Implementation of DPA-Resistant Circuit for FPGA

Pengyuan Yu

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Dr. Patrick Schaumont, Chair
Dr. Sandeep Shukla
Dr. Michael Hsiao

April 24, 2007
Blacksburg, Virginia

Keywords: FPGA, Secure Circuit, Differential Power Analysis

Implementation of DPA-Resistant Circuit for FPGA

Pengyuan Yu

ABSTRACT

In current Field-Programmable-Logic Architecture (FPGA) design flows, it is very hard to control the routing of submodules. It is thus very hard to make an identical copy of an existing circuit within the same FPGA fabric. We have solved this problem in a way that still enables us to modify the logic function of the copied submodule. Our technique has important applications in the design of side-channel resistant implementations in FPGA. Starting from an existing single-ended design, we are able to create a complementary circuit. The resulting overall circuit strongly reduces the power-consumption-dependent information leaks. We will show all the necessary steps needed to implement secure circuits on a FPGA, from initial design stage all the way to verification of the level of security through laboratory measurements. We show that the direct mapping of a secure ASIC circuit-style in an FPGA does not preserve the same level of security, unless our symmetrical routing technique is employed. We demonstrate our approach on an FPGA prototype of a cryptographic design, and show through power-measurements followed by side-channel power analysis that secure logic implemented with our approach is resistant whereas non-routing-aware directly mapped circuit can be successfully attacked.

Dedication

To My Parents

Acknowledgments

I would like to thank my advisor, Dr. Patrick Schaumont, for giving me the opportunity to conduct research in the Secure Embedded Systems Group. I truly appreciate and am grateful for his guidance and support on every aspects throughout my study. Without his assistance and inspiration, the work in this thesis wouldn't be possible.

I would also like to thank my colleague, Eric Simpson, for providing valuable suggestions at the time of difficulty during this research work.

Lastly, I would also like to thank my wonderful parents, for their constant support throughout my life, academically and emotionally.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's world, almost everyone has come in contact in one way or another with some form of embedded systems, such as smart-phones, smart-cards, RFID systems, ATMs, and laptops. Many of these embedded systems have already become part of their user's everyday lives: vast amount of personal information are processed by them, stored in them, and communicated through them. Sometimes, when these information are stolen, they can be used against us, such as identity theft, malicious threats/harassment or even personal endangerment from dangerous predators. Imagine losing your cellphone which you use everyday, on which you could have stored many information about yourself and also your family and friends, such as addresses and phone numbers. Without protection, the minute you lose your cellphone is the minute you relinquish not only your privacy but also the privacy of people close to you. Losing important systems such as cellphone happens quite often. In recent years, it was reported that many federal government branches have lost many of their government laptops, which stores not only personal information of the user, but critical information concerning many people such as their social security numbers, past tax returns, or even FBI profiles if they have one. It is impossible to prevent people from losing their valuables or having personal possessions stolen. However, it is possible to provide a level of security so that the information within these stolen or lost devices are not accessible by people with

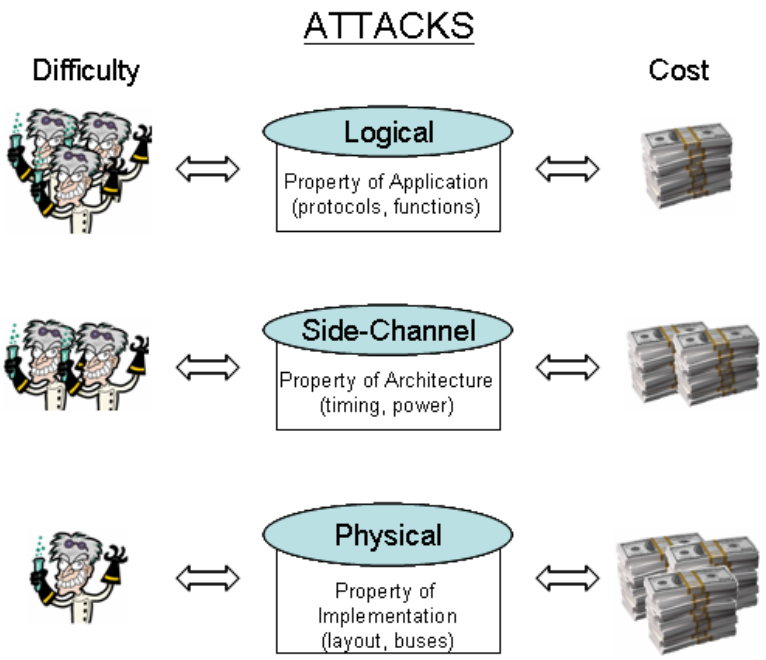


Figure 1.1: Different attacks and their cost and difficulty

malicious intent.

There exist many method for protection of embedded systems, one way is to use cryptographic functions. Information processed by the embedded system are encrypted by the crypto-system so that attackers who are trying to access this information do not understand the encrypted data they can easily obtain. All crypto-systems relies on the use of "secret key" to encrypt the data passing through it. This form of protection entrust the weight of maintaining security on the correct operations of the cryptographic systems used and the absolute secrecy of the key used. Unfortunately, nothing has 100% bullet-proof security. The crypto-system can be attacked in various ways such that the underlying "secret key" are extracted.

1.1 Security Risks of Cryptographic Systems

Based on the nature of various attacks on a cryptographic systems, we can distinguish three broad categories of attacks on embedded electronics [1]. Figure 1.1 can best summarize the attacks and their associated cost and difficulty levels.

1. Logical Attacks: exploit bugs in the embedded software, or weaknesses in the interfaces [2]. It takes advantages of loopholes that exist in security protocols, such as executing valid commands in an unexpected sequences can potentially give attacker certain access privileges. Economically they are the least costly, since no specialized equipment is needed to mount and attack. The difficulty, on the other hand, presents challenges to the attacker. Successful attacks are not always possible.

2. Side-Channel Attacks: exploit physical phenomena of the electronics, such as power-consumption and execution time [3]. Side-channel attacks are more economical than physical attacks and they are more systematic than logical attacks. Recent research efforts have provided many feasible scenarios for these side-channel attacks. Cache-timing attacks on Advanced Encryption Standard (AES) have been demonstrated in [4]. Another type of Side-Channel attack is based on power consumption. Cryptographic processors implemented in hardware have certain power consumption profiles during operation, it is possible to extract secret key information based on analysis of the power consumption profile.

3. Physical Attacks: exploit the physical implementation itself and rely on probe stations, chemical solvents, and microscopes to gain inside knowledge of a chips' operation [5]. They are invasive attacks that require many hours of work to mount and results in the destruction of the packaging of the device. They are however the most systematic kind of attacks, and require little knowledge of the circuit under attack. [6]. The cost of implementing such attacks are usually very high compared to others.

Fortunately, circuit-level defenses against side-channel attacks do exist. Using constant-time, constant-power design techniques, side-channel information leakage can be hidden from the external observer. In this work, we will focus on preventing Side-Channel attacks, specifically attacks based on power leakage profile.

1.2 Motivation

Virtually all constant-power secure circuit techniques proposed so far are specifically targeted towards ASIC. However, also FPGAs are an excellent platform for secure circuit design. Reconfigurable fabrics have an excellent resistance against physical attacks since the underlying platform is regular and does not reveal information on the actual design content. In an SRAM-based FPGA, a design exists only as long as the device is configured and powered.

In this thesis we will show that it is also possible to develop efficient side-channel resistant circuits in an FPGA. It is unfortunately not possible to port secure circuit styles from ASIC directly onto FPGA. We will demonstrate this by successfully attacking an FPGA prototype of a well-known secure logic style, based on building complementary-switching circuits. To remedy this effect, we propose a place-and-route (P&R) technique to complement the selected logic style. Our P&R technique is able to create an identical duplicate of a previously placed-and-routed module, such that all routing information is preserved. This way, we can create two modules which will have identical capacitive loading, and consequently identical power profiles. The symmetric P&R technique is then used to create precisely-matched complementary modules with symmetrical routing. We are able to demonstrate that our symmetrical routing technique is necessary in order to achieve security against power analysis attacks on a FPGA.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 presents the basics of a pre-charged differential logic style. In Chapter 3, we provide an overview of our approach to implementing secure logic on FPGA. Then, we demonstrate how to implement secure logics on FPGA and introduce our routing technique. Chapter 4 presents security evaluations of circuits implemented with and without our routing approach, and we conclude and point out some future research directions in Chapter 5.

Chapter 2

Power-Attack-Resistant Logic

In our research, we are specifically concerned with power-based side-channel attacks. Every circuit has a power consumption signature during operation, shown through current drawn from the power supplies. Based on this signature, powerful techniques such as Differential Power Analysis (DPA) can be employed to extract important information during circuit operation. Power consumption characteristics prove to be especially devastating for cryptographic implementations.

2.1 Sources of Information Leakage

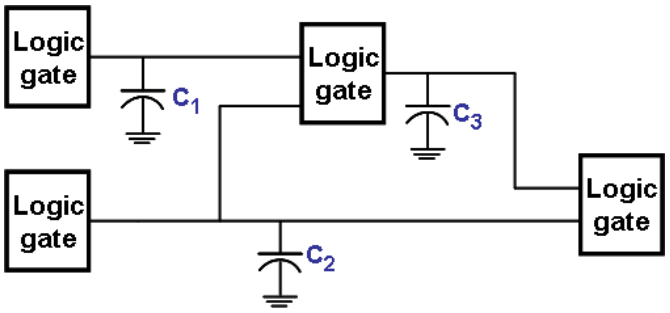
For current CMOS technology, power consumption of a circuit is mainly contributed by dynamic power consumption and static power leakage consumption. Analyzing these types of power consumption helps us to find solution on better controlling the total power consumption.

The sources of dynamic power information leakage come mainly from two categories:

- A digital circuit functions by evaluating input voltage level and set the output voltage level based on the input through a set of logic gates. In current Complementary Metal-

Transition	Current Draw
0 => 0	No
0 => 1	Yes
1 => 0	No
1 => 1	No

(a) Power consumption for each type of transitions



(b) Capacitive loading for different nets

Figure 2.1: Side-Channel Power Information Leakage Source

Oxide Semiconductor (CMOS) technology, the logic value of a gate actually depends on the amount of charge stored on the parasitic capacitor at the output of the logic gate. A fully-charged capacitor represents logic-high(logic-1) whereas a depleted capacitor represents logic-low(logic-0). For each binary logic gate, there can only be four types of transitions on the gate output wire, as shown in 2.1(a). Only one transition, from logic-0 to logic-1, actually draws current from the power supply to charge up the parasitic capacitor. By monitoring the amount of current consumed by the digital circuit at all times, we can get an idea on the relative amount of logic gate that are switching at any given time. This gives us some information about the circuit based on power consumption.

- Parasitic capacitance are not uniform for every gate. They depend on the type of gate, fanout of the gate, and also the length of the wire or net in between current gate and its fanout gates 2.1(b). Taking the length of wires as an example, even if two exactly same gates with same number of fanout are connected to same set of successor gates, if the routing of the wires are different, the capacitance will differ. If both had a power-consuming transition, the amount of power consumed will be different, thus leaking important power information about the circuit.

Static power leakage consumption is a tricky matter. It is a characteristic of the process used

to manufacture the circuit. The exact amount of leakage for a given gate within circuit is not controllable by a logic designer. Assuming the gates are manufactured exactly the same, then the static power leakage does not pose a threat. But this is not the case in the real world. Process variation plays an important role in the balancing of static power leakage. As process variation increase, the variation of the amount of charge leaked for every gate during a fixed period of time also increases. Unfortunately, the effect of static power leakage due to process variation can not be evaluated at design time, thus can only be seen through actually measurements on the finished a product, whether it's a ASIC design or a design for FPGA.

In our research, we will find a solution to problem posed by characteristics we can control. We then go through actual power measurement to evaluate our solution as a whole including uncontrollable factors such as static power leakage.

The single most important parameter used for a cryptographic processor is the secret key used for encryption and decryption of data. While the key stays constant for the duration of encryption, the input values for each sub-round of encryption are always changing. The input-dependent characteristic of regular digital circuit will leak enough power consumption information for a skilled adversary to successfully obtain the secret key.

There are two major classes of secure circuits, both of which attempt to remove the correlation between the overall circuit power consumption and the secret data values at selected circuit nodes.

2.2 Masking-Based Logic

Masking-based logic attempts to remove the correlation between power consumption and secret data by randomizing the power consumption such that the correlation is destroyed [7], as shown in Figure 2.2. Masking-logic works by XOR-ing signals with a masking-bit, then later remove the mask by doing another XOR operation. One such masking-based logic style

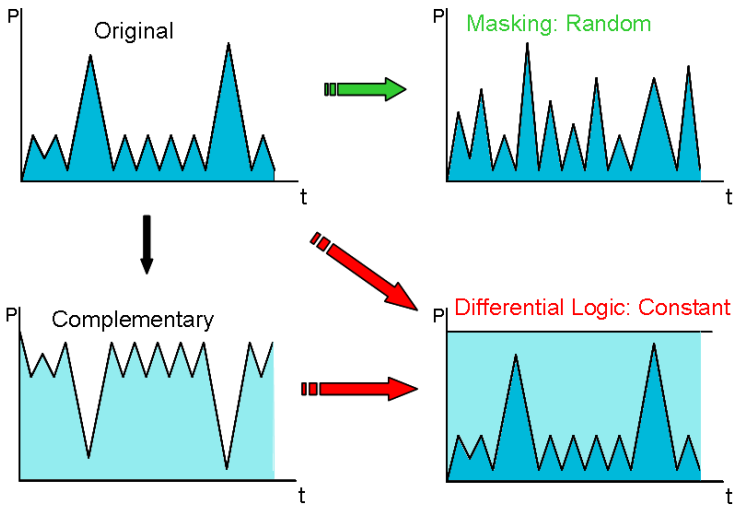


Figure 2.2: Two classes of secure circuits: Masking and Differential Logic

is Random Switching Logic (RSL) [8].

