REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
29-05-2015	Final Report	1-Aug-2014 - 30-Apr-2015

4. TITLE AND SUBTITLE
Final Report: Side-Channel-Proof Circuit Design for Hardware Security

5a. CONTRACT NUMBER
W911NF-14-1-0355

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS
Jia Di

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of Arkansas
210 Administration Building
1 University of Arkansas
Fayetteville, AR 72701 -1201

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
64830-CS-II.1

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
This STIR project was highly successful. The team has significantly reduced the overhead of side-channel-proof Multi-Threshold Dual-Spacer, Dual-Rail, Delay-Insensitive (MTD3L) asynchronous logic, without compromising the resistivity against power-based side-channel attacks. The improvements of the new MTD3L architecture compared with the old MTD3L architecture in implementing an AES core is 353% in power, 388% in speed, and 546% in registration size.

15. SUBJECT TERMS
Side Channel Attack, MTD3L, Asynchronous Logic, Overhead Reduction

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jia Di
UU	UU	UU	UU		19b. TELEPHONE NUMBER 479-575-5728

Report Title

Final Report: Side-Channel-Proof Circuit Design for Hardware Security

ABSTRACT

This STIR project was highly successful. The team has significantly reduced the overhead of side-channel-proof Multi-Threshold Dual-Spacer, Dual-Rail, Delay-Insensitive (MTD3L) asynchronous logic, without compromising the resistivity against power-based side-channel attacks. The improvements of the new MTD3L architecture compared with the old MTD3L architecture in implementing an AES core is 353% in power, 388% in speed, and 546% in registration size.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received Paper

TOTAL:

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received Paper

TOTAL:

Number of Manuscripts:

Books

Received Book

TOTAL:

Received Book Chapter

TOTAL:

Patents Submitted

Patents Awarded

Awards

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Jean Habimana	0.50	
FTE Equivalent:	**0.50**	
Total Number:	**1**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Jia Di	0.15	
FTE Equivalent:	**0.15**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period: 0.00

The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:..... 0.00

The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:..... 0.00

Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):..... 0.00

Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:..... 0.00

The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense 0.00

The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields: 0.00

Names of Personnel receiving masters degrees

NAME
Total Number:

Names of personnel receiving PHDs

NAME
Total Number:

Names of other research staff

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Sub Contractors (DD882)

Inventions (DD882)

Scientific Progress

See Attachment

Technology Transfer

STIR Project Final Report

SIDE-CHANNEL-PROOF CIRCUIT DESIGN FOR HARDWARE SECURITY

Jia Di, University of Arkansas, jdi@uark.edu

Summary: This STIR project was highly successful. The team has significantly reduced the overhead of side-channel-proof Multi-Threshold Dual-Spacer, Dual-Rail, Delay-Insensitive (MTD^{3}L) asynchronous logic, without compromising the resistivity against power-based side-channel attacks. The improvements of the new MTD^{3}L architecture compared with the old MTD^{3}L architecture in implementing an AES core is **353% in power, 388% in speed, and 546% in registration size**.

1. Task 1: Optimization of the MTD^{3}L Registration Mechanism

Consider the registration cost for the previous version as shown in table 1.

	Old version **MTD^{3}L**	
128 bit register	Regular Register	16212
	Spacer Filter Register	27516
	Spacer generator Register	32124
AES Core registration	242264	

Table 1: Old Version MTD^{3}L Registration transistor repartition

From the table above, it is clear that the Spacer Filter and the Spacer generator Registers make for most of the registration overhead. The Spacer generators were used to generate All Zero Spacers alternatively with All One Spacers, and the Spacer Filters were used to distinguish the Spacers from DATA depending on whether it is a DATA wave passing through the register or a NULL wave (Spacer). The new MTD^{3}L architecture intends to reduce the registration overhead by replacing the Spacer Generators and Spacer Filters with different mechanisms that assure the same functionalities with a lesser cost in terms of transistor count, power consumption and delay.

1.1 Spacer Generator Registers Elimination

The Spacer Generators can be replaced with Sleepable Registers. Sleepable registers generate All One Spacers (A1s) and All Zero Spacers (A0s), respectively, by sleeping to 1 or to 0. Each rail of a dual-rail signal has its own register logic, and when both logics are slept to logic 1, it is the equivalent of an All One Spacer; the same way, when both logics are slept to logic 0, it is the equivalent of an All Zero Spacer.

Like other MTD^{3}L gates, the new sleepable registers have a sleep to '0' and a sleep to '1' control inputs. However, because of the nature of the register circuit, instead of passing **s1** to the register, its inverse **ns1 (not s1)** is used. Figure 1 shows the schematic of the new register circuit.

The sleep to 0 **(s0)** signal when asserted, the output of the circuit is logic 0; the sleep to 1 **(s1)** when asserted, the circuit output is logic 1. Table 2 shows the relationship between **s0, s1, ns1** and the register's output.

s0	s1	ns1	Register Output
1	0	1	All Zeros Spacer
0	1	0	All Ones Spacer
0	0	1	Data

Table 2: Sleep Signals and Register's Output relationship

Notice that **s0** and **s1** are to be asserted exclusively. This is explained by the circuit diagram in Figure 2.

