*Masters Project Synopsis*

# Scan Based Attack and Prevention Technique on stream cipher Hardware & Power Profile Modeling of Sequential Circuits

*Author:*

Mukesh Agrawal

Roll No: 03CS3008

*Supervisors:*

Prof. Dipanwita Roy Chowdhury

Prof. Indranil Sengupta

Department of Computer Science & Engineering

Indian Institute of Technology

Kharagpur

April 30, 2008

## 1 Introduction and Motivation

This work is divided into two parts.

Scan based design for test (DFT) is a famous and powerful testing mechanism, but it can be used to extract information from the cryptographic hardware which is undesirable. The weakness of crypto-chip does not lie in the algorithm it implements but in the scan design intended for efficient testability. A recent work has been done proposing a hardware solution to avert this scan based attack. This work presents an attacking scheme against the above work and suggests a modification in the testing hardware to avoid the proposed attack. Further, a scan based attack on Trivium: a hardware profile in 3rd phase of e-stream is shown and security of the modified testing hardware against this attack is demonstrated. [Section 2-5].

For each core of the SOC, there are different wrapper configurations and for each wrapper configuration is associated a testing time. So, each core can be represented as a set of rectangles with one side as the wrapper width and other side being the width dependent testing time. For shortening test time, concurrency in test scheduling is desirable. This concurrency in test scheduling leads to high power consumption. So, power dimension is added to this rectangle to get a 3D rectangular block. Thus, entire problem is modeled as a restricted 3D bin packing problem with constraints being the power and SOC pins. Power of SOC is modeled in different ways. This work suggest a power profile model aiming at minimizing the false power. [Section 6-7].

## 2 Attack on the Flipped Scan Chain Architecture

Firstly, a brief overview of this architecture is given below and then, an attack is proposed. The overview is directly taken from [7].

### 2.1 The Architecture

Details of the architecture can be obtained from [7].In [7], the inverters or NOT gates were introduced (Fig. 1) at the input of the scan in pin of the scan D-flip flop (SDFF) and these were termed as flipped scan DFF or FSDFF. In the architecture, scan chain consists both of SDFFs and FSDFFs. The presence of inverters aimed at preventing the scan data from being analyzed in order to determine the intermediate values stored in the flip flops.

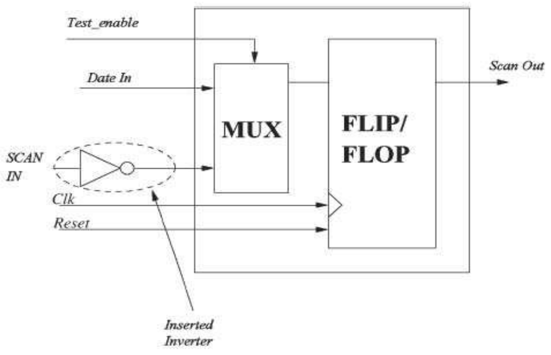

Figure 1: Flipped Scan DFF (FSDFF).

So the contention made in [7] is that, using this architecture, the attacker can only know the parity of the inverters by observing the difference in polarity of the scan in and scan out data and he cannot ascertain the exact positions of the inverters in the chain.

## 2.2 Attack

There is a RESET signal going to each SDFF which resets each of them to zero. So what an attacker can do is use this reset signal and obtain the scan out pattern by operating the crypto chip in test mode. The scan out pattern would look like pattern of zeroes interleaved with patterns of ones i.e. pattern of zeroes and ones would appear alternatively. The places where polarity is reversed are the locations where an inverter has been inserted. This will become more clear with the following example.