RSL replaces traditional logic gates such as NAND, NOR, and XOR gates with their RSL version respectively. Each RSL gate has four inputs as opposed to two of their traditional counterparts. The two extra inputs are *enable* and *RandomBit*. The *RandomBit* input are used to alter transition probabilities of the circuit and are used to achieve the randomized switching property. *enable* signal on the other hand is used to suppress spurious transitions. The circuit starts operating when *enabled* is asserted, otherwise, circuit is driven to logic-0.

One critical condition for RSL circuit to be secure is that *enable* signal is to be asserted only after ALL input signals are fixed. One question comes to mind is that since *enable* is connected to every gate, how can the signal be coordinated such that the circuit still operates? Imagine the following scenario: RSLGate1 is driving RSLGate2. Assuming they use the same *enable* signal, one of the input of RSLGate2, the one driven by RSLGate1, won't arrive until RSLGate1 is enabled. This means RSLGate2 must also be enabled. Following the condition for security, RSLGate2 shouldn't be enabled because output of RSLGate1 hasn't arrived yet. The implication of this conflict is that every gate or level of gates must have a different enable signal. If so, the cost of enabling every gate could be prohibitively

high.

Assuming RSL *enable* signal can be created and routed, it is shown through simulated attacks in [9], that RSL can still be attacked fairly effectively. It is shown that, indeed, when RSL is in normal operation, the power consumption correlation with secret key is removed. However, further analysis by using a threshold filter can removed the randomizing effect caused by the random bits used, after the random bits is removed, the stripped RSL circuit becomes just like the original circuit with no countermeasure and can be attacked using power analysis to single out the secret key.

2.3 Differential Logic

Differential logic on the other hand attempts to make the resulting power consumption constant [10] [11]. As shown in Figure 2.2, differential logic works by generating a second circuit based on the original one. The power consumed by the second circuit will complement that of original at any point in time. When combined, the overall logic circuit will exhibit overall constant power consumption. Since the power consumption is constant, monitoring power supply will not reveal any secret information about the circuit under attack.

In this work, we will focus on the constant power approach for FPGA. Next, we will discuss in detail the characteristics of a constant power circuit by analyzing a proven ASIC logic style called Wave Dynamic Differential Logic (WDDL).

2.3.1 Wave Dynamic Differential Logic

The most prominent of constant-power technologies is Wave Dynamic Differential Logic(WDDL) [11]. It is logic style designed first for ASIC, based on SABL (Sense-Amplifier Based Logic) technology. To adopt to standard cell ASIC design, it has been improved to form the current WDDL logic style. It can be characterized by the following properties.

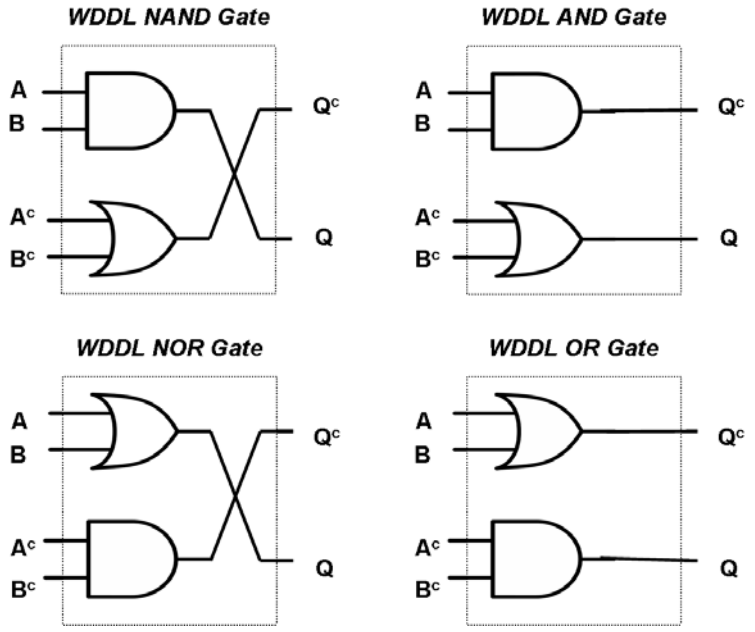


Figure 2.3: A set of basic WDDL gates: NAND, NOR, OR, AND

- Consistent Switching Activities: In order to keep power consumption constant, dual-rail differential logic is used. WDDL replaces the standard gates with their WDDL versions, as shown in Figure 2.3. Each WDDL gates has four inputs and two outputs. Two of the inputs corresponds to the original gate inputs and the other two are their logical complements. The two outputs also complement each other. These logic gates guarantee an opposite switching behavior for the two complementary outputs. As can be seen by the following truth table, Table 2.1 for a WDDL NAND gate, when the direct logic block switches high (consuming current from power supply), the complementary logic block will switch low (discharging the capacitive load).

- Pre-charge Wave Generation: The gates introduced previously will not provide a transition if the inputs are not changing. This behavior renders the gates data-dependent. To fix this problem, in the absence of input data changes, a transition is provided by means of a pre-charge circuit, as shown in Figure 2.4. When the clock is high, the signal connected to the precharge circuit enter a pre-charge phase, where the connected net is

A	B	Ac	Bc	Q	Qc
0	0	1	1	0	1
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	0

Table 2.1: WDDL NAND gate true table

driven to logic 0. When the clock becomes low, the circuit enters the evaluation phase, where actual computation is done. The operation can be more accurately described as pre-discharge since the logics are driven to logic low. We will keep the terminology to precharge since the idea behind it is the same. The reason to assign clock high as precharge phase as oppose to evaluation phase is because regular flip-flops latch in data value at clock up-edge, this means when clock is low, the circuit should be in evaluation phase.

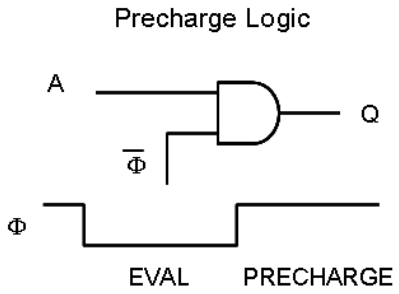


Figure 2.4: Generic WDDL precharge circuit

There are two ways to integrate the precharge logic into the overall circuit. Both has advantages and disadvantages.

– Before Every Gate: if every gate has its own precharge logic, we can precharge every gate almost instantly. The drawbacks are that clock net will have to be routed to every gate. Since precharge logic does not come at no cost, area and delay incurred by precharge logic inserted this way will be significant.

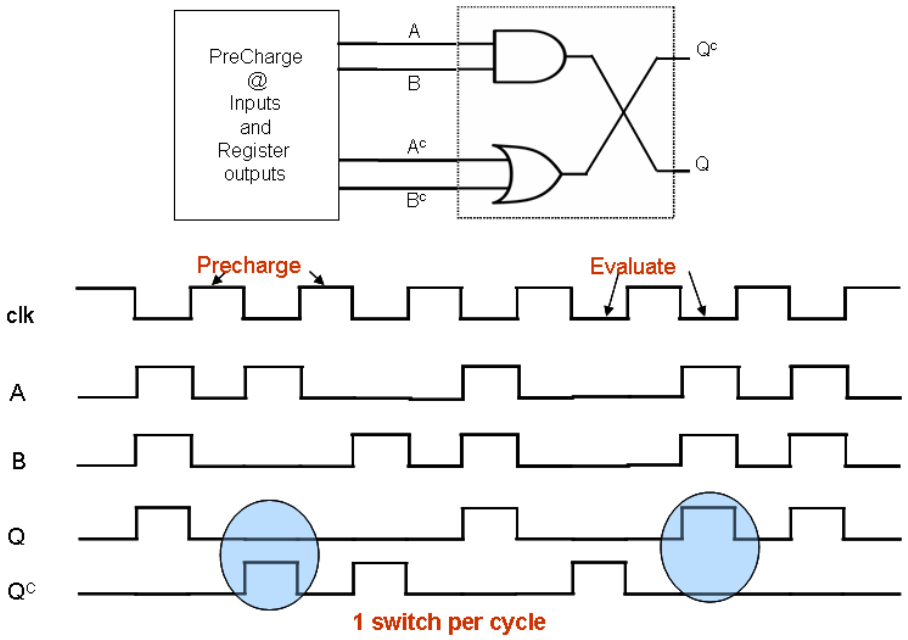


Figure 2.5: WDDL Operation Behavior: 1 Transition per Cycle

– Only At Primary Inputs: circuit inputs and register outputs are considered primary inputs. Inserting precharge logic only at these locations translates to very small area overhead. Since during precharge, every gate must be driven to logic low, this means it takes a lot of time for every gate to evaluate to logic 0. Whether this ripple propagation is more or less than actually logic propagation in the previous case will depend on the circuit itself. One thing is for sure, the area overhead will be much less than inserting precharge logic everywhere.

WDDL follows the second approach and implements the pre-charge circuit only at the register outputs and system inputs [11], and allows a logic-0-wave ripple through the whole module. Hence the name "Wave" Dynamic Differential Logic.

The dual-rail logic gates of WDDL and precharge logic together forms the most important feature of WDDL logic style: during every clock cycle, there are EXACTLY ONE switching activity per WDDL gate. This can be seen clearly from Figure 2.5. This feature is the basis for implementing constant power consumption secure circuits.

- Negative Logic: Notice that, all the WDDL gates have no inverters in them, even though they still implement negative logic, for example, WDDL NAND gate. Inverters disrupt the pre-charge wave propagation because wave front is inverted. Therefore, WDDL uses only positive logic and implements inverters by cross-coupling wires from the direct gate with those from the complementary gate.

  Note that above three properties address the first source of information leakage from power consumption. Rather than altering the probability of power-consuming transitions like masking-based logic, WDDL make such transitions happen on every clock cycle. This effectively deflects any attempt to correlate number of transitions to the secret data.

- Routing Procedure: One major hurdle for WDDL to overcome is the routing of complementary nets. As discussed previously, any routing asymmetry between complementary nets results in unbalanced parasitic capacitive loading and in a residual power variation between direct and complementary transitions. Current techniques to control routing keep the direct and complementary gates close to each other, so that resulting nets are as symmetrical as possible[12].

All four properties of WDDL have been addressed for ASIC designs, in which layout features (circuit elements and routing) can be fully controlled. The same thing can not be said for FPGA. We therefore adapted an existing FPGA design flow to address each of the four aspects.

Chapter 3

Implementation of Secure Circuits for FPGA

Using FPGA as an implementation platform has gained significant popularity in recent times, especially when the FPGA technology is rapidly improving in terms of speed, the amount of logic it can potentially implement, the amount of power consumed, and the speed of prototyping and making real laboratory testing as oppose to pure simulations.

As FPGA has become more readily available to researchers and used more and more for industry products, it becomes important to evaluate whether FPGAs can be used to implement secure logic and to what level of success can we achieve on a FPGA in terms of security against power attacks.

3.1 FPGA Approach to Secure Logic

Based on the success of WDDL for ASIC, we decided to go with the constant-power approach for FPGAs. We would like to minimize the power consumption variations of the combined direct and complementary circuit. We have chosen to start from pre-charged differential logic

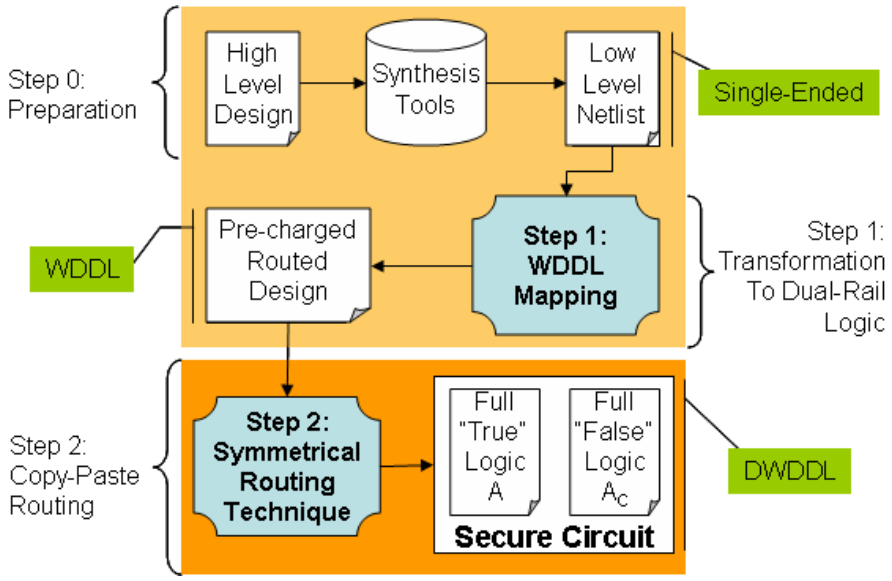


Figure 3.1: FPGA Secure Logic Implementation Flow

to implement side-channel-resistant logic into an FPGA. Current state-of-the-art results have shown that the key challenge for this secure logic style is to maintain the symmetry between the direct and complementary sides of the circuit. Maintaining this symmetry in an FPGA requires precise control over the placement and routing of the differential circuits. Once we have mastered this, we can expect the results to be better than what can be obtained with an ASIC.

The single best way to maintain symmetry for nets of a circuit is to simply copy all the routing information of the original circuit. As will be shown later, the task of maintaining such control in FPGA is very difficult, but not impossible. Therefore, we have separated the implementation of secure circuits on FPGA into two major steps:

1. Find an efficient transformation to map constant power logic styles, such as WDDL, onto FPGA.

2. Develop a symmetrical routing technique to copy all routing information of mapped logic in order to obtain overall constant power consumption.