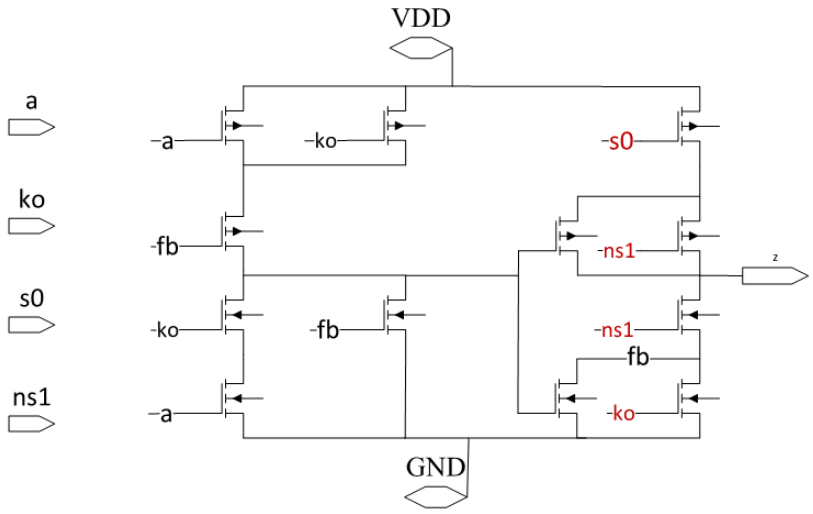

Figure 1: New Version MTD^{3}L Register Circuit

Sleep signals generation

The new architecture uses a **Sleep Early Completion** mechanism. The Completion Logic checks all the inputs going to the Registers, and asserts a **Completion Signal** when all the inputs are valid data or set it to logic 0 when all inputs are NULL (All Zero or All One Spacer). The Completion Logic then uses Previous Spacer signal **(ps)** to determine which value the register should sleep to or which Spacer to generate next. The **ps** signal shows which spacer was used the previous time the register was in a sleep mode. If the previous spacer was All One, then the next spacer should be All Zero; therefore, **s0** is asserted. Otherwise, if the previous spacer was All Zero, the **s1** signal is asserted. Table 2 shows the relationship between the previous spacer, **s0** and **s1**.

ps	s0	s1	Generated Spacer
1	1	0	All Zero Spacer
0	0	1	All One Spacer

Table 3: Relationship between previous spacer, sleep signal and the next spacer

The circuit diagram in Figure 2 explains how the values in the table above are generated.

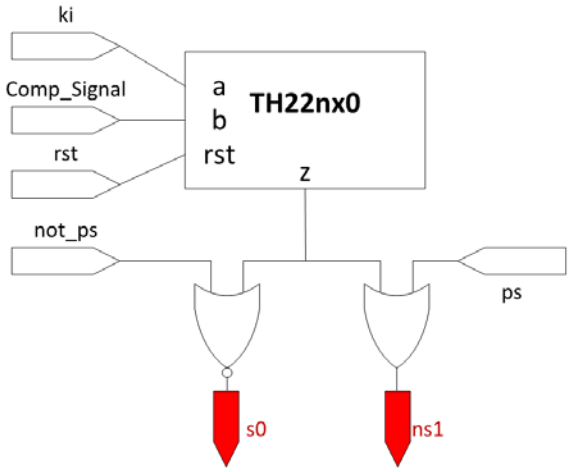

Figure 2: Sleep Signals Generation Circuit

Notice: The Sleepable registers used with a Completion Logic Unit able to generate **s0** and **s1** signals alternatively, assure the generation of the needed Spacer. This helps eliminate the need for Spacer Generator Registers, which contributes to a big overhead reduction given the number of transistors that were needed to implement the Spacer Generator Registers.

1.2 Spacer Filter Elimination

Compared to other dual-ail logic such as NCL or MTNCL, MTD^{3}L architecture has to overcome the challenge of handling the **All One Spacer** as a NULL wave. In fact, for NCL and MTNCL, logic 1 means that the rail of interest is presenting data, but in the MTD^{3}L architecture, logic 1 could mean that the rail is presenting data or an All One Spacer depending on the value of the other rail of the signal. This complicates the MTD^{3}L Registration, which was why Spacer Filter Registers were used to solve in the Old version. The new MTD^{3}L version intends to get around the All One Spacer problem without using a high cost mechanism like Spacer Filters.

Dual-rail circuits including MTD^{3}L use handshaking signals to control data propagation. **ko** is a signal used to request DATA or a Spacer from the previous stages of the circuit. Table 2 shows the relationship between **ko, s0, s1** and the **outputs** of the stage registers.

ko	s0	s1	Register outputs
1	1	0	All Zero Spacer
1	0	1	All One Spacer
0	0	0	Valid DATA

Table 4 Relationship between ko, s0, s1 and Register Outputs

Notice that **s0** and **s1** are all logic 0 when **ko** is logic 0. The architecture is designed that way to allow the register output to be valid data when **ko** is logic 0 (Request For NULL).

The fact that **ko** is logic 0 when the register is passing DATA can be used for more than sending the request for NULL (Request for a Spacer). In fact, instead of using a **Spacer Filter** to block All One Spacer from interfering with data, **ko** is added in the Register's circuit to prevent an All One Spacer to cause the register's output to change to logic 1 before all registers in the current stage are ready to change their output from DATA to NULL. Figure 3 shows the schematic of a one rail register circuit and highlights how **ko** is used to control All One Spacer.

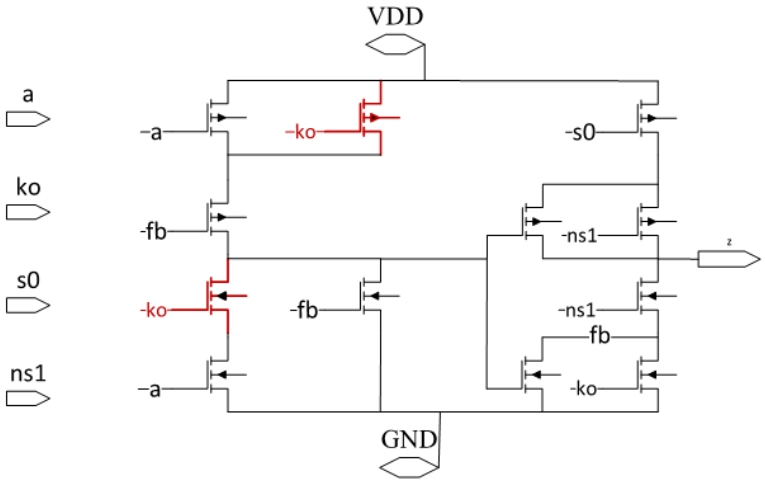

Figure 3: All One Spacer control mechanism

The circuit in the diagram above shows how the register's output is kept unchanged until the register is slept to 1 or to 0. Notice that for input **'a'** to pull the register's output to logic 1, both **a** and **ko** need to be logic 1. However, as shown in Table 4, when **ko** is logic 1, either **s0** or **s1** is asserted; therefore, the register is generating a Spacer, which means the input value **(a's value)** does not matter. On the other hand, when **ko** is logic 0 (both **s0** and **s1** are not asserted), the register's output cannot rise to logic 1 even if the input is logic 1. The only time the output can rise is exactly when **ko** is switching from logic 1 to 0. The architecture makes sure that the internal node controlled by the input **a** and **ko** has risen before **ko** falls. This way, All One Spacers will not interfere with data on the output node of the register because the spacer is presented when **ko** is logic 0 (request for spacer).

