CHAPTER-3
Proposed counter measure for Side Channel Attack

Side channel attacks are becoming a major threat to the security of embedded systems. Countermeasures proposed to overcome Simple Power Analysis (SPA) and Differential Power Analysis (DPA), are data masking, table masking, current flattening, circuitry level solutions, dummy instruction insertions and balancing bit-flips. All these techniques are either susceptible to multi-order side channel attacks, not sufficiently generic to cover all encryption algorithms, or burden the system with high area cost, run-time or energy consumption.

A HW/SW based randomized hardware module injection technique is proposed in this chapter to overcome the pitfalls of previous countermeasures. Proposed technique injects random hardware modules at random places during the execution of an application which protects the system from both SPA and DPA. Further, it is devised for systematic method to measure the security level of a power analysis attack.
CHAPTER THREE: Proposed countermeasure for Side Channel Attack

3.1 Introduction

Cryptographic algorithms are generally designed to cost thousands of years of crypto analysis computation time [196, 201, 202] to break the code, but when they are actualized as a technology, software implementation or hardware details are easier to glean, allowing leakage of information that can help crack them in days, not centuries. Knowledge of execution time, power dissipation pattern or even magnetic radiation and its spectral content can help infer key parameters, narrowing the options to guide the search for hacking successfully. This information is not the main information like key, password etc. So it is called side channel, and the modus operandi is called side channel attack (SCA). Counter measure approach to cover-up details of software implementations by emulating hardware implementations are discussed in this chapter. Such counter measures maintain the generality of secure circuits, but do not succumb to the limitations of specific algorithm, thereby increasing the flexibility of the approach.

A closer look at implementation options reveals embedded systems are common in many low end as well as high end systems, and they play an essential role in our society. The attacks against these systems are achieving higher levels of sophistication and critical mass. Current technologies are having difficulty coping with dual requirement of higher performance and enhanced security, thereby necessitating new solutions that enable a clear definition of ‘secure embedded system’. In such systems, architectures are not only tamper resistant and reliable, but also enable high performance and energy efficient requirements. Such secure embedded systems prevent attacks or react when an attack has been detected while guaranteeing the energy and computational efficiency of the system. FPGAs can address these requirements and provide efficient security primitives [198].

But before delving into the solution space, we highlight current vulnerabilities of FPGA at all levels of the security pyramid (in Figure 3-1). While studying and designing solutions, it is best to identify the weakest point in the whole security pyramid, as there is no additional value of newer solutions, without addressing the weakest link in the security chain. Severe attacks can be mounted by exploiting simple weaknesses in a system, and make massive brute force computational attacks unnecessary.
Figure 3-1 Security pyramid: toward a defence-in-depth [198]

Any information that gives hints or details about a cryptographic device can be useful to mount and attack, and over the years it has become necessary to guard against quite a few types of threat patterns. For example, it does not take high competency to extract power consumption or electromagnetic emissions, and careful study of spectral content can indicate the size of the encryption key, and other inference on class of algorithm used. In other cases, errors can be induced during encryption or decryption process. Thus it is necessary to protect cryptographic devices against fault injection or leakage of information. Counter measures for ASICs are well established over the years, but this is not the case for FPGA devices. Few academic or industrial studies have been conducted so far to investigate the integrity and confidentiality of FPGA devices.

Security in Reconfigurable Hardware Devices
Many design criteria have to be considered while selecting an implementation platform in a digital system. There is need to look beyond obvious criteria like system performance, speed and area cost of a system, to additional security criteria like physical security (against key recovery and algorithm manipulation) flexibility of security algorithm,
power consumption that could end up effecting design selection. The theory and research publications on cryptographic implementation on FPGA devices are extensive, but few paper have anything to say about the convenience levels of using FPGA devices as a target device for security applications from a system point of view. There are some reports of resilience of FPGA against physical or system attacks, which too are quite dangerous compared to brainy algorithm attacks [144-146]. A comprehensive analysis of FPGA security [146-147] shows that FPGA technology can provide security levels quite reasonable when used properly.

The fourth generation design security of Xilinx Virtex-4 family is equipped with bit-stream encryption/decryption technology based on 256-bit Advanced Encryption Standard [AES]. The user generates the encryption key and encrypted bit-stream using Xilinx ISE software. In the second step, during configuration, the Virtex-4 device decrypts the incoming bit-stream using a decryption logic module with dedicated memory for storing the 256-bit encryption key [148].