*Example.* Suppose there is a chain of 10 SDFFs and inverters are placed at position number 1, 3, 6, 7, and 11 out of 11 valid positions. We apply a reset signal and scan out the pattern. We will get a pattern $X = \{X_1, X_2, ...., X_{10}\}$ where $X_i = a_{i+1} \oplus a_{i+2} \oplus ... \oplus a_{11}$. We will get this pattern in the order $X_{10}, X_9, ..., X_1$.In this example,
$X_{10} = a_{11}$
$X_9 = a_{10} \oplus a_{11}$
$X_8 = a_9 \oplus a_{10} \oplus a_{11}$
.
.
$X_1 = a_1 \oplus a_2 \oplus \cdots \oplus a_{11}$
On closer look, one can always tell that the inverters are placed at positions 1,3,6 and 7.

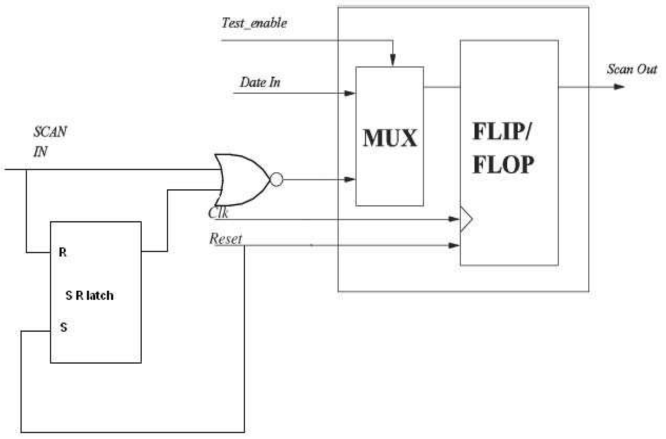

Figure 2: Replacement of inverter with a latched NOR gate.

## 3 Solution

In this section, we first propose a solution to overcome the above problem. This includes replacing each inverter with a digital circuit without compromising with the security that the inverter had provided. Refer to figure 2. The digital circuit consists of a NOR gate and a SR latch. The SR latch can be synthesized using two NOR gates. The Q output of the SR latch is fed into a 2 input NOR gate. The other input of this NOR gate is driven by the scan in line. This scan in line also goes to the R input of SR latch. The reset signal which resets every flip flop in the scan chain is wired to the S input of the SR latch.

The deactivation signal for a NOR gate is 1. This means that when one of the inputs of a multiple input NOR gate is set to 1, it continues to produce a 0 output irrespective of the other input values. So, when reset signal is applied, output of the SR latch becomes 1 and it deactivates the NOR gate driven by it. The output of the NOR gate becomes 0 and it remains in that state until output of SR latch does not change. For this to change, R input must get a 1 thereby activating the NOR gate. From here onwards, the NOR gate starts functioning like a NOT gate. It is to be noted that invalid state of SR latch is not reached i.e. both S and R inputs cannot become 1 simultaneously. Because, as soon as reset signal is applied, all flip-flops return to zero state. With this all scan in line which is scan out of previous flip flop in the scan chain becomes 0. Hence, R line becomes zero.

With this design, the attacker cannot do anything but guessing to determine the position of these *latched nor gates*. After applying the reset signal, the attacker remains unable to determine the exact position of these NOR circuits in the chain because he will get a sequence of all zeroes due to deactivation of all NOR gates.

Here, a problem arises when the Reset signal is applied and the user wants to use the device either in the test mode thereafter. There is a problem because all the NOR gates are deactivated and there is no mechanism to activate them so that the user could use it for analyzing the scan out data. In this scenario, we send in an *activation sequence* through the scan in line of the scan chain which goes on activating all the NOR gates in its path. The user should be careful about application of this sequence after every use of the reset signal. This is so because, if the reset is applied and the user wants to operate it under the test mode after operating it in normal mode, then, the data captured by the device after normal mode will be all destroyed even with the application of a single clock in test mode. So, the user should make it sure to apply an activation sequence after every application of reset signal or to be safe, after the device is switched on.

