*IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*
*Volume 3, Issue 3 (Sep. – Oct. 2013), PP 38-43*
*e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197*
*www.iosrjournals.org*

# Fault injection test bed for clock violation or metastability based Cipher attacks on FPGA hardware

Rashmi Singh, S. Latha,
*( Electronics and Communication Engineering, Aurora Technological and Research Institute/ JNTU, India)*
*HOD and associate professor in ECE department, Aurora Technological and Research institute, JNTU Hyderabad, India.*

***Abstract:*** *In this project an FPGA based test bed is realized for injecting faults through clock glitches, to result in setup and hold violations. The UART interface is realized on FPGA to provide PC based controlling for this fault injection. The pre-build serial International Data Encryption (IDEA) algorithm synthesis models will be used as test encryption algorithm. The Xilinx Digital clock manager (DCM) component will be used for generation clocks of different frequencies and phase shifts. The encryption module output with faults introduced and without fault introduced is compared as a function of ratio of used clock frequency and maximum frequency of operation reported by synthesis tool. The modules for clock generation, clock switching, interface adopter to IDEA core and UART interface will be realized and tested in FPGA hardware in integrated form. From PC using HyperTerminal commands will be sent to FPGA firmware. Xilinx simulation and synthesis tools are used to this project. Xilinx Spartan family FPGA board along with serial communication with PC will be used for hardware level testing. Xilinx chipscope tools will be used for verifying the output at various levels in FPGA hardware.*

***Keywords:*** *Brute force attacks, digital clock manager, invasive techniques, international data encryption algorithm, non invasive techniques.*

## I. Introduction

Cryptanalysis is the field of research where the cryptographic algorithms are studied to break them and further extract the information. This subject gains more and more power with high speed running processors and FPGAs as the time required to iterate for millions of key word (password) combinations comes down with high speed running hardware. There are several types of cryptanalysis algorithms apart from brute force attacks on dictionary based searching. In recent days the new class of cryptanalysis algorithms is evolved based on clock violations and metastable conditions of flip-flops which are part of encryption algorithms.

Current problem is that the Attack technologies are being constantly improved and there is growing demand for secure chips. One of the proposed solution is to use abnormal working conditions to generate malfunctions in the system which provides a way to analyze the susceptibility of FPGA to faults. The main aim of this paper is to inject the faults into FPGA based logics using clock glitching mechanism and to thoroughly analyze the impact of injected faults on the operation of the circuit logic.

A long term objective of our research is to develop an efficient method for protecting FPGA-based implementations of cryptographic algorithms through effective concurrent testing of various types of faults, including faults injected by the attackers. An essential part of this research is to develop a method and tool for the evaluation of susceptibility of FPGA based circuits to fault injection attacks. In this paper, we present such a method and tool. It allows us to examine an FPGA-based circuit, in particular an implementation of a cryptographic algorithm, subjected to a fault injection attack based on clock glitching. The effectiveness of the proposed approach is assessed for the IDEA implementation. This paper consists of four sections namely hardware used, the testing procedure, the results obtained followed by conclusion.

## II. Hardware

### 2.1Block diagram

We will be using Spartran 3E board for our implementation and the IC number is XC3S500E-4fg320 and VHDL will be used as programming language. The block diagram of our implementation is as shown in Figure below. Here the CUT is IDEA and the test vectors will be generated using DCM that is digital clock manager and using clock division principle. As per the select line selected the appropriate frequency will be given to the CUT.

