Securing Light Weight Cryptographic Implementations on FPGAs Using Dual Rail with Pre-Charge Logic

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science at George Mason University

By

Rajesh Velegalati
Bachelor of Science
SIR C.R.R College of Engineering, 2006

Director: Dr. Jens-Peter Kaps, Professor
Department of Electrical and Computer Engineering

Summer Semester 2009
George Mason University
Fairfax, VA

Dedication

I dedicate this thesis to my parents, Gopal and Janaki Velegalati. Thanks for supporting me through various stages of my life.

Acknowledgments

There are many people who have helped me through various stages of my thesis. First of all, to my advisor Dr. Jens-Peter Kaps without whom this thesis would not be possible. Thanks for taking me as your student. I am really looking forward to do my Ph.D under you!!

I would like to thank Dr. David Hwang and Dr. Kris Gaj for their valuable support and suggestion. I would also like to thank Dr. Alok Berry for patiently listing to my presentation and providing valuable comments Additionally in no particular order i would like to acknowledge the following people:

- The people at CERG, for providing a stimulating environment and also for goofing around with me !!!
- My friends Sabari, Mahi, Venu, and Panci for supporting me through this thesis.
- Mrs.Kaps, the curry you made was awesome!
- To Xilinx tools developers, for making my life interesting!!!
- To Nick Ton who helped me a lot, and made my work much simpler.

Table of Contents

List of Tables

List of Figures

Abstract

SECURING LIGHT WEIGHT CRYPTOGRAPHIC IMPLEMENTATIONS ON FPGAS USING DUAL RAIL WITH PRE-CHARGE LOGIC

Rajesh Velegalati, MS

George Mason University, 2009

Thesis Director: Dr. Jens-Peter Kaps

Recent advances in Field Programmable Gate Array (FPGA) technology are bound to make FPGAs a popular platform for battery powered devices. Many applications of such devices are mission critical and require the use of cryptographic algorithms to provide the desired security. However, Differential Power Analysis (DPA) attacks pose a severe threat against otherwise secure cryptographic implementations.

Current techniques to defend against DPA attacks such as Dual rail with Pre-Charge Logic (DPL) lead to an increase in area consumption of factor 4 or more which is not suitable for Light Weight implementations. Current secure implementations using DPL require ASIC tools and a special ASIC library. In this thesis we show that moderate security against DPA attacks can be achieved for DPL secured implementations using only FPGA CAD tools augmented by some scripts. The resulting circuit has an area increase of not much more than a factor two over standard FPGA implementations. We demonstrate our approach by implementing a cryptographic algorithm on Spartan3E FPGA and assessing the security it provides against DPA. We also study one of the Xilinx FPGA specific intrinsic features - Wide Dedicated Multiplexer (WDM) -with respect to DPA.

Chapter 1: Introduction

With ever increasing miniaturization and ubiquity of computing devices such as smart cards, WSN sensor nodes, RFIDs etc., security threats against them have become a growing concern. Smart cards can be used for identification, authentication etc., as such they will contain sensitive information like biometric data, personal information or cryptographic keys. RFIDs are used just about anywhere, from pet-tags to food-tags, from clothing-tags to missiles, anywhere that an unique identification system is needed. They are also used in hospitals to control access to drugs or even to keep track of locations of doctors and patients. The shared radio medium of the RFIDs allows for evasdropping which poses threat to individual privacy. WSN nodes are generally used for applications involving monitoring, tracking and controlling for example, WSN nodes could be deployed over a battle field to detect enemy intrusion or to control a nuclear reactor. If an adversary manges to obtain these devices either by stealing or purchasing, information inside these devices gets compromised.

Even though these devices protect confidential information using cryptographic algorithms that withstand rigorous cryptanalytic attacks, an adversary can obtain the secret information by observing the so called side channel leakage from the cryptographic device. These side channels can be power consumption [1, 2], execution time [3], or electromagnetic field [4,5] of the device. These side-channels leak information whenever the device performs an operation using the secret data. Attacks which make use of such inherent physical leakage are called side-channel attacks (SCA). SCA pose a major threat because the physical implementations of the cryptographic devices are difficult to control and often result in unintended leakage of information.

Amongst these passive non-invasive techniques, the power analysis attack has received the most amount of attention by the research community because it is very powerful, can

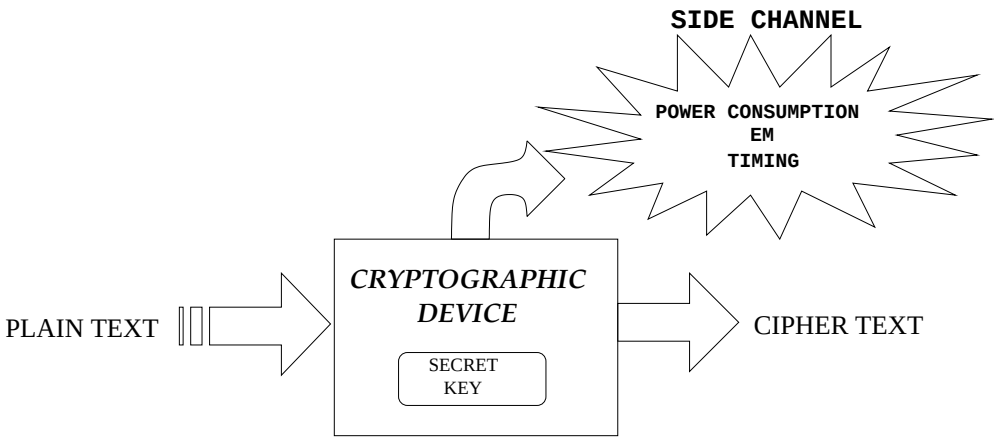


Figure 1.1: Side Channel Leakage from Cryptographic Device

easily be conducted, and has been used successful many times. It can be applied to dedicated cryptographic processors as well as to general purpose processors running a cryptographic software. The fact that ultra-low power implementations of cryptographic algorithms perform most operations in a serial fashion, in order to conserve power, makes them even more susceptible to SCA.

1.1 Motivation

Field Programmable Gate Arrays (FPGA) are a popular choice for applications where low up-front cost, fast time to market and flexibility are important. Many of those applications are based on battery powered mobile devices which require low power consumption. FPGAs are less energy efficient than ASICs or embedded processors and hence they are not often found in such applications. Recent advances in FPGA technology [6] however, will enable FPGAs to become more popular in this market.

Light weight or low area consuming implementations of cryptographic algorithms facilitate the use of smaller and hence less expensive FPGAs or enable their use in battery powered applications. Unfortunately Differential Power Analysis (DPA) attacks are a threat to otherwise secure cryptosystems on embedded devices.

The goal of this thesis is two fold. The primary goal is to introduce a secure design

flow that enables us to achieve resistance to DPA attacks through Dynamic Differential Logic(DDL) for light-weight implementations of cryptographic algorithms on FPGAs. The secondary goal is to explore the Xilinx FPGA specific intrinsic features — Wide Dedicated Multiplexer (WDM) — with respect to DPA.

1.2 Power Analysis

In the year 1999, Kocher et al. proposed power analysis attacks [1] on cryptographic devices. The authors have discovered that the power consumption of the cryptographic devices can be exploited to reveal the secret information. Power analysis attacks exploit the dependence between the instantaneous power consumption of the cryptographic device and the data it processes. These attacks are very powerful, do not require expensive equipment, and are almost always are successful.

This thesis discusses the security of cryptographic implementations on FPGA and most present FPGAs are build using Complementary Metal Oxide Semiconductor (CMOS) devices . Hence it is necessary to understand power consumption in CMOS circuits.

1.2.1 Power Consumption in FPGAs

The power consumption in CMOS devices is the sum of *Static or leakage power*, which is caused by the leakage current of each gate, and *dynamic power*, caused by switching activity.

The static power consumption is the power that is dissipated when the device is powered up but in idle state. The major component of static power is the leakage current, that "leaks" either from source to drain or through the gate oxide, even when the transistor is off. It is given by

$$P_{Static} = V_{DD} * I_{leakage} \tag{1.1}$$

These leakage currents are dependent on temperature and process variation.

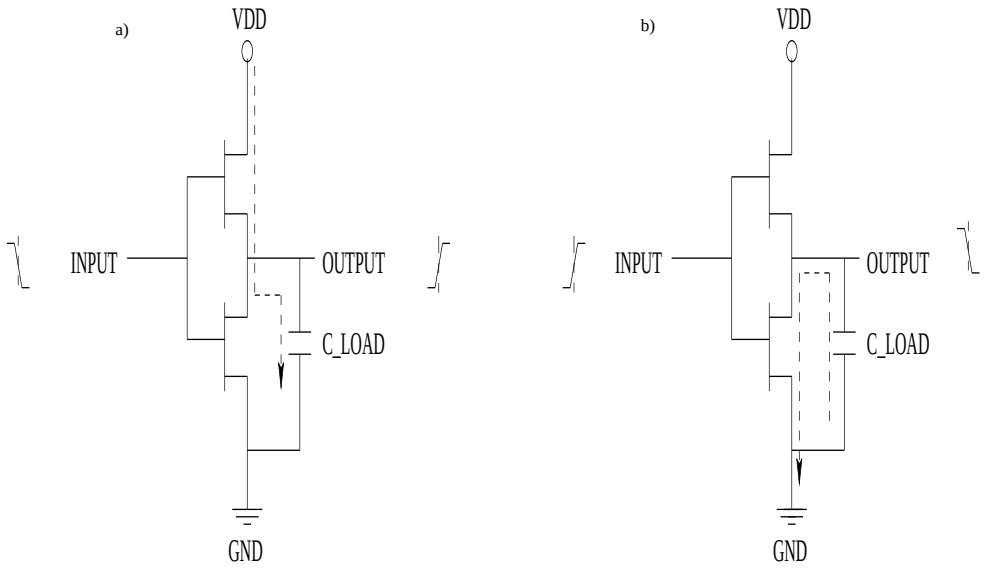


Figure 1.2: Current Flow in a CMOS inverter

Dynamic power consumption takes up major share of total power consumed. It is the switching power which is data dependent [7] and is given by

$$P_{Dynamic} = C_{load} V_{DD}^2 N f \tag{1.2}$$

where C_{load} is the gate load capacitance, N is the number of bits switching per clock cycle and f is the clock frequency.

For each binary logic there can be only four types of transitions on the gate output. Input transitions and corresponding power consumption of the CMOS inverter is shown in Table 1.2.1.

Only when the output of a CMOS gate transitions from low to high, a certain amount of energy is drawn from the power supply and partially stored in the load capacitor as shown in Figure 1.2a) . On a high to low transition the load capacitor is discharged and the energy is dissipated as shown in Figure 1.2b). Transitions from low to low and high to high do not consume any dynamic power. The dynamic power consumption for FPGAs depends upon the switching event of the core and the I/O of the FPGA and is a function of frequency as

Table 1.1: Power consumption in a CMOS inverter

Input transitions	Output
$0 \rightarrow 0$	0
$0 \rightarrow 1$	Discharge
$1 \rightarrow 0$	Charge
$1 \rightarrow 1$	0

shown in Equation 1.2.

This fact is used by power analysis to determine the secret key.

Power analysis attacks are classified into two types, Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [1].

1.2.2 Simple Power Analysis

Simple power analysis technique involves direct interpretation of power traces generated by an algorithm run on hardware. By observing the trace one can gain information about the hamming weight of a certain byte of the key, and thus construct the key. For example looking at Fig 1.3, a power trace measured from an FPGA implementing AES one can guess the number of rounds (which by the way is 11 rounds). It is also possible to identify the operation performed by the processor as, different operations have different power consumption characteristics [8, 9]. Generally the SPA attacks are quite challenging since the adversary needs to have detailed knowledge about the implementation of the cryptographic algorithm that is being executed by the hardware.

1.2.3 Differential Power Analysis

Differential power analysis attacks (DPA) are the most popular and powerful type of power analysis attack due to the fact that they do not require detailed knowledge about the attacked device. DPA exploits the data dependency of the power consumption of cryptographic device. Unlike SPA which requires few traces, DPA attack works with large number of power traces.

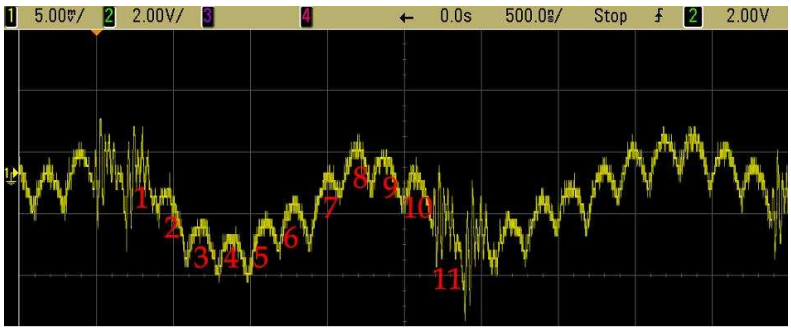


Figure 1.3: SPA trace showing individual rounds in AES

In subsequent years many successful DPA attacks against cryptographic algorithms implemented in software and on ASICs were published. However, four years later the first results on successful DPA attacks against DES and RSA [10] and ECC [11] implementations on FPGAs were reported.

DPA attack methodology can be generalized in four steps.

1. *STEP:1* Choose an intermediate result of the algorithm being executed which, should be function of input data and key.

2. *STEP:2* Measure the power consumption when the intermediate algorithm is being executed.

3. *STEP:3* Calculating the hypothetical values and building a power model.

4. *STEP:4* Comparing the hypothetical power model with power consumption by performing statistical tests

1.3 Thesis Organization

This thesis is organized as follows. In chapter 2 countermeasures against DPA and the basics of Dual rail with pre-charged logic is described. Chapter 3 provides an overview of the FPGA fabric and details about the editing of FPGA resources. Chapter 4 presents our approach to implementing SDDL on FPGAs. Chapter 5 Covers the statistical tests

conducted over the course of this thesis. Chapter 6 presents the security evaluation of the test circuits implemented with our approach and we conclude and point out some future research ideas in chapter 7.