We call this approach a copy-modify-and-paste approach, as illustrated in the design flow of Figure 3.1. We start with preparation of our initial design for mapping onto FPGA. The starting point is high-level Register-Transfer-Level (RTL) design, from which we will obtain a lower-level design file, either gate-level netlist or LUT-level netlist, through the use of synthesis tools. After preparation, we begin the transformations of low-level design. During transformation, precharge logic will be inserted at the primary inputs. If the netlist is not already in LUT format, we convert it to LUT format at this stage. After the transformation step, we obtain a precharged design. We then pass the design through our symmetrical routing technique to produce the final design. The final design will be the combination of two modules: True module, or direct module, is our precharged design after transformation step; False module, or complementary module, is the duplicated module that will compensate the power consumption of the direct module. Since both modules have an identical routing pattern, transitions in true module are indistinguishable from transitions in false module, and the complete circuit appears to have a constant power consumption. As a result the power consumption is uncorrelated with the internal data activity.

The following sections will discuss the transformations done in step 1 and the different routes that we can take to reach the precharged logic before step 2.

3.2 Understanding Our FPGA Platform

Low level transformations or manipulations requires detailed understanding of the underlying platform, in this case, the physical structure of our FPGA. There is a very wide variety of FPGAs on the market, with the majority share grabbed by Xilinx Inc. Due to the popularity of their FPGAs, and the fact that later laboratory experiments are done using their FPGAs, the structure of Xilinx Spartan3E FPGA will be discussed. We note that FPGAs from other companies like Altera are similar in structure.

On a typical FPGA fabric, there are arrays of neatly organized and mostly identical compo-

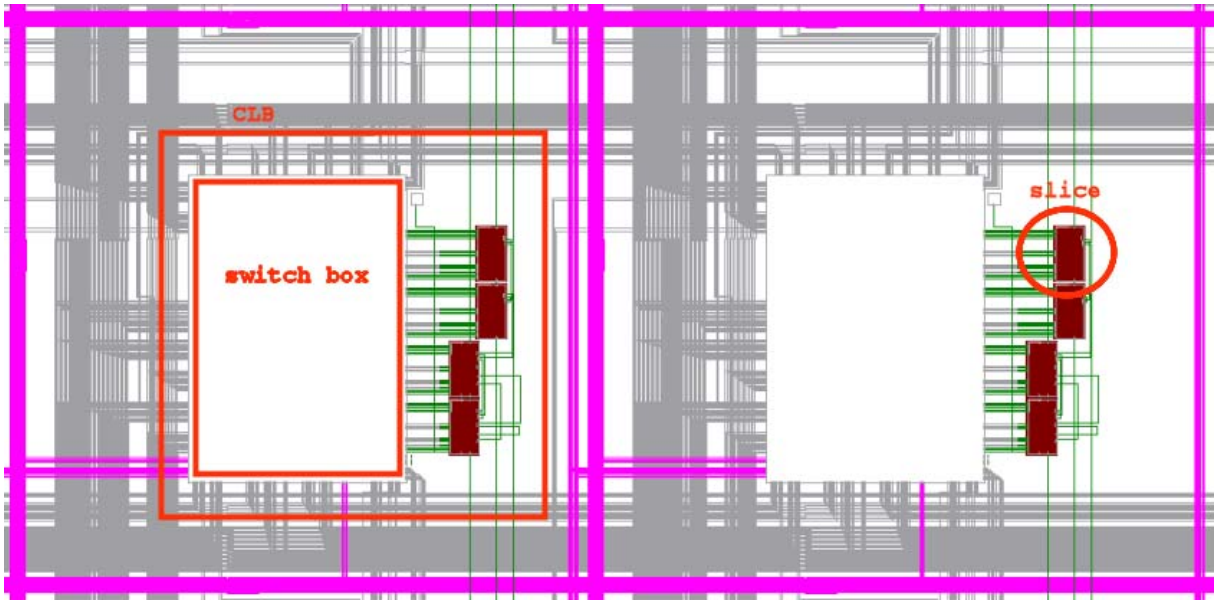


Figure 3.2: Spartan 3E FPGA fabric

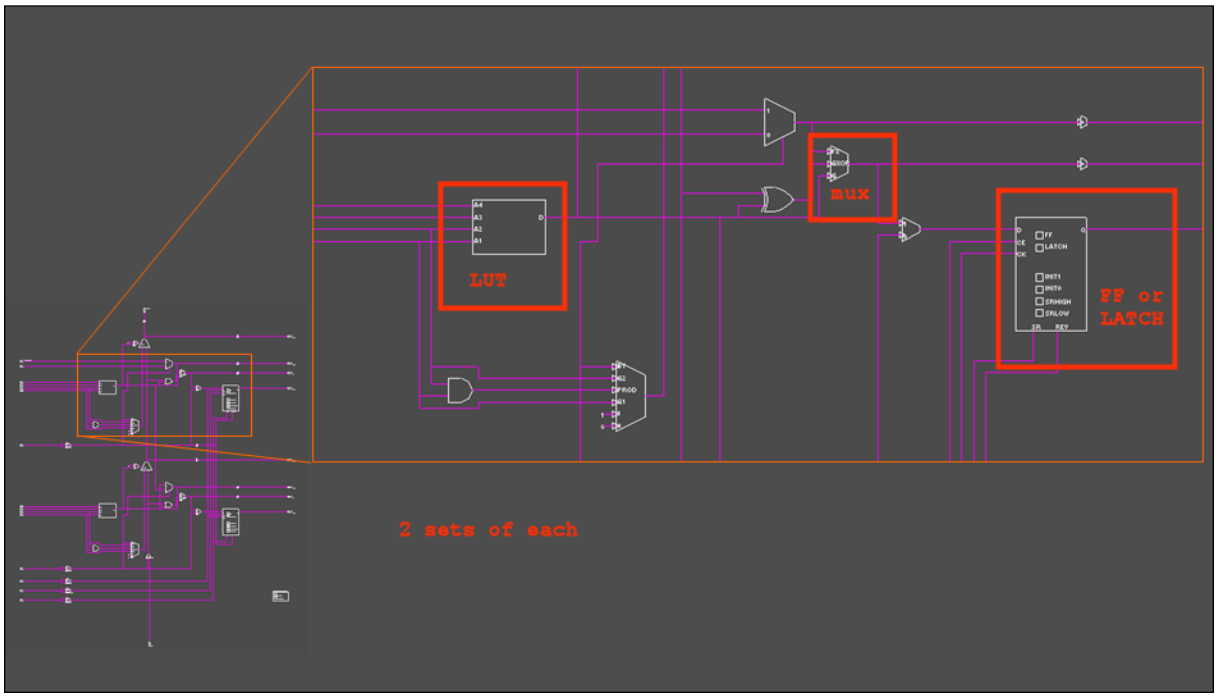


Figure 3.3: Spartan 3E Slice Components

nents called Configurable Logic Blocks (CLB). In Figure 3.2, two such CLBs are shown. We can see that major components within a CLB are four slices, 1 switch box, and many rows and columns of wires. Only the slices are used for logic implementation. The rest are for routing. The 4 slices communicate with each other and other CLBs through the single switch box to the left. The pink wires are horizontal and vertical long wires, they carry singles to far away CLBs. The shorter wires are colored gray and green, they are for communication with neighboring CLB 2 blocks and 6 blocks away and also for local CLB routing. All routing will need to use these routing resources.

For logic implementation, slices within the CLB are used. Within each slice, as shown in Figure 3.3, there are 2 Look-Up-Tables (LUT), 2 dedicated multiplexors right after LUT output, 2 memory elements that can be configured to act as edge-sensitive flip-flops or level-sensitive latches, and other miscellaneous logic elements.

3.3 Step 1: Mapping Techniques for FPGA

In this section, we explain our approach to the implementation of secure logic for FPGA. We will discuss the methods used to map existing ASIC WDDL logic style onto FPGA. As discussed in Section 2.3.1, the first 3 property completely defines the logic style of WDDL. Therefore, the first task of implementing secure logic for FPGA is to use basic FPGA components or a combination of them to accurately reflect the 3 properties of ASIC WDDL.

There are basically two alternatives to defining such logic style for FPGA. Both alternatives share the same type of techniques used for mapping, the difference is in the compromises made to achieve maximum security with little overheads. Both alternatives fit well into the overall design flow defined in Section 3.1, as can be seen from Figure 3.4.

The first scheme is Separated Dynamic Differential Logic (SDDL). SDDL has the same characteristics as WDDL except for the processing of negative logic. SDDL is dual-railed dynamic logic that utilizes precharge circuits to achieve one functional transition per clock

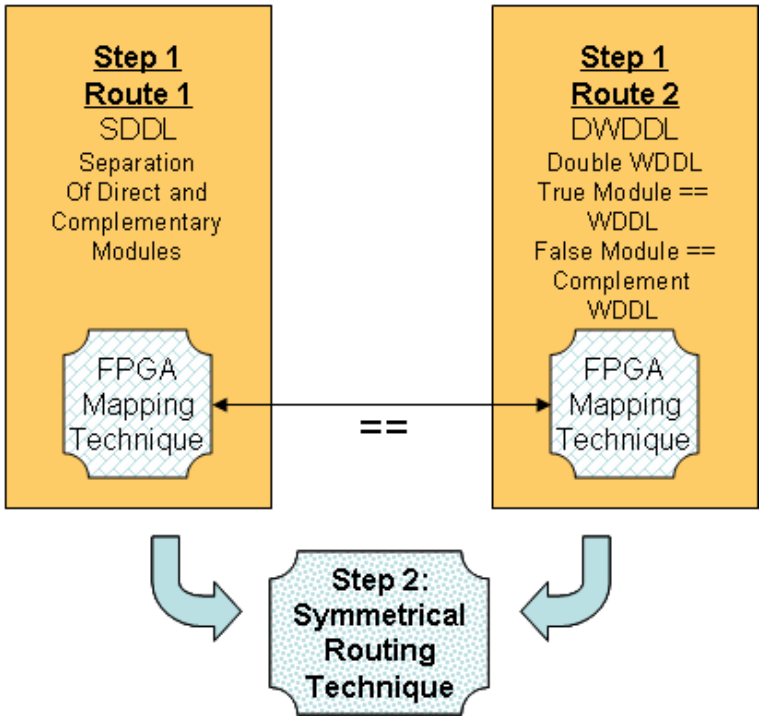


Figure 3.4: Two Alternative Implementation Schemes

cycle. SDDL allows the existence of negative logic in the final circuit without cross-coupling complementary nets. Instead, the precharge circuits are inserted at locations where logical-0 wave are inverted to keep the whole circuit precharged. Hence the removal of "W" for wave for the name of the logic style.

The second scheme is Double Wave Dynamic Differential Logic (DWDDL). DWDDL has all three characteristics of original WDDL logic style. It shares some of the logic mapping technique with SDDL for mapping from ASIC to FPGA. The difference from original WDDL is that both direct and complementary module of DWDDL is a full WDDL logic circuit by itself. This has the benefit of much-relaxed placement constraints for complementary components in each WDDL logic. This translates to unconstrained routing of individual WDDL module, which is the equivalent of elevating the responsibility and constraints of routing to the symmetrical routing technique (step 2) of DWDDL.

In the next few subsections, mapping technique used in both implementation schemes will

be introduced.

3.3.1 When To Insert Pre-Charged Logic

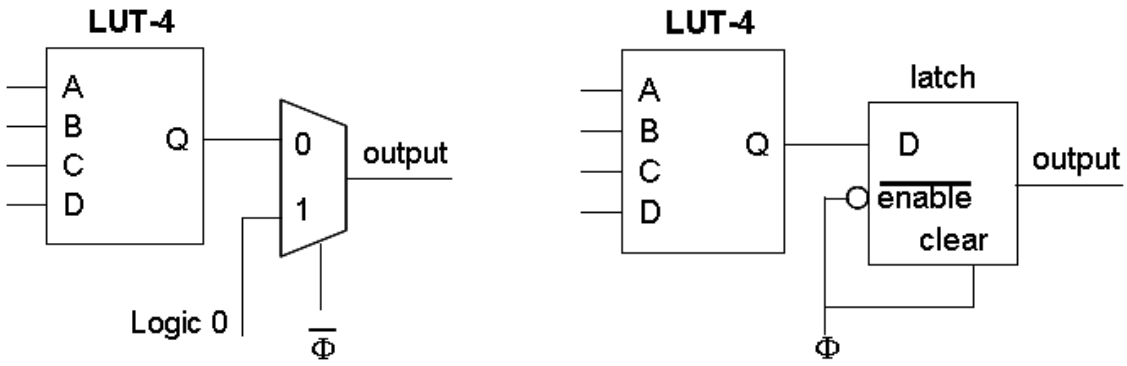
There are two possible abstraction levels at which a pre-charge circuit can be inserted into the netlist.

- Gate-Level: This is the case if design-entry is a gate-level netlist. In the original netlist, insert a pre-charge circuit after every node that halts the pre-charge wave. This will be the case at register outputs and system inputs. If using SDDL logic scheme for implementation, pre-charge are also inserted at inverter outputs.
- FPGA-Level: This is the case if design-entry is a LUT-level netlist. Pre-charge logic are inserted also at system inputs and register outputs. For SDDL, we need to determine which LUT halts the pre-charge wave and insert precharge at the output of those LUTs. This will be further discussed in Section 3.3.3.

We opted for the second approach because this allowed us to obtain a denser circuit and better LUT usage. Indeed, after inserting the precharge logic on the gate-level netlist, one final translation to LUT is required before mapping the circuit onto FPGA. If instead insertion of pre-charge circuit is done directly at LUT-level, we can make use of features of the FPGA fabric itself, such as multiplexors, rather then mapping precharge logic into LUTs. The result is that, for experimental netlists, we obtained 50% less resource usage.

The design-entry for SDDL is to synthesize a LUT-level netlist from high-level design directly. This is possible because SDDL is not concerned about negative logic, thus, if the tool determines it is optimal to synthesize negative logic into a LUT, we will allow it. With this design-entry, the second approach works out very well.