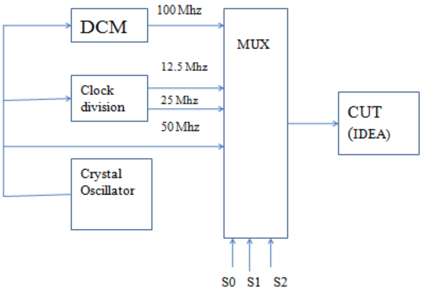


Figure1 Block diagram of implementation

### 2.2 Fault injection

Several fault injection techniques have been proposed and practically experimented. They can be grouped into hardware-based fault injection, software-based fault injection, simulation-based fault injection, emulation-based fault injection and hybrid fault injection. From another point of view, the fault injection techniques can be grouped into invasive and noninvasive techniques. The problem with sufficiently complex systems, particularly time dependant ones, is that it may be impossible to remove the footprint of the testing mechanism from the behavior of the system, independent of the fault injected. Invasive techniques are those which leave behind such a footprint during testing. Non-invasive techniques are able to mask their presence so as to have no effect on the system other than the faults they inject.

### 2.3 Clock glitching

With more and more multi-frequency clocks being used in today's chips, especially in the communications field, it is often necessary to switch the source of a clock line while the chip is running. This is usually implemented by multiplexing two different frequency clock sources in hardware and controlling the multiplexer select line by internal logic. The two clock frequencies could be totally unrelated to each other or they may be multiples of each other. In either case, there is a chance of generating a glitch on the clock line at the time of the switch. A glitch on the clock line is hazardous to the whole system, as it could be interpreted as a capture clock edge by some registers while missed by others. Fig shown below has a simple implementation of a clock switch, using an AND-OR type multiplexer logic and also gives the timing signals.

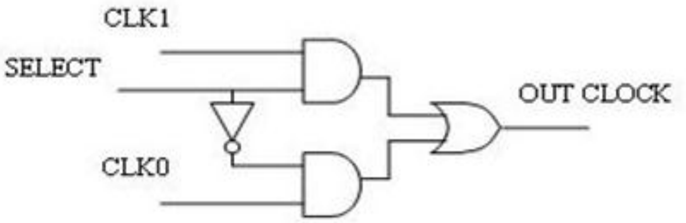


Figure2 Clock switching multiplexer

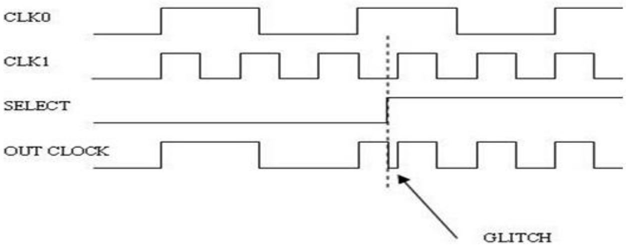


Figure3 Clock glitch generation

The multiplexer has one control signal, named SELECT, which either propagates CLK0 to the output when set to "zero" or propagates CLK1 to the output when set to "one." A glitch may be caused due to immediate switching of the output from Current Clock source to the Next Clock source, when the SELECT value changes. Current Clock is the clock source currently selected while Next Clock is the clock source corresponding to the new SELECT value. The problem with this kind of switch is that the switch control signal can change any time with respect to the source clocks, thus creating a potential for chopping the output clock or creating a glitch at the output.

### 2.4 Circuit under Test

IDEA is a block cipher that uses a 128-bit key to encrypt 64-bit data blocks. The 52 subkeys are all generated from the 128-bit original key. IDEA algorithm uses 52, 16-bit key sub-blocks, i.e. six subkeys for each of the first eight rounds and four more for the ninth round of output transformation. The following figure shows encryption with 52 subkeys