If the total number of *latched nor gate* being used is k and the total number of flip flops is n, then my *activation sequence* would look like:

$$\overbrace{1010....}^{k\ bits}\ \underbrace{00...}_{n-k\ bits}$$

The way this sequence works is very simple. The first bit which is 1 is undoubtedly going to reach the first NOR gate thereby making it active. Now, this 1 gets inverted and similarly, all the following bits get inverted on successive clocks. So the second bit of the sequence now becomes 1 which activates the next NOR gate in the chain. This bit and the following bits get inverted due to this effect. By then, third bit of the sequence becomes 1 which activates the next NOR gate. This continues until every NOR gate gets activated. This procedure takes a total of n clock cycles. In this way, the user can work with the hardware in normal mode or test mode. This *activation sequence* does not depend on the position of the NOR circuit placed in the scan chain but somewhat depends on their total number. Also, this sequence need not be unique. For example, the trailing n-k bits need not be all zeroes. It can be anything. And, the leading sequence of ones and zeroes need not be that too. Intuitively, one can see that the only necessary condition is that there should be change in polarity of bits at least k times. Hence, there is no way the attacker can ascertain the total number or the positions of NOR circuits.

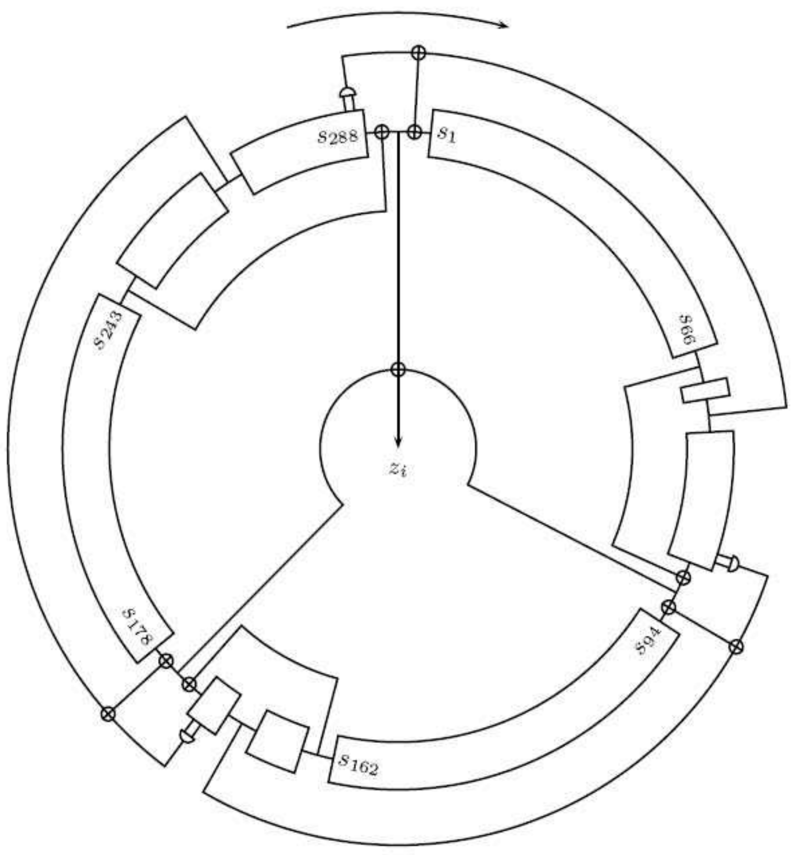

Figure 3: TRIVIUM

## 4 Scan based Attack on Trivium

Trivium is a synchronous stream cipher. It is one of the hardware candidates in phase 3 of *estream*[1]. Its details can be obtained from [3].

### 4.1 Attack

The objective of the attacker is to obtain the message from the stream of ciphertexts[5]. He observes the cryptograms $c_1, c_2, ..., c_l$. he then gets possession of the device and his intention is to obtain the plaintext $m_1, m_2, ..., m_l$ using scan based side channel attack. As illustrated in [5, 7], this attack works in two phases. The first phase aims at ascertaining the position of different registers in the scan chain. In the second phase, this information is

[1]`http://www.ecrypt.eu.org/stream/triviump3.html`

used to obtain the sequence of plaintext. In the context of trivium, the first phase involves determining the positions of s-bits namely $s_1, ...., s_{288}$.

