913. 2.3.1

[49] B. den Boer, K. Lemke, and G. Wicke. A DPA Attack against the Modular Reduction within a CRT Implementation of RSA. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 228–243, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-36400-9. 2.3.1

[50] K. Tiri, M. Akmal, and I. Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Dower Analysis on Smart Cards. In *Proceedings of the 28th European Solid-State Circuits Conference (ESSCIRC)*, pages 403–406, Sept 2002. 2.3.2, 2.3.2, 4.3.2

[51] M. Bucci, L. Giancane, R. Luzzi, and A. Trifiletti. Three-phase Dual-rail Pre-charge Logic. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 232–241. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-46561-4. 2.3.2, 2.3.2

[52] P. Schaumont and K. Tiri. Masking and Dual-Rail Logic Dont Add Up. In *Cryptographic Hardware and Embedded Systems - CHES*, pages 95–106, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74735-2. 2.3.2

[53] F.X. Standaert, E. Peeters, and J.J. Quisquater. On the Masking Countermeasure and Higher-order Power Analysis Attacks. In *International Conference on Information Technology: Coding and Computing (ITCC'05)*, volume 1, pages 562–567 Vol. 1, April 2005. doi: 10.1109/ITCC.2005.213. 2.3.2

[54] K. Tiri and P. Schaumont. Changing the Odds Against Masked Logic. In *Selected Areas in Cryptography (SAC)*, pages 134–146, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74462-7. 2.3.2

[55] N. E. C. Akkaya, B. Erbagci, R. Carley, and K. Mai. A DPA-resistant Self-timed Three-phase Dual-rail Pre-charge Logic Family. In *2015 IEEE International Symposium on*

Hardware Oriented Security and Trust (HOST), pages 112–117, May 2015. doi: 10.1109/HST.2015.7140248. 2.3.2

[56] K. Tiri and I. Verbauwhede. Charge recycling sense amplifier based logic: securing low power security ICs against DPA . In *Proceedings of the 30th European Solid-State Circuits Conference (ESSCIRC)*, pages 179–182, Sept 2004. doi: 10.1109/ESSCIR.2004.1356647. 2.3.2

[57] D. Suzuki and M. Saeki. Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style. In *Cryptographic Hardware and Embedded Systems - CHES*, pages 255–269, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46561-4. 2.3.2, 2.4.2

[58] C. Tokunaga and D. Blaauw. Securing Encryption Systems with a Switched Capacitor Current Equalizer. *IEEE Journal of Solid-State Circuits (JSSC)*, 45(1):23–31, Jan 2010. ISSN 0018-9200. doi: 10.1109/JSSC.2009.2034081. 2.3.2, 4.3.2

[59] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski. Side-channel Attacks on the Bitstream Encryption Mechanism of Altera Stratix II: Facilitating Black-box Analysis Using Software Reverse-engineering. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, pages 91–100, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1887-7. doi: 10.1145/2435264.2435282. URL `http://doi.acm.org/10.1145/2435264.2435282`. 2.4.1, 3.7.4

[60] A. Moradi, A. Barenghi, T. Kasper, and C. Paar. On the Vulnerability of FPGA Bitstream Encryption Against Power Analysis Attacks: Extracting Keys from Xilinx Virtex-II FPGAs. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 111–124, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0948-6. doi: 10.1145/2046707.2046722. URL `http://doi.acm.org/10.1145/2046707.2046722`. 2.4.1, 3.7.4

[61] S. Skorobogatov and Woods C. In the blink of an eye: There goes your AES key. *IACR*

Cryptology Archive, 2012. 2.4.1

[62] A. Schlosser, D. Nedospasov, J. Kramer, S. Orlic, and J. P. Seifert. Simple Photonic Emission Analysis of AES. *Journal of Cryptographic Engineering*, 3(1):3–15, Apr 2013. ISSN 2190-8516. doi: 10.1007/s13389-013-0053-7. URL `https://doi.org/10.1007/s13389-013-0053-7`. 2.4.1, 3.7.3

[63] J. Ferrigno and M. Hlavac. When AES blinks: introducing optical side channel. *IET Information Security*, 2(3):94–98, September 2008. ISSN 1751-8709. doi: 10.1049/iet-ifs:20080038. 2.4.1, 3.7.3

[64] S. Skorobogatov. Using Optical Emission Analysis for Estimating Contribution to Power Analysis. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 111–119, Sept 2009. doi: 10.1109/FDTC.2009.39. 2.4.1, 3.7.3