The design entry for DWDDL, on the other hand, is a gate-level netlist obtained from synthesis tools such as Synopsys Design Compiler. DWDDL is concerned with replacing



(a) Precharge Logic using Multiplexer (b) Precharge Logic using Latch

Figure 3.5: Precharge Logic Implementations on FPGA

negative logic with cross-coupled wires, so the ability to identify inversions in the netlist is critical, requiring gate-level synthesis. Note that, the gate library used for synthesis does not have to be limited to all positive logic. Negative logic is allowed because they can be identified and replaced easily. Even though the design entry is a gate-level netlist, it will be translated to LUT-level netlist before insertion of precharge logic to take advantage of above empirical observation for SDDL.

3.3.2 Pre-Charge Logic on FPGA

The purpose of the pre-charge circuit is to force the output of a slice to logic low during the pre-charge phase. There are several ways to do this. In a Xilinx slice, Figure 3.3, each LUT is followed by a dedicated multiplexer and a memory element that can be configured as a register or a latch. We considered the use of the multiplexer and the memory element for pre-charging, as illustrated in Figure 3.5.

Each has advantages and disadvantages:

- By implementing pre-charge with a clock-controlled multiplexer, Figure 3.5(a), we keep the memory element in each slice free for use as a register. However, due to the structure of the multiplexer in the slice, we need to supply and distribute an inverted

clock in addition to the regular clock used by registers.

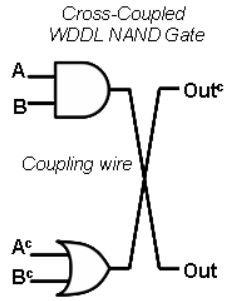
- We can also use the memory-element as an asynchronously-cleared transparent latch with inverted enable input, Figure 3.5(b). This way we only need to supply a single clock signal to both the registers and the pre-charge latches. The pre-charge functionality is implemented by connecting the clock to both the inverting enable input and the clear input of the latch. The disadvantage of this approach is that design-specific register storage now will require a separate slice.

We opted for the second approach because duplication of the clock signal into a direct and complementary form would significantly increase the power consumption and area of the circuit, and it would complicate the routing.

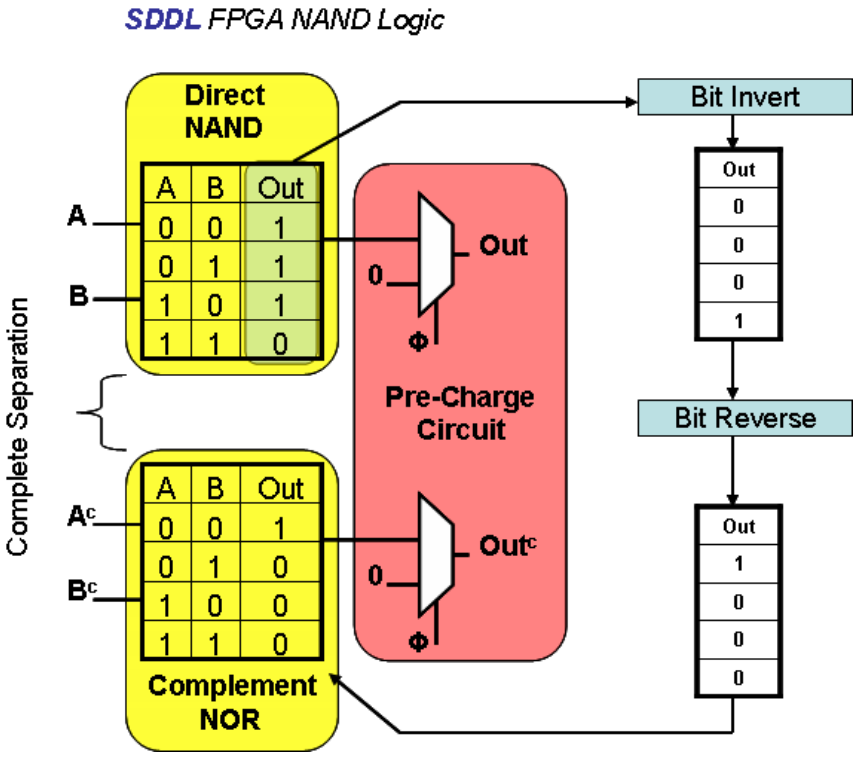
3.3.3 SDDL Complementary LUT

Besides the pre-charge circuits, we also need to create a complementary netlist out of an existing direct LUT-mapped netlist. As illustrated in Figure 3.6, this is done by inverting every bit and reversing the ordering of significance of the inverted bits[11].

To determine whether a LUT will halt pre-charge wave propagation, we only need to look at the least-significant bit and the most-significant bit from the direct-LUT. The most-significant LUT-bit is the entry for which all inputs are logic-1. The least significant LUT-bit is the entry for which all inputs are logic-0. If the least significant bit of the direct-LUT is logic-1, then the direct-LUT will halt the pre-charge wave. Similarly, the complementary-LUT can halt the wave propagation when its least significant bit is logic-1. This will be the case when the most-significant bit of the direct-LUT is logic-0. So the only case when an LUT will not halt wave propagation is when least significant output bit is logic-0 and most significant bit is logic-1. On the average, no more than 1/4 of the synthesized LUT have this property. We therefore implemented a pre-charge circuit after every LUT, and we implemented the pre-charge circuit using the level-sensitive latch approach.



(a) Original WDDL NAND Gate



(b) SDDL NAND Gate

Figure 3.6: Basic SDDL NAND gate implementation

Notice that after this transformation, the direct and complementary modules are completely separated, this conforms to our implementation flow where we will pass the direct module to our symmetrical routing technique to copy and paste the routing information.

3.3.4 DWDDL Complementary LUT

To map DWDDL onto FPGA, we start from a single-ended gate-level netlist, which can be obtained from a synthesis tool such as Synopsys Design Compiler. When negative logic is being replaced by cross-coupling wires, it becomes impossible to keep the complementary nets in the circuit in perfect symmetry, this is especially true on FPGA, which has limited and rigid routing resource. However, we can create a complementary WDDL logic of the orignial direct WDDL logic. These two module will complementary each other in terms of switching activities and they are completely separated, this can be best seen in Figure 3.7.

To create the complementary WDDL, we need a direct WDDL module to start with. From the single-ended gate-level netlist, there are two ways to transform it into WDDL LUT-level netlist.

- Gate-level substitution: we can substitute each negative logic gate in the singled-ended netlist with a positive logic gate representation. For example, substitute each NAND gate with 1 AND gate and 1 OR gate. After every one of the negative logic is replaced with appropriate combination of positive logic and cross-coupled wires, we can use standard FPGA synthesis tool to obtain a LUT-level netlist.

- LUT-level substitution: instead of using positive gates as replacements, we can substitute each negative logic with 2-input LUTs that represent the appropriate positive gate. This requires us to figure out the correct LUT configuration for each type of positive logic used. Standard FPGA synthesis tool will not be able to optimize the circuit, but standard FPGA component mapping tool can do some optimization to pack 2-input LUTs into 4-input LUTs.

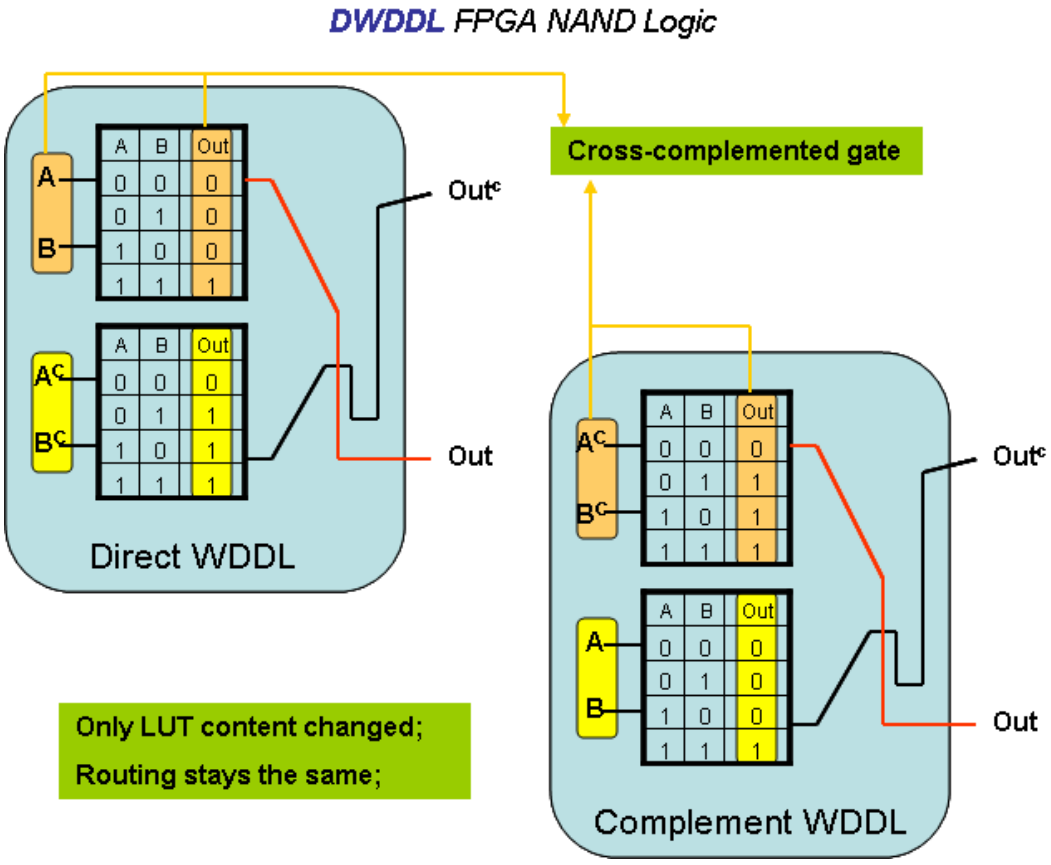


Figure 3.7: DWDDL Complementary LUT Implementation

We used the second substitution methods. After substitution, first method relies on FPGA synthesis to obtain LUT-level netlist. FPGA synthesis sometimes infers multiplexors from the gate-level netlist. This is undesirable because multiplexors are in nature negative logic because it needs inversions on the select line. The second method gives us full control on the type of LUT used and it produce consistent results that do not contain negative logic.

Same as in the case of SDDL, the contents of complementary LUT are created the same way through bit-inversion followed by bit-reversal. The WDDL circuit obtained after previous transformation steps satisfies all the requirements defined in section 2 except for the routing. To overcome this problem, we will next develop an automated placement and routing control methodology and show how to apply it to building secure circuit.

3.4 Step 2: Symmetrical Routing Technique

Modular design is a technique that is frequently used for hierarchical hardware design. It allows reuse of lower-level hardware blocks, and per-module synthesis of complex design. Current high-end FPGA design flows support this modular design for complex, high-performance applications. We have developed a design flow based on existing FPGA synthesis tools that achieves precise control of placement as well as routing in modular designs. While this technique can be useful in many different situations, we applied it in this work to the placement and routing of the secure logic style developed in previous section.

In current FPGA design flows, the above procedure is described as the creation of a "hard macro". Present tools only preserve the relative placement of CLB logic function, but not the interconnections [13]. As a result, the relocation of a hard macro to another place in the FPGA fabric will lose guaranteed timing properties. In addition, the process of making a hard macro for a large functional module requires manual interaction of the designer in an FPGA editor to manually remove a design from physical input/output pin constraints. This process soon becomes laborious and impractical as the design size increases. Our technique,

on the other hand, offers an easy, automatic and transparent copy process of modules for the designer.

3.4.1 Secure Routing Design Flow

Figure 3.8 presents the symmetrical routing design methodology for FPGA. All tools used in this flow are standard FPGA synthesis tools. We used Xilinx FPGA synthesis suite (Xilinx ISE Foundation). The design flow requires 3 design iterations, including synthesis, place and route. In between the iterations, we added two more processing steps.

The first iteration only synthesizes the initial arbitrary single-ended module to be relocated or duplicated, and produces a LUT-level netlist. The design module is synthesized as "closed" design in the sense that all input/output ports have been disconnected from I/O pins.

Before the second iteration, the module netlist is preprocessed by adding identifier tags to component and net names. This is useful for quickly identifying which components and nets are part of the initial single-ended module. The module is area-constrained within the FPGA fabric. This can be done automatically based on the structure of the target FPGA platform or manually using floorplanner software. The second iteration then produces a fully closed but routed module based on the area constraints. The module netlist is then converted to an ASCII representation for post-processing.

Figure 3.9 and Figure 3.10 are examples of design in its ASCII representation. Figure 3.9 shows that within each slice, up to 2 LUT configuration can be specified, their values can be directly modified in ASCII and converted back to Xilinx design formats. Associated with each slice are 2 parameters which determines the location of the slice on the FPGA fabric: Slice Location and CLB Location. In Figure 3.10, specific routing of each net is shown. As can be seen, each net routed is defined by the Programmable-Interconnect-Point(PIP) it passes through. The PIP are then specified by the CLB locations indicating which CLB on the fabric they belong to.