1.3 Register Circuit Simplification

After eliminating Spacer Generator and Spacer Filter registers, the new MTD^{3}L registration depends solely on the actual register circuit. It is important to keep this circuit to its minimum size, power consumption and delay in order to achieve a significant overhead reduction. Notice from the schematic in Figure 3 that 12 transistors are used for one rail's register. Table 3 shows improvements made in terms of transistor count reduction.

128 bit register	**Old version registration**		**New version registration**	
	Regular Register	16212	1 rail	12
	Spacer Filter Register	27516	1 bit	24
	Spacer generator Register	32124		
	Total	**75852**	128 bits	**3072**

Table 5: Transistor count comparison between Old and New MTD^{3}L

1.4 AES core Results

The new MTD^{3}L architecture was used to design an AES core, and the circuit was compared to the old MTD^{3}L and NCL AES core circuits in terms of energy, transistor count (size) and delay. The comparison results are shown in the following tables.

Design	**Delay (ns)**	**Energy (nJ)**
New MTD^{3}L	85	1.085
Old MTD^{3}L	330	3.84
NCL	462	2.208

Table 6: Speed and Energy comparison

Size comparison (In terms of transistor count)

Design	**Registration transistors**	**Combination Logic**	**Total**
New MTD^{3}L	44344	220072	264416
Old MTD^{3}L	242364	Not Available	Not Available
NCL	49140	Not Available	Not Available

Table 7: Registration transistor count comparison

The data in the table above suggests that the new MTD^{3}L architecture brings a **546.6 %** improvement over the old architecture in terms of registration transistor count.

Registration Energy Consumption

Design	**Registration**	**Logic**	**Total**
NCL	0.252	1.512	2.208
Old MTD^{3}L	1.836	1.26	3.84
New MTD^{3}L	0.414	0.671	1.085

Table 8: Registration and logic power consumption in (nJ)

New MTD^{3}L Improvements

Design Component	**Old MTD^{3}L**	**New MTD^{3}L**	**Improvement**
Speed	330	84.9	388.7 %
Registration power consumption	1.836	0.414	443.5 %
Total power consumption	3.84	1.085	353.9 %
Registration transistor count	242364	44344	546.6 %

Table 9: New MTD^{3}L Improvement

2. Task 2: Develop an Automated MTD^{3}L Design Flow

An automated design flow for MTD^{3}L circuits has been developed as shown in Figure 4.

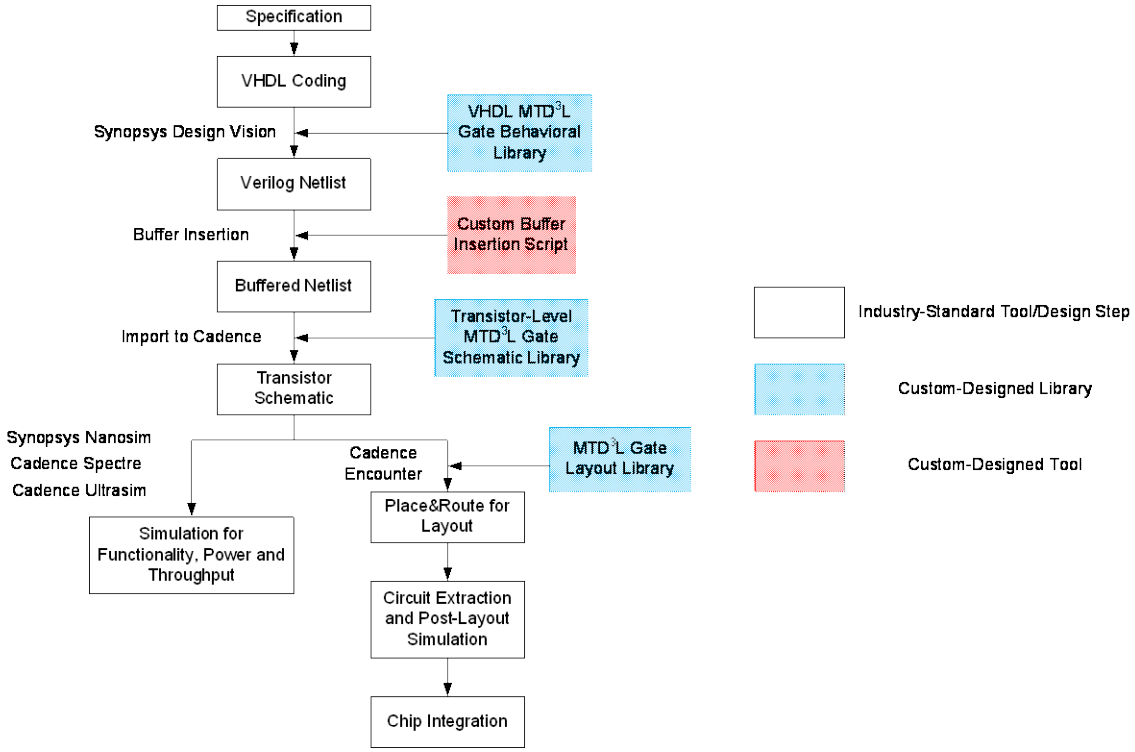

Figure 4: Automated MTD^{3}L Design Flow

In Figure 4, the design entry is structural VHDL instead of RTL behavioral VHDL because asynchronous synthesis algorithms are not available. A MTD^{3}L gate behavior library is used for assisting the functional simulation. After the design is imported to Cadence and transistor-level schematics are generated, transistor-level simulation will be performed with the aid of a MTD^{3}L transistor-level gate schematic library. The automatic place, route and layout will be based on a MTD^{3}L gate layout library. This flow is applicable to any customized digital logic circuit as long as the circuit can be described in a structural fashion.

3. Task 3: Evaluate the SCA Mitigation Effectiveness and Efficiency of the Optimized MTD^{3}L AES Core

A DPA (Differential Power Attack) was applied to the new MTD^{3}L AES core to check if the ability and effectiveness of the MTD^{3}L architecture to mitigate Side Channel Attacks was lost or compromised during size, power and speed optimization process.

3.1 DPA Overview

DPA Definition

DPA or Differential Power Attack is a Side Channel Attack that relies on the tie between a circuit's power consumption and the data being processed to deduct information, otherwise

hidden by safe algorithms, from power consumption information inevitably available to the attacker.