[65] J. Kramer, D. Nedospasov, A. Schlosser, and J. P. Seifert. Differential Photonic Emission Analysis. In *Constructive Side-Channel Analysis and Secure Design (COSADE)*, pages 1–16, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40026-1. 2.4.1, 3.7.3

[66] R. Velegalati and J. Kaps. DPA Resistance for Lightweight Implementations of Cryptographic Algorithms on FPGAs. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 385–390, Aug 2009. doi: 10.1109/FPL.2009.5272260. 2.4.2

[67] K. Baddam and M. Zwolinski. Divided Backend Duplication Methodology for Balanced Dual Rail Routing. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 396–410, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-85053-3. 2.4.2

[68] R. Maes, D. Schellekens, P. Tuyls, and I. Verbauwhede. Analysis and design of active IC metering schemes. In *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 74–81, July 2009. doi: 10.1109/HST.2009.5224964. 3.1

[69] J. Guajardo, T. Guneysu, S. S. Kumar, and C. Paar. Secure IP-block distribution for hardware devices. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pages 82–89, July 2009. doi: 10.1109/HST.2009.5224965. 3.1

[70] E. Simpson and P. Schaumont. Offline Hardware/Software Authentication for Reconfigurable Platforms. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 311–323, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46561-4. 3.1, 3.7.4, 3.7.4

[71] J. X. Zheng and M. Potkonjak. A Digital PUF-based IP Protection Architecture for Network Embedded Systems. In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 255–256, Oct 2014. 3.1

[72] B. Erbagci, M. Bhargava, R. Dondero, and K. Mai. Deeply Hardware-entangled Reconfigurable Logic and Interconnect. In *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–8, Dec 2015. doi: 10.1109/ReConFig.2015.7393295. 3.2

[73] S. M. Trimberger and J. J. Moore. FPGA Security: Motivations, Features, and Applications. *Proceedings of the IEEE*, 102(8):1248–1265, Aug 2014. ISSN 0018-9219. doi: 10.1109/JPROC.2014.2331672. 3.2

[74] K. Wilkinson. Using Encryption to Secure a 7 Series FPGA Bitstream. 2018. URL `https://www.xilinx.com/support/documentation/application_notes/xapp1239-fpga-bitstream-encryption.pdf`. 3.2

[75] G. E. Suh and S. Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *ACM/IEEE Design Automation Conference (DAC)*, pages 9–14, June 2007. 3.2

[76] Intrinsic-ID: Products webpage: www.intrinsic-id.com/products. . 3.2

[77] Verayo Inc. Products webpage: www.verayo.com/products. . 3.2

[78] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. A technique

to build a secret key in ICs for identification and authentication applications. In *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, pages 176–179, June 2004. doi: 10.1109/VLSIC.2004.1346548. 3.2

[79] S. Yu and P. Chen. Emerging Memory Technologies: Recent Trends and Prospects. *IEEE Solid-State Circuits Magazine*, 8(2):43–56, Spring 2016. ISSN 1943-0582. doi: 10.1109/MSSC.2016.2546199. 3.2

[80] M. Bhargava. *Reliable, Secure, Efficient Physical Unclonable Functions*. PhD thesis, Carnegie Mellon University, 2013. 3.2, 3.4.1, 5.4.2, 5.4.2

[81] A. Zygmontowicz, J. Dworak, A. Crouch, and J. Potter. Making It Harder to Unlock an LSIB: Honeytraps and Misdirection in a P1687 Network. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014. doi: 10.7873/DATE.2014.208. 3.3

[82] B. Ege, A. Das, S. Gosh, and I. Verbauwhede. Differential Scan Attack on AES with X-tolerant and X-masked Test Response Compactor. In *Euromicro Conference on Digital System Design*, pages 545–552, Sept 2012. doi: 10.1109/DSD.2012.44. 3.3

[83] A. Das, U. Kocabas, A. Sadeghi, and I. Verbauwhede. PUF-based secure test wrapper design for cryptographic SoC testing. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 866–869, March 2012. doi: 10.1109/DATE.2012.6176618. 3.3

[84] M. Bhargava, C. Cakir, and K. Mai. Comparison of bi-stable and delay-based PUFs from measurements in 65nm bulk CMOS. In *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, Sept 2012. doi: 10.1109/CICC.2012.6330625. 3.4.1