The prime threat for cryptographic applications is disclosure of the confidential cryptographic key, either a symmetric key or the private key of an asymmetric algorithm. It has been found that FPGA implementations are also vulnerable to side-channel attacks [142,146], in which information is gained directly from the physical implementation. As mentioned earlier, extraction from power consumption, timing behaviour, and electromagnetic radiation are discussed extensively [136-140, 143]. The concept of Power analysis attacks were introduced in 1998 by Kocher et al. [150], where power consumption of the FPGA device is measured closely during the execution of a cryptographic operation. Thereafter, that power consumption can be analyzed in an effort for finding regions in the power consumption trace of a device that are correlated with algorithm's secret key.

The first experimental setup in a lab, and the results of a power analysis attack [142] showed a methodical approach to glean side band information from elliptical curve
cryptosystem implementation. Later, RSA, AES and DES based FPGA implementations were also shown to be vulnerable [144-145]. They showed that SCA can crack a crypto implementation much faster than a brute force crypto-analysis approach. The main advantage of crypto implementation is exorbitant computational cost imposed due to exponential complexity. But SCA based methodology brought unexpected parameters to aid cracking, thereby reducing the complexity quite drastically. It showed the crypto key could be methodically inferred bit by bit, or piece by piece.

Let’s take a more detailed look at the working of an SCA approach with a graphical description of the concept in figure 3-2. Let’s label the cryptographic algorithm as $f$, which responds to input plaintext($PT$), and the key ($K$) to generate an Encryption Response [ER.] Thus $ER=f(PT,K)$, and the goal of the SCA is to infer the key $K$, that is otherwise not directly observable through the ports of the FPGA device.

![Diagram](image)

**Figure 3-2 An example of Side channel Attack on crypto-device and uncovers the key enclosed [198]**

We already know that each bit of $ER$ is related to the individual bit combinations of $PT$ and $K$. In the case that $K$ has 128 bits; a brute force attack needs to consider $2^{128}$ possible values of the key that would cost an exorbitant computational resource for a traditional brute force attack, hence aiding the security of the cryptographic algorithm. But an SCA attack breaks the 128 bit monolithic key into smaller pieces, even as small as 1 bit each.
Hence the search space of the key is reduced from $2^{128}$ to smaller $16 \times 2^k = 2^{16}$, which can be seen as a drastic reduction in complexity. The simplest way to understand this is that not only the full correct key value correlates with the power consumption signal traces, but correlations can be found to infer intermediate values of key also. This the breaking of the key $K$, into intermediate values $[IV]$, allows for partitioning of key, hence complexity of breaking the crypto algorithm. Although every single bit of the output ER is guaranteed to be related to every bit of $K$, this is not true for the intermediate values. It can always find intermediate values that are only related to a small part of $K$, e.g. one byte of $K$. For example, assume that it can find an intermediate value $IV$ which depends on a single key byte $K[7 : 0]$ and the plaintext $PT$. It can be written as $IV = g(PT,K[7 : 0])$. Then $K[7 : 0]$ can be discovered with only $2^8$ guesses. To test which guess is correct, wit is required to observe $IV$. This variable is inside the implementation, but it is indirectly observable through its power dissipation. Indeed, the power dissipated by $IV$ is a part of the power dissipated by the entire device, which can be measured by the attackers. Hence, by measurement of the chip power dissipation, there is a way to test the guesses the attacker makes on a single key byte. Through proper correlation techniques, the chip’s overall power dissipation can be used in place of the power dissipation from $IV$, while the power dissipated by unrelated components can be treated as noise. Power SCA techniques have emerged in recent years, and a Unified Framework for the Analysis of Side-Channel Key Recovery Attacks: extended version is shown in Figure 3-3.
3.2 Detailed Problem Statement

Published literature has already demonstrated need to protect against Side Channel Attacks to protect detection of encryption key and algorithm from power consumption signature. Countermeasures proposed to overcome Simple Power Analysis (SPA) and Differential Power Analysis (DPA), are data masking, table masking, current flattening, circuitry level solutions, dummy instruction insertions and balancing bit-flips. All these techniques are either susceptible to multi-order side channel attacks, not sufficiently generic to cover all encryption algorithms, or burden the system with high area cost, runtime or energy consumption.