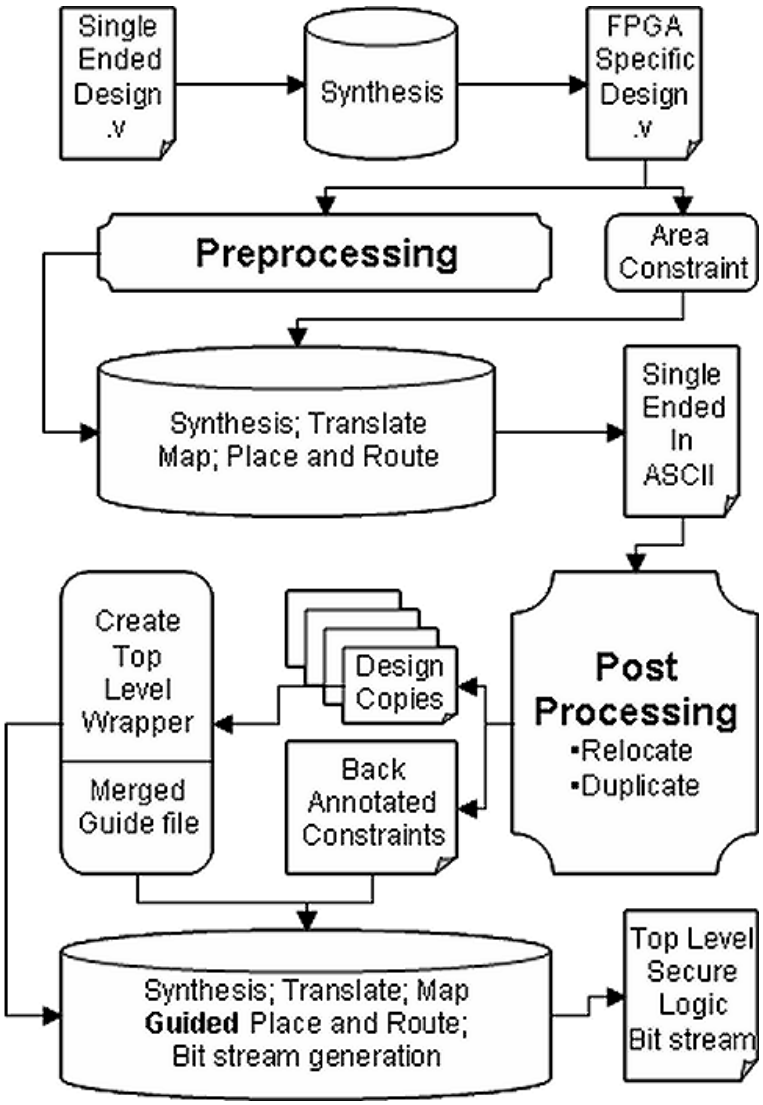
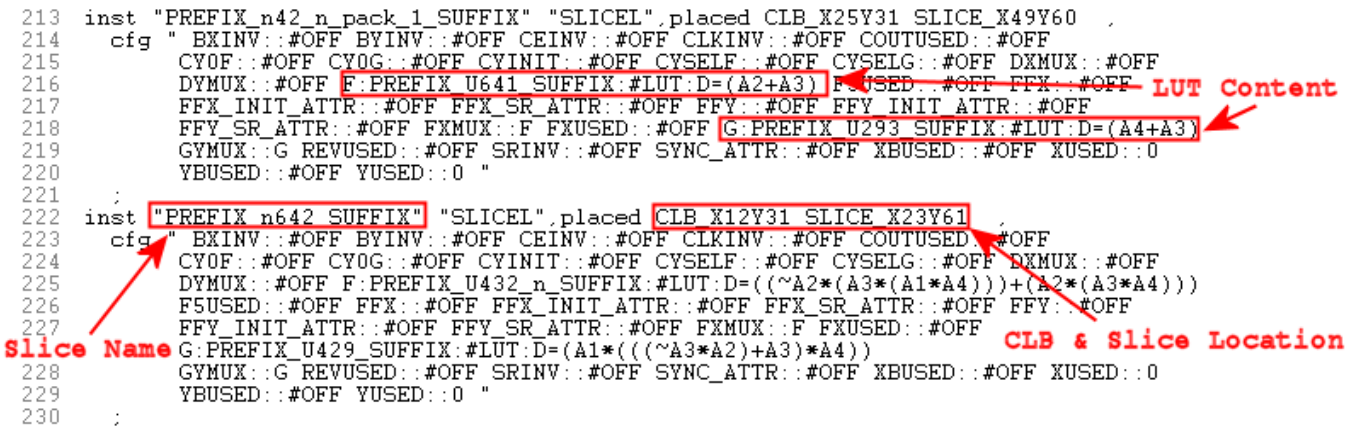
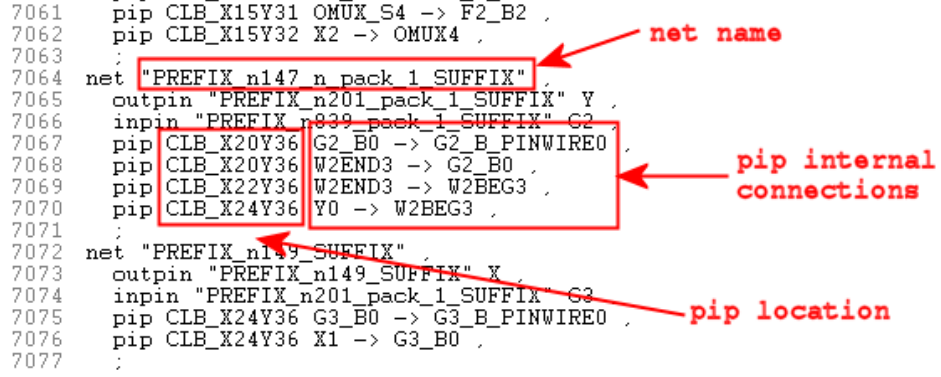


Figure 3.8: Symmetrical Routing Flow




```
inst "PREFIX_n42_n_pack_1_SUFFIX" "SLICEL",placed CLB_X25Y31 SLICE_X49Y60 ,
  cfg " BXINV::#OFF BYINV::#OFF CEINV::#OFF CLKINV::#OFF COUTUSED::#OFF
       CY0F::#OFF CY0G::#OFF CYINIT::#OFF CYSELF::#OFF CYSELG::#OFF DXMUX::#OFF
       DYMUX::#OFF F:PREFIX_U641_SUFFIX:#LUT:D=(A2+A3) F5USED::#OFF FFX::#OFF
       FFX_INIT_ATTR::#OFF FFX_SR_ATTR::#OFF FFY::#OFF FFY_INIT_ATTR::#OFF
       FFY_SR_ATTR::#OFF FXMUX::F FXUSED::#OFF G:PREFIX_U293_SUFFIX:#LUT:D=(A4+A3)
       GYMUX::G REVUSED::#OFF SRINV::#OFF SYNC_ATTR::#OFF XBUSED::#OFF XUSED::0
       YBUSED::#OFF YUSED::0 "
  ;
inst "PREFIX_n642_SUFFIX" "SLICEL",placed CLB_X12Y31 SLICE_X23Y61 ,
  cfg " BXINV::#OFF BYINV::#OFF CEINV::#OFF CLKINV::#OFF COUTUSED::#OFF
       CY0F::#OFF CY0G::#OFF CYINIT::#OFF CYSELF::#OFF CYSELG::#OFF DXMUX::#OFF
       DYMUX::#OFF F:PREFIX_U432_n_SUFFIX:#LUT:D=((~A2*(A3*(A1*A4)))+(A2*(A3*A4)))
       F5USED::#OFF FFX::#OFF FFX_INIT_ATTR::#OFF FFX_SR_ATTR::#OFF FFY::#OFF
       FFY_INIT_ATTR::#OFF FFY_SR_ATTR::#OFF FXMUX::F FXUSED::#OFF
       G:PREFIX_U429_SUFFIX:#LUT:D=(A1*(((~A3*A2)+A3)*A4))
       GYMUX::G REVUSED::#OFF SRINV::#OFF SYNC_ATTR::#OFF XBUSED::#OFF XUSED::0
       YBUSED::#OFF YUSED::0 "
  ;
```

Figure 3.9: ASCII representation of logic in slices




```
net "PREFIX_n147_SUFFIX" ,
  outpin "PREFIX_n147_SUFFIX" X ,
  inpin "PREFIX_n646_pack_1_SUFFIX" F2 ,
  pip CLB_X15Y31 F2_B2 -> F2_B_PINWIRE2 ,
  pip CLB_X15Y31 OMUX_S4 -> F2_B2 ,
  pip CLB_X15Y32 X2 -> OMUX4 ,
  ;
net "PREFIX_n147_n_pack_1_SUFFIX" ,
  outpin "PREFIX_n201_pack_1_SUFFIX" Y ,
  inpin "PREFIX_n839_pack_1_SUFFIX" G2 ,
  pip CLB_X20Y36 G2_B0 -> G2_B_PINWIRE0 ,
  pip CLB_X20Y36 W2END3 -> G2_B0 ,
  pip CLB_X22Y36 W2END3 -> W2BEG3 ,
  pip CLB_X24Y36 Y0 -> W2BEG3 ,
  ;
net "PREFIX_n149_SUFFIX" ,
  outpin "PREFIX_n149_SUFFIX" X ,
  inpin "PREFIX_n201_pack_1_SUFFIX" G3 ,
  pip CLB_X24Y36 G3_B0 -> G3_B_PINWIRE0 ,
  pip CLB_X24Y36 X1 -> G3_B0 ,
  ;
```

Figure 3.10: ASCII representation of routing switchbox

During post-processing, the module can be modified in ASCII in the following two ways:

- Relocation: We can modify the module by changing the location of every component and routing resource used. The module can move freely within the FPGA fabric as long as the target location provides sufficient CLB and routing resources, and as long as the area constraints do not move out of bound.
- Duplication: We can also create a second module based on the design information in the first one. Internal components, net names and LUT contents can be modified while creating this second module. If both modules exist in the final system, we need to make sure they do not overlap.

Post-processing also extracts design information that will be back-annotated into original constraints file. This is necessary for the following reason: Overall place and route heavily depends on the actual instance and net name used. Instance name such as "Slice Name" depends on the name of the second (bottom) outgoing wire of the slice. Back-annotating placement information into a constraint file forces later synthesis steps to place the correct LUT and its associated wire into the bottom location within a slice. This maintains consistency between two separate synthesis run, which is required for successful final routing.

With the post-processing step, an arbitrary number of duplications can be produced as long as the overall design still fits on the FPGA fabric.

In the third iteration, we need to complete the routing of the closed modules to input/output pins. Each module is instantiated in a top level file, which acts as a wrapper to external logic blocks. The top design then goes through a regular synthesis flow using the complete constraints generated during post processing. The design is placed and routed using the combined design guide file after post-processing.

Following the copy-modify-and-paste approach explained in section 3.1, we can duplicate the asymmetric WDDL just developed to create an overall symmetrical secure circuit; we

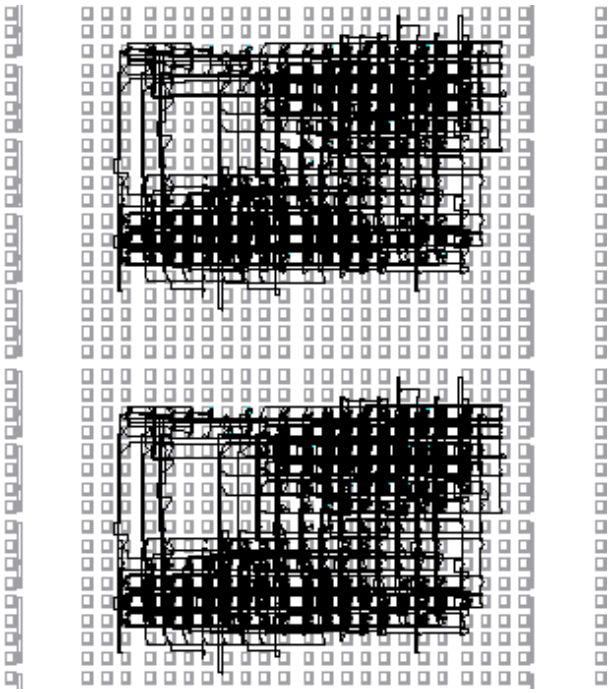


Figure 3.11: Example of symmetrically routed and duplicated overall design

call it, simply, double WDDL (DWDDL) logic, see Figure 3.11. During the duplication, we can switch the contents of direct and complementary LUT for the second duplicated module. This way, the LUT at the exact same location of both modules will form a direct and complementary pair and the nets after such pairs become complementary nets that are routed exactly the same way, as shown in Figure 3.7.

Chapter 4

Results and Power Analysis

To validate the approach of constant-power circuits, we implemented a sample design on a Digilent Spartan3E Starter board. The sample design in Figure 4.1 includes a cryptographic S-Box which is driven with test patterns out of an 8-bit Linear Feedback Shift Register (LFSR). The S-Box table was taken from the advanced encryption standard (AES). The output of the S-box is combined with a secret key-byte to produce the output samples.

4.1 Measurement Equipment, Setup and Methodology

Table 4.1 provides a summery of the equipment we used for measurement. We used an Agilent DSO3062A digital storage oscilloscope which has a bandwidth of 60Mhz with maximum sampling rate of 1Gsample/sec. To obtain enough sample points per cycle, we lowered our circuit speed to 5MHz.

We made measurement on a Digilent Spartan3E Starter board. We bypassed the onboard power control IC and used an external linear-mode power supply for minimal power line noise. We then attached a 1mV/mA current probe in series with the power supply to convert current consumption variation into voltage variation to be measured on the oscilloscope. Note that

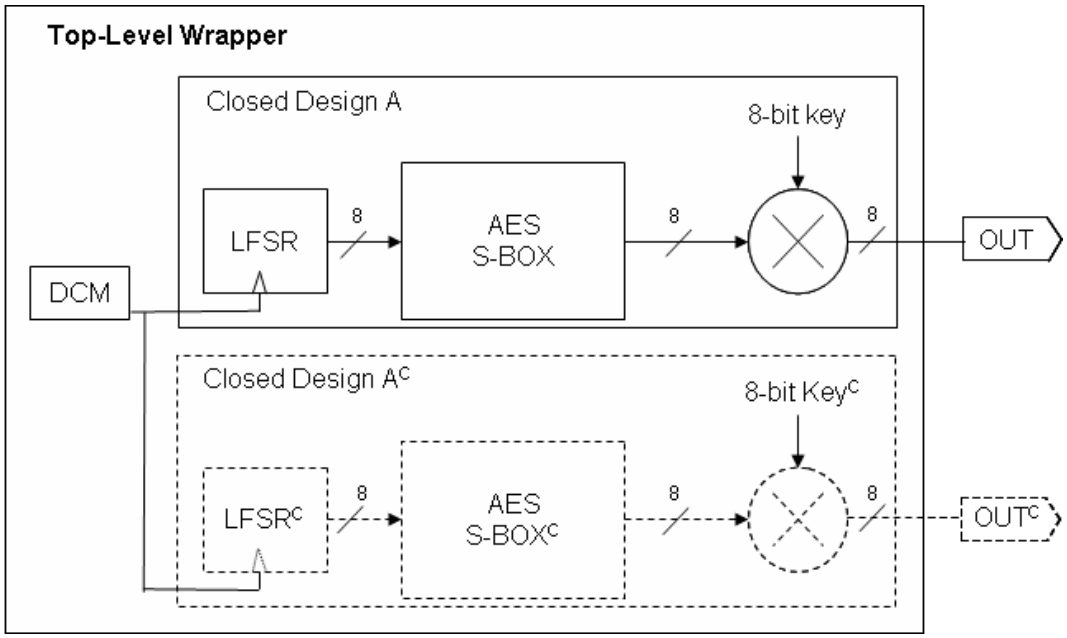


Figure 4.1: Sample Design Block Diagram

Type	Model	Characteristics
Power Supply	B&K Precision	Linear-mode PSU
Oscilloscope	Agilent DSO3062A	60Mhz BW, 1 Gsample/sec
Current Probe	Tektronix CT-2	freq. resp. 1.2kHz - 200MHz
Test Board	Digilent Spartan3E	decoupling capacitors de-soldered

Table 4.1: Test Equipment Summary

the current probe used has a bandpass frequency response that will reject any DC signal, but will pass variable AC signals from 1.2kHz to 200MHz.