[85] M. Bhargava and K. Mai. A High Reliability PUF Using Hot Carrier Injection Based Response Reinforcement. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 90–106, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40349-1. 3.4.1, 3.4.2, 3.3, 3.4.3, 5.4.2, 5.4.2, 5.4.2, 5.4.2

[86] W. Rankl and W. Effing, editors. *Smart Card Handbook*. Wiley, 2010. ISBN

9780470743676. 3.4.3

[87] J. Luu et al. VTR 7.0: Next Generation Architecture and CAD System for FPGAs. *ACM Trans. Reconfigurable Technology and Systems (TRETS)*, 7(2):6:1–6:30, July 2014. ISSN 1936-7406. doi: 10.1145/2617593. URL `http://doi.acm.org/10.1145/2617593`. 3.5.1, 3.5.2, 4.2.2, 4.2.3

[88] C. Chiasson and V. Betz. Coffe: Fully-automated transistor sizing for FPGAs. In *International Conference on Field-Programmable Technology (FPT)*, pages 34–41, Dec 2013. doi: 10.1109/FPT.2013.6718327. 3.5.1, 3.5.2

[89] C. Chiasson and V. Betz. Should FPGAS abandon the pass-gate? In *23rd International Conference on Field programmable Logic and Applications (FPL)*, pages 1–8, Sept 2013. doi: 10.1109/FPL.2013.6645511. 3.5.1

[90] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(3):288–298, March 2004. ISSN 1063-8210. doi: 10.1109/TVLSI.2004.824300. 3.5.1

[91] Y. Cao, A. Balijepalli, S. Sinha, C. C. Wang, W. Wang, and W. Zhao. The Predictive Technology Model in the Late Silicon Era and Beyond. *Foundations and Trends in Electronic Design Automation*, 3(4):305–401, April 2010. ISSN 1551-3939. doi: 10.1561/1000000012. URL `http://dx.doi.org/10.1561/1000000012`. 3.5.2

[92] Intelligence Advanced Research Projects Activity. Trusted integrated circuits program. URL `https://www.fbo.gov/utils/view?id=b8be3d2c5d5babbdffc6975c370247a6`. 3.5.3

[93] R. W. Jarvis and M. G. McIntyre. Split manufacturing method for advanced semiconductor circuits. In *US Patent No. 7195931, 2002*. 3.5.3

[94] J. Rajendran, O. Sinanoglu, and R. Karri. Is split manufacturing secure? In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1259–1264, March 2013.

doi: 10.7873/DATE.2013.261. 3.5.3

[95] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi. Building trusted ICs using split fabrication. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 1–6, May 2014. doi: 10.1109/HST.2014.6855559. 3.5.3

[96] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Security analysis of logic obfuscation. In *Design Automation Conference (DAC)*, pages 83–89, June 2012. doi: 10.1145/2228360.2228377. 3.7.1, 3.7.2

[97] C. Yu, X. Zhang, D. Liu, M. Ciesielski, and D. Holcomb. Incremental SAT-Based Reverse Engineering of Camouflaged Logic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 36(10):1647–1659, Oct 2017. ISSN 0278-0070. doi: 10.1109/TCAD.2017.2652220. 3.7.2

[98] M. Yasin, O. Sinanoglu, and J. Rajendran. Testing the Trustworthiness of IC Testing: An Oracle-Less Attack on IC Camouflaging. *IEEE Transactions on Information Forensics and Security*, 12(11):2668–2682, Nov 2017. ISSN 1556-6013. doi: 10.1109/TIFS.2017.2710954. 3.7.2

[99] W. Vandervorst, T. Clarysse, and P. Eyben. Spreading resistance roadmap towards and beyond the 70 nm technology node. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, 20(1): 451–458, 2002. doi: 10.1116/1.1446455. URL `https://avs.scitation.org/doi/abs/10.1116/1.1446455`. 3.7.3

[100] C. C. Williams. Two-Dimensional Dopant Profiling by Scanning Capacitance Microscopy. *Annual Review of Materials Science*, 29(1):471–504, 1999. doi: 10.1146/annurev.matsci.29.1.471. URL `https://doi.org/10.1146/annurev.matsci.29.1.471`. 3.7.3

[101] E. Volkl, A. Edgar, F. Lawrence, and D. C. Joy. *Introduction to Electron Holography*. Springer, 1999. ISBN 9781461548171. 3.7.3

[102] Y. Huang, C. C. Williams, and H. Smith. Direct Comparison of Cross sectional Scanning Capacitance Microscope Dopant Profile and Vertical Secondary Ion mass Spectroscopy Profile. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, 14(1):433–436, 1996. doi: 10.1116/1.588489. URL `https://avs.scitation.org/doi/abs/10.1116/1.588489`. 3.7.3