For an AES core, DPA targets to deduct the encryption key from the power consumption information. In fact, during AES encryption process there are points where the key value influences the circuit power consumption. The round key addition (also known as "AddRoundKey") and the S-box substitution steps are the most vulnerable points to DPA. These points are the most likely to reveal the encryption key information through power consumption because these are the steps where the key is being used in computations. With the assumption that energy consumed by a node to switch to logic 1 is different from the energy consumed by the same node to switch to logic 0, it is implied that the energy consumption at a time by a circuit is related to the number of nodes switching to logic 1 and nodes switching to logic 0 at that time. The same way, the energy consumption during the AddRoundKey and S-box steps is related to the number of nodes switching to logic 1 and the number of those switching to logic 0, and these numbers are strongly correlated to the encryption key.

DPA Process

Due to the correlation between power consumption and data (encryption key in this case), to deduct the key, DPA runs a guessing process and uses different mechanisms to check whether the guessed value is the actual encryption key.

The guessing process consists of trying all the possible values until one is proved to be the real key. However, for an AES core with a 128 bits key, it would take too long to process all the possible values (2^{128} values). One way to solve that problem is to target one byte of the key at a time. In that case only 256 guesses are required to exhaustively cover all the possible cases; therefore, 16×256 (4096) key guesses are required to get the whole encryption key.

A key guess is verified by checking if the guessed key could produce power traces similar to the ones produced by the circuit while processing the same plain texts. For this process, a software model of the AES core has to be designed, and it should be able to emulate the AES functionality at least up to the point of the attack; in other words, the model has to cover the AddRoundKey and S-box steps. The software model helps divide power traces in 2 groups: Group 1 and Group 0. Group 1 consists of power traces corresponding to plain texts that should cause the circuit to dissipate more power due to the switching activities that result from those plain texts. Group 0 on the other hand, consists of power trace corresponding to plain texts that result in switching activities that should use less power during key processing operations (AddRoundKey and S-box substitution). This software model and the grouping process will be explained later.

By adding all traces in each group together, it is anticipated that the difference in power consumption at points where AddRoundKey and S-box substitution are performed will accumulate. Finally, this difference can be illustrated by averaging the 2 groups' power traces and subtracting one Group's average from the other's average. In other words, if the resulting Differential Power Trace has a spike, it is a sign that a significant difference in power consumption was accumulated between Group 1 and Group 0 traces, which means that the key

used to generate the Groupings is the same one as the actual encryption key in circuit, so the key has been successfully guessed.

On the other hand, if a different key was used for the Grouping, power traces in each group has nothing in common that relates to the encryption key. Therefore, when traces are added together, there is no correlation being built up, and later when the resulting averages are subtracted on from the other, there will be no spike on the Differential Power trace.

3.2 Performed DPA Details and Results

For this project, DPA was performed to check/prove that the optimized design is still able to mitigate Side Channel Attack. Therefore, for the performed attack the key was already known, so the purpose was to show that the resulting Differential Power trace using the right key as a guess does not show any signs of correlation or spike, which would mean that the circuit is safe against Side Channel Attack.

First Round Attack vs Last Round attack

An AES core can be attacked from the First Round or from the Last Round. For the First Round attack, DPA targets the First Round AddRoundKey step and the first S-box substitution in the AES encryption. Different plain texts are used to run simulations that generate enough power traces for the attack to be successful, and a software model that emulates the AddRoundKey and S-box steps is used for the power traces grouping. On the other hand, the Last Round Attack targets the Last Round S-box and AddRoundKey steps. For this, a software model is built to reverse the encryption process, thus starting from the cipher texts, and reversing the encryption process from the output back to the Last Round S-box to compute the information used for the power trace grouping.

In general, the First Round attack has an advantage in that when the key value is correctly guessed, it is the actual encryption key, whereas for the Last Round attack, when the key is guessed, it is the Last Round key, which requires additional computations to deduct the actual encryption key. Moreover, for our case, the First Round attack presents another advantage in terms of simulation time. In fact, since the First Round attack targets the first AddRoundKey and S-box steps, it is not necessary to run the whole encryption for each plain text in order to perform this attack. In contrary, for the Last Round attack the whole encryption has to be run in order to get a power trace including the Last Round S-box and Add Round key steps. For those 2 reasons, the First Round attack was used in this project.

Circuit Simulation and power trace gathering

From different DPA projects, it is noted that at least 1000 power traces are required for the attack to be successful. For this project, 4000 power traces were used in order to perform a strong attack. Cadence UltraSim was used for simulation. Obviously, it would be impossible or very time consuming to run 4000 simulations one by one, so a vector file that runs all the 4000 plain texts at once was created. The graph in Figure 4 shows power traces for consecutive random plain texts.

Notice from Figure 4 that there are patterns being repeated every **0.01 us** or **10ns**. The simulation was run for 10 ns for each plain text because it was proved that 10 ns is enough time to reset the circuit, feed the circuit with a new plain text, and run it to cover at least the first AddRoundKey and S-box operations. In other words, every 10 ns on the graph correspond to a power trace for 1 plain text. Also, notice that the X-axis is in uS units.