Chapter 2: Countermeasures Against DPA

When it turned out that the cryptographic devices are vulnerable against power analysis, there has been great motivation in development of countermeasures against DPA.

The countermeasures proposed against DPA can be grouped into two categories, *Masking* and *Hiding*.

2.1 Masking

The principle behind Masking is to De correlate the relation between processed data and power consumption. It makes use of random variable called masks, which randomizes the intermediate data being processed while executing the algorithm. As long as the mask value is not known, the circuit is DPA resistant.

The basic structure of the masking countermeasure is shown in Figure 2.1. The mask m is XORed with the data inputs, and the final result O is obtained by XORing the masked output O_m with the mask m again.

Depending upon the number of masking bit used, there are two types of masking namely Boolean and Algorithmic Masking.In Boolean Masking, the random masks are at bit level i.e. a single bit. Logic styles like Random switching Logic (RSL) [12] and Masked Dual-rail with Pre-charge Logic [13] belong to this category.. In contrast Algorithmic Masking uses random masks at word level i.e. multiple bits. [14]. The Boolean Masking scheme can be broken by first estimating the random mask value and then mounting a DPA attack using these estimated value [15]. The Algorithmic Masking can also be broken by estimating the the masked value which is dependent on the probability density function of the power consumption [16].

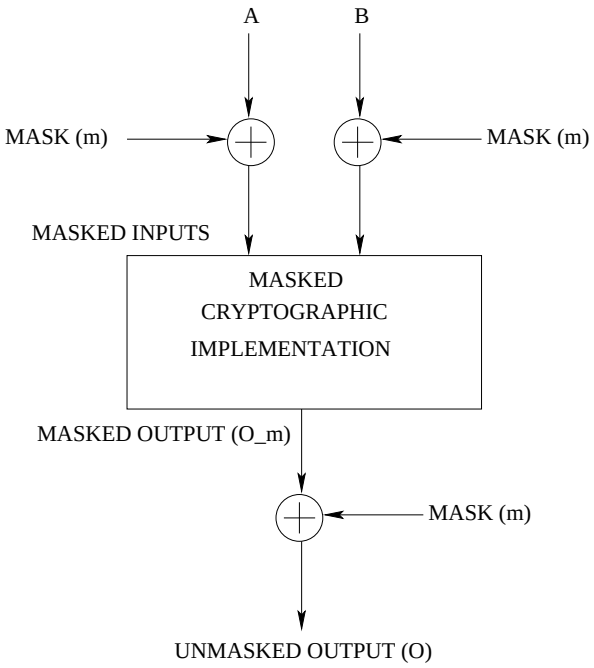


Figure 2.1: Masking Technique

2.2 Hiding

Hiding countermeasures on the other hand, were proven to be more secure than masking [17] as they requires more measurements to disclose the key compared to other counter measures. Hiding makes use of the so called Dynamic and Diffrential Logic (DDL) or Dual rail with Pre-Charge Logic (DPL) to make the resultant power consumption constant for each clock cycle. We make use of the either notations alternatively through out this thesis.

DPL style was introduced by Tiri in 2004 [18]. The goal of DPL is to eliminate the correlation between the data being processed and the power consumption of the circuit by achieving constant power consumption per cycle, hence making a DPA attack infeasible. This is accomplished through duplication of the original circuit into a direct and a complementary logic which follow two basic principles:

1. *Constant switching activity:* This guarantees a single switching event per clock cycle and gate output. During each clock cycle either a gate output in the direct path switches or the corresponding gate in the complementary path.

2. *Constant load capacitance:* The capacitive loads driven by the gates in the direct path is equal to the load driven by the gates in the complementary path.

2.2.1 Dual-Rail with Pre-Charge Logic

The dynamic power consumption in a CMOS circuit depends on the output transitions of the logic gates. This dependency is not symmetric i.e a $0 \rightarrow 1$ or $1 \rightarrow 0$ transition will consume power whereas $0 \rightarrow 0$ or $1 \rightarrow 1$ will not. This makes the power consumption dependent on the hamming distance of the data. The information thus obtained can be exploited to reveal the secret key. The Dual rail with Pre-Charge logic style attempts to remove the relation between power consumption and data.

2.2.2 A dynamic logic

will have two phases, so called *Pre-charge* and *Evaluation* phase which are alternating. During pre-charge phase the output of logic gates is forced to a constant value (0 or 1). Pre-charging of the output signal is done in order to have a transition at the gate output when the inputs do not change, Thus making the power consumption data independent. The original transition of the logic gates will occur in evaluation phase. The pre-charge signal is generally the clock of the system. When the clock goes high, the pre-charge phase occurs and when the clock goes low, the circuit evaluates to original output.

2.2.3 A differential logic

will have two circuits, direct and complimentary whose outputs will be inverse of each other during evaluation phase. Lets assume that all outputs are pre-charged to 0. If an output of the direct logic evaluates to 1 then the complementary logic will evaluate to 0 leading to a single switching event. Similarly vice versa is also true thus achieving constant switching activity.

Constant load capacitance means that the gates in the direct logic drive the same load as the gates in the complementary logic. This requires that routing in both parts to be

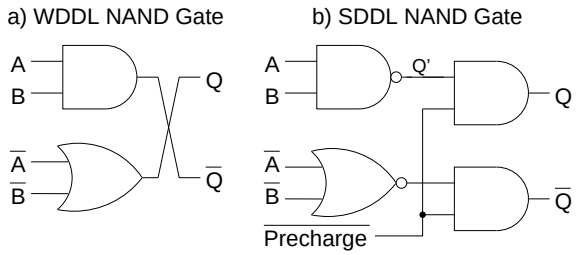


Figure 2.2: WDDL and SDDL version of NAND gate

same, so called symmetrical routing.

There are two different DDL styles called Wave Dynamic Differential Logic (WDDL) and Simple Dynamic Differential Logic (SDDL).

2.3 Wave Dynamic Differential Logic

WDDL is being used successfully for secure cryptographic implementations in ASICs. It is an all positive logic which guarantees one transition per clock cycle. A pre-charge circuit is added only at the register outputs and system inputs. Inverters are implemented by cross connecting the outputs of direct and complementary circuits. A WDDL NAND gate is shown in Fig. 2.2a). This DDL allows for a logic 0 wave to pass through the entire combinational logic hence, the name "wave" is added. WDDL logic style guarantees one switching event per cycle. This can be seen from Figure 2.3. The waveform shows the behavior of NAND gate using DPL logic style.

2.4 Separated Dynamic Differential Logic

SDDL allows the use of negative logic thus making it more flexible than WDDL. However, each time negative logic is used a pre-charge circuit should be added as shown in Fig. 2.2b) because negative logic stops the precharge wave . Thus, unlike WDDL, SDDL needs pre-charge circuit to be added not only at register outputs and at system inputs but also a every gate output. In ASIC circuits this leads to an increase in the area compared to

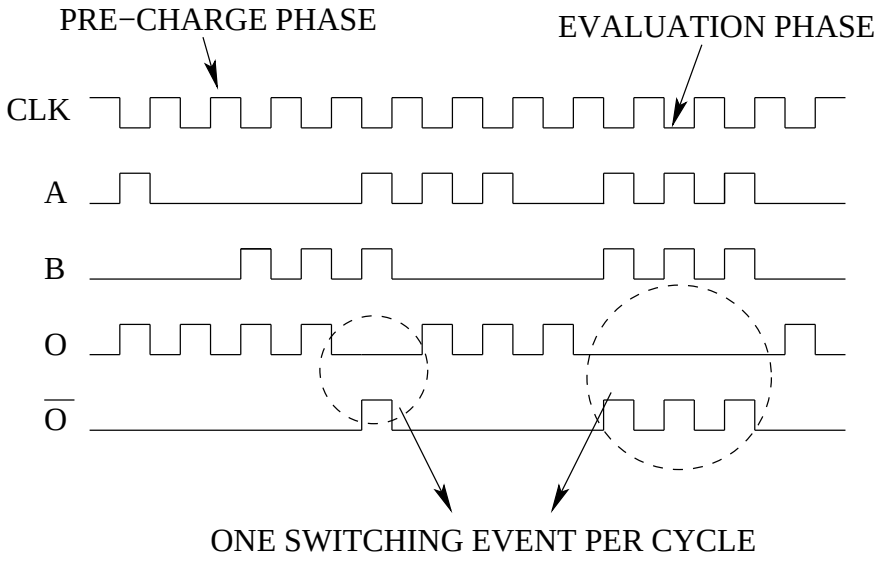


Figure 2.3: WDDL NAND gate operation

WDDL. In implementations on Xilinx FPGAs there is additional area over WDDL because registers in logic only slices are not being used and can be used to insert pre-charge. Unfortunately SDDL is not considered secure as WDDL because negative logic can produce glitches therefore SDDL cannot guarantee one switching event per clock cycle.

2.5 Previous Work on DPL

Recent implementation results of WDDL on FPGAs show two major drawbacks of this technique [19]. WDDL requires the use of glitch free positive logic and duplication, which leads to an increase in area consumption of more than a factor five over a single ended design on FPGAs. Furthermore, balanced load capacitances cannot be guaranteed on an FPGA because the required cross connections between the direct and the complementary logic lead to unbalanced paths. Therefore, Yu proposes in [19] to use Double Wave Dynamic Differential Logic (DWDDL) which allows for balanced load capacitances on FPGAs. However, it leads to an area increase of more than eleven times.

In [20] Guilley et al. evaluate methods which reduce the size of WDDL implementations on FPGAs. They were able to reduce the size of a WDDL implementation of Triple DES

by 23% [21] through new synthesis methods. However, this design is still much larger than a single ended design due to the use of only positive logic as required by WDDL. Current FPGA tools can not produce net lists with only positive logic, hence Guilley uses ASIC tools and a special ASIC library containing hundreds of cells.

The power dissipation of a Xilinx Virtex^{TM}-II FPGA consists to more than 60% of power consumed by the routing resources [22]. This illustrates that it is important to balance the paths of the direct and complementary logic. As this is not a trivial problem on FPGAs, masking schemes have been proposed specifically to overcome the routing problem [13]. However, it has been shown in 2005 that circuits protected only by masking are not secure [23]. Later publications demonstrated that masking does not remove the need for balanced routing in WDDL designs [15, 24].

The impact of current place-and-route methods on the security of a WDDL design was explored in [25] with respect to balancing the timing delays. In [26] the authors propose a new switch box design for FPGAs that enables balanced routing and is secure against power as well as EM attacks.

2.6 SDDL for Light Weight Implementations ?

Unfortunately, applying WDDL to FPGAs is not straight forward. Firstly, FPGA CAD tools cannot be restricted to use only positive logic when synthesizing an implementation. Therefore, ASIC synthesizers are commonly used. Yu and Schaumont also show in [19] that the replacement of inverters by cross connection will result in unsymmetrical routing, which is undesirable.

In positive only architectures optimal usage of intrinsic features, for example fast interconnects, dedicated multiplexers, fast carry logic, shift registers, etc inside the FPGA fabric cannot be exploited.

SDDL can be implemented using only FPGA CAD tools. A negative logic style requires no cross connections between direct and complementary logic, hence symmetric routing is

possible in FPGAs. SDDL allows optimal usage of the intrinsic features inside the FPGA thereby reducing the slice count. WDDL is considered to be a more secure logic style than SDDL because it is glitch free. Hence, SDDL becomes an obvious choice.

Chapter 3: Looking Into FPGA Fabric

There are a wide verity of FPGAs manufactured by different vendors like Xilinx, Altera, Actel etc. This section describes the underlying structure of Xilinx Spartan3E FPGA which we are using for laboratory testing. We also note that the architecture is similar to that Altera FPGAs.

3.1 CLBs and Slices

The Configuration Logic Blocks (CLBs) contains the main logic resources for implementing a wide variety of logic functions as well as for storing information. CLBs are arranged in arrays of rows and columns and their densities varies from one FPGA to another. In Xilinx Spartan3E FPGAs, a CLB is comprised of four slices, each containing two look-up tables (LUT) and two storage elements that can be used as either flip-flops or latches. It also contains two wide function multiplexers, fast carry logic and other miscellaneous elements. The slice internal configuration is shown in Figure 3.1

The slices are further divided into two types, SLICEL (slice logic) and SLICEM (slice memory) as shown in figure 3.2. The LUTs in SLICEM are used not only to implement logic but can also be configured as 16-bit Distributed RAM or 16-bit Shift Register LUT (SRL).

3.2 Switch Matrix and Routing

Each CLB is associated with programmable interconnect switch box or matrix that connects the CLB to the adjacent and nearby CLBs (as shown in fig(to be added)). The routings inside an FPGA fabric is categorized according to the distance of the output wire or line.