[103] V. Vartanian et al. Metrology Challenges for 45 nm Strained-Si Devices. *IEEE Transactions on Semiconductor Manufacturing*, 19(4):381–390, Nov 2006. ISSN 0894-6507. doi: 10.1109/TSM.2006.884603. 3.7.3

[104] M. V. Stangoni. *Scanning Probe Techniques for Dopant Profile Characterization*. PhD thesis, ETH, 2005. 3.7.3

[105] P. De Wolf, R. Stephenson, T. Trenkler, T. Clarysse, T. Hantschel, and W. Vandervorst. Status and review of two-dimensional carrier and dopant profiling using scanning probe microscopy. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, 18(1):361–368, 2000. doi: 10.1116/1.591198. URL `https://avs.scitation.org/doi/abs/10.1116/1.591198`. 3.7.3

[106] T. E. Williams. *Self-timed rings and their application to division*. PhD thesis, Stanford University, 1991. 4.1

[107] Z. Xia, M. Hariyama, and M. Kameyama. Asynchronous Domino Logic Pipeline Design Based on Constructed Critical Data Path. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(4):619–630, April 2015. ISSN 1063-8210. doi: 10.1109/TVLSI.2014.2314685. 4.1

[108] D. E. Muller. Asynchronous logics and application to information processing . In *Switching Theory in Space Technology*, 1963. 4.1

[109] I. Kuon and J. Rose, editors. *Quantifying and Exploring the Gap Between FPGAs and*

ASICs. Springer, 2010. ISBN 9781441907387. 4.2

[110] P. Yu. Implementation of DPA-Resistant Circuit for FPGA. Master's thesis, Virginia Tech, 2007. 4.3.1, 5.5, 5.5.1, 5.5.1

[111] R. Chaware, G. Hariharan, J. Lin, I. Singh, G. O'Rourke, K. Ng, S. Y. Pai, C. Li, Z. Huang, and S. K. Cheng. Assembly Challenges in Developing 3D IC Package with Ultra High Yield and High Reliability. In *IEEE 65th Electronic Components and Technology Conference (ECTC)*, pages 1447–1451, May 2015. doi: 10.1109/ECTC.2015.7159787. 4.3.1

[112] I. Bolsens. 2.5D ICs: Just a Stepping Stone or a Long Term Alternative to 3D? 2011. URL `https://www.xilinx.com/publications/about/3-D_Architectures.pdf`. 4.3.1

[113] K. Saban. Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency. 2012. URL `https://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf`. 4.3.1

[114] D. Schultz et al. FPGA configuration circuit including bus-based CRC register, 1999. URL `https://patents.google.com/patent/US6191614`. 5.2

[115] Microsemi IGLOO2 FPGAs Product Brief. 2018. URL `https://www.microsemi.com/document-portal/doc_view/131877-pb0121-igloo2-fpga-product-brief`. 5.4.1

[116] Intel Cyclone III Device Data Sheet. 2012. URL `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyc3/cyc3_ciii52001.pdf`. 5.4.1

[117] Reliability of Intrinsic ID SRAM PUFs. 2017. URL `http://go.intrinsic-id.com/reliability-sram-puf-white-paper-lp`. 5.4.2

[118] ST Microelectronics Reliability Report. 2007. URL `https://escies.org/download/webDocumentFile?id=48684`. 5.4.2

[119] Intel Reliability Report, First Half 2017. Technical report, Intel, 2018. URL `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/rr/rr.pdf`. 5.4.2

[120] Xilinx Device Reliability Report, First Half 2018. Technical report, Xilinx, 2018. URL `https://www.xilinx.com/support/documentation/user_guides/ug116.pdf`. 5.4.2

[121] Xilinx 7 series FPGAs Data Sheet: Overview. 2018. URL `https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf`. 5.4.2

[122] Intel Stratix V FPGA Features. 2015. URL `https://www.intel.com/content/www/us/en/products/programmable/fpga/stratix-v/features.html`. 5.4.2

[123] T. C. Ong, M. Levi, P. Ko, and C. Hu. Recovery of Threshold Voltage After Hot-Carrier Stressing. *IEEE Transactions on Electron Devices*, 35(7):978–984, July 1988. ISSN 0018-9383. doi: 10.1109/16.3354. 5.4.2, 6