Most Xilinx FPGA needs 3 power supplies: one for IO blocks on the periphery of FPGA; one for auxiliary components such as digital clock manager(DCM); the last one for internal FPGA fabric such as logic and routing resources. We only need to measure the power consumption variation for the internal logic power supply. We also removed all the decoupling capacitors on the internal fabric power input of the FPGA to maximize the success of detecting power consumption variations.

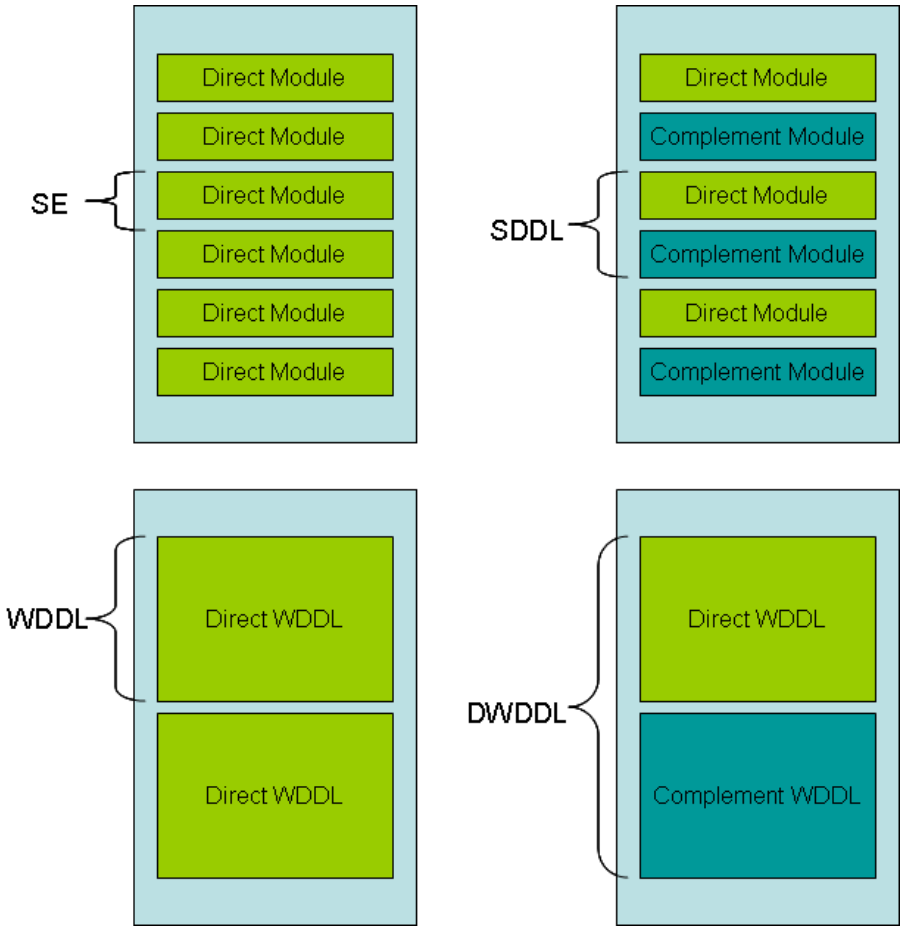


Figure 4.2: Four Test Setups

Note that a resistor is not used for measurement because using it creates a negative feedback system. When current going through a resistor increase, the voltage drop across it increases. With power supply fixed at certain value, the voltage at the FPGA power input will decrease. If FPGA power supply decreases, the current drawn by the FPGA will decrease. This negative feedback loop requires us to choose carefully the value of the resistor to use in our test setup. Bigger resistor causes bigger variations while small resistor won't cause enough voltage drop for the signal to be registered on an oscilloscope. Thus, a resistor is not used in our setup.

Following the design idea of Figure 4.1, we prepared 4 different test cases for comparison,

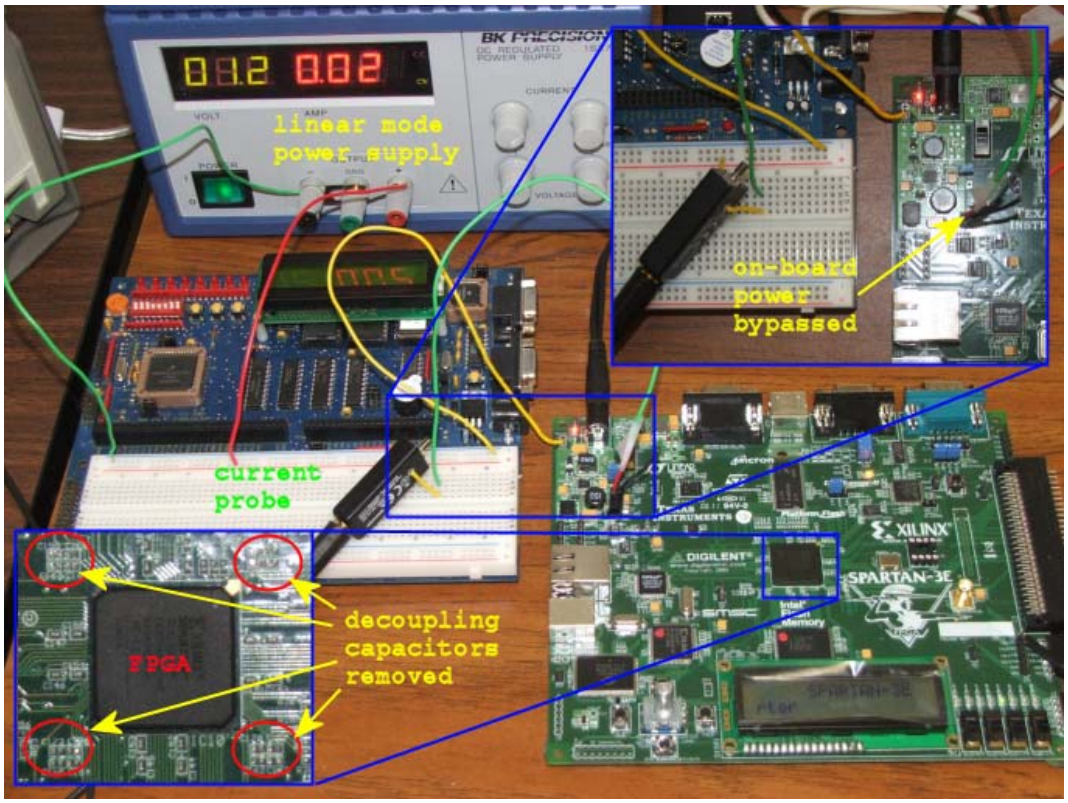


Figure 4.3: Test Equipment Connections

as shown in Figure 4.2. In order to consume enough current for the oscilloscope to detect any signal, we need to increase power consumption. Thus, we placed multiple designs on the same module to increase the magnitude of power consumption. This will not affect our ability to mount power attacks since the attacks we mount is sensitive to the shape of variation rather than the magnitude of the variation. With this in mind, the first setup is half-SDDL, only the direct logic is used, we call it Single-Ended design and we placed 6 of them on the FPGA fabric. The second setup is a SDDL version and we placed 3 of them on the fabric. The third one contains 2 non-routing-aware WDDL modules while the fourth one is a symmetrically routed DWDDL setup. All test cases use an 8-bit fixed key with hex value of 8'hAE (decimal 174).

After test setups are ready, we load them onto the FPGA one at a time to make measurements. The equipments are connected as shown in Figure 4.3.

4.2 Power Variation Measurement Results

Figure 4.4 shows the current variation for the first 100 cycles in a 255-cycle period. A 10-cycle snapshot, corresponding to the shaded area, is shown in the inset. The single-ended (SE) variation is over 10 times larger than those of other test cases and is scaled down to fit inside of the inset. SDDL variation is also scaled down by 1/2 to fit inside the inset. Visually, the variations of DWDDL look smaller than those of WDDL. Both of them are much smaller than the variations of SE and SDDL.

To test the effectiveness of our SDDL and DWDDL secure circuits, we mounted a differential-power-analysis (DPA) attack for all four cases.

4.3 DPA-Attack Results

In a DPA attack, we estimate the circuit power consumption using a power model that combines the observed output values (OUT) with an estimate for the key value (KEYguess). We then correlate the estimated power, for each key guess, with the measured power. The highest correlation over all key guesses will return the correct key. For the single-ended and SDDL design, our power model is the Hamming weight (Hw) of the input value of the Sbox. This value depends on the key guess and the output, since

$$Hw(IN) = Hw(SBOX^{-1}(OUT \bigotimes KEY_{guess})); \tag{4.1}$$

where Hw() represents the Hamming Weight function. The resulting correlation of the Single-Ended and SDDL design over all key guesses 0 ... 255 are shown in Figure 4.5 and Figure 4.6, respectively. The DPA-attack on both cases results in a very sharp correlation peak at key guess 174. This means that the single-ended and SDDL design can be easily broken.

WDDL and DWDDL are both dual-rail circuits. Side-channel leaks on these circuits are caused by imbalanced routing. We therefore mounted a DPA attack on WDDL and DWDDL