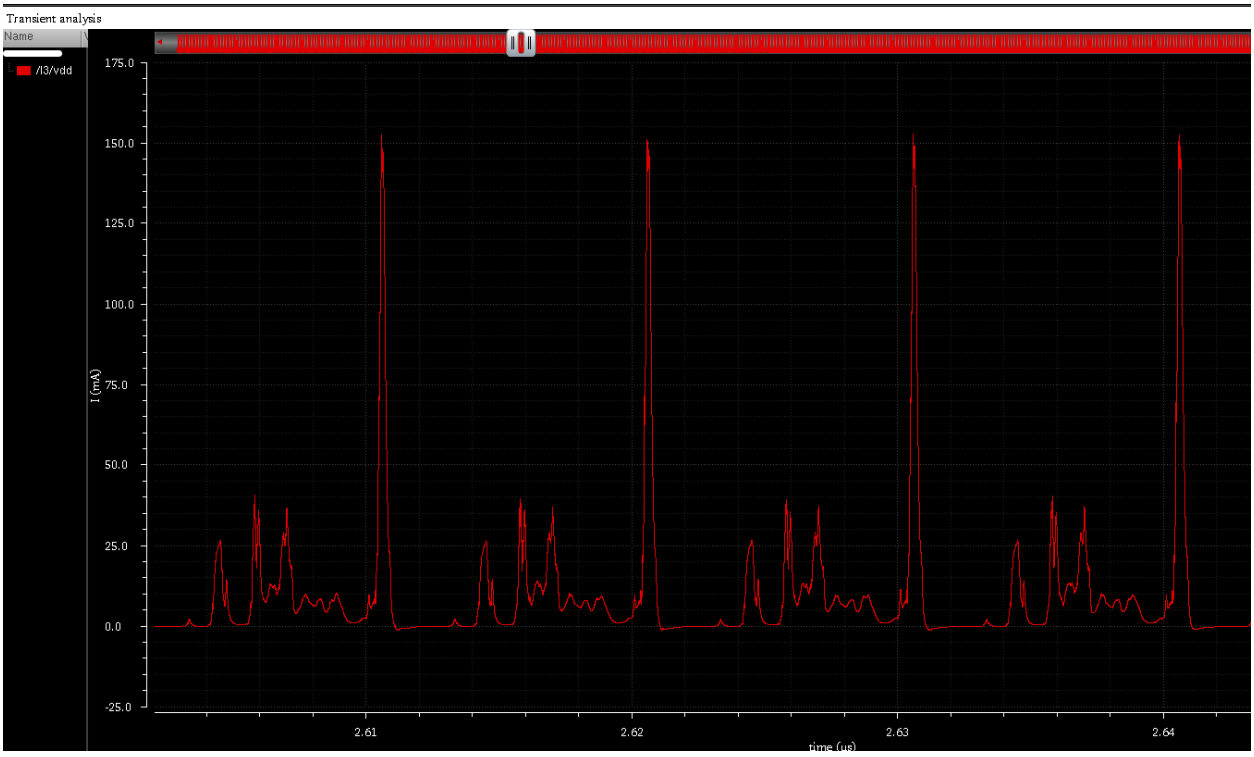

Figure 4: Power traces from different random plain texts

Clipping and shifting power traces

For the power traces to be added together, they need to be of the same length (same simulation time) and same offset (simulation starting and ending at the same time). In order to have these type of wave forms, the power trace including all 4000 individual traces was divided into 4000 10 ns traces, and each individual power trace was left shifted by the right offset (time) in order to have all traces starting at 0 and ending at 10 ns. Cadence Virtuoso Calculator was used for this purpose. The following is an example of equations used to perform the Clipping and Shifting of power traces.

*artistResMgrCalcExprSetup("false" "plot" "data1" "**lshift**(clip(i("/I3/vdd" ?result "tran" ?resultsDir "/home/jhabiman/simulation/AES_testbench_vector_1/UltraSim/schematic/psf") 0n 10n) **0n**)")*

- First, the clip function is used to clip from a power trace going from 0 to 4000 ns, an individual trace that goes from 0 to 10 ns. This new trace is the power trace corresponding to the first plain text.

- The Left Shift function **(lshift)** is then used to shift left the clipped trace so that all the resulting individual traces are aligned at 0. For this example, the offset is 0, because it computes the first power trace, but for the next trace the offset will be 10 ns, and so on.

Figure 5 shows, data1, data2, and data3 clipped and aligned.

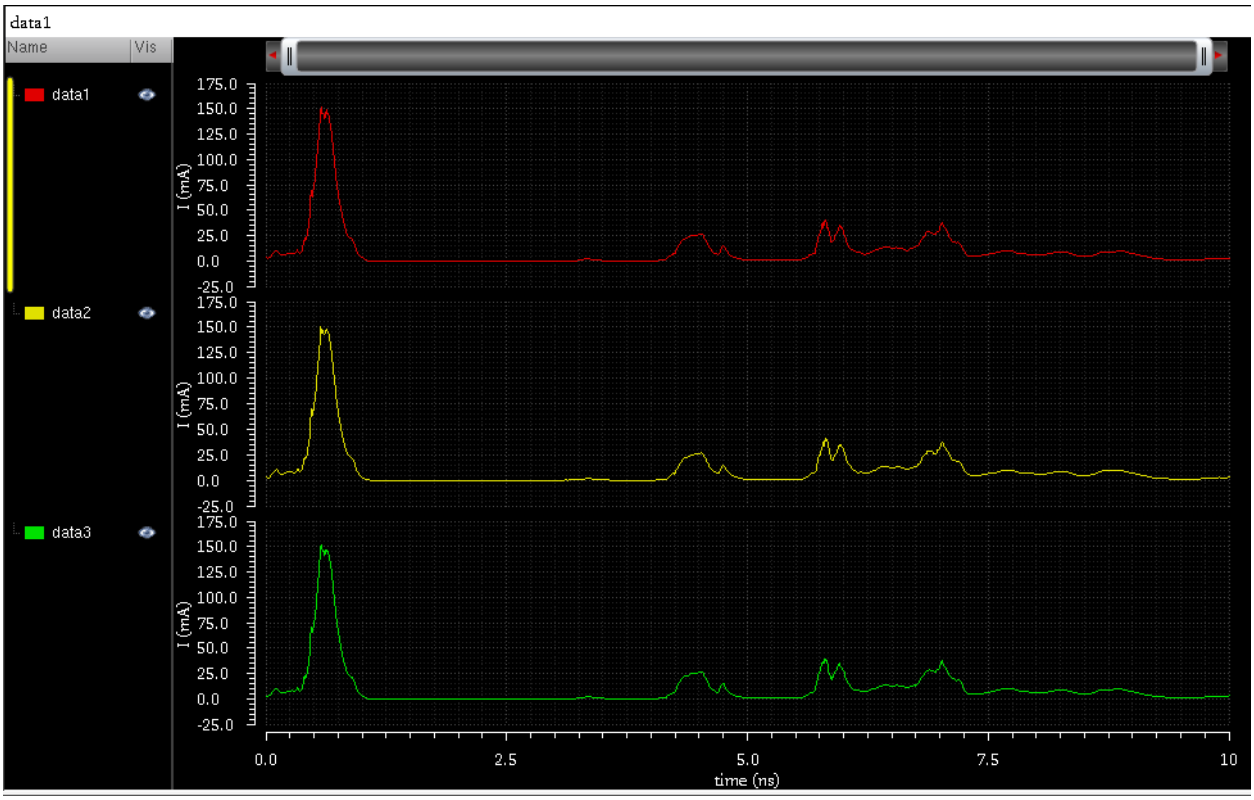

Figure 5: Data1, data2 and data3 deducted from the simulation results by clipping and shifting

Splitting power traces into Group 1 and Group 0 (Grouping)

The grouping is the part that establish a relationship between the guessed key and the Differential Power Trace. For this project a selection function based on **Hamming Distance** was used. This Selection Function is built in a software model that represents AddRoundKey and S-box functions of the AES algorithm. In addition to AddRoundKey and S-box functionalities, this software model computes the Hamming Distance corresponding to each plain text throughout AddRoundKey and S-box operations, and power traces corresponding to plain texts that result in higher Hamming Distance are put into Group 1, while power traces corresponding to plain text with lower Hamming Distances are put into Group 0.