Methods like RIJID have demonstrated ability to mask these through inserting dummy instructions as discussed. But these solutions require compiler support, knowledge of specific algorithm, and an assumption that the system can be contained in the chip memory. This approach violates the earlier goal for a unified solution space independent of application and platform specifics.
3.3 Method of Solution

Countermeasures for SCA are one of the important requirements of mission critical applications.

A HW/SW based randomized hardware module injection technique is proposed in this chapter to overcome the pitfalls of previous countermeasures. Proposed technique injects random hardware modules at random places during the execution of an application which protects the system from both SPA and DPA. Further, it is devised for systematic method to measure the security level of a power analysis attack.

The research in this thesis presents SCA modules that are inserted at random interval of time. This is done at run time using partial reconfiguration feature of FPGAs. The hardware modules are pre generated and stored in external memory. These modules are dynamically loaded during run time while cryptographic algorithms are running on the system. In addition to being application independent implementation, it shows more than 50X more difficult in key decodeability.

This chapter concludes with the discussion on the implementation with respect to area overhead, power consumption of the device. It also highlights that the proposed approach is cryptographic algorithm independent.

3.4 Power Analysis Attacks

In these attacks the power consumption of devices, especially cryptographic modules, is studied by the attackers [196,199]. Close proximity to a sensor node to measure the power consumption of the sensor node is required to mount a power analysis attack. There are two types of power analysis [200]: simple power analysis (SPA) and differential power analysis (DPA).
3.4.1 SPA

The attacker directly observes the device’s power consumption. The working principle is that power consumed by the device varies depending on the data processed on and the instructions performed during different parts of an execution. A power trace is a set of power consumption measurements during a cryptographic operation. By closely studying power traces, it is possible to determine the characteristic details of a cryptographic device and the algorithm being used.

3.4.2 DPA

It exploits characteristic behavior using statistical analysis to extract hidden information from a large sample of power traces obtained during controlled cryptographic computations [149]. In a dynamic power analysis based attack, the attacker does not need to know so many details about how the algorithm is implemented like in a SPA. Secret key is obtained bit by bit. It is based on the different amount of power consumption that the algorithm needs to manipulate a certain bit (if it is zero or one) during a specific time.

3.5 SCA Countermeasures

There are generally two kinds of macro strategies to counter sorts of SCAs, by physical tactics or by structural improvements on the functional level [153].

3.5.1 Physical protection

Encapsulate the sensor node with some special casing could avoid or at least reduce the strength of certain kinds of SCAs. A tamper-proof casing, on the other hand, could also prevent wear and tear due to brutal environment factors. In allusion to different scenarios, a variety of casings are deployed.

3.5.2 Functional Protection

Another macro category of SCA-countermeasures is based on the functional protection. Functional protection is the generic term which refers to all the SCA-countermeasures
except for the physical methods (i.e. Casing). In this kind of protection, various techniques are used to counter different sorts of SCAs, commonly or respectively.

3.6 The Importance of Countermeasuring Differential Power Analysis

Among all of these SCAs, the attack towards power characteristics is particularly devastating for cryptographic implementations. SPA (Simple Power Analysis) was firstly introduced by Paul Kocher et al in 1998 [150]. SPA exploits the facts that different operations and different data values processed by a component must have different power consumption characteristics. In the SPA attacks, the attacker only focuses on single power trace to look for such power characteristics and was proved to be successful in attacks to SMART cards [151]. DPA (Differential Power Analysis), on the other hand, depends on a large number of power traces to analyze the intermediate value of the cryptographic algorithm and then guess and verify the secret key of the cryptographic algorithm by means of statistical methods [150]. DPA attack looks at (multiple) specific intermediate values of the implementation, thus it’s especially hard to be defeated. The current countermeasures are generally based on normalizing the power difference or randomizing the power characteristics. Reconfigurable device, to some extent, offers a preferable implementation platform to meet the requirements of the balance between high-security insurance and low-power control.

Side Channel Attack (SCA) can be applied to unprotected implementations of any algorithm. It works because an attacker can confirm an intermediate value of the secret key which correlates with a small part of the secret key that occurs during the execution of decryption – encryption algorithm. Since the processed data produces a strong signature in the power consumption of the implementation, there is a strong correlation between the power consumption and the intermediate values. So an attacker can quickly check the quality of correctness of their partial prediction by calculating this correlation. This observation also hints at a way to counteract SCA is to randomize the intermediate values that occur during the execution of the algorithm by adding another secret value, which is called mask. When this mask value is changed for every execution of the algorithm, the power dissipation signature becomes difficult to correlate for intermediate
values, and the attacker is thus unable to predict intermediate values. In this work, experimental setup for power analysis on FPGA is developed as shown in Figure 3-4. Different case studies have been described here.