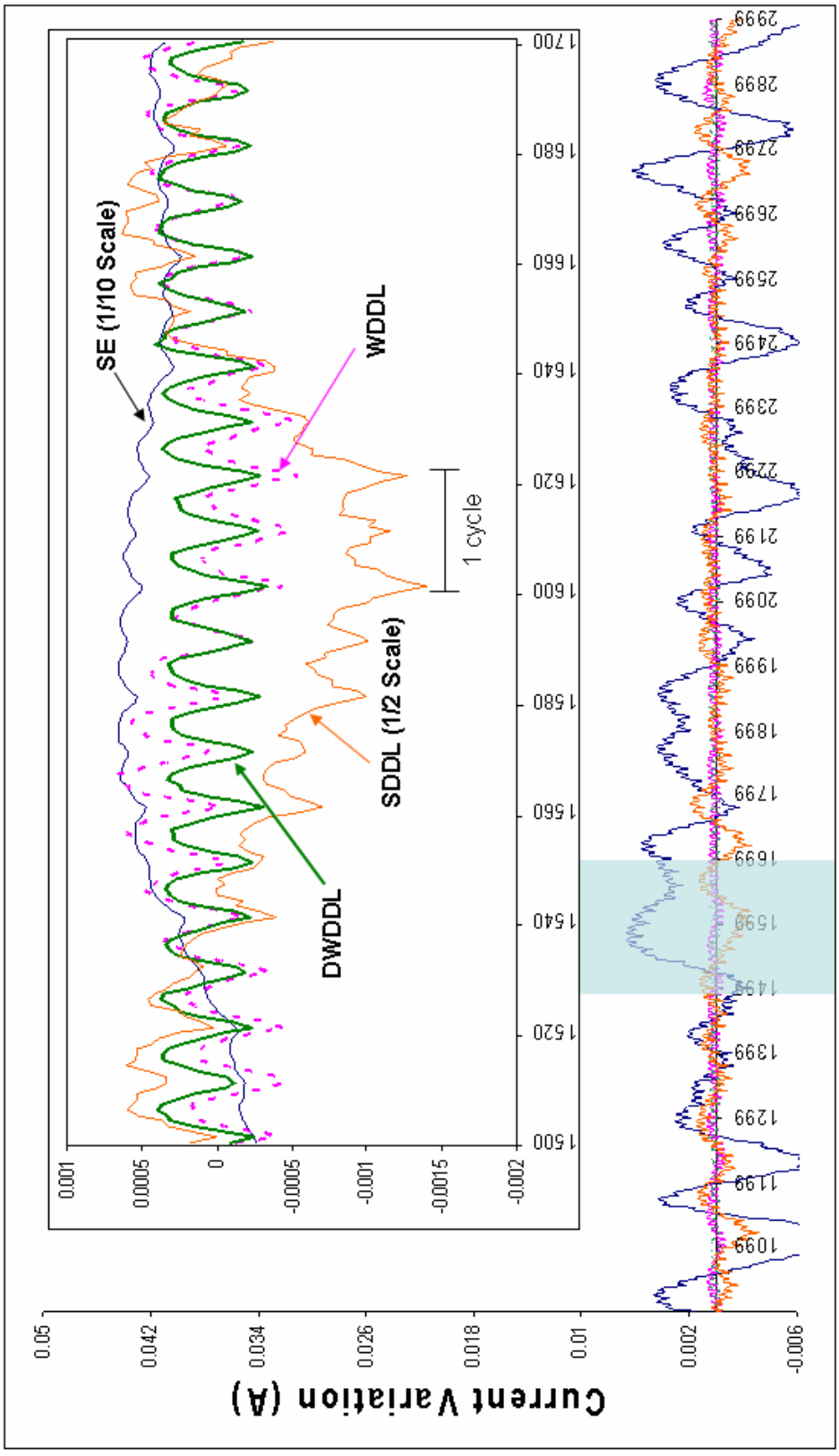


Figure 4.4: Power Variation Measurements

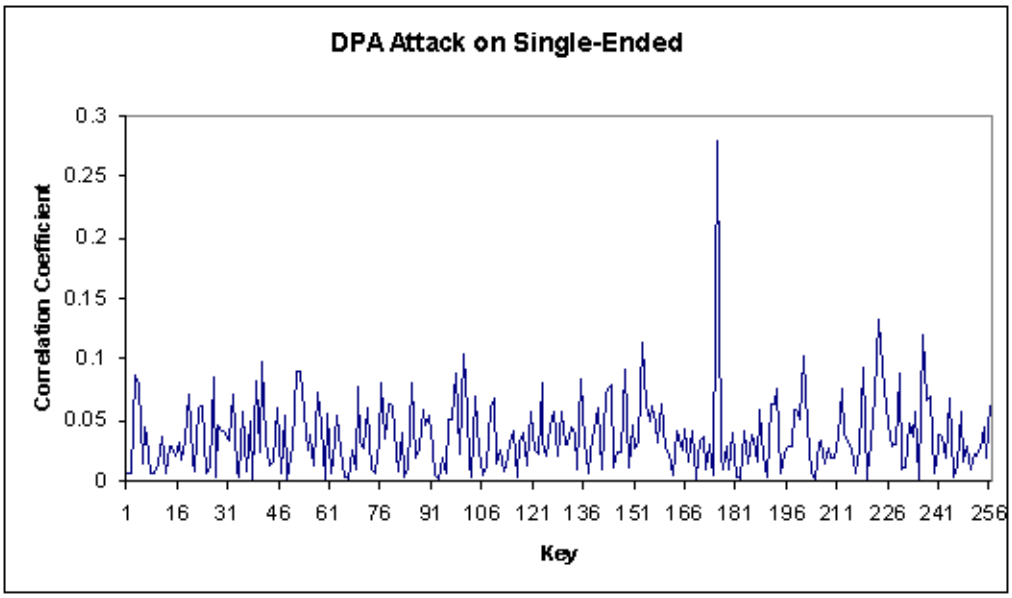


Figure 4.5: DPA Attack on Single-Ended

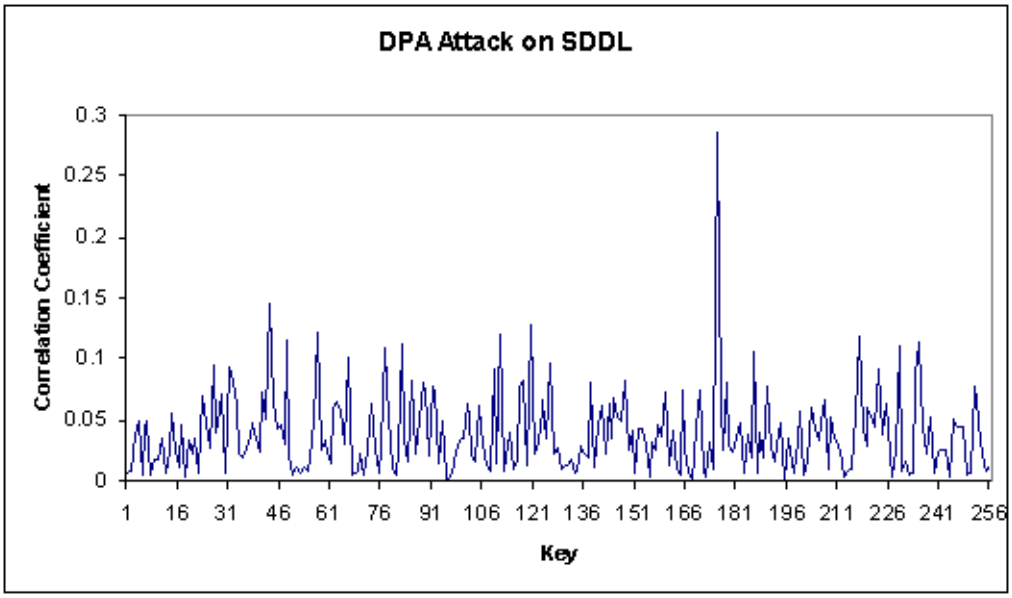


Figure 4.6: DPA Attack on SDDL

that exploits those imbalances. Our power model in this case is the Hamming Weight of a single bit on the input of an SBOX.

$$Hw(bit(IN, i))|_{i:0...7} = Hw(bit(SBOX^{-1}(OUT \bigotimes KEY_{guess}), i))|_{i:0...7} \tag{4.2}$$

Indeed, the single-bit power model allows each bit of the byte-wide circuit to have a different routing imbalance and a different leakage. We then obtain a correlation plot for each of the 8 bits, find the maximum in each plot and use majority voting among all bits to arrive at the final key.

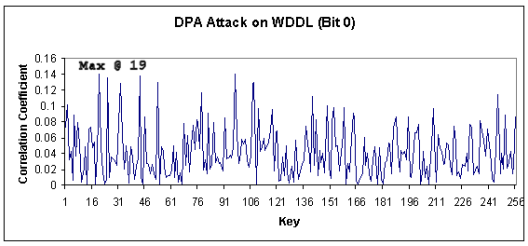
Figure 4.7 and Figure 4.8 shows these correlation plots for WDDL and DWDDL, respectively. By majority vote on bit 3, 4, 6 and 7 for WDDL, we are able to obtain the correct key value for WDDL. In contrast, we cannot find a correct key for DWDDL. All bits lead to a different key in the correlation process, all of them equally likely.

In our measurements, we obtained 20 sample points per clock cycle. The above attacks for WDDL and DWDDL are mounted using the middle sample, sample 10.

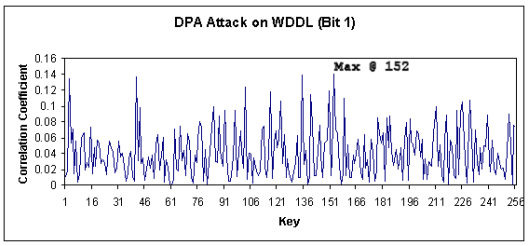
Figure 4.9(a) shows the 3D mesh plot of the key value obtained from attacking on each of the 20 samples points. The table in Figure 4.9(b) shows the numerical values. From Figure 4.9(c), we can see that most key value obtained settle on key value 174.

Figure 4.10(a), 4.10(b), and 4.10(c) showed the same set of information for the DWDDL cases. As can be seem, the key value extracted are very random, with no single outstanding value dominating. In fact, from Figure 4.10(c), the key value extracted most often is actually 113, which is an incorrect key.

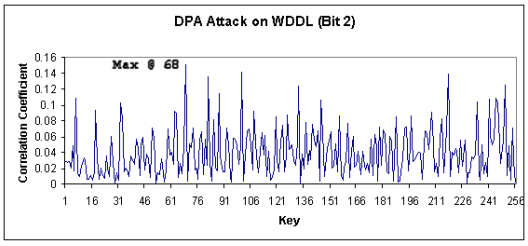
All these data further proved that WDDL on FPGA does not withstand a DPA attack, while DWDDL does.



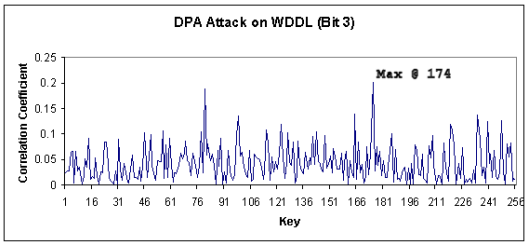
(a) DPA Attack on Bit 0



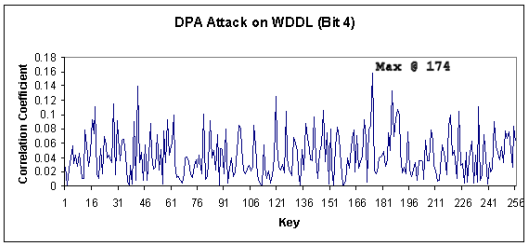
(b) DPA Attack on Bit 1



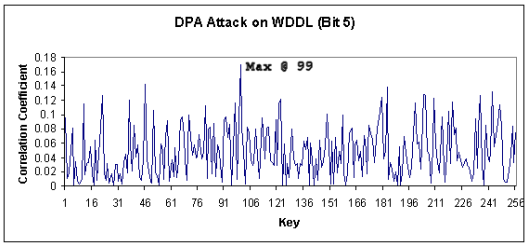
(c) DPA Attack on Bit 2



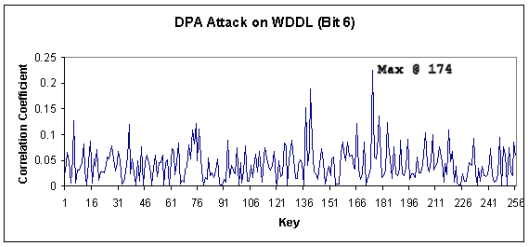
(d) DPA Attack on Bit 3



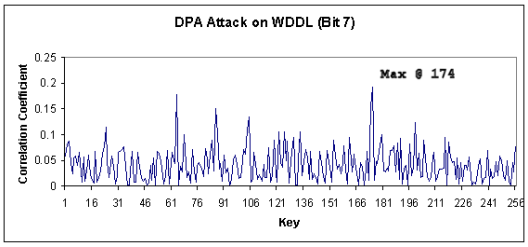
(e) DPA Attack on Bit 4



(f) DPA Attack on Bit 5

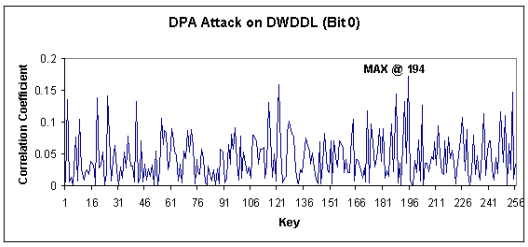


(g) DPA Attack on Bit 6

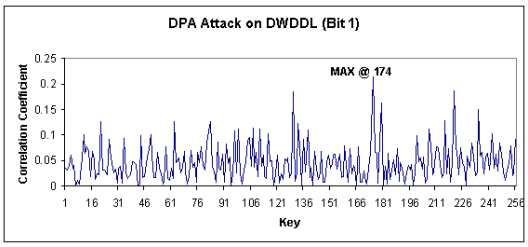


(h) DPA Attack on Bit 7

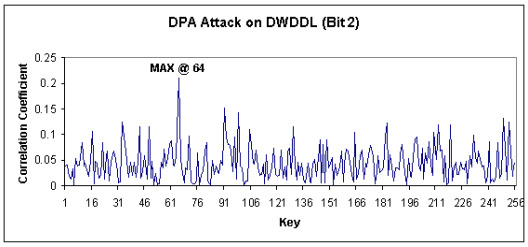
Figure 4.7: DPA Attack on WDDL @ Sample 10



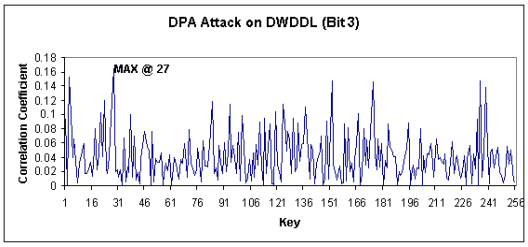
(a) DPA Attack on Bit 0



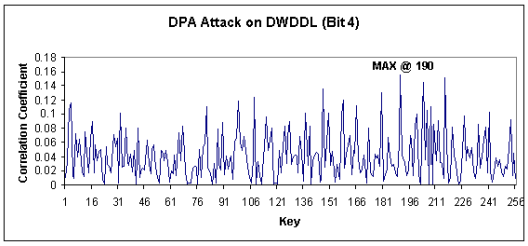
(b) DPA Attack on Bit 1



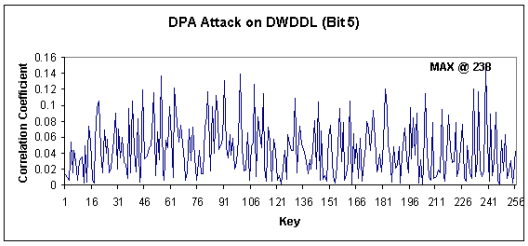
(c) DPA Attack on Bit 2



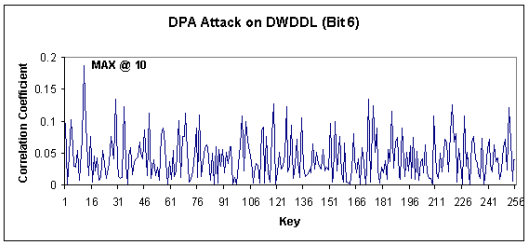
(d) DPA Attack on Bit 3



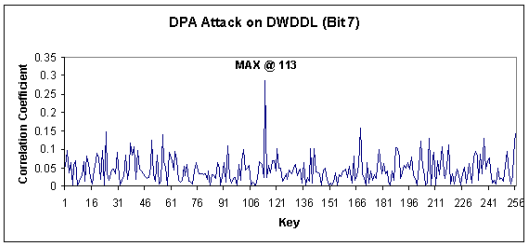
(e) DPA Attack on Bit 4



(f) DPA Attack on Bit 5

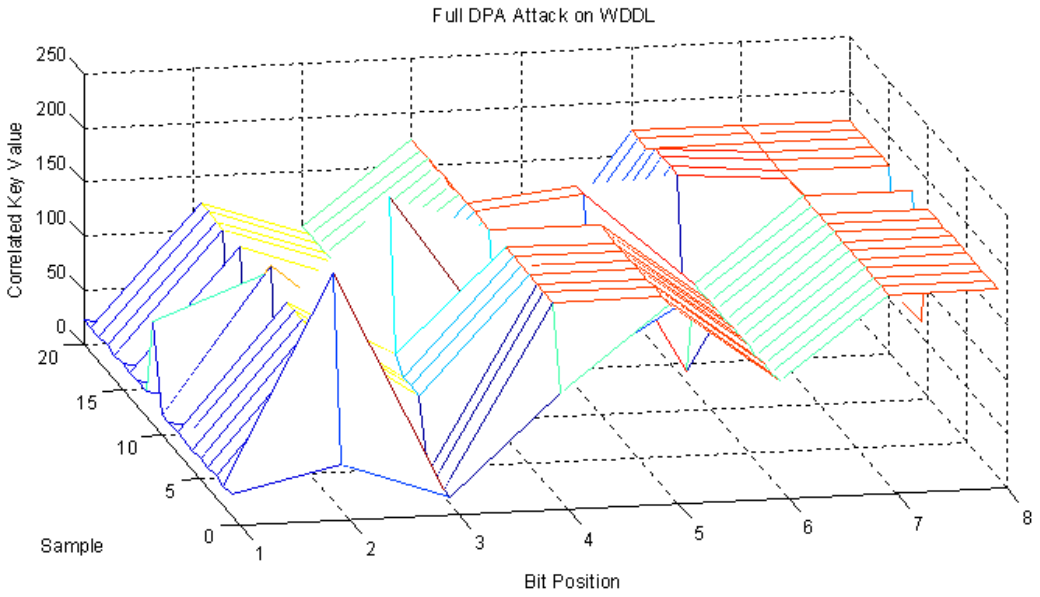


(g) DPA Attack on Bit 6



(h) DPA Attack on Bit 7

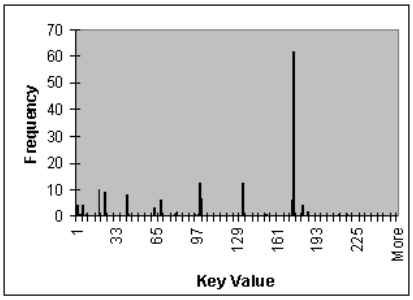
Figure 4.8: DPA Attack on DWDDL @ Sample 10