Adding, Averaging and Subtracting power traces

Again, Cadence Virtuoso Calculator was used for this task. All power traces in each grouping are added together to generate a sum power trace for each Group. The graph in Figure 6 shows the sums for Group 1 and Group 0 power traces.

Notice from the graph in Figure 6 that,

- SumG1 and SumG0 graphs still presents the same patterns as the individual power traces.
- The scale on the Y axis has changed from mA to Amps.
- SumG1 and sumG0 are almost the same as the 2 graphs overlaps almost completely.

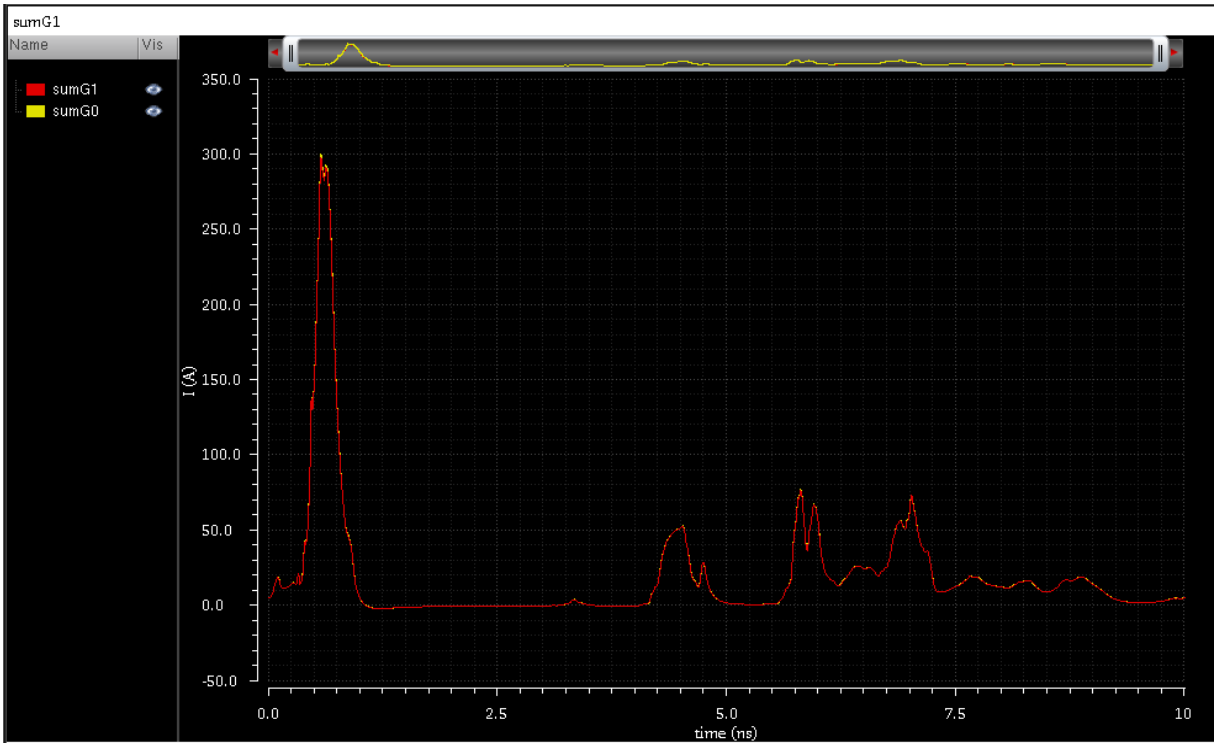

Figure 6: Graphs showing power trace sums for Group1 and Group 0

As shown in Figure 7, the average graphs also keep the original power traces patterns.