Figure 3-4 DPA attack flow for our AES circuit

3.7 Novel Approach to Counter Measure using Partial Reconfiguration

- Published literature already extensively discussed in chapter one and it is demonstrating the need to protect against Side Channel Attacks to protect detection of encryption key and algorithm from power consumption signature. Methods like RIJID have demonstrated ability to mask these through inserting dummy instructions as discussed. But these solutions require compiler support, knowledge of specific algorithm and an assumption that the system can be contained in the chip memory. This approach violates the earlier goal for a unified solution space independent of application encryption algorithm and architectural specifics.

- The research in this thesis presents SCA modules that are inserted at random interval of time. This is done at run time using partial reconfiguration feature of FPGAs. The hardware modules are pre generated and stored in external memory.
These modules are dynamically loaded during run time while cryptographic algorithms are running on the system. In addition to being application independent implementation, it shows more than 50X more difficult in key decode ability.

- The proposed technique inserts random hardware modules at random places in FPGAs at the time of application execution, which protects the system from power analysis attacks, both SPA and DPA. Further, systematic method is devised to measure the security level of a power trace.

### 3.7.1 General Overview of Proposed Approach:

A PR design platform using Microblaze embedded processor is designed as shown in Figure 3-5. The FPGA is configured with Static region and partial region. The static region has local logic as well as softcore processor Microblaze. The Cryptographic algorithm written in C/C++ is executed in Microblaze. The code is stored in External memory. This is base system without the countermeasure. This we refer as software version. The power is measured for the given algorithm as per the experimental setup shown in Figure 3-10.

SCA countermeasure is made active by injecting Hardware module in the Partial Reconfiguration region identified in the system as shown in Figure 3-6 and 3-9.
The hardware modules to be configured are of different size, in terms of area and power consumption signature. The hardware modules are composed of free running counters. The counters are of 4 bit, 8 bit, 16 bit and 32 bit. Depending on the size of counters, the area and power of hardware modules varies. So their configuration bit file size and configuration time varies with reference to their size. The configuration bit files of these are pre generated for partial reconfiguration region defined in FPGA. This configuration bit files are stored in Compact FLASH or BRAM of the FPGAs as shown in Figure 3-9. The Hardware modules are changed during runtime as shown in Figure 3-7, As per the software design flow in Figure 3-8 and the algorithm is implemented. The cryptographic software [200] is to be modified as per software design flow. Critical points of software are to be identified, which leaks significant information for power analysis. Here software code is to be added as shown in the algorithm flow below, which randomly chooses the hardware module and insert it in the FPGA. The pre designed hardware modules of various sizes are kept in memory as bit files. This hardware modules have different sizes in terms of configuration bit file size as well as different /varying power signatures. It actually now decides the power signature of the FPGA device. These modules are randomly selected and inserted at run time. The seed of the selections is left to the software written in C. It actually uses the random function of C library and chooses the hardware module number. The Crypto algorithms are iterative process, so in each round, it will have come across another hardware module in the runtime. This is how the power consumption of device is varied and power analysis is masked. The proposed method of
hardware module insertion is independent of algorithm as it is to be updated in the
software part of the crypto algorithm. This helps in improving the mechanism against
RIJID where we require the knowledge of Crypto algorithm to find location for
instruction injection and special compiler support.

Algorithm To Implement Hardware Module

Execution of Cryptographic Algorithm.

// Insert the following at critical points of cryptographic algorithm.

SET FLAG: Insert hardware module.

Choose hardware module randomly form given list.

Read configuration bit file of chosen hardware module.

Load the configuration bit file into the FPGA partial
reconfiguration module position.

RESET FLAG:

Figure 3-6 Block diagram of system with multiple PRRs (Courtesy: Xilinx)

PRR – Partially Reconfigurable Regions
e.g. Initially a basic system is loaded, where only Microprocessor and basic I/O are available. After some time two hardware modules A and B are loaded. After certain amount of time the Hardware module B is stopped and another module C is loaded in place of module B as shown in Figure 3-7.