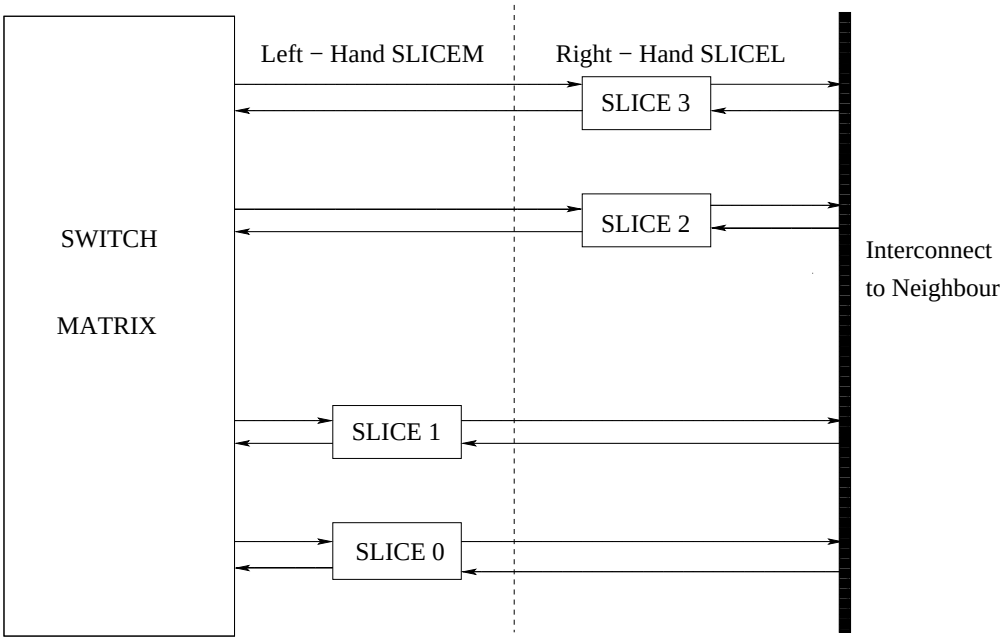


Figure 3.1: Configuration Logic Block

Long lines are bidirectional lines that distribute signals across the device. Hex lines are connected to every third and sixth CLB, double lines connect every first or the second CLB and the direct lines span one CLB.

An exception to these routing resources are the local interconnects, so called fast interconnects that exist between slices and between CLBs. These fast interconnects are used to create Wide Dedicated Multiplexers (WDMs).

3.3 Wide Dedicated Multiplexers

WDMs are one of the special intrinsic features inside the Xilinx FPGA [27]. They are used to effectively combine complex logic operations. These WDMs are implemented using the dedicated two input multiplexers in the slice and the fast interconnects, that exist between slices and between CLBs. Using conventional LUTs as multiplexers will add to the logic delay and also have an additional routing delay and increase the area consumption. This effect can be minimized by using WDMs. Also wide input logic functions can be easily

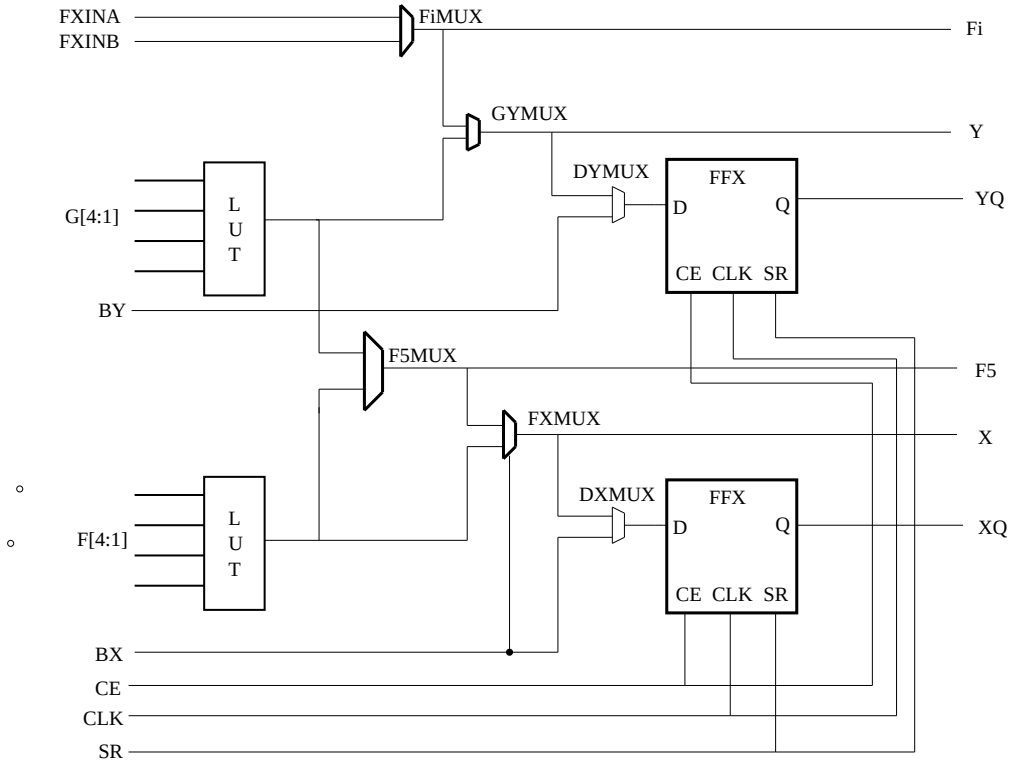


Figure 3.2: Slice Internal Structure

realized using WDMs.

Every pair of LUTs in a slice is followed by a two input multiplexer called F5MUX (see Figure 3.2) which effectively combines the output of the two LUTs to create a four input multiplexer or a five input function as shown in figure 3.4. The F5MUX is present at the bottom part of each slice. The second two input multiplexers at the top part of each slice is called FiMUX (see Figure 3.2), where i can be 6, 7 and 8 depending on the level of logic or multiplexer implemented. Each FiMUX drive the multiplexers of next higher number. F6MUX combined with two slices can implement an 8 input multiplexer or a 6 input logic function as shown in Figure 3.5. F7MUX and four slices i.e. one CLB can implement an 16 input multiplexer or a 7 input function. F8MUX along with 2 CLBs can implement an 32 input multiplexer or an 8 input function as shown in Figure 3.6. In this thesis we study the effect of three levels of multiplexers or logic function, the F5MUX (4:1 MUX), F7MUX

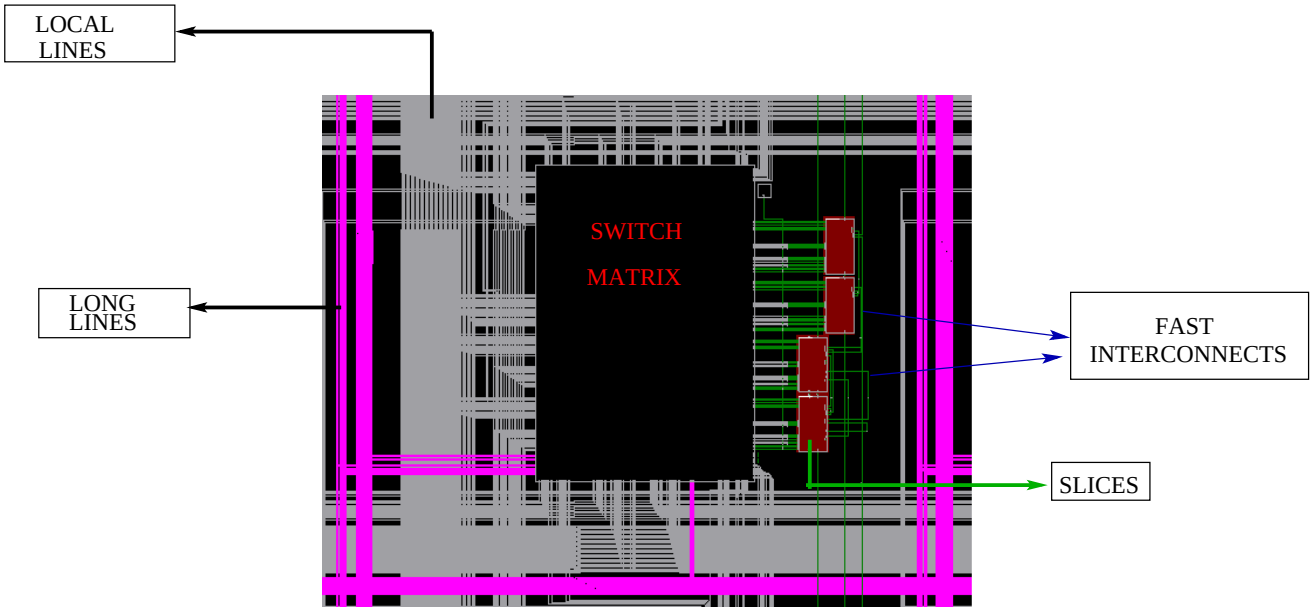


Figure 3.3: Switch Matrix and Routing

(16:1 MUX) and the F8MUX (32:1 MUX).

3.4 Editing FPGA Resources

We will be inserting pre-charge circuit and duplicating the original part after the design is implemented i.e. we will be editing the FPGA resources at LUT level. There are two methods to edit the resources in a FPGA at slice level, FPGA editor and Xilinx Design Language (XDL). FPGA editor offers graphical user interface for the designer to the circuit. Using FPGA editor small design changes can be made at ease but for an extensive change in the design like inserting pre-charge circuit becomes impractical and cumbersome.Hence we use XDL.

XDL describes the resources used in a design in ASCII format. Xilinx ISE contains XDL tool which converts xilinx proprietary Native Circuit Description (NCD) file to XDL file and vice versa. An XDL file will typically contain slice configuration information called Instances and the routing between such instances called Nets.

An example of an Instance is shown in Figure 3.7. From such an instance description

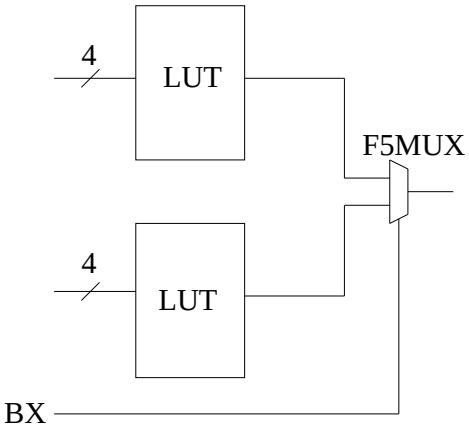


Figure 3.4: A 4 by 1 Input Mux using Two LUTs

Table 3.1: Operators used in LUT equation

Operators	Symbols
Logical AND	*
Logical OR	+
Logical XOR	@
Unary NOT	~

one can know slice specific information for example the logic equation LUT implements etc. Table 3.4 lists the operators used in an LUT equation.

The slice internal component representation format is always the same

$$Component_{name} = \#ParameterValue$$

If the value is other than #OFF, it indicates that the corresponding component is used in the slice as shown in Figure 3.7.

The routing information is described in the *net* part of the XDL file. As shown in Figure 3.8 Programmble Inter connects (PIPs) are always written as