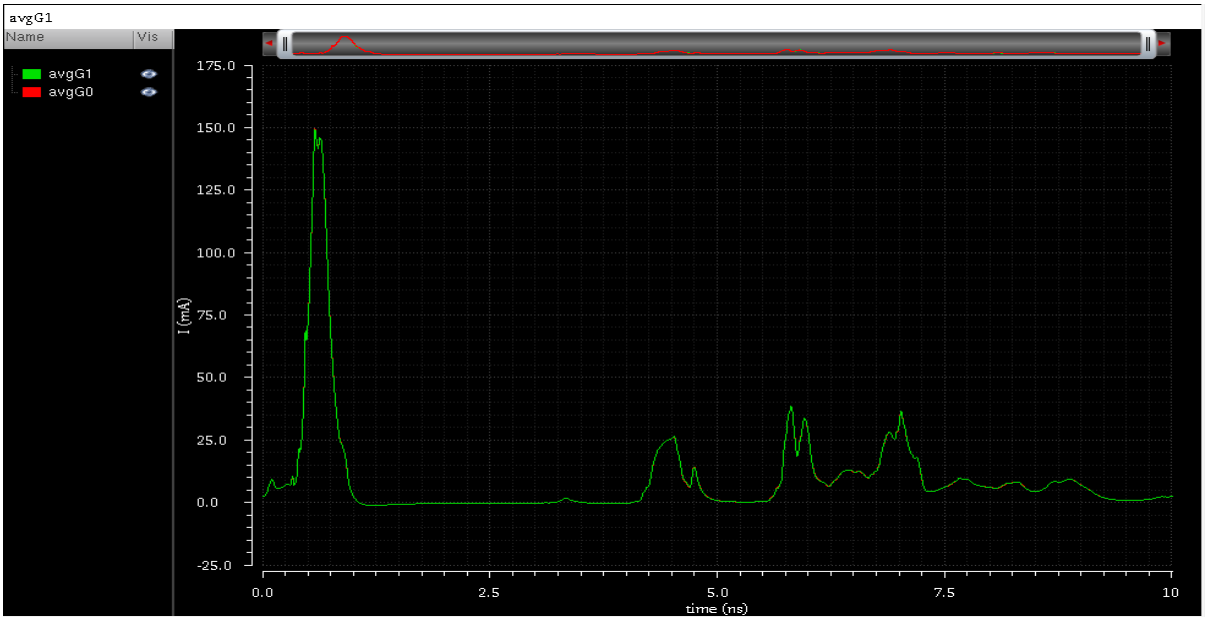

Figure 7: Group 1 and Group 0 average Power Traces

On the contrary, the Differential Power trace does not keep the power consumption patterns from the original traces. In fact, if it was not for noises and tool inefficiencies, the Differential Power trace should be close to a straight line except for the points where the circuit performs encryption key related operations where the trace should present a spike in case of a successful key guess. Figure 8 shows the Differential Power trace.

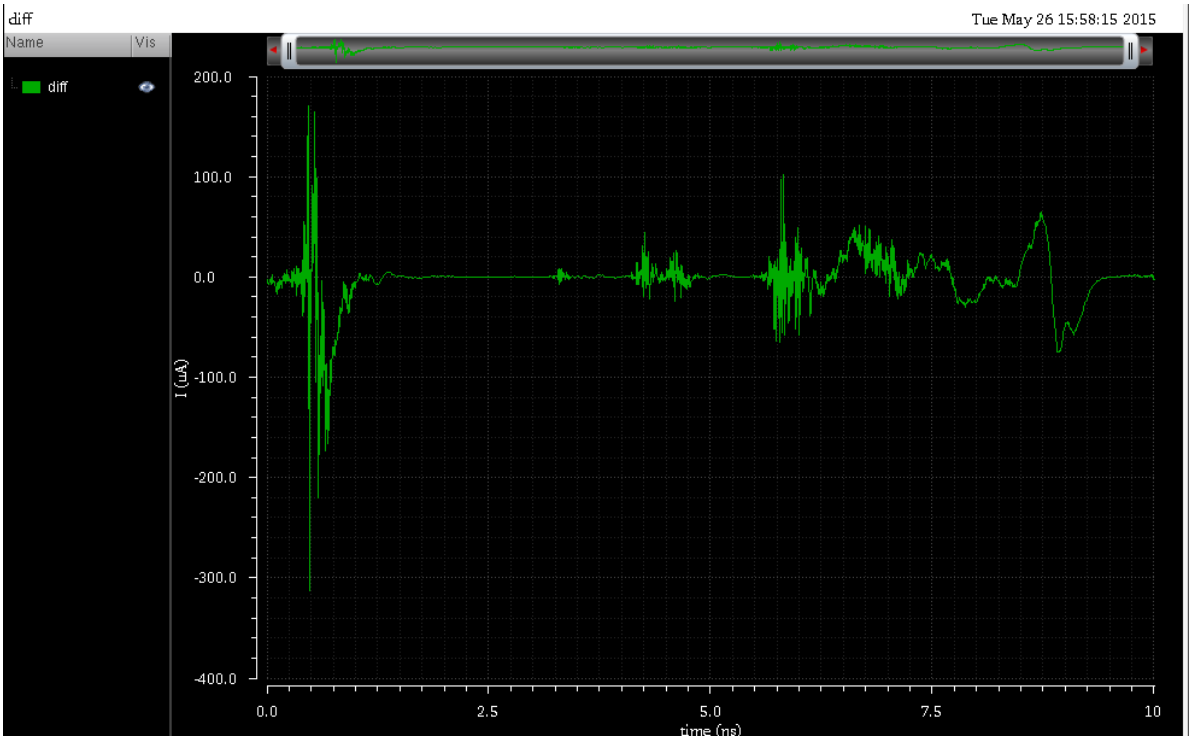

Figure 8: Differential Power trace resulting from a correct key guess

Result Analysis

Looking at the Differential Power trace in Figure 8, one may be confused as whether the trace presents a spike or not. The answer is: the trace does not present a spike.

Y axis Scale Observation

It is true that the trace in Figure 8 has some higher points than others; however, considering the current scale (micro Amps) on the Differential Power trace compared to the mA scale for the original traces, it is possible that the visible peaks on the Differential Power trace are a result of noises and lack of precision of the tools. In fact, the highest peak on the Differential Power trace in Figure 8, is found at around 1ns on the time axis. However, considering our simulation set up, at that time the circuit is not performing any encryption key related operation because the circuit is in reset mode from 0 ns to 3 ns. Therefore, it is clear that the peak at 1 ns cannot be attributed to any correlation to the encryption key. Moreover, if this peak at 1 ns is proved to be a result of noises, and it is the highest peak, it is implied that other peaks are also a result of noises.

Another observation is that there is a correlation between peaks on the Differential Power trace and the amount of power being used by the circuit at the time of the peak. Compare the

Differential Power trace in Figure 9 below (left) to the power trace average in Figure 10 (right). It is clear that noises (peaks) on the Differential Power trace correspond to points where the circuit was using higher power.