***Phase 1. Ascertaining the location of s-bits*** First, when the attacker takes control of the chip, he scans out the patterns which the internal state of the trivium registers. But he does not the required positions which he decides in the following steps.

1. The attacker sets the K and IV input lines to be zero. Therefor value of $K_i$ and $IV_i$ is zero. One clock is then given in normal mode which sets $s_1$ to $s_{285}$ to 0 and $s_{286}$ to $s_{288}$ to 1. The pattern is scanned out in test mode to know the positions of $s_{286}$ to $s_{288}$ collectively. To know the exact position of $s_{286}$, the same procedure is repeated and one more clock is given in normal mode. This makes $s_{286}$ zero (due to right shifting operation in the trivium registers) and thus, when the pattern is scanned out, position of $s_{286}$ is known. Similarly, the respective positions of $s_{287}$ and $s_{288}$ are also known by giving one more extra clock in normal mode.

2. Set $K_1 = 1$ and all other input lines to be zero. One clock in normal mode lead us to the position of $s_1$. Similarly, $s_2$ to $s_{80}$ can be known by setting the corresponding $K_i$ line to be one and applying one clock. Also, $s_{94}$ to $s_{173}$ is located by setting IV lines to 1 individually one-by-one.

3. $s_{81}$ to $s_{93}$ can be located by setting $K_{80} = 1$ and all other input lines to be 0 and applying repeated clocks and scanning out in an iterative manner. Locating $s_{81}$ will require 2 clocks , $s_{82}$ will require 3 clocks and so on other than the clocks required for scanning out the entire pattern. Similarly, by setting $IV_{80}$ to be 1 and all other input lines to be 0 positions of $s_{174}$ to $s_{177}$ can be known.

4. Since we know the position of $s_{177}$ we can set this bit to 1 either by scanning in the required pattern in test mode or by setting $IV_{80} = 1$ and giving 5 clock cycles in normal mode. After doing this, one more clock is given in normal mode which sets $s_{178} = 1$. Its position is known by scanning out the pattern. Remaining s-bits namely $s_{179}\ to\ s_{285}$ can be located similarly in an iterative fashion.

This completes the first phase of the attack. It should be noted that the total number of clocks required for knowing the position of each bit is $O$(n) where n is the total number of flip-flops. Overall the time taken is $O(n^2)$. Now we show how can we decipher the cryptogram.

***Phase 2. Deciphering the cryptogram*** The attacker had scanned out the internal state of Trivium after getting hold of device. Now, he has ascertained the positions of s-bits in the scan chain. This information will be used to decipher the cryptogram and obtain the plain text. We proceed by knowing the previous state from the current state (Refer to table 3). As is clear from the encryption algorithm, current state is a right shift of previous state with first bit being the non-linear function of some other bits. So, our task remains to calculate 'a', 'b' and 'c'. Observe the following equation:

$s_{94} = s_{67} + s_{92}.s_{93} + a + s_{172}$.

This equation is deduced from the encryption algorithm and referring to table 3. From this we can deduce the value of 'a' as follows:

$a = s_{94} + s_{67} + s_{92}.s_{93} + s_{172}$.

Similarly, 'b' and 'c' can be deduced by following equations:

$b = s_{178} + s_{163} + s_{176}.s_{177} + s_{265}$.

$c = s_1 + s_{244} + s_{287}.s_{288} + s_{70}$.

In this way, we can compute the previous state from a given current state. That means, we can compute all previous states. Once obtained a state, it is loaded on to hardware by scanning the required pattern. Applying one clock in normal mode then gives us the key stream bit which when xored with ciphertext bit of that state produces corresponding plaintext bit.