(a) 3D Mesh Plot of DPA Attack on WDDL

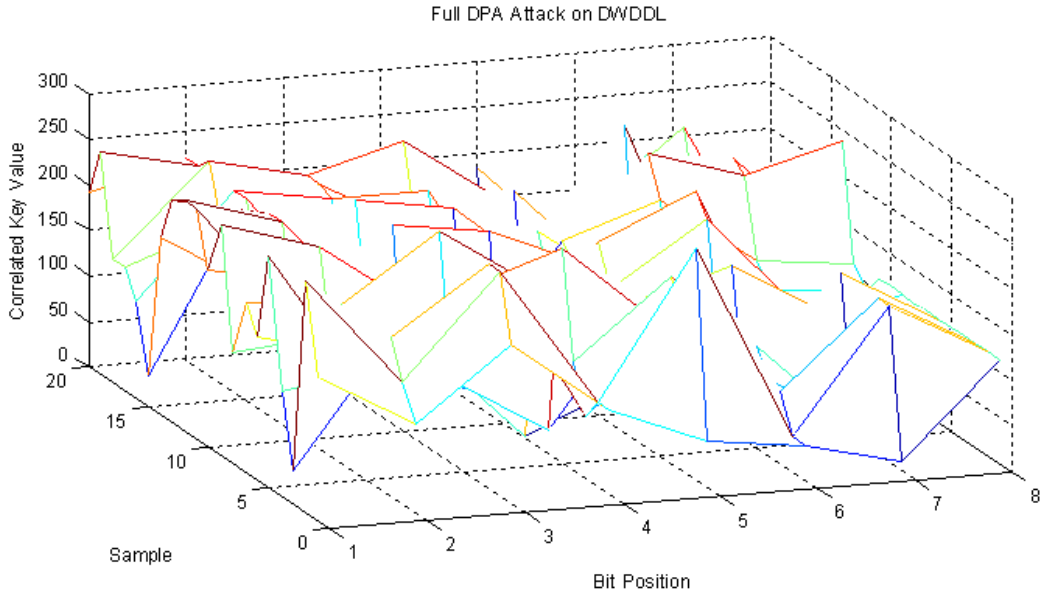
Sample	Bit Position							
	0	1	2	3	4	5	6	7
0	19	42	6	98	174	99	174	174
1	19	210	6	174	174	99	174	174
2	19	134	6	174	174	99	174	174
3	19	134	6	174	174	99	174	174
4	19	134	68	174	174	99	174	174
5	19	134	68	174	174	99	174	174
6	19	134	68	174	174	99	174	174
7	19	134	81	174	174	99	174	174
8	19	2	217	79	174	99	174	174
9	19	152	68	174	174	99	174	174
10	96	134	81	174	41	99	174	63
11	24	2	68	174	185	99	174	174
12	24	2	68	174	185	10	174	63
13	24	134	100	174	41	182	174	63
14	24	2	100	174	41	182	174	173
15	24	134	100	174	41	182	174	173
16	24	134	100	174	41	182	174	173
17	24	134	100	174	41	174	174	173
18	24	134	100	174	41	174	174	173
19	24	4	100	174	41	174	174	173

(b) Data Points of DPA on WDDL



(c) Histogram of WDDL DPA Data Point

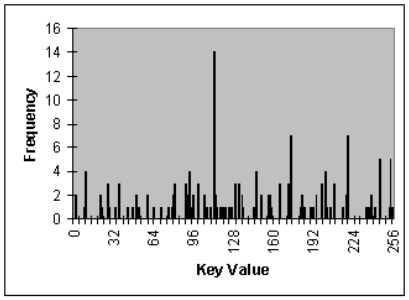
Figure 4.9: Full DPA Attack on WDDL



(a) 3D Mesh Plot of DPA Attack on DWDDL

Sample	Bit Position							
	0	1	2	3	4	5	6	7
0	156	96	174	93	50	36	10	113
1	253	132	245	93	252	36	172	113
2	36	174	245	70	159	79	172	113
3	115	132	174	119	205	81	28	113
4	253	96	199	235	146	109	10	113
5	156	129	90	27	165	44	172	113
6	185	174	245	174	146	181	118	113
7	121	132	105	2	50	81	100	113
8	253	220	144	93	146	215	100	113
9	194	174	64	27	190	238	10	113
10	253	129	218	201	157	199	10	113
11	253	220	105	22	146	99	100	92
12	245	220	94	22	165	199	125	113
13	194	129	218	201	114	241	209	237
14	34	220	90	93	157	81	219	113
15	107	220	202	201	27	255	9	92
16	135	174	209	150	183	80	47	113
17	135	234	209	238	165	52	127	59
18	245	220	90	201	2	136	219	23
19	191	220	114	150	183	80	76	59

(b) Data Points of DPA on DWDDL



(c) Histogram of DWDDL DPA Data Point

Figure 4.10: Full DPA Attack on DWDDL

4.4 Result Analysis

In summary, only our DWDDL secure logic implementation provided adequate protection against DPA attacks.

We were able to determine the failure of SDDL logic design. Because of the separation of complementary logics, we have to allow the existence of inverting negative logic in our LUT configuration. This in turn created a situation where during the evaluation phase, glitches are produced. Through further simulation, we determined that glitches with width greater than 1000ps will be transported. Any glitches with width less than 1000ps will be suppressed by inertial delay of the gate. Furthermore, because of the fact that SDDL can be successfully attacked, we can conclude that the glitches between the complementary modules are not in sync, meaning glitches might appear on the direct net and not on the complementary net, vice versa. These create an imbalance on the number of switching activities for the complementary nets, thus leaking power consumption information.

DWDDL, on the other hand, is based on WDDL, which removed all inverting logics. This does not necessarily translates to no glitches on the LUT outputs. However, since DWDDL uses precharging phases to bring every net to logic-0, this pattern of evaluations translated to exactly 1 single switching per cycle for WDDL or exactly 2 switchings per cycle for DWDDL, guaranteed.

From all these observations, we can safely conclude that in order to build a secure logic on FPGA, our circuit needs to have two most important properties:

- No Glitch: glitches destroys the perfect balance on the number of switching activities. We need an all positive logic logic style, such as WDDL to achieve this.
- Symmetrical Routing: asymmetrically routed design that has no glitches such as WDDL on FPGA from our test case, do not provide any higher security than a design that produces glitches, such as our SDDL design. To achieve symmetry in routing of

Design	Glitch	Symmetrical Routing	Security
SE	Yes	No	**No**
WDDL	No	No	**No**
SDDL	Yes	Yes	**No**
DWDDL	No	Yes	**YES**

Table 4.2: Test Cases Security Summary

Design	Slice Count	Delay(ns)
Original-SE	70	3.99
SE-Precharge	197	11.49
WDDL	409	19.54
SDDL	394	11.49
DWDDL	818	20.88

Table 4.3: LFSR+SBOX Implementation Results

complementary modules, our symmetrical routing technique is absolutely necessary.

Table 4.2 contains a summary of design requirements for designing secure logic on FPGA.

4.5 Implementation Summary

All secure-logic implementations incurred considerable area and delay overhead, as demonstrated in 4.3. As shown in the table, both WDDL and DWDDL are considerably bigger and slower than the most compact implementation of a single-ended SBOX+LFSR. DWDDL is essentially twice the size of WDDL as expected, but in return, our DWDDL technique is the ONLY module that withstands DPA attacks.

Chapter 5

Conclusion and Future Work

As seen from actual measurement and actual DPA attacks, our approach to implementing secure logic for FPGA can achieve a better result than the current, non-routing aware WDDL approach. The tradeoff between area/performance overhead to the level of security obtained will have to be considered by the circuit designer based on the overall system requirement.

From here, we can identify some of the future tasks or research directions:

1. Overhead Improvement: This is an immediate task. We think the area and delay overhead can be potentially reduced. The packing done by standard synthesis tool might need to be improved, through development of specialized tools.

2. Logic Style Development: This is a near-future task. A thorough evaluation of possible methods that can remove glitches at design- and synthesis-stage should be investigated. Possible alternatives are to develop logic style using asynchronous logic such as asynchronous Muller gates. If this is possible, our SDDL approach might be a better alternative to DWDDL approach.

3. Application of DWDDL: This is a long-term task. We can apply this secure logic flow to other applications. For example, in [14], a methodology has been developed to

convert a hardware design into software and executing the software on a network of Picoblaze processors. This provide a way to evaluate the performance of cryptographic algorithm implemented on hardware or software or a set of software running on multi-core processor networks. Applying DWDDL to Picoblaze processor, we can transform the network of Picoblaze processors to a network of secure Picoblaze processors, the software running on them are thus immune to low-level power analysis attacks.

Bibliography

[1] M. Witteman, "Advances in smartcard security," *Information Security Bulletin*, pp. 11–12, July 2002.

[2] M. Bond and R. Anderson, "Api-level attacks on embedded systems," in *IEEE Computer Magazine*, vol. 34, no. 10, 2001, pp. 67–75.

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology: Proceedings of CRYPTO'99*, ser. LNCS, M. Wiener, Ed., vol. 1666. Springer-Verlag, 1999, pp. 388–397.

[4] D. J. Bernstein, "Cache-timing attacks on aes," Online, http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[5] S. Skorobogatov, "Semi-invasive attacks: a new approach to hardware security analysis," University of Cambridge, Tech. Rep. UCAM-CL-TR-630, April 2005.

[6] H. Bar-El, "Known attacks against smartcards," Discretix Technology, Tech. Rep., http://www.discretix.com/PDF/Known

[7] E. Oswald, S. Mangard, C. Herbst, and S. Tillich, "Practical second-order dpa attacks for masked smart card implementations of block ciphers," in *RSA Conference Cryptographers Track*, ser. LNCS, vol. 3860. Springer, 2006, pp. 192–207.

[8] D. Suzuki, M. Saeki, and T. Ichikawa, "Random switching logic: A countermeasure against dpa based on transition probability," Tech. Rep., 2004, cryptology ePrint Archive, Report 2004/346.

[9] K. Tiri and P. Schaumont, "Changing the odds against masked logic," in *13th Annual Workshop on Selected Areas in Cryptography*, 2006.

[10] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Design and analysis of dual-rail circuits for security applications," in *IEEE Transactions on Computers*, vol. 54, no. 4, April 2005, pp. 449–460.

[11] K. Tiri and I. Verbauwhede, "A digital design flow for secure integrated circuits," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 7, July 2006, pp. 1197–1208.

[12] ——, "Place and route for secure standard cell design," in *6th International Conference on Smart Card Research and Advanced Applications(CARDIS)*, August 2004, pp. 143–158.

[13] "How to create a hard macro," Online, http://www.cse.ucsc.edu/classes /cmpe225/Fall01/hardmacro.html.

[14] P. Yu and P. Schaumont, "Executing hardware as parallel software for picoblaze networks," in *16th International Conference on Field Programmable Logic and Applications*, 2006.

Vita

Pengyuan Yu was born on April 5th, 1983 in Shenyang, China. He received his Bachelor of Science degree in Electrical Engineering from University of California, San Diego, in June 2005.

He started working toward his Master of Science degree in Computer Engineering from August 2005 at Virginia Tech and joined Secure Embedded Systems Group at that time. In April 2007, He completed all the requirements and obtained his Master of Science degree.

His research interests include design methodology and secure embedded codesign. His most recent research topic was on implementing side-channel resistant circuit on FPGA.

Publications:

- **(Under Review)** P. Yu, P. Schaumont, "Secure FPGA Circuits using Controlled Placement and Routing," CODES 2007.
- P. Yu, P. Schaumont, "Executing Hardware as Parallel Software for Picoblaze Networks," 16th International Conference on Field Programmable Logic and Applications (FPL 2006), Madrid, Spain, August 2006.
- P. Yu, P. Schaumont, D. Ha, "Securing RFID with Ultra-Wideband Modulation," 2nd Workshop on RFID Security (RFIDSec 2006), Graz, Austria, July 2006.