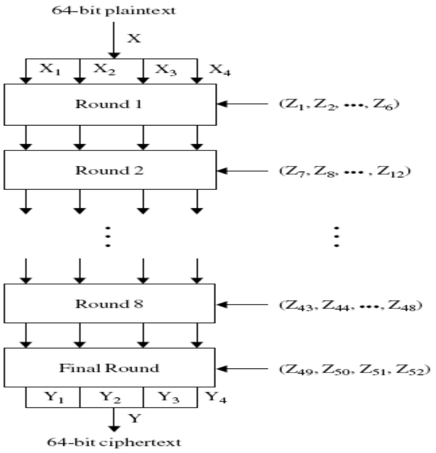

64-bit plaintext
X
$X_1$
$X_2$
$X_3$
$X_4$
Round 1
$(Z_1, Z_2, \cdots, Z_6)$
Round 2
$(Z_7, Z_8, \cdots, Z_{12})$
Round 8
$(Z_{43}, Z_{44}, \cdots, Z_{48})$
Final Round
$(Z_{49}, Z_{50}, Z_{51}, Z_{52})$
$Y_1$
$Y_2$
$Y_3$
$Y_4$
Y
64-bit ciphertext

Figure4 Encryption rounds

The 128-bit encryption key is divided into eight 16-bit subkeys. The first eight subkeys, labeled Z1, Z2, . . . , Z8 are taken directly from the key. After that, the key is circularly shifted 25 bits to the left and again divided into eight subkeys. This procedure is repeated until all 52 subkeys have been generated. In IDEA, the plaintext is 64 bits in length and the key size is 128 bits long. The design methodology behind the IDEA algorithm is based on mixing three different operations. These operations are:

⊕ Bit-by-bit XOR of 16-bit sub-blocks
⊞ Addition of 16-bit integers modulo $2^{16}$
⊙ Multiplication of 16-bit integers modulo $2^{16} + 1$

The following figure shows the IDEA structure

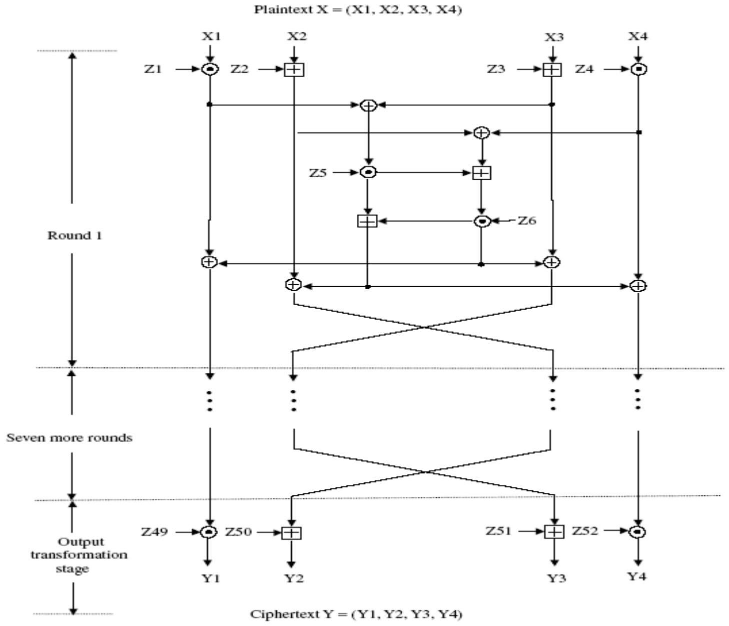

Plaintext X = (X1, X2, X3, X4)
X1
X2
X3
X4
Z1
Z2
Z3
Z4
Z5
Z6
Round 1
Seven more rounds
Output transformation stage
Z49
Z50
Z51
Z52
Y1
Y2
Y3
Y4
Ciphertext Y = (Y1, Y2, Y3, Y4)

Figure5 IDEA structure

### 2.5 Clock division and DCM

As our CUT operates at 20 MHz, we will be generating different frequencies using clock division mechanism and DCM. Using clock division we will generate 12.5 MHz and 25 MHz frequency. In our implementation we are using DCM for generation of higher clock frequencies (Greater than 50 MHz) since maximum clock frequency of Spartan 3e is 50Mhz. we are just going to configure the required clock and the vhdl code will generated. this module will be instantiated in the top level module for using the higher frequencies.