Figure 3-7 Run Time Reconfiguration of Hardware Module injection
(courtesy: Xilinx)

This is how the power consumption of system is varied. The Power measurements are carried out using simple cryptographic algorithms executed in Microblaze with no hardware modules and with random insertion of hardware modules of same size or different size, at fixed and random intervals. The software design flow is as shown in Figure 3-8.
3.7.2 Experimental Setup

Xilinx soft core processor Microblaze (MB) is used to build a PR based SoC Platform. Xilinx Virtex5 LX110T FPGA based board is used for prototype implementation. The MB processor is clocked at 100 MHz. The software application is written using SDK and stored on-board DDR2 SRAM memory. The DDR memory also stores an AES
encryption key which cannot be read out of the board. The experimental setup consists of FPGA board hooked, Tektronix Oscilloscope (Tektronix 2014) and a PC, as shown in Figure 3-10. For the data transfer between FPGA board and PC UART is used via RS-232 cable. The current measurement is carried out by measurement through 1 ohm resistor mechanism. The current flowing into the FPGA core is measured by the current probe (Tektronix CT-2) hooked to an oscilloscope. Power consumption signature of the FPGA is rated through current. JTAG cable is used for on chip data transfer between FPGA board and PC through Chip Scope for online debugging. The data transfer across the PC and oscilloscope is through USB cable communication between the oscilloscope and the PC. The PC is thus able to send command to the oscilloscope; for example - to require the oscilloscope to sample current traces with the current probe. After one sampling is finished, the PC is also able to obtain the current sample trace also through the USB cable.
Power measurement for SCA: For every measurement, the following steps of operations are performed:

- The PC sends a random PlainText message (16 bytes) (PT) to the FPGA board through the RS-232 cable. The PT is also recorded in the PC.
- The Microblaze receives PT from the RS-232 cable, calculates the cryptographic program with PT; and PT as per the software implementation. Cipher is the cryptographic application under attack.
- After sending out PT, the PC sends command to the oscilloscope to sample the current trace when the Microblaze is running cryptographic algorithm and also
inserting Hardware module run time in the FPGA when it is performing encryption.

- When sampling is done, the oscilloscope sends the current trace TR[999:0] (1000 sampling points) back to PC for further analysis.

### 3.7.3 Correlation Power Attack

As the name suggests, a SCA called Correlation Power Attack (CPA) [153] is mounted with the above measurement results. In the experiments, CPA focuses on the SubBytes step (16 bytes) outputs in the first round of the AES algorithm. CPA is a searching process and every output is only related to one byte of the key. To attack one key byte, it goes through every possible value of that key byte ($2^8 = 256$ possible values). Different outputs of the SubBytes operation are used for different key guesses, which leads to different power dissipation are expectations, called power hypothesis. A decision function calculates the correlation coefficient of the power hypothesis and the actual current measurement traces, to identify the correct key value that is used by the encryption, where the correct key guess results in larger correlation coefficient than any other incorrect key guess. Every sample trace has 1000 sample points that cover a certain range of time (500 us in experiments) which includes the calculation of the first round of the AES. Although the attackers do not know which points correspond to the instruction execution related to the SubBytes’s outputs, brute forcing through every point is still possible by increasing the analysis complexity, but the analysis time is still acceptable.

The key point (Table 3-1) shows that the protected AES (with hardware module insertion) is much harder to attack. Experiments are first compared with the unprotected AES (AES) and then to the protected AES (referring to Random Hardware module insertion) implementations. The attack results on one of AES’s key bytes with 5120 measurements as an example are shown in Figure 3-11, where the time that each measurement covers is represented on the x-axis. The first round execution of AES occurs within this time range. The correlation coefficient between the power hypothesis and the actual current measurement is represented on the y-axis. The CPA analysis is
mounted at each sampling point (1000 in total), which results in 256 correlation coefficients for 256 possible key values. Figure 3-11 is thus obtained after finishing SCA analysis at 1000 different sampling points.

![Figure 3-11 Attack result on unprotected AES](image)

**Figure 3-11 Attack result on unprotected AES**

Attack result on unprotected AES: Correlation between the sampled current and the power expectations with 5120 measurements is shown. Correct key’s trace is plotted in black here, while all other key’s traces are plotted in gray. The emerged black trace represents successful attack.
Figure 3-12 Attack result on protected AES system

Attack result on protected AES system: Correlation between the sampled current and the power estimations with 5120 measurements is shown again. Correct key’s trace is plotted in black here, while all other key’s traces are plotted in gray. The buried black trace represents unsuccessful attack.