### 4.2 Security induced by the proposed Architecture

Using the design of our proposed architecture using NOR circuits(*latched NOR gates*), the attacker cannot control or observe the values of the internal state registers of the trivium hardware through the scan inputs and outputs due to unknown position of the NOR circuits. Hence, the steps of the above attack are not successful in breaking this hardware stream cipher. Trivium uses 288 flip flops, so if we place $288/2 = 144$ NOR circuits at certain locations in the scan chain, as per [2], the probability to guess the correct structure is approximately $\frac{1}{2^{288}}$, which is too less. Thus, we can avoid the scan based attack. We may place different number of NOR circuits, but placing half the number of total flip flops provides the maximum security. This security analysis is based on probability theory. Interested readers are suggested to read [7] for more details.

## 5 Performance Comparison

In this section, we will evaluate the area overhead that would be incurred while implementing the design given in this paper and compare it with the

| ISCAS'89 circuit | Total flip-flops | Logic gates in the ckt. | Total gate count | NOR circuits used | Gate overhead | Overhead as in [5] |
|---|---|---|---|---|---|---|
| s298 | 14 | 119 | 287 | 5 | 5.23% | 21% |
| s344 | 15 | 160 | 340 | 5 | 4.42% | 18% |
| s382 | 21 | 158 | 410 | 7 | 5.13% | 19% |
| s400 | 21 | 162 | 414 | 7 | 5.08% | 19.4% |
| s5378 | 179 | 2779 | 4927 | 60 | 3.66% | 17% |
| s9234 | 228 | 5597 | 8333 | 76 | 2.74% | 17.7% |
| s13207 | 669 | 7951 | 15979 | 223 | 4.19% | 16.4% |
| s15850 | 597 | 9772 | 1936 | 199 | 3.53% | 17% |
| s35932 | 1728 | 16065 | 36801 | 576 | 4.7% | 15.8% |
| s38417 | 1636 | 22179 | 41811 | 546 | 3.92% | 16.4% |

Table 1: Hardware Overhead

| AES Architecture | scheme in [8] | | | scheme in [7] | | | our scheme | | |
|---|---|---|---|---|---|---|---|---|---|
| | Gates | Overhead | | Gates | Overhead | | Gates | Overhead | |
| | | Gate | % | | Gate | % | | Gate | % |
| with KS | 273,183 | 412 | 0.15 | 184,209 | 80+159 | 0.12 | 184,209 | 240 | 0.13 |
| without KS | 282,120 | 4620 | 1.64 | 57,017 | 80+159 | 0.41 | 57,017 | 240 | 0.42 |

Table 2: Overhead analysis in AES implementation

results obtained in [5]. In [5], the authors have considered ISCAS'89 benchmark circuits [2]. They generated scan tree after finding compatible scan cells from the test set, as in [1]. The output from this scan tree was then fed into aliasing free compactor to match the output with the expected output. This compactor is supposed to be designed by the design engineer given he knows the test patterns and test responses. Area overhead for this design was computed using Synopsys Tetramax and Design Compiler tool. We evaluated the overhead over the same set of circuits because these are already tested benchmark circuits. Refer to table 1. Next, the results for these two cases are compared. Since, area overhead and gate overhead are comparable figures, we have computed gate overhead in this work. Using the fact that 12 NAND gates are needed to synthesize a D-flip-flop, we have computed total number of gates required for the synthesis of ISCAS'89 circuits. Also, total number of NOR gates used in our design is 3 for each of the NOR circuits. And, we have taken the count of total NOR circuits to be one-third of total number of flip flops as these many are sufficient from security perspective. As can be seen, we have a fair improvement from the results in [5].

| **Present State** | **Previous State** |
|---|---|
| $(s_1, s_2, ..., s_{93})$ | $(s_2, s_3, ..., s_{93}, a)$ |
| $(s_{94}, s_{95}, ..., s_{177})$ | $(s_{95}, s_{96}, ..., s_{177}, b)$ |
| $(s_{178}, s_{179}, ..., s_{288})$ | $(s_{179}, s_{180}, ..., s_{288}, c)$ |

Table 3: Internal states of Trivium

In [6], end to end design of AES is done using 0.18-$\mu$m CMOS technology. The work in [7] is based on this design. On the other hand, the work in [8] is based on the AES hardware implementation given in [4]. We have considered the same implementation as in [7] and got our results which is shown in table 2. Overhead analysis is done under two conditions: with KS (key scheduling) (number of flip-flops 6336) and without key scheduling (number of flip-flops 4048). 80 inverters were used in [7]- 10 in each of the 8 scan chains each containing equal number of flip-flops. What we have done is replaced these inverters with our designed NOR circuits - one for one - and computed the overhead.