$$pip\ tile\ wirex \rightarrow wirey$$

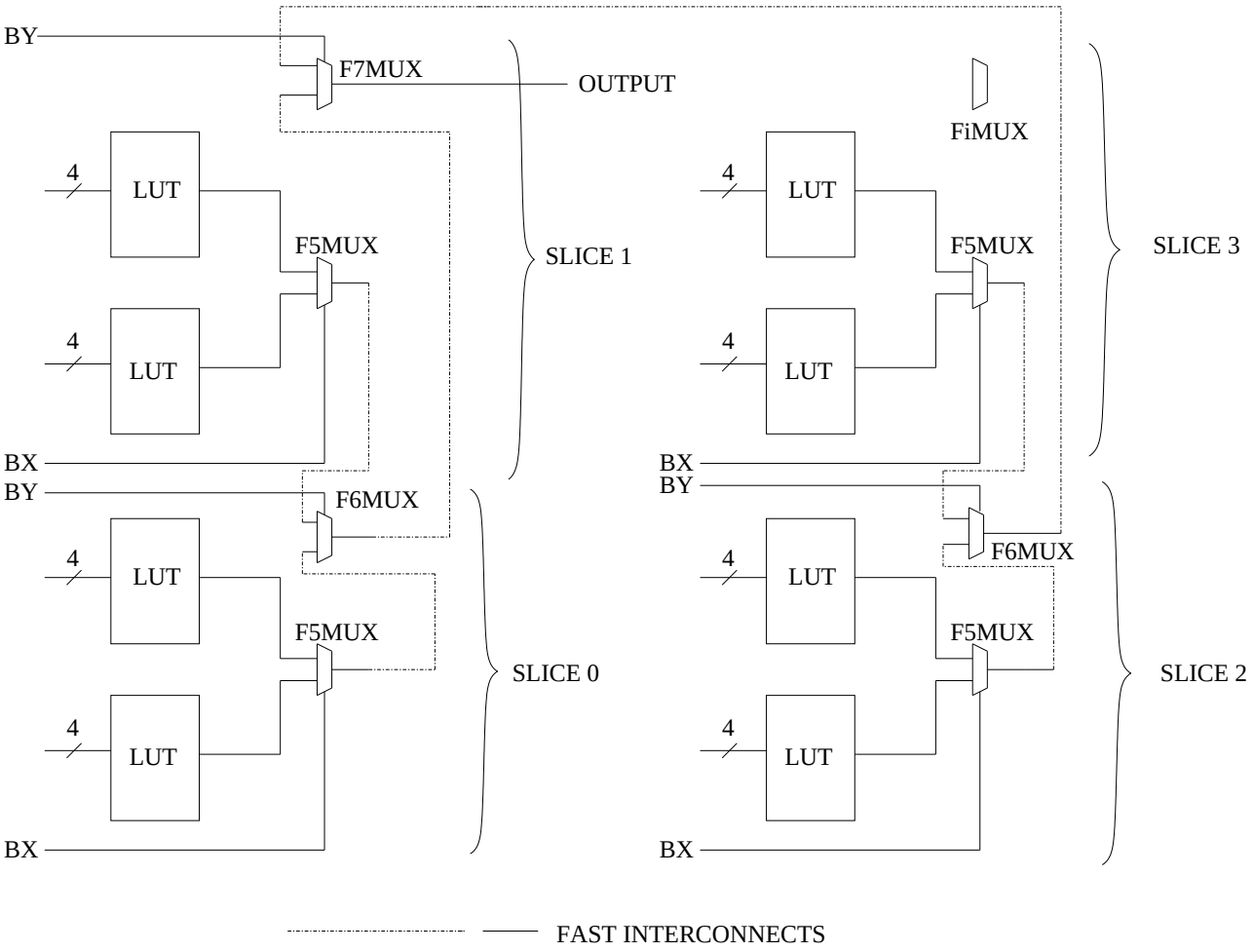


Figure 3.5: A 16 by 1 Mux using Four Slices

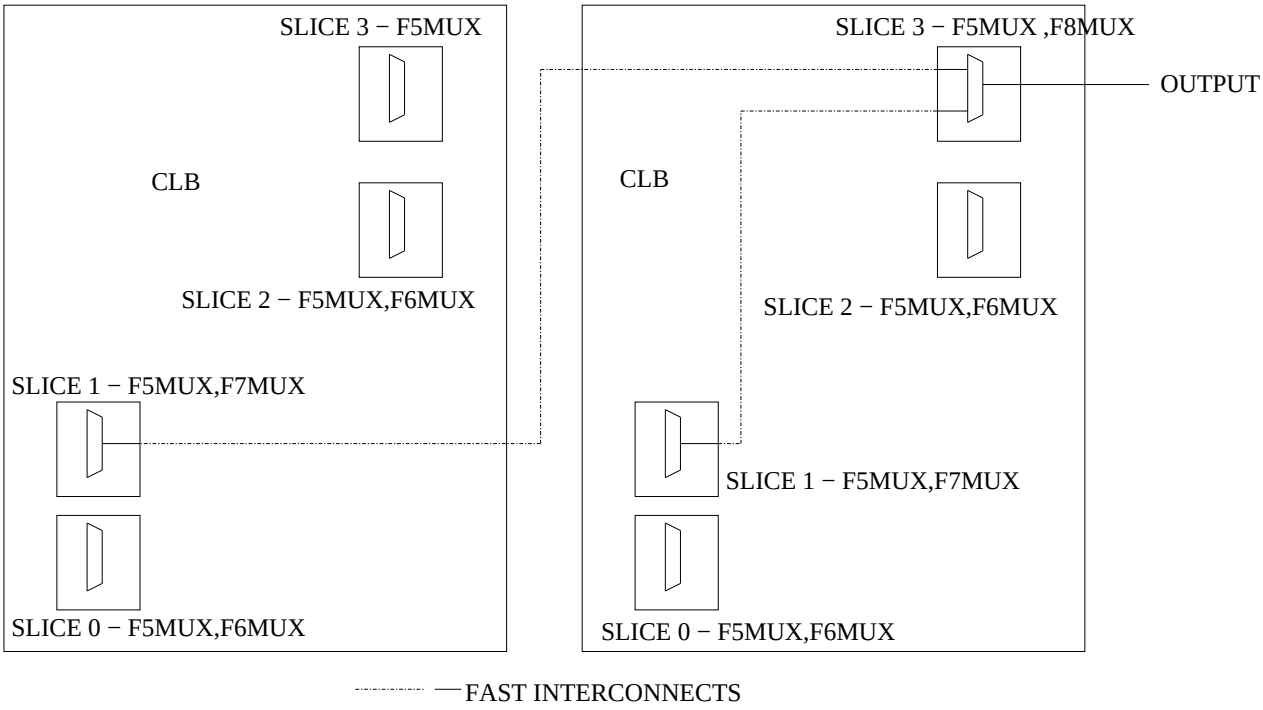


Figure 3.6: A 32 by 1 Mux using Two CLBs

```
                Instance Name                    CLB Location   Slice Location
inst "aes/datastore/N34" "SLICEL",placed CLB_X17Y39 SLICE_X32Y77 ,
  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:aes/datastore/datac_14_mux0000_SW1:#LUT:D=(~A1*(A4*~A2))+(A1*(A4+A2)))
       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::#OFF
       GYMUX::#OFF REVUSED::#OFF SRINV::#OFF SYNC_ATTR::#OFF XBUSED::#OFF
       XUSED::0 YBUSED::#OFF YUSED::#OFF "
  ;
                                                                  LUT Logic Equation
```

Figure 3.7: Slice Internal Components Description in XDL

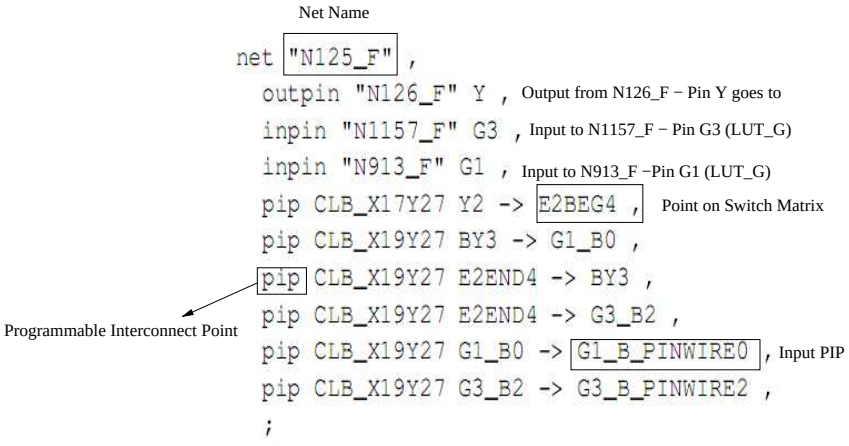


Figure 3.8: Routing Description in XDL

Chapter 4: Implementing SDDL on FPGA

In our proposed SDDL model, FPGA CAD tools are given the maximum flexibility to optimize a given design for the target FPGA. Such an optimized design will allow logic packing in LUTs and also make use of all the intrinsic features present in the FPGA. In this thesis we are exploring the usage of Wide Dedicated Multiplexer (WDM). Using WDMs reduces the area consumed by our design however, their effect on DPA resistance has not been explored yet.

We apply SDDL only to the data path of the circuit, because it handles the secret data. We allow the controller to leak, as it manipulates the public information about the algorithm. This approach is similar to the security partitioning in cryptographic ASICs.

We also make the following assumptions in our SDDL model

- The wire connecting the LUT and the flip-flop/latch *will not leak* any information.
- The wire connecting two slices in the same CLB *will not leak* any information.
- Any wire between CLBs i.e. wire connecting a slice in one CLB to a slice in another CLB *will leak* information.

Our proposed SDDL model is shown in Figure 4.1. Instead of applying De-morgan's law to obtain the duplicate part we simply invert the inputs and outputs of the original logic gate. This technique is not suitable for ASICs because placing a inverter on both inputs and outputs will increase the area consumption. Where as in FPGAs, we need to modify only the logic equation in the LUTs, which will not cause any area overhead.

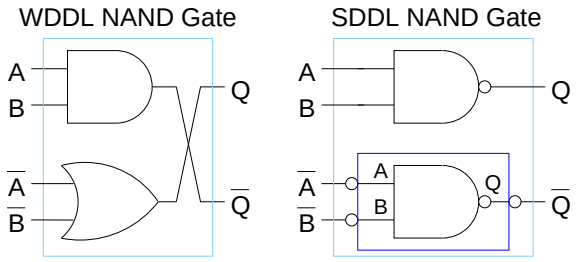


Figure 4.1: Propsed SDDL model

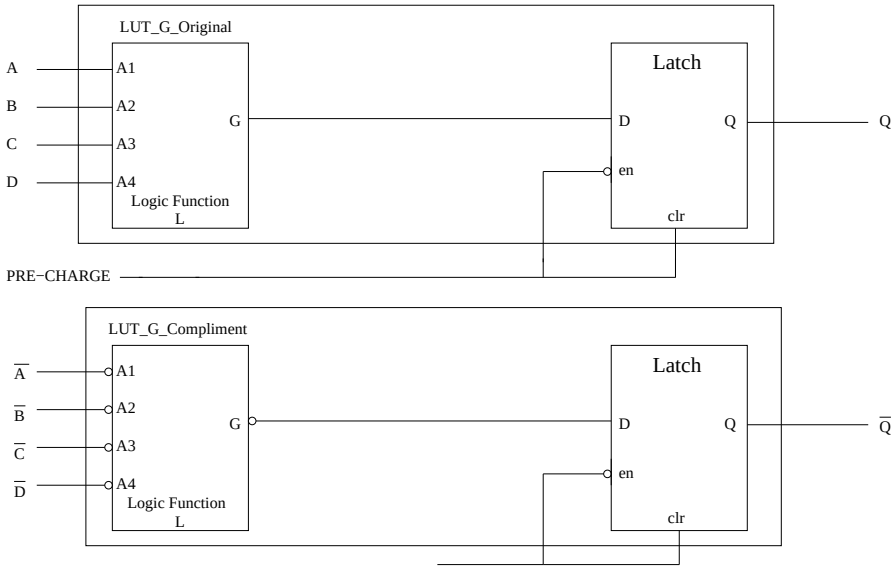


Figure 4.2: Pre-Charge circuit implementation in FPGA

4.1 Pre-Charge Logic

Pre-charge insertion is done using the technique introduced by Yu and Schaumont in [19, 28].A flip-flop/latch is following every LUT. The pre-charge circuit is implemented using an asynchronously cleared latch with an inverting enable input, which forces the output of the LUT to logic '0' during the pre-charge phase as shown in Figure 4.2. If a flip-flop is already used in the design then the pre-charge circuit should be inserted in a slice as near as possible to the flip-flop so that the routing between the two slices is kept at minimum.

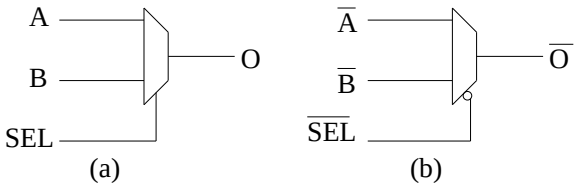


Figure 4.3: Duplicating Multiplexers in FPGA

4.2 Duplication

The first step in creating the complementary path is duplication of the original path. Before this can be done, appropriate CLB locations for the duplicate design must be chosen such that they have the same routing resources as the original design. Then the original design is copied (including routing), the components and nets are renamed and moved to the chosen locations. This is done using the Xilinx Design Language described in the previous section

4.3 Complementing the Logic

The complemented path should produce outputs inverse to that of the direct path. If $f(x)$ is the equation which defines a LUT in the direct path, then it's complimentary equation $g(\bar{x})$ is given by

$$g(\bar{x}) = \overline{f(\bar{\bar{x}})} = \overline{f(x)} \tag{4.1}$$

WDMs use LUTs and slice internal multiplexers. Equation 4.1 holds for LUTs however, for the slice internal multiplexers only the select lines should be inverted as shown in Figure 4.3.

Since we do not apply SDDL to the control block of the implementation, the control signals are to be inverted before they are attached to the complementary block. This method causes area overhead. Therefore, care must be taken, so that the control signal are not inverted in the logic equation as shown in figure 4.4.

Figure 4.4: Complementing Control Logic

4.4 Secure Design Flow

Our design flow for implementing SDDL on FPGAs uses Xilinx ISE Design suite 10.1 and Perl scripts. It consists of three phases as show in Fig.

In the first phase, the single ended design is synthesized and implemented. Area constraints to be applied should perform the following tasks

- Constraints are to be applied to limit the design to one section of the FPGA fabric.
- It should also specify that the locations near registers should be left empty as they will be needed to insert pre-charge in the next phase.
- The controller block is to be constrained in such a way that so that the distance form the control signals to the direct and complementary path should as similar as possible, so that there would not be any delay of operation between the direct and complementary circuits.

In the second phase, the circuit description file from the first phase is converted into ASCII representation format with help of XDL (Xilinx Design language) tool . Perl scripts interpret the XDL file and insert pre-charge. Subsequently only Place and Route is executed.

In the third phase, the I/O connections are removed and the design is converted into XDL format. Perl scripts duplicate and complement the original circuit resulting in an SDDL implementation. However, as the I/O pins are still disconnected, and all the routing has to be preserved, we use re-entrant routing only.

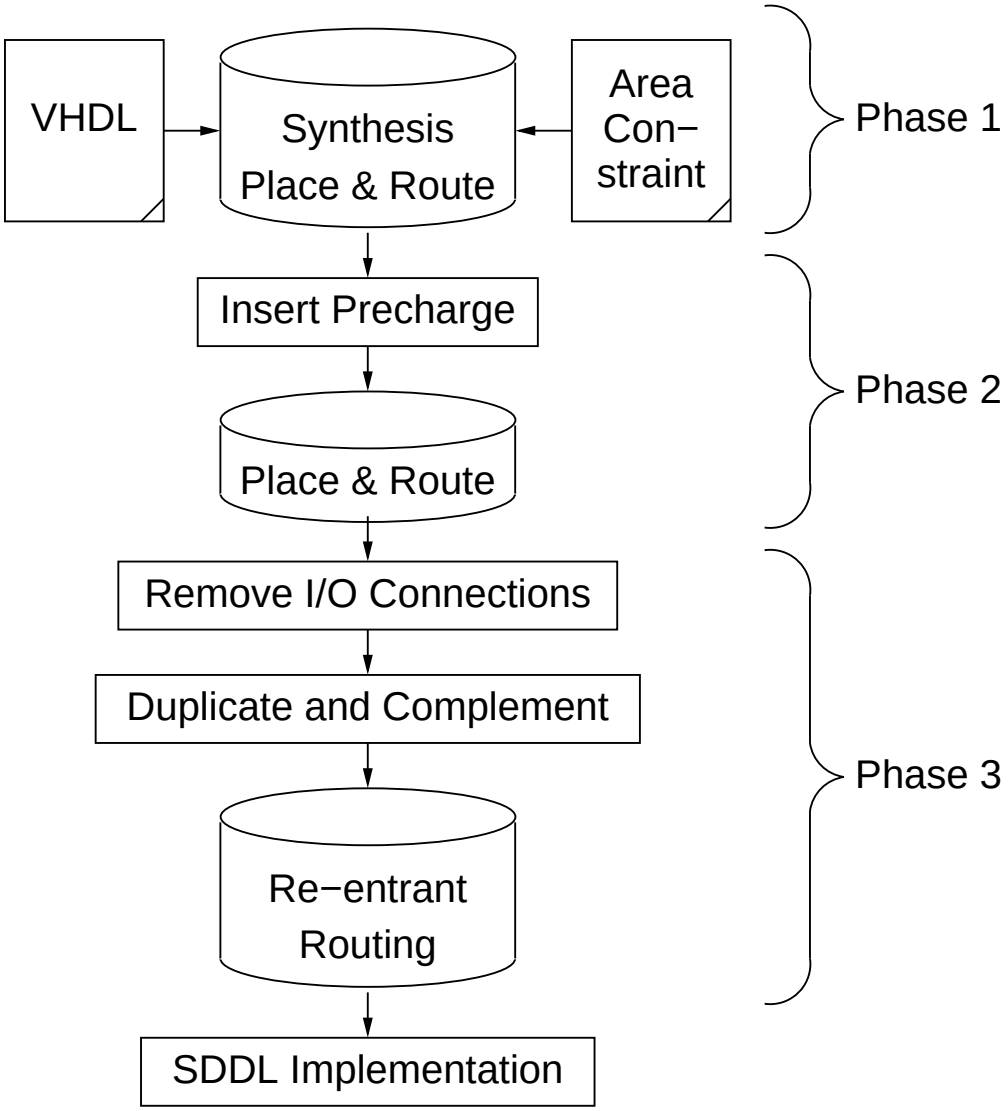


Figure 4.5: SDDL Design Flow

Chapter 5: Statistics

This section describes the power models and statistical tests conducted over the course of the thesis.