Correlation coefficient values of AES for different keys:
Figure 3-13 Correlation coefficient values of AES for different keys. There are more than one significant peaks.

Figure 3.13 is the plot of correlation coefficient values of AES for different keys. The correlation peaks indicates in this time frame the presence of key. These correlation peaks are further analyzed for identification of key numbers. Figure 3.14 shows the correlation coefficient of key “43” identified from the analyzed data. It is observed that key no. “43” out of 256 keys of AES has highest value of correlation; it implies that key “43” is identified.

Figure 3.15 shows the output of correlation coefficient of different keys for masked power traces with our proposed SCA-CM mechanism. It is clearly seen that no significant peak is observed in any of the 256 keys. It is obvious form the plots and analysis that our proposed scheme offers Side channel attack Counter Measure.
Figure 3-14 Correlation coefficient values of AES for different keys. There is a significant peak corresponding to correct key 43.

Figure 3-15 Correlation coefficient values of masked AES for different keys. There is no such significant peak so that correct key can be guessed.
The black trace corresponds to the correlation coefficients at 1000 different time points when correct key byte value is used to calculate the power hypothesis. The gray traces correspond to other incorrect key byte values. It turns out that the black trace differentiates itself from all the other gray traces (around 200μs). This means that the correct key byte value can be identified and the attack is successful. The experiment also mounted the same CPA attack with hardware module insertion of random size and also at random time with fixed size implementation. The example of the attack on the same key byte is shown in Figure 3-11. The black trace is totally buried into the gray traces. So the correct key byte value is not identifiable. Therefore the attack is not successful. Figure 3-11 and 3-12 give two visualized attack examples when the sampling trace number is 5120. To quantify the security improvement, researchers usually use Measurement to Disclosure (MTD). MTD gives the number of measurements required for a successful attack. Larger MTD indicates better security. Here, we define the successful attack to be uncovering all 16 bytes of the AES key. Table 3-1 presents more detailed attack results. For AES, 1280 measurements are sufficient to uncover all 16 key bytes. For AES with hardware module insertion, the number of measurements for full attack increases to 25600. The improvement is as high as 20. All these designs could be broken with a similar effort to break AES, as shown in Table 3-1. Table 3-1 also shows that AES with hardware module (random size) insertion, pays 10.7 times decrease of throughput and 1.1 times increase of footprint for higher capability of resisting CPA. The Power consumption increases by almost 90-100%. Since the implementation is having larger power but it has flexibility of using a platform for any encryption algorithm without modifying any hardware details of designed system. Just by changing the cryptographic algorithm to be implemented in the software is to be modified by inserting code of Hardware module injection at the critical point of cryptographic algorithm. It is also tested on DES and 3DES cryptographic algorithm.
### Table 3-1 Attack Results summary for AES algorithm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>AES</th>
<th>AES with Hardware Module insertion of random size</th>
<th>AES with Hardware Module insertion of Fixed size at random interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (kb/s)</td>
<td>1100</td>
<td>103</td>
<td>144</td>
</tr>
<tr>
<td>Footprint of software (kB)</td>
<td>45.7</td>
<td>48.3</td>
<td>48.3</td>
</tr>
<tr>
<td>Key bytes not found(^1)</td>
<td>1</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>@ 512</td>
<td>0</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>@ 1280</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>@ 12800</td>
<td>0</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>@ 25600</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

\(^1\) Number of measurements

### Table 3-2 Synthesis Results –Device Utilization summary: Xilinx Virtex -5LX110T-1ff1136 for AES

<table>
<thead>
<tr>
<th>Resources</th>
<th>Original: Without Random Hardware insertion</th>
<th>Random Hardware insertion</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Available</td>
<td>Used</td>
<td>Utilization</td>
</tr>
<tr>
<td>Slice Registers</td>
<td>69120</td>
<td>2599</td>
<td>3.76 %</td>
</tr>
<tr>
<td>Number of LUTs</td>
<td>69120</td>
<td>2220</td>
<td>3.76 %</td>
</tr>
<tr>
<td>Number of Bonded IOBs</td>
<td>640</td>
<td>48</td>
<td>7.5 %</td>
</tr>
<tr>
<td>Number of Block RAMs/FIFO</td>
<td>148</td>
<td>66</td>
<td>0.62 %</td>
</tr>
<tr>
<td>Number of BufG/BufGCTRLS</td>
<td>32</td>
<td>4</td>
<td>12.5 %</td>
</tr>
<tr>
<td>Number of ICAPs</td>
<td>2</td>
<td>0</td>
<td>0 %</td>
</tr>
</tbody>
</table>