## III. Testing Procedure

In our implementation we will be having clock and reset signals along with three user input switches. These switches are used for switching the input clock frequency of the CUT (Idea algorithm in this case). Thus we will be observing the behavior of the algorithm by operating at the different clock frequencies (12.5Mhz,25Mhz,50Mhz,100Mhz). Depending on the switch input one of the clock frequencies is given as input to the encryption algorithm. Crystal oscillator on the FPGA generates 50 MHz frequency, 12.5 MHz and 25 MHz frequencies will be generated using clock division technique and 100 MHz will be generated using DCM. Depending on the select lines selected MUX gives the appropriate frequency to the input of CUT, this is as shown in the table below.

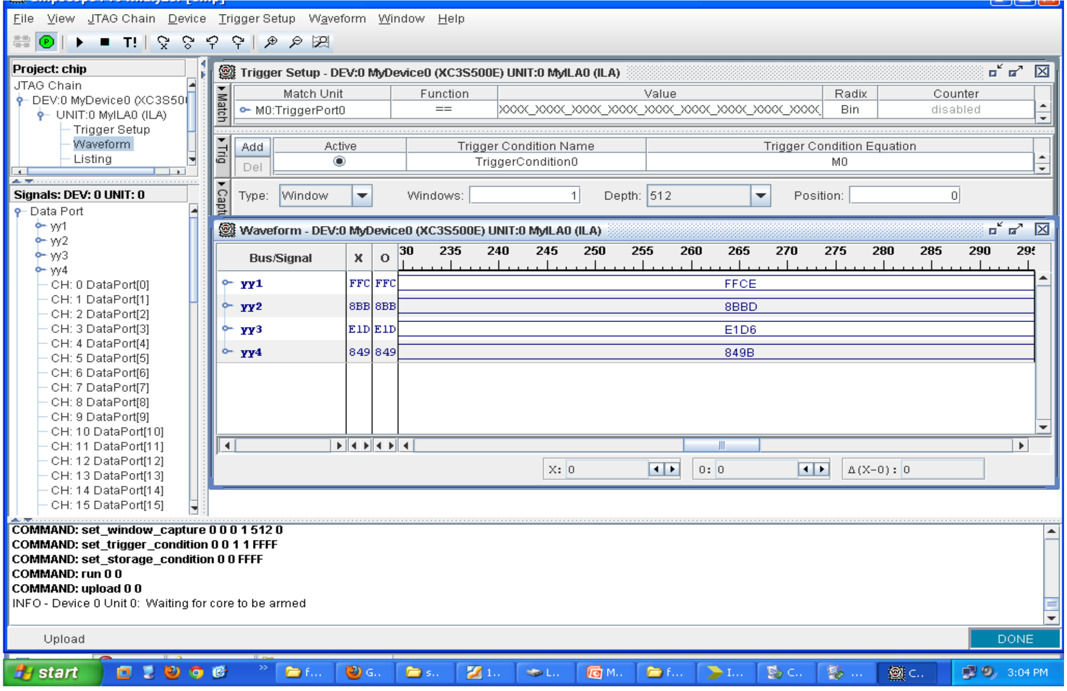
Table: Output of MUX depending on select lines

| S0 | S1 | S2 | Output |
|---|---|---|---|
| **0** | **0** | **0** | **12.5Mhz** |
| **1** | **0** | **0** | **25Mhz** |
| **0** | **1** | **0** | **50Mhz** |
| **0** | **0** | **1** | **100Mhz** |

## IV. Results

IDEA algorithm has a maximum freqency of approximately 19MHZ frequency. So the clock freqency applied should be less than this freqency or approximately around this frequency. If we apply the freqency more than this freqency then the metastability condition takes place and the output will be corrupted depending on the applied clock freqencies. The below chipscope results show how the idea encryption algorithm is behaving with the corresponding different clock freqencies.