5.1 Power Models

In order to simulate the power consumption of the cryptographic device i.e. to relate the power consumption with the data being processed we use the following two generic models.

5.1.1 Hamming Weight Model

It is a simple power model, which assumes that the power consumptions is directly proportional to the Hamming weight (HW) i.e. number of bits that are set in the corresponding data word [1, 29]. If W is binary data of length 'i' given by

$$W = \sum_{i=0}^{i-1} w_i 2^i \tag{5.1}$$

then the Hamming weight of W is the number of bits that are set to '1', shown in Equation 5.2.

$$HW(W) = \sum_{i=0}^{i-1} w_i \tag{5.2}$$

This model is typically used when the attacker knows only the data value at a given time, but not it's previous or the next state value. The hamming weight model is generally

not suited to describe the power consumption in CMOS devices as the reference state from which the bits are switched may not necessarily be zeros.

5.1.2 Hamming Distance Model

Hamming Distance (HD) Model generalizes the hamming weight model. If D and W are two consecutive states of the processed data then the Hamming distance between D and W is given by

$$HD(D, W) = HW(D \oplus W) \tag{5.3}$$

HD is basically the number of bits flipping from state D to state W. This model assumes that $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions have same power consumption, and it ignores the static power consumption. It also assumes that there is a linear relationship between power consumption and the HD. It is one of the most commonly used models and is well suited to describe the power consumption in registers and data buses.

5.2 Correlation Power Analysis

Correlation Power Analysis (CPA) is generally based on the correlation between the power consumption and the power model. The correct key is the one which maximizes the correlation between power consumption and the power model. CPA is less prone to validate false key hypothesis [30]. Figure 5.1 describes the attack methodology using CPA. Power traces are obtained from the cryptographic devices which are then correlated with the power model using a key hypothesis. The highest peak of the correlation plot gives us the correct key hypothesis.

In this thesis we perform CPA using both Pearson's Correlation [30,31] and Rank correlation [32] in conjunction with either Hamming distance model or Hamming weight model (depending upon the situation).

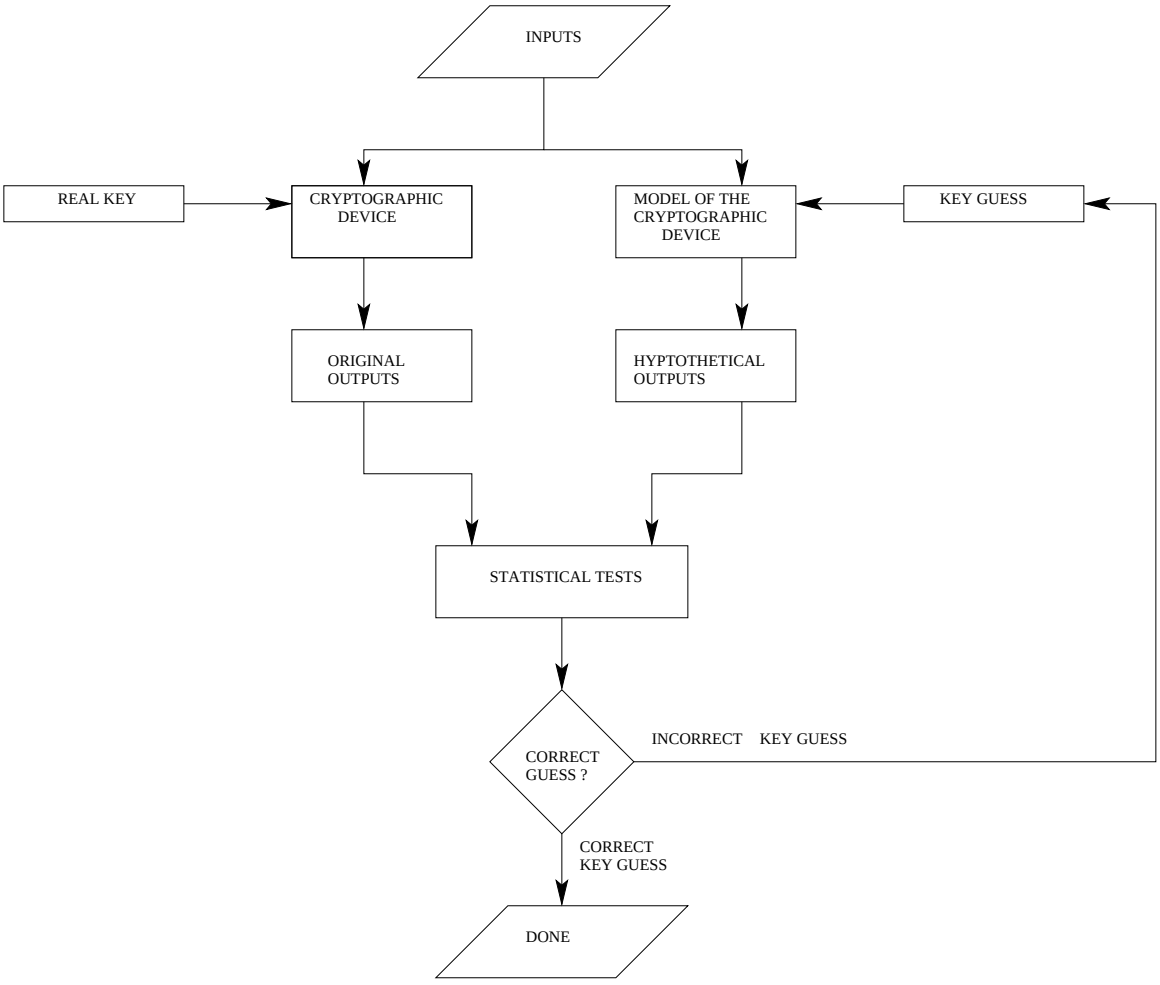


Figure 5.1: CPA Attack Flow

5.2.1 Pearson's Correlation

The correlation between two variables reflects the degree to which the variables are related. The most common measure of correlation is the Pearson Product Moment Correlation (called Pearson's correlation for short). When measured in a population the Pearson Product Moment correlation is designated by the Greek letter rho (ρ). When computed in a sample, it is designated by the letter "r" and is sometimes called "Pearson's r." Pearson's correlation reflects the degree of linear relationship between two variables. It ranges from +1 to -1. A correlation of +1 means that there is a perfect positive linear relationship between variables. A correlation of -1 means that there is a perfect negative linear relationship

between variables. A correlation of 0 means there is no linear relationship between the two variables.

The Pearson's Correlation between power consumption measurement samples P and power consumption hypothesis samples G is given by

$$r(P,G) = \frac{n\sum_i^n P_i G_i - \sum_i^n P_i \sum_i^n G_i}{\sqrt{n\sum_i^n P_i^2 - (\sum_i^n P_i)^2}\sqrt{n\sum_i^n G_i^2 - (\sum_i^n G_i)^2}} \tag{5.4}$$

where n is the number of samples or measurements.

Pearson's Correlation assumes that the power consumption has a linear relationship with the transitions on a data bus, which is true in case of platforms like micro controllers. However on other platforms like FPGA and ASICs the relationship may not be linear [32]. Therefore, we also use Rank correlation to perform CPA on our test circuit.

5.2.2 Spearman Rank Correlation

The Spearman Rank correlation is a measure of monotonic relationship between two variables, in this case between power consumption and power model. It is used as an alternate to Pearson's Correlation when the data does not meet the assumption of linearity. Spearman correlation works by converting each variable to ranks and then correlating the ranked variables. Ranks are given in the ascending order of the data value i.e. the lowest value of the data gets a rank value of 1 and so on. If there is a tie in the rank then a *mean rank* is given to tied data values. For example if there are three variables tied at rank 2 then the three variables will receive a rank of 3(mean of 2,3 and 4 ranks).

Rank Correlation is represented by either ρ or r_s. The Rank correlation between power consumption samples P and power consumption hypothesis samples G is given by

$$r_s(P,G) = 1 - \frac{6\sum_i^n d_i^2}{n(n^2-1)} \tag{5.5}$$

where d_i =P_i - G_i, and P_i, G_i are the ranks of the variables P and G.

If there are tied ranks then the Rank correlation is computed as pearson's correlation between the rank values of the two variables.

Chapter 6: Test Circuit Description and Results

6.1 Experimental Setup

We implemented our designs on a Xilinx Spartan 3e starter board containing a XC3S500eFG320-4 FPGA. We removed the capacitances of the core voltage net and connected it to an external regulated power supply. Power consumption is measured using a Tektronics CT-1 current probe and an Agilent DSO6054A oscilloscope, which has a bandwidth of 500MHz and samples at 4GSa/sec.

6.2 Test Circuits Description

6.2.1 LFSR-SBOX

We wanted to test our secure design flow on a smaller scale and also look at Wide Dedicated Multiplexer (WDM) feature, before moving on to AES. The circuit shown in Fig. 6.1. consists of some of the main building blocks of AES i.e. SBOX and key XORing. The test circuit allows us to replicate a DPA attack on AES on a smaller scale. An 8-bit LFSR is used to supply inputs to the SBOX. The output of the SBOX is XORed with key and stored in register FF1. The dashed line in Fig. 6.1 indicates the part of the circuit that we want to protect. The register FF2 drives the outputs of the chip and is implemented in I/O blocks (IOB).

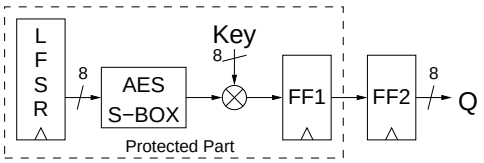


Figure 6.1: Block Diagram of Test Circuit

$$Power_{guess} = \text{HD}\left(lfsr - output_{(i-1)}, (\text{SBOX}^{-1}(Key_{guess} \oplus Output))_i\right) \quad (6.1)$$

$$Power_{guess} = \text{HD}\left(0x00, (\text{SBOX}^{-1}(Key_{guess} \oplus Output))_i\right) \quad (6.2)$$

The AES SBOX maps 8 input bits to 8 output bits using a substitution table. The Xilinx tool implements this function by default as a mixture of boolean logic and multiplexers. The usage of WDMs and the maximum size of the multiplexers can be controlled by the Xilinx ISE tool. In our design we explore three AES SBOX implementations, SBOX implemented using 4:1 multiplexer, 16:1 WDMs and 32:1 WDMs. The output of a 4:1 multiplexer can be pre-charged within the same slice. On the other hand, the 16:1 WDM consists of 4 slices and only the output of the last slice can be pre-charged. In case of 32:1 WDM which consists of 8 slices and only the output of the last slice can be pre-charged. The input LUTs to the WDMs can contain negative logic and hence might produce glitches and disrupt the pre-charge wave. These signals travel through local interconnects and might make this design susceptible to DPA attacks. This leads to a trade off between security vs area.

The power model for the single ended cases is given by Equation 6.1. It calculates the Hamming distance between the previous output of the LFSR and the estimated following output. We estimate the following output of the LFSR for all possible key guesses.We use a different power model to mount a DPA attack on SDDL designs, given by Equation 6.2. The pre-charge phase sets all logic outputs to 0 therefore, the Hamming distance is computed between 0 and the estimated outputs of the LFSR for all possible key guesses. This is equal to their Hamming weight.