### Table 3-3 Details about the random Hardware Module Sizes and their Configuration bit size with configuration time

<table>
<thead>
<tr>
<th>Hardware modules for Partial Reconfiguration</th>
<th>Size in terms of CLBs</th>
<th>Configuration File size</th>
<th>Configuration time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size 1 x</td>
<td>5 x 5</td>
<td>14 KB</td>
<td>0.14 ms</td>
</tr>
<tr>
<td>2 x</td>
<td>10 x 5</td>
<td>29 KB</td>
<td>0.29 ms</td>
</tr>
<tr>
<td>3 x</td>
<td>10 x 10</td>
<td>56 KB</td>
<td>0.56 ms</td>
</tr>
<tr>
<td>4 x</td>
<td>15 x 10</td>
<td>80 KB</td>
<td>0.80 ms</td>
</tr>
</tbody>
</table>
Full Configuration of FPGA:
3799KB (Configuration Bit File size) so configuration time is 37.99 ms at clock frequency of 100 MHz.

**Table 3-4 Power consumption report for various Cryptographic algorithms**

<table>
<thead>
<tr>
<th>Strategy / Cryptographic Algorithm</th>
<th>AES</th>
<th>DES</th>
<th>3DES</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Countermeasure</td>
<td>1231</td>
<td>1020</td>
<td>1365</td>
</tr>
<tr>
<td>Software Counter measure</td>
<td>1274</td>
<td>1042</td>
<td>1402</td>
</tr>
<tr>
<td>With random size of Hardware module with regular interval</td>
<td>1822</td>
<td>1657</td>
<td>2033</td>
</tr>
<tr>
<td>With random Time insertion of Hardware module with fixed size (10 x 10)</td>
<td>1820</td>
<td>1650</td>
<td>2010</td>
</tr>
</tbody>
</table>

**Figure 3-16 Power Consumption comparison for various Crypto algorithms with proposed approach**
It can be observed from the graph shown in Figure 3-16 that there is no significant difference between no countermeasure and software counter measure. But Hardware counter measure power consumption is increased by 56 % or so. But it gives the safety almost 20 times as discussed earlier.

RIJIJD [95] framework is using Random Code injection to Mask power analysis but it requires special hardware changes in the processor architecture along with compiler support. We are comparing our proposal with it in terms of energy overhead, hardware overhead and run time overhead for various encryption algorithms in the following table 3-5.

<table>
<thead>
<tr>
<th>Types of Overheads</th>
<th>SCA Counter Measure methods</th>
<th>Types of Encryption Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AES</td>
<td>DES</td>
</tr>
<tr>
<td>Hardware Overhead in %</td>
<td>RIJID</td>
<td>1.98</td>
</tr>
<tr>
<td></td>
<td>Our proposal</td>
<td>1.15</td>
</tr>
<tr>
<td>Run Time Overhead in %</td>
<td>RIJID</td>
<td>24.67</td>
</tr>
<tr>
<td>Energy Overhead in %</td>
<td>RIJID</td>
<td>38.2</td>
</tr>
<tr>
<td></td>
<td>Our Proposal</td>
<td>67</td>
</tr>
</tbody>
</table>

3.8 Summary
A HW/SW based randomized hardware module injection technique is proposed here, to overcome the drawbacks of previous countermeasures. The Proposed technique injects random hardware modules at random places in FPGAs during the execution of an application which protects the system from both SPA and DPA. For AES with hardware module insertion, the number of measurements for full attack increases to 25600. The improvement is as high as 20 time compare to no countermeasure. It also shows that AES with hardware module (random size) insertion, pays 10.7 times decrease of throughput and 1.1 times increase of footprint for higher capability of resisting CPA. The Power consumption increases by almost 60 %. The hardware overhead is 5-10 % depends on number of hardware modules used. The proposed implementation is having larger power
but it has flexibility of using platform for any encryption algorithm without modifying any hardware details of designed system. Here modification is required in software by inserting code for Hardware module injection at the critical point of cryptographic algorithm.