## 6 Power Model

### 6.1 Global Peak Power Approximation Model(GP-PAM)

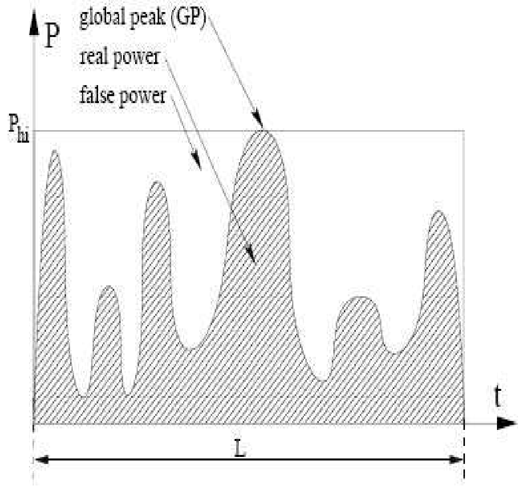

Figure : Global peak power approximation model

This approximate power model flattens the power profile of a block to the worst case instantaneous power dissipation value, i.e., its peak value. Evi-

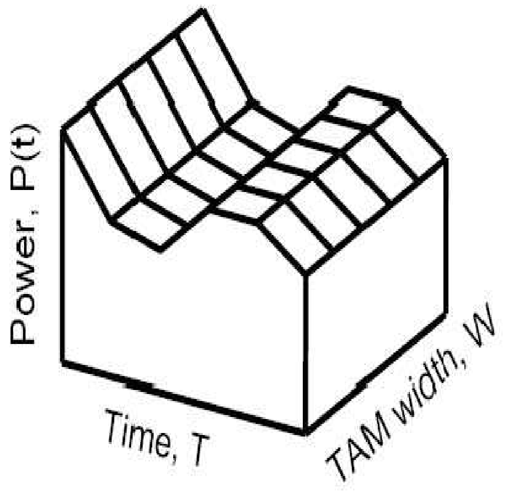

Figure 4: A core with Cycle Accurate Power Model

dently, the simplicity and reliability requirements for a good power approximation model are satisfied by the GP-PAM. However, the low complexity of the GP-PAM is achieved at the expense of a relatively high approximation error, as indicated by the large false power region in the figure above.

## 6.2 Cycle Accurate Power Model

In this model, power is computed on a per clock basis and no approximation is done. So our cube takes the form of The disadvantage of this power model is its huge complexity for storing the power generated in each cycle.

## 6.3 Proposed Power Model

In this approach, we are following the procedure outlined below:
**STEP 1**: In this power model, we are splitting the power profile based on the number of test patterns. For example, say dividing the power profile into blocks of 50 or 100 test patterns each.
**STEP 2:** With in each block so formed, we try to arrange the test vectors in such a way that the initial sequence of test vectors leads to comparatively low power consumption followed by a sequence of vectors leading to relatively high power consumption.
**STEP 3:** We then find out $T_{OPT}$ such that P1*L1 + P2*L2 is minimum. Also, the part L2 usually remains small. And, we have the relation P1*L1 +