6.3 AES

The Advanced Encryption Standard (AES) [33] is one of the most widely used block ciphers. It was designed to be resistant towards linear and differential cryptanalysis. AES is a block cipher of fixed input size of 128 bits and key length of either 129 or 192 or 256 bits. For our AES implementation we chose to use a key length of 128 bits. AES applies the same round function ten times to its inputs during encryption. The round function consists of

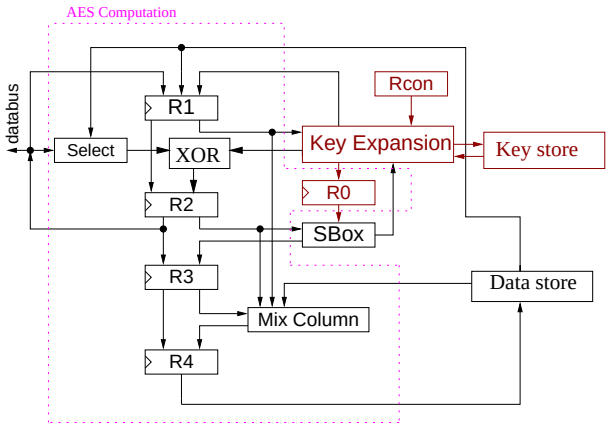


Figure 6.2: Block Diagram of AES Module

four different transformations *sub bytes, shift rows, mix columns and add round key* each changing the input by applying linear, non linear and key dependent transformations. Our AES implementation also assumes that the data input and the secret key are stored in memory. The AES transformations are grouped into four stages

1. Initial AddRoundKey-SubBytes-ShiftRows
2. MixColumns
3. AddRoundKey-Subbytes-Shiftrows
4. FinalAddRoundKey

The data path of out AES implementation is shown in Figure 6.2. It is characterized by pipelined architecture for stages 1 and 3 which reduces the number of clock cycle. Five registers R_0, R_1, R_2, R_3, R_4 are used of which R_0 is used exclusively for *RotWord* operation. R_1 is used for key computation and state computation in MixColumns operation, R_2, R_3, R_4 are used for state computation. The boxes labeled as *Keys* and *Data* are 128 bit registers used for Round keys and State Memory respectively.

In order to provide different plain text and key as input to the AES module we built an wrapper circuit as shown in figure 6.3. A 128-bit Linear Feedback Shift Register (LFSR)

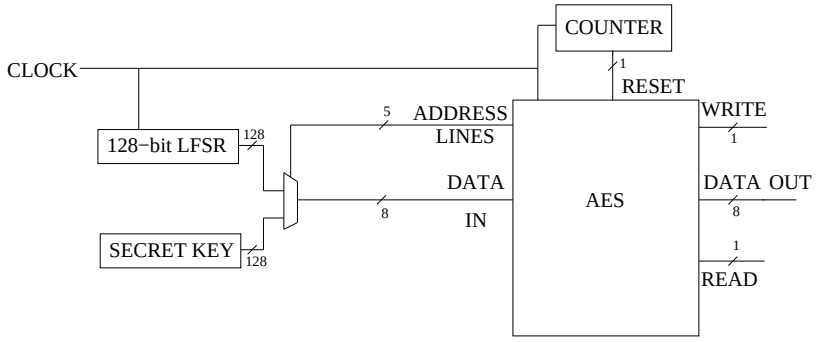


Figure 6.3: AES module with a Wrapper Circuit

provides input data to AES module. A 32:1 multiplexer is used to select the data or key depending upon the address lines from the AES module.

Consider the data flow from R_2, R_3 and R_4. Resetting the AES module changes the data value these registers to 0x00. In the first clock cycle, the output of the register R_3 changes from 0x00 to 0x63 i.e. SBOX value(R_2) and the data value in R_2 changes to the input data XORed with key. In the subsequent clock cycle the data value in the register R_3 changes from 0x63 to SBOX value(R_2). This sequence of change in the data values of the register R_3 i.e $0x00 \rightarrow 0x63 \rightarrow SBOX(Key \oplus Inputdata)$ occurs every time the AES module is reseted. Thus we know two consecutive data values of a register and can apply HD model to simulate the power consumption of the register. Therefore, we use a counter to reset the AES module after every 10 clock cycles. The power model for the single ended cases is given by Equation 6.3.

It calculates the Hamming distance between the SBOX value of key guess xor data and the hex value of 0x63. We estimate the following output of the SBOX for all possible key guesses. We use a different power model to mount a DPA attack on SDDL designs, given by Equation 6.4. The pre-charge phase sets all logic outputs to 0 therefore, the Hamming distance is computed between 0 and the estimated outputs of the SBOX for all possible key guesses. This is equal to their Hamming weight.

$$Power_{guess} = \text{HD}\,(0x63, (\text{SBOX}((Key_{guess} \oplus Output))_i) \tag{6.3}$$

$$Power_{guess} = \text{HD}\,(0x00, (\text{SBOX}((Key_{guess} \oplus Output))_i) \tag{6.4}$$

Table 6.1: Results of LFSR and SBOX implementations

Design	Slices	Delay (ns)	MTD
MUX-4 SE	134	9.08	500
MUX-4 SDDL	283	18.16	$> 15,000$
MUX-16 SE	80	7.51	500
MUX-16 SDDL	168	14.59	$> 10,000$
MUX-32 SE	70	8.088	500
MUX-32 SDDL	148	16.71	> 4000

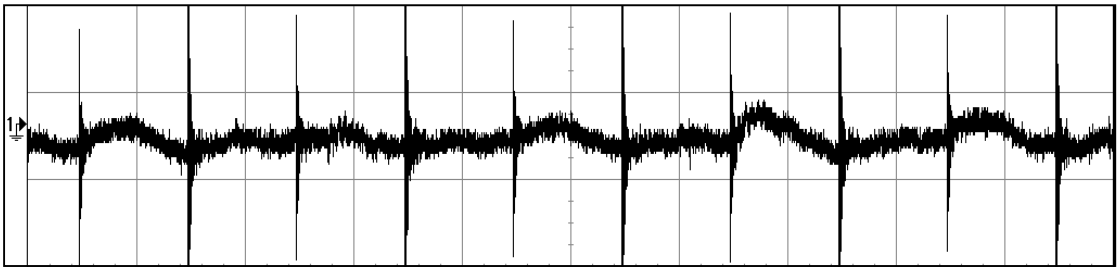
6.4 Results and Analysis

6.4.1 Analysis of LFSR-SBOX test circuit

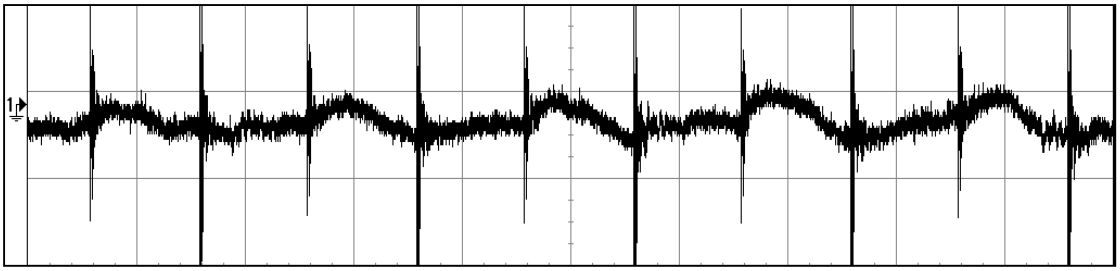
We have implemented three different designs of our test circuit. MUX-4 SE, MUX-16 SE and MUX-32 SE are single ended implementations of our test circuit which use 4:1 multiplexer, 16:1 WDM and 32:1 WDM for the AES SBOX respectively. MUX-4 SDDL , MUX-16 SDDL and MUX-32 SDDL are three symmetrically routed SDDL designs of the said single ended. Table 6.6 shows the results of our implementations and number of measurements to disclosure (MTD) of the key.

The MUX-32 and MUX-16 designs are much smaller than the MUX-4 design and also have a shorter critical path delay. All the SDDL designs are a little bit more than a factor 2 larger than the single ended designs. This is due to the fact that the outputs of the flip-flops need to be pre-charged. This increases the area by one slice per two flip-flops. The delay of the SDDL designs is roughly 2 times larger than the single ended designs because all computations have to be performed during the evaluation phase which is half a clock cycle in length.

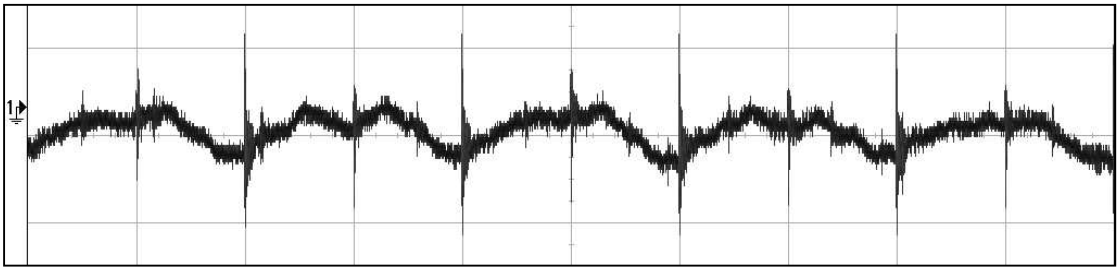
Figure 6.10 shows the power consumption traces for all six designs. The single ended wave forms have their peak near the rising edge of the clock. All the SDDL designs show lower peaks during pre-charge phase and higher peaks during the evaluation phase. It can also be clearly seen that the peaks of the SDDL designs are more uniform compared to the



a) MUX−4 SE



e) MUX−16 SE (duplicated to increase signal strength)



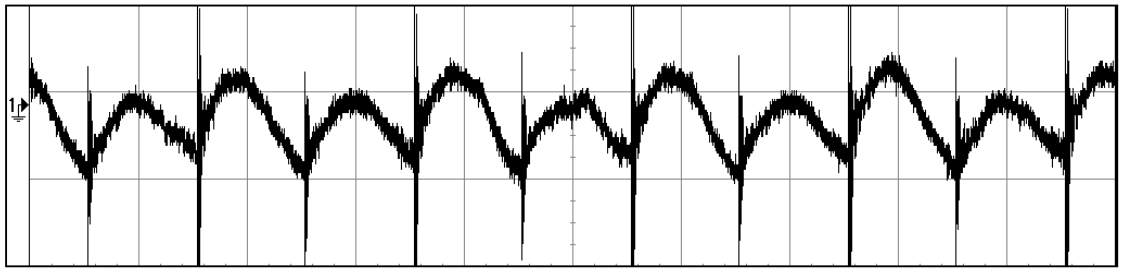
c) MUX−32 SE (duplicated to increase signal strength)

Figure 6.4: Power Traces (5 mV/div, 1 μs/div)

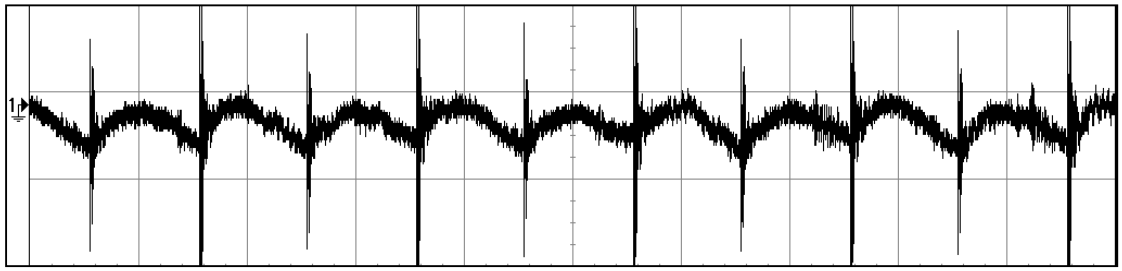
ones of the single ended. Therefore, they are less correlated to the data being processed. This suggests that the SDDL designs are more difficult to attack.

We used a fixed 8-bit key value of 174 for all designs. The correlation plots between power guess and power measured for the MUX-4 SE implementation (Fig. 6.6), the MUX-16 SE implementation (Fig. 6.7) and the MUX-32 SE implementations (Fig. 6.8) taken over 500 measurements shows a sharp peak at the key guess 174. Therefore all single ended designs were broken.

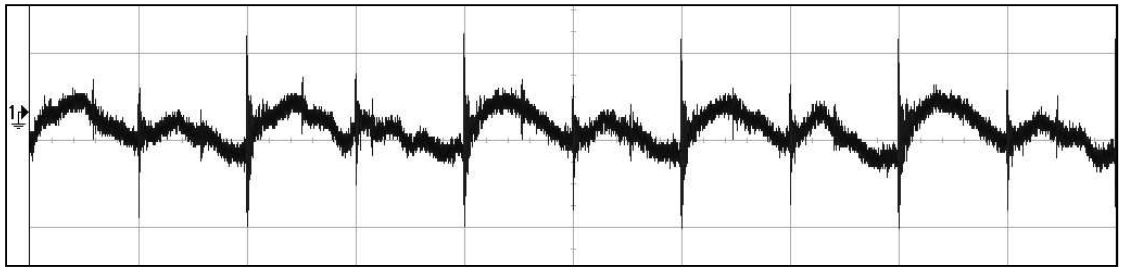
The correlation plots for the SDDL design did not show a definite peak after 1024 measurements. Therefore, we had to take multiple sets of measurements. A set spans 1024



d) MUX−4 SDDL



e) MUX−16 SDDL



f) MUX−32 SDDL

Figure 6.5: Power Traces (5 mV/div, 1 μs/div)