The chipscope results are shown as follows



Trigger Setup - DEV:0 MyDevice0 (XC3S500E) UNIT:0 MyILA0 (ILA)

| Match Unit | Function | Value | Radix | Counter |
|---|---|---|---|---|
| M0:TriggerPort0 | == | XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX | Bin | disabled |

| Active | Trigger Condition Name | Trigger Condition Equation |
|---|---|---|
| ● | TriggerCondition0 | M0 |

Type: Window Windows: 1 Depth: 512 Position: 0

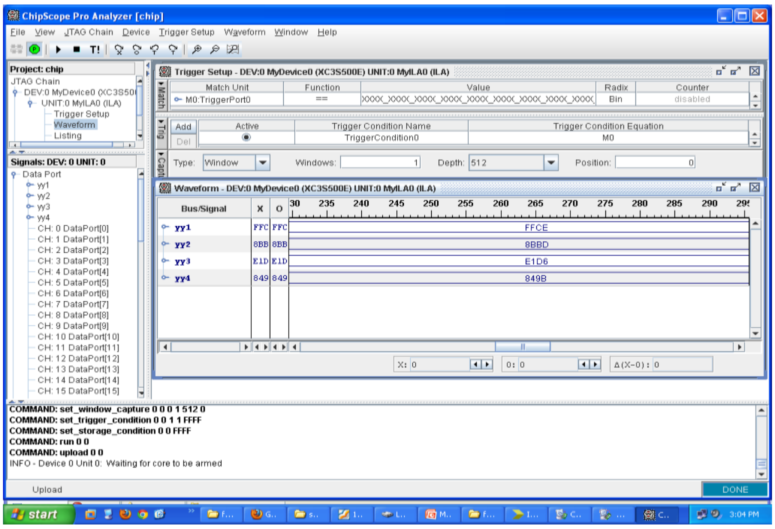
Waveform - DEV:0 MyDevice0 (XC3S500E) UNIT:0 MyILA0 (ILA)

| Bus/Signal | X | O | 230 – 295 |
|---|---|---|---|
| yy1 | FFC | FFC | FFCE |
| yy2 | 8BB | 8BB | 8BBD |
| yy3 | E1D | E1D | E1D6 |
| yy4 | 849 | 849 | 849B |

X: 0 O: 0 Δ(X-O): 0

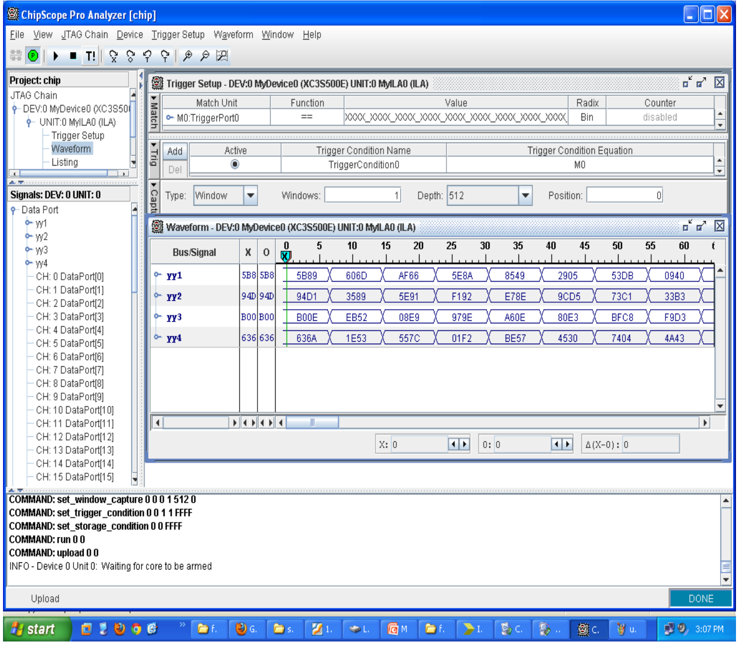
```
COMMAND: set_window_capture 0 0 0 1 512 0
COMMAND: set_trigger_condition 0 0 1 1 FFFF
COMMAND: set_storage_condition 0 0 FFFF
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0:  Waiting for core to be armed
```