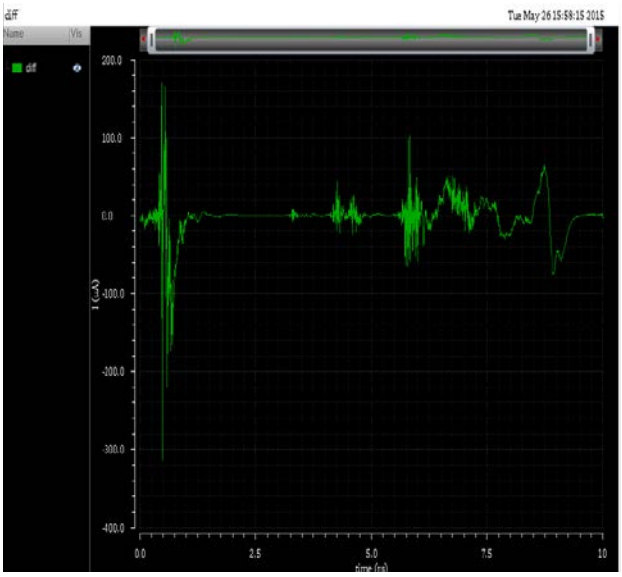

Figure 9: Differential Power Trace

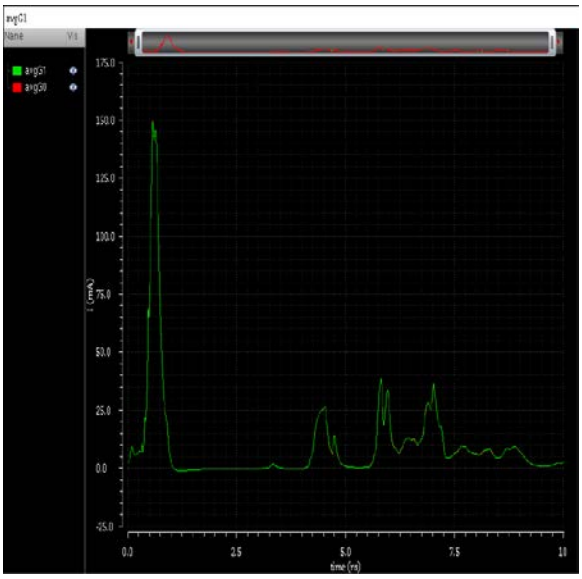

Figure 10: Average Power Traces by Group

Comparing a wrong key guess Differential Power trace to a successful key guess one

DPA guesses all possible key values and compares the resulting Differential Power traces to decide which key was a hit. Therefore, another way to prove that the Differential Power trace in Figure 8 does not represent any valid spike, is to compare it to a Differential Power trace resulting from a wrong key guess.

To generate a Differential Power trace from a wrong key guess, the select function software was run using a wrong key. Different power trace groupings were generated, added together and averaged. The resulting Differential Power trace is presented in Figure 11.

Notice that the Differential Power trace in Figure 11 (generated by a wrong key guess) presents almost the same peaks at the same times as the correct key Differential Power trace. This confirms that the peaks on these Differential Power traces are not a result of a correct key guess, but a result of noises that are accentuated at points where the circuit uses more power than it uses at others.

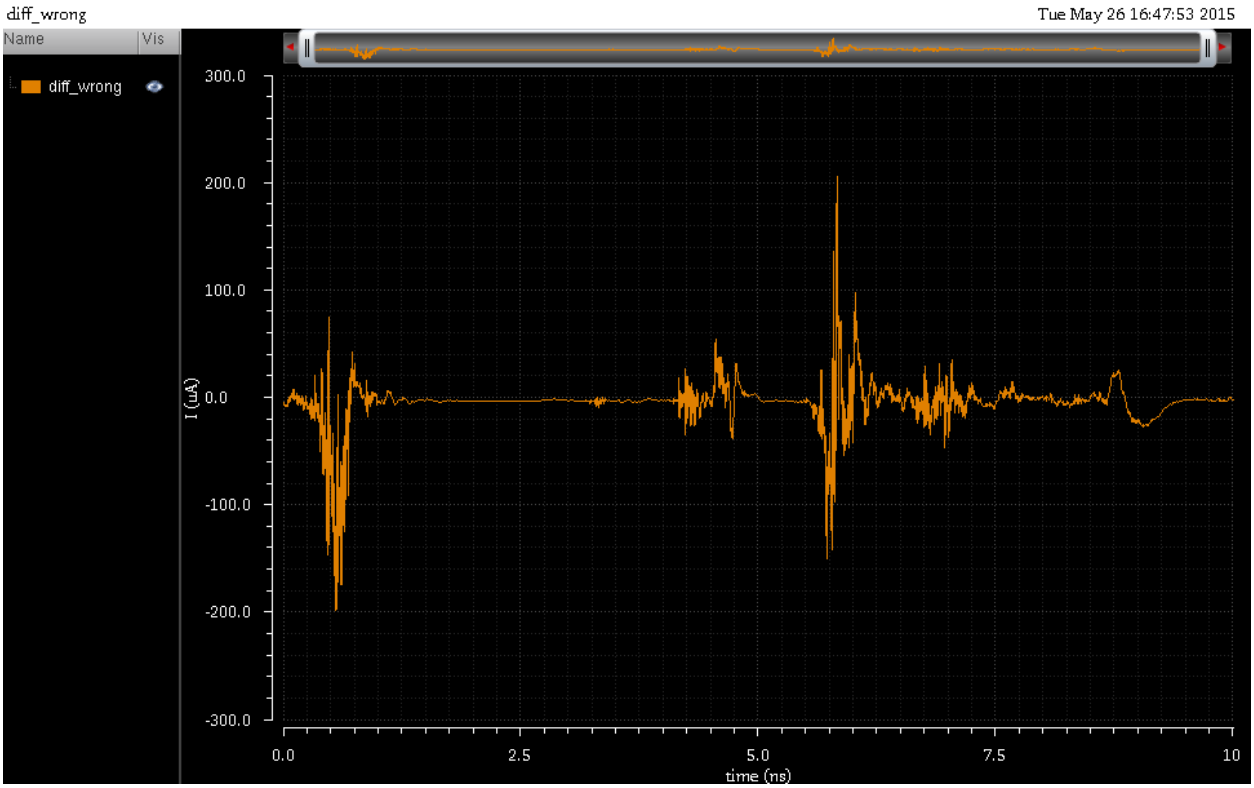

Figure 11: Differential Power Trace resulting from a wrong key guess

Conclusion

To show that the optimization of the MTD^{3}L architecture did not compromise the architecture's ability and effectiveness in mitigating Side Channel Attack, a DPA attack was performed on the new MTD^{3}L AES core. The attack was performed to the best of our knowledge and ability. Given that we knew what the key was, it was enough to perform the attack once using the right key, and check if the resulting Differential Power trace has any sort of spike, in which case the key would have been successfully guessed. After the attack was completed we decided that even though the Differential Power trace had some sort of peaks, they were not valid spikes. To support that argument, the Attack was repeated using a wrong key guess in order to compare a Differential Power trace resulting from a correct key guess to one resulting from a wrong key guess. The comparison proved that the 2 Differential Power traces were almost identical in terms of peaks they present and times peaks are presented at. Therefore, it was concluded that the peaks were a result of noises rather than a sign of correction. That way, it was proved that the new MTD^{3}L AES core is safe against DPA.