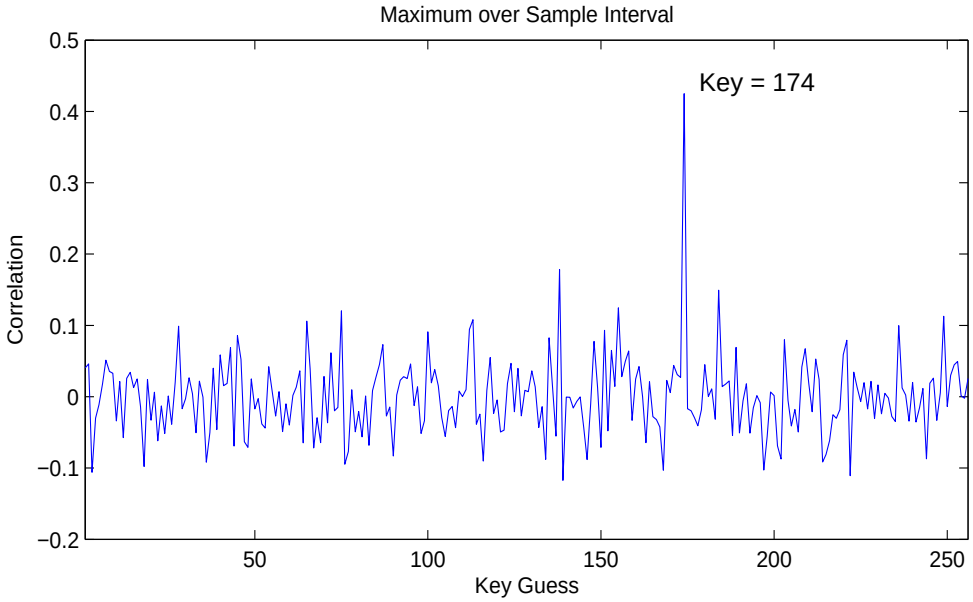


Figure 6.6: DPA Attack on MUX-4 SE Implementation

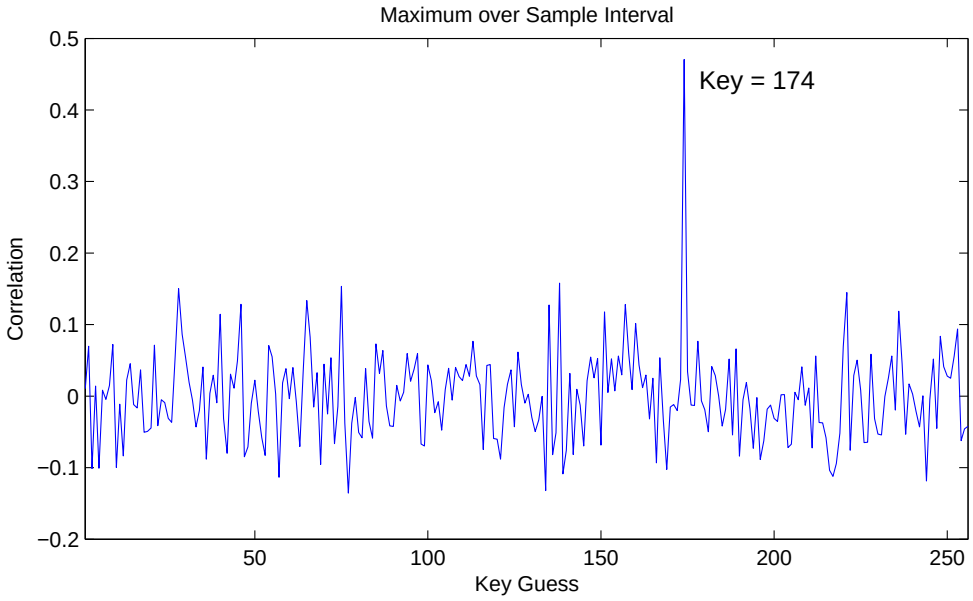


Figure 6.7: DPA Attack on MUX-16 SE Implementation

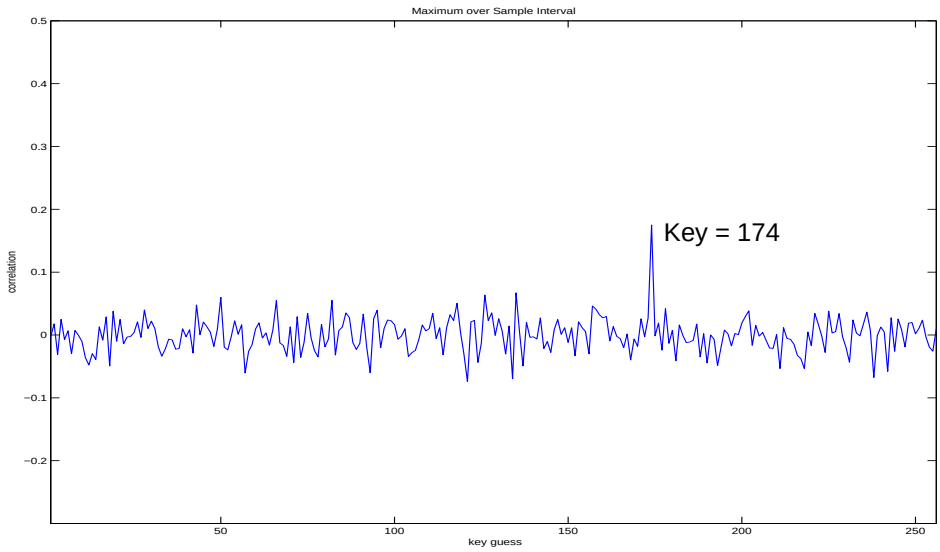


Figure 6.8: DPA Attack on MUX-32 SE Implementation

clock cycles (one measurement per clock cycle) each containing 800 samples. We sub-divide each clock cycle into 10 intervals. These intervals are shown as rows in Tables 6.3, 6.2 and 6.4. For each interval we compute the maximum measured value. We correlate these maximum values of each set with the power model using Equation 6.2. The correlation peaks of 15 sets for MUX-4 SDDL design, 10 sets for MUX-16 SDDL design and 4 sets for MUX-32 SDDL design and each interval are shown in the Table 6.2, 6.3 and 6.4 respectively.

The tables 6.3 and 6.2 show that the correct key of 174 appears sporadically with no obvious pattern. From the Table 6.2 we estimate that the MTD to be larger than 15000 for MUX-4 SDDL design. Similarly it took 10 sets and 4 sets to confirm the correct key i.e. 174 for MUX-16 SDDL design and MUX-32 SDDL designs respectively. Hence, we estimate that the MTDs to be larger than 10,000 and 4,000 in case of MUX-16 SDDL design and MUX-32 design respectively. The reason for the decrease in MTDs is the lack of a pre-charge signal on the fast interconnects between the slices and also between the CLBs. These interconnects are longer compared to the fast interconnects between the slices as used

Table 6.2: Maximum Correlations for MUX-4 SDDL

Interval	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	Set 11	Set 12	Set 13	Set 14	Set 15
0	43	174	238	71	3	174	21	174	238	208	118	128	90	99	88
1	193	247	203	26	201	175	12	44	174	208	22	51	190	212	105
2	228	175	203	170	107	182	219	174	29	11	142	254	110	28	11
3	84	126	184	149	242	247	161	100	29	174	90	50	202	185	212
4	177	115	74	235	167	143	151	15	97	29	174	121	149	174	212
5	59	204	185	119	158	94	232	19	201	221	130	113	246	213	82
6	94	110	199	161	242	154	94	100	99	159	171	58	106	154	180
7	100	199	180	85	170	161	128	99	91	100	154	91	40	242	167
8	43	180	94	161	91	100	161	222	31	87	118	94	173	139	102
9	112	46	172	42	2	56	20	130	253	212	104	111	1	225	23

Table 6.3: Maximum Correlations for MUX-16 SDDL

Interval	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10
0	164	52	20	23	209	160	94	115	245	174
1	119	2	55	88	117	188	194	142	132	14
2	169	202	140	212	40	142	58	120	174	17
3	12	164	51	141	72	95	160	51	140	37
4	223	249	76	177	123	62	168	236	161	215
5	150	212	174	216	204	79	46	140	79	200
6	82	58	208	247	230	28	174	248	231	122
7	192	252	89	72	199	136	230	214	104	83
8	93	14	60	57	190	147	26	213	85	209
9	155	25	96	30	106	197	69	21	160	174

in the MUX-16 design and hence they leak more information

6.4.2 Analysis of AES test circuit

We have implemented two different designs of our AES circuit. AES-SE is a single ended design which use 32:1 WDMs. AES-SDDL is the symmetrically routed design of the said single ended.

Design consideration for AES-SDDL

From our earlier experiments, we concluded that MUX-4 SDDL design was the most secure of the three, though it consumed more area compared to the other SDDL design. Therefore,

Table 6.4: Maximum Correlations for MUX-32 SDDL

Interval	Set 1	Set 2	Set 3	Set 4
0	142	185	60	43
1	240	98	146	59
2	138	34	99	61
3	138	210	68	24
4	174	174	174	174
5	141	230	217	157
6	78	34	22	91
7	194	103	116	213
8	11	169	58	27
9	194	194	141	127

Table 6.5: Slice consumption of Individual Components in AES

AES-Component	slices count before pre-charge	Slice count after pre-charge
Data store + Mix columns	106	178
Key store + Key expansion	98	98
AES Computations	51	71
SBOX	64	129
Controller and address	74	74
Total	393	550

we restricted our AES design to use only 4:1 multiplexers. Due to this restriction the slice consumption of SBOX component increased to twice that of the original value (refer Table 6.5. Pre-charging increases the slice consumed by a register by a factor 2. For example the data store register which is 128-bits in length can be implemented in 64 slices. After pre-charging the area consumed by the data store component becomes 128 slices, which is same in the case of key store register also. Duplicating the pre-charged data store register again increases the area consumption of the final data store - SDDL design by a factor 2. The resulting area overhead due to pre-charging and duplications of a total factor 4 greatly effects the light weigh implementations which are severely area constraint.

Key register is a probable attack point, although the attack itself will be difficult i.e. it will probably require more MTDs compared to attacking the registers R_3 or R_4. The reason is that we cannot compute an HD model as the key store register output is always

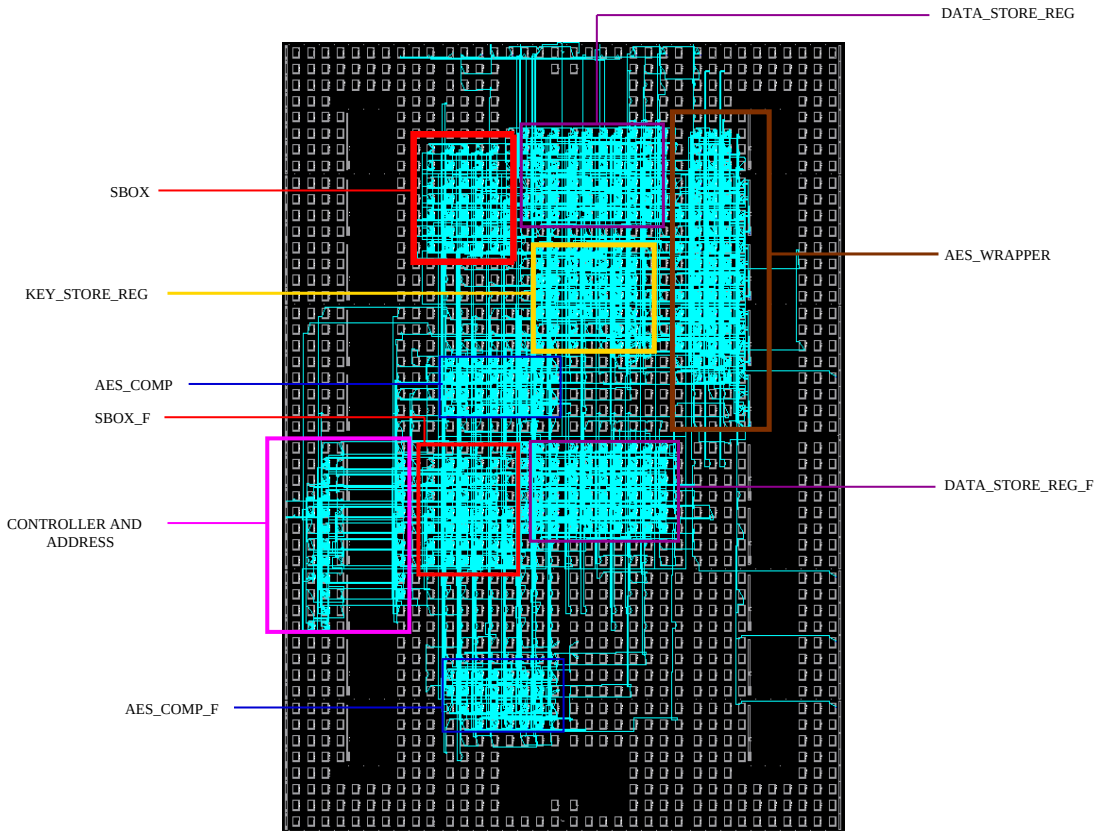
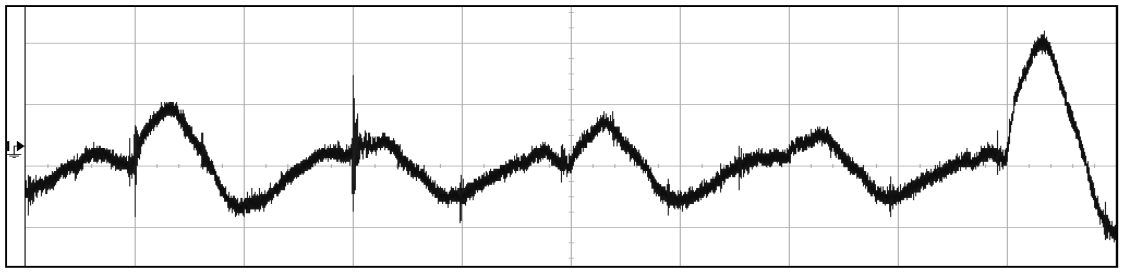


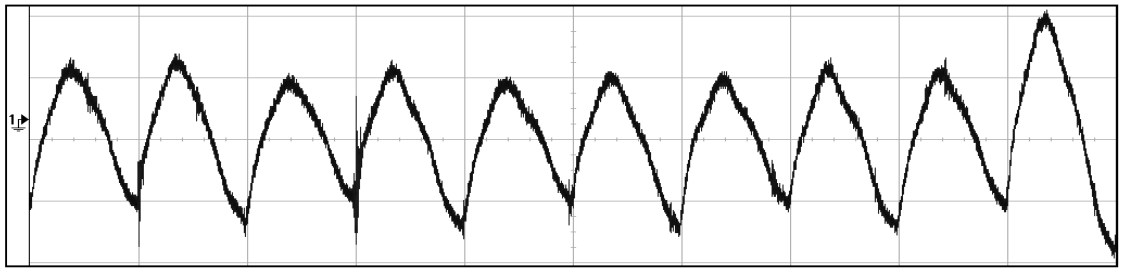
Figure 6.9: Placement of AES modules on FPGA fabric

an unknown value. We have to used the HW model in this case, which ultimately increases the MTDs. Due to this reason and the fact that the SDDL version of key store register will cause a large area over head, we decided not to duplicate the key store register. This is a risk we are taking to limit the area over head although the security of the final SDDL design may decrease. We also decided not to duplicate the controller as it leaks only the public information of the algorithm. the Controller does not manipulate any sensitive information. The parts which are not precharged and duplicated are shown in Figure 6.2 indicated by the red lines and boxes.