Figure6 Chipscope result for a frequency of 12.5 MHz



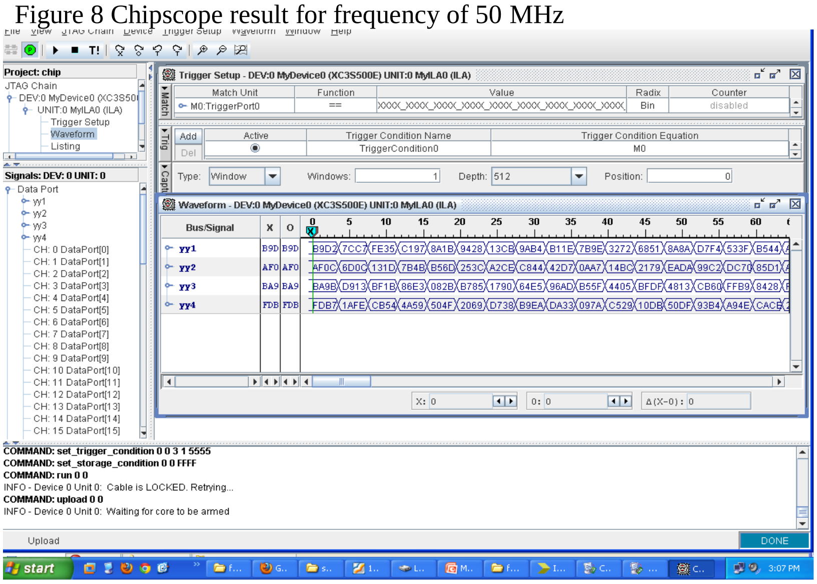
```
COMMAND: set_window_capture 0 0 0 1 512 0
COMMAND: set_trigger_condition 0 0 1 1 FFFF
COMMAND: set_storage_condition 0 0 FFFF
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0:  Waiting for core to be armed
```

Figure7 Chipscope result for a frequency of 25 MHz



```
COMMAND: set_window_capture 0 0 0 1 512 0
COMMAND: set_trigger_condition 0 0 1 1 FFFF
COMMAND: set_storage_condition 0 0 FFFF
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0:  Waiting for core to be armed
```

Figure 8 Chipscope result for frequency of 50 MHz



```
COMMAND: set_trigger_condition 0 0 3 1 5555
COMMAND: set_storage_condition 0 0 FFFF
COMMAND: run 0 0
INFO - Device 0 Unit 0:  Cable is LOCKED. Retrying...
COMMAND: upload 0 0
INFO - Device 0 Unit 0:  Waiting for core to be armed
```

Figure9 Chipscope result for frequency of 100 MHz

## V. Conclusion

The presented method and tool for injecting faults in an FPGA-based circuit, based on clock glitching, has some unique features that allow us to thoroughly examine and analyse the impact of such faults on the operation of the circuit. In particular, through precise adjustment of the frequency of an external clock generator, we can control the number of faults occurring at the output of the circuit under test.

The results obtained by applying the proposed technique to the IDEA implementation lead to a number of practical guidelines that may be essential when planning experimental studies on fault injection in FPGA-based circuits.

## References

[1] D. Saha, D. Mukhopadhyay and D. RoyChowdhury, "A Diagonal Fault Attack on the Advanced Encryption Standard", IACR Cryptology ePrint Archive, vol. 2009, p. 581, 2009.

[2] J. Balasch, B. Gierlichs and I. Verbauwhede, "An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs", Proc. Workshop on Fault Diagnosis and Tolerance in Cryptography, 2011.

[3]. http://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm

[4]. http://www.limited-entropy.com/fault-injection-techniques

[5]. http://en.wikipedia.org/wiki/Metastability_in_electronics