Placement of controller inside the FPGA fabric is important. Controller provides signals to both original and duplicate part. The length of the routings from controller to original and duplicate path must be similar so that there is no delay in the operation of the original and the duplicate part. We placed the controller as shown in Figure 6.9 as it mainly controls



a) AES−SE



b) AES−SDDL

Figure 6.10: Power Traces (5 mV/div, 1 μs/div)

Table 6.6: Results of AES-SDDL Implementation

Design	Slices	Delay (ns)	MTD
AES SE	393	14.213	500
AES SDDL	928	28.321	$> 12,000$

the AES computation module.

The key byte which we are attacking is a fixed value of 10 for both designs. The correlation plots between power guess and power measured for the AES SE implementation taken over 500 measurements show a sharp peak at the key guess 10 as shown in Figure 6.11.

The correlation plots for the SDDL design did not show a definite peak after 500 measurements. Therefore, we had to take multiple sets of measurements. We obtained a clear peak for the correct key after 12,000 measurements as shown in Figure 6.12.

The final AES SDDL implementation is larger than the AES SE by a factor of 2.3. The security provided by the AES SDDL is 20 times to that of the AES SE design.

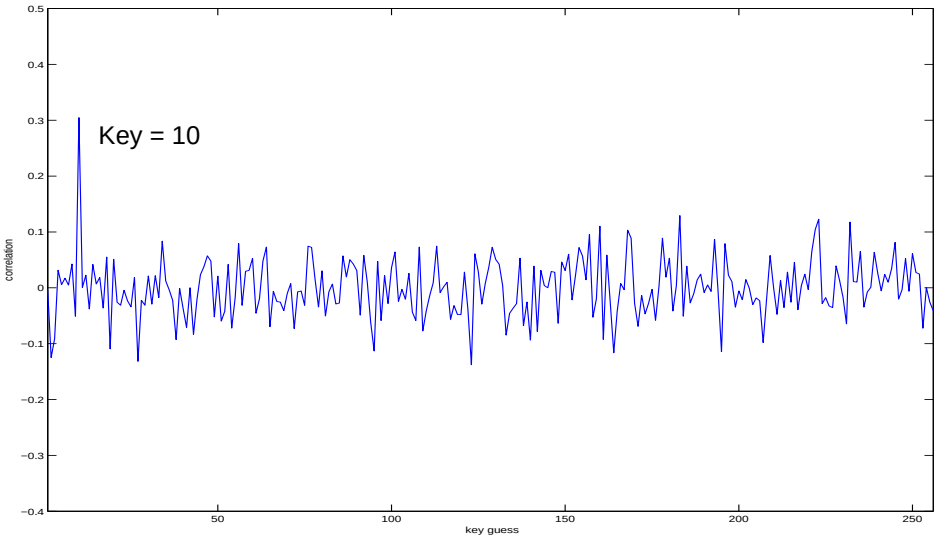


Figure 6.11: DPA Attack on AES SE Implementation

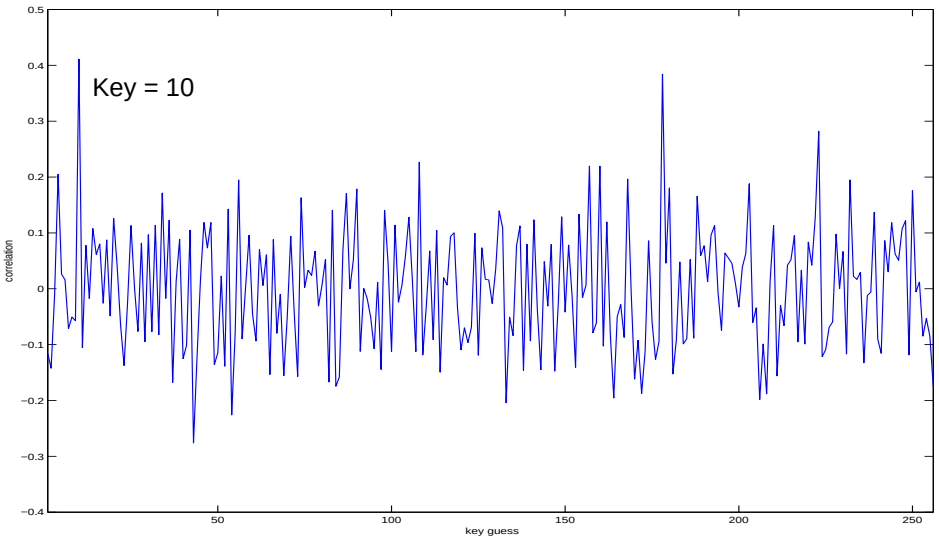


Figure 6.12: DPA Attack on AES SDDL Implementation

Chapter 7: Conclusion and Future Work

Perfect security does not exist. A high level of security is achievable but at the cost of large area consumption. Our results show that we were able to achieve a moderate level of security by using our design flow at an area increase by a factor of just greater than 2.3. Thus showing that it is possible to apply Dynamic and Differential logic styles to low area implementations on FPGAs. Our SDDL can still be broken mainly due to glitches. For future work we plan to reduce the glitches and also examine the DPA resistance of other intrinsic features provided in FPGAs.

Bibliography

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO'99*, ser. LNCS, vol. 1666. Berlin: Springer Verlag, Aug 1999, pp. 388–397.

[2] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks, Revealing the Secrets of Smart Cards*. Springer, 2007.

[3] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology - CRYPTO 96*, ser. Lecture Notes in Computer Science (LNCS), vol. 1109. Berlin: Springer-Verlag, 1996, pp. 104–113.

[4] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems–CHES 2001*, ser. Lecture Notes in Computer Science (LNCS), c. K. Koç, D. Naccache, and C. Paar, Eds., vol. 2162. Springer-Verlag, 2001, pp. 251–261.

[5] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): Measures and counter-measures for smart cards," in *Smart Card Programming and Security, Proceedings of E-smart*, ser. Lecture Notes in Computer Science (LNCS), vol. 2140. Berlin: Springer-Verlag, 200–210 2001.

[6] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, "A 90nm low-power FPGA for battery-powered applications," in *FPGA '06*, ACM/SIGDA. New York, NY, USA: ACM, 2006, pp. 3–11.

[7] J. M. Rabaey, A. Chandrakasan, and B. Nokolic, *Digital Integrated Circuits, A Design Perspective*, 2nd ed. Pearson Education, 2003.

[8] C. Giraud, "An RSA implementation resistant to fault attacks and to simple power analysis," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1116–1120, Sep 2006.

[9] X. Lu and H. Howard M, "A simple power analysis attack against the key schedule of the camellia block cipher," in *Information Processing Letters-IPL 2005*, 2005, pp. 409–412.

[10] F.-X. Standaert, L. van Oldeneel tot Oldenzeel, D. Samyde, and J.-J. Quisquater, "Power analysis of FPGAs: How practical is the attack?" in *FPL 2003*, ser. LNCS, P. Y. K. Cheung, G. A. Constantinides, and J. T. de Sousa, Eds., vol. 2778. Berlin / Heidelberg: Springer, 2003, pp. 701–711.

[11] S. B. Örs, E. Oswald, and B. Preneel, "Power-analysis attacks on an FPGA – first experimental results," in *CHES 2003*, ser. LNCS, C. D. Walter, Çetin K. Koç, and C. Paar, Eds., vol. 2779. Berlin: Springer-Verlag, Sep 2003, pp. 35–50.

[12] J.-S. Coron and L. Goubin, "On Boolean and Arithmetic Masking against Differential Power Analysis," in *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems CHES 2000*, ser. Lecture Notes In Computer Science, vol. 1965. Springer-Verlag, 2000, pp. 231 – 237.

[13] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: DPA-resistance without routing constraints," in *CHES 2005*, ser. LNCS, J. R. Rao and B. Sunar, Eds., vol. 3659. Heidelberg: Springer, 2005, pp. 172–186.

[14] J. Blmer, J. Guajardo, and V. Krummel, "Provably secure masking of aes," in *Cryptology ePrint Archive, Report 2004/101*, ser. Lecture Notes in Computer Science. Springer Berlin, 2004.

[15] P. Schaumont and K. Tiri, "Masking and dual-rail logic don't add up," in *CHES 2007*, ser. LNCS, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Heidelberg: Springer, 2007, pp. 95–106.

[16] Z. Chen and P. Schaumont, "Slicing up a perfect hardware masking scheme," in *Hardware-Oriented Security and Trust HOST, IEEE International Workshop*. IEEE Computer Society, 2008, pp. 21–25.

[17] K. Tiri, D. Hwang, A. Hodjat, B. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "A side-channel leakage free coprocessor IC in 0.18μm CMOS for embedded AES-based cryptographic and biometric processing," in *42nd Design Automation Conference*, 2005, pp. 222–227.

[18] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proc. Design, Automation and Test in Europe (DATE'04)*. IEEE Computer Society, Feb 2004, pp. 246–251.

[19] P. Yu and P. Schaumont, "Secure FPGA circuits using controlled placement and routing," in *CODES+ISSS '07*. New York, NY, USA: ACM, 2007, pp. 45–50.

[20] S. Guilley, L. Sauvage, J. Danger, T. Graba, and Y. Mathieu, "Evaluation of power-constant dual-rail logic as a protection of cryptographic applications in FPGAs," in *SSIRI '08*. IEEE, Jul 2008, pp. 16–23.

[21] S. Guilley, L. Sauvage, J.-L. Danger, and P. Hoogvorst, "Area optimization of cryptographic co-processors implemented in dual-rail with precharge positive logic," in *FPL 2008*, U. Kebschull, M. Platzner, and J. Teich, Eds. IEEE, Sep 2008, pp. 161–166.

[22] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex$^{\text{TM}}$-II FPGA family," in *FPGA '02*, ACM/SIGDA. New York, NY, USA: ACM, 2002, pp. 157–164.

[23] D. Suzuki, M. Saeki, and T. Ichikawa, "DPA leakage models for CMOS logic circuits," in *CHES 2005*, ser. LNCS, J. R. Rao and B. Sunar, Eds., vol. 3659. Heidelberg: Springer, 2005, pp. 366–382.

[24] D. Suzuki and M. Saeki, "Security evaluation of DPA countermeasures using dual-rail pre-charge logic style," in *CHES 2006*, ser. LNCS, L. Goubin and M. Matsui, Eds., vol. 4249. Heidelberg: Springer, 2006, pp. 255–269.

[25] S. Guilley, S. Chaudhuri, L. Sauvage, T. Graba, J.-L. Danger, P. Hoogvorst, Vinh-Nga, and M. Nassar, "Place-and-route impact on the security of DPL designs in FPGAs," in *HOST 2008*. IEEE, 2008, pp. 26–32.

[26] S. Chaudhuri, S. Guilley, P. Hoogvorst, J.-L. Danger, T. Beyrouthy, A. Razafindraibe, L. Fesquet, and M. Renaudin, "Physical design of FPGA interconnect to prevent information leakage," in *ARC 2008*, ser. LNCS, R. Woods, K. Compton, C. Bouganis, and P. C. Diniz, Eds., vol. 4943. Berlin / Heidelberg: Springer, 2008, pp. 87–98.

[27] X. Inc., *XAPP466 - Using Dedicated Multiplexers in Spartan-3 Generation FPGAs*, v1.1 ed., May 2005, application note.

[28] P. Yu, "Implementation of DPA-resistant circuit for FPGA," Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2007, Masters Thesis.

[29] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *USENIX Workshop on Smartcard Technology*, 1999, pp. 151–161.

[30] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems – CHES 2004*, ser. Lecture Notes in Computer Science, vol. 31. Berlin / Heidelberg: Springer, Aug 2004, pp. 135–152.

[31] S. Aumônier, "Generalized correlation power analysis," in *Ecrypt Workshop Tools For Cryptanalysis 2007*, 2007.

[32] L. Batina, B. Gierlichs, and K. Lemke-Rust, "Comparative evaluation of rank correlation based dpa on an aes prototype chip," in *Proceedings of the 11th Information Security Conference (ISC 2008)*, ser. Lecture Notes in Computer Science, C.-L. Lei, V. Rijmen, and T.-C. Wu, Eds., vol. 5222. Springer-Verlag, 2008, pp. 341–354.

[33] *Advanced Encryption Standard (AES)*, National Institute of Standards and Technology (NIST), FIPS Publication 197, Nov 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

Curriculum Vitae

Rajesh Velegalati was born on May 5th, 1985 in Hyderabad, India. He received his Bachelor of Engineering Degree from SIR C.R.Reddy College of Engineering, Andhra Pradesh, India in April 2006. He started working towards his Master of Science degree in Computer Engineering from January 2007 at George Mason University. He was involved in teaching various undergraduate courses at George Mason University both as an Teaching Assistant and Lab Instructor. He is a Research student at Cryptographic Engineering Research Group (CERG). His research interest include Light weight implementations of cryptographic algorithms, Side channel resistance design, Tamper sensing and developing custom FPGA back end tools.