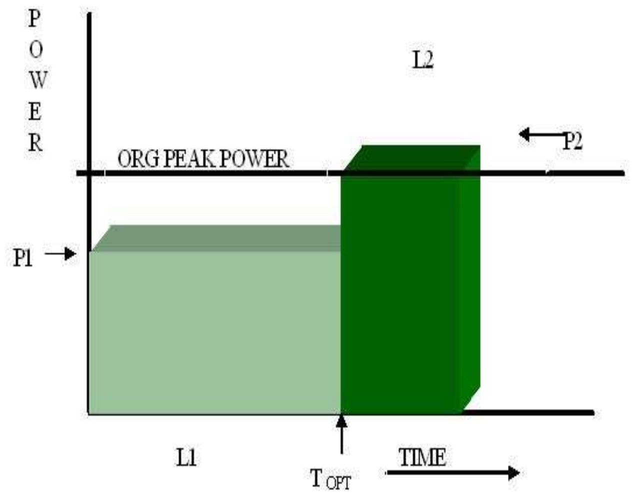

Figure 5: Modified Model

P2*L2 < ORG. PEAK POWER * (L1 + L2). The term on the right hand side is that obtained by Global Peak Approximation Model.

## 7 Implementation

I have used ISCAS89 circuits in place of cores because ITC'02 benchmark SOC circuits does not provide us with the knowledge of test vectors or the internal circuits design. And, for generating the power profile we need test vectors. So, tetramax tool provided by Synopsis has been used to generate the test vectors.

### 7.1 Generating the power profile

For generating the power profile, weighted transition count has been used. Suppose, we want to compute the power dissipated between the shifting out the test response $V_j$ and shifting in the input vector $V_i$. The WTC values corresponding to Vi are:

$$WTC_{scanin}(V_i) = \sum_{j=1}^{m-1} (V_i(j) \oplus V_i(j+1))(m-j))$$

Table 4: Comparison with GP-PAM.

| Circuit | Proposed | GP-PAM | Difference | improvement(%) |
|---|---|---|---|---|
| s344 | 3204 | 3861 | 657 | 17.02 |
| s420 | 9361 | 13932 | 4571 | 32.81 |
| s444 | 11140 | 13676 | 2537 | 18.55 |
| s1423 | 198686 | 225638 | 26952 | 11.95 |
| s5378 | 4360449 | 4700752 | 340303 | 7.24 |
| s9234 | 9369731 | 9995040 | 625309 | 6.26 |
| s13207 | 108851211 | 112105128 | 3253917 | 2.91 |
| s15850 | 65095828 | 70414911 | 5319083 | 7.56 |
| s35932 | 101927819 | 105108126 | 3180307 | 3.03 |
| s38584 | 758351027 | 806073609 | 47722582 | 5.93 |

$$WTC_{scanout}(V_i) = \sum_{j=1}^{m-1} (V_i(j) \oplus V_i(j+1))j)$$

$$Weight(V_i, V_j) = WTC_{scanout}(V(i)) + WTC_{scanin}(V(j))$$

where j is the bit position.

## 7.2 Reordering the test vectors

The test vectors are reordered using a Greedy approach. A graph is formed with vertices as test patterns and edges between them as Weight(Vi,Vj). As a result of which a bidirectional clique is formed. Now, the problem at hand is reduced to finding a Hamiltonian tour of low cost. Any vertex is selected as the starting node and then next best node is sought.Table 4 shows the Approximated error improvement of the proposed model over GP-PAM model for ISCAS89 circuits.The block size for this table is kept at 30 test patterns each block.

The following table shows the effect of increasing the block size on s15850.

| Block Size | Proposed | Difference | Improvement(%) |
|---|---|---|---|
| 30 | 65095828 | 5319083 | 7.56 |
| 40 | 65353153 | 5061758 | 7.12 |
| 50 | 65349776 | 5065135 | 7.20 |
| 60 | 65602782 | 4812129 | 6.84 |
| 70 | 65606352 | 4808559 | 6.83 |
| 80 | 65866707 | 4548204 | 6.46 |

## 8 Conclusion

In the first part of this work, with the new design, security is not compromised and proposed attack is prevented. Hardware overhead is also found better than the existing work. Moreover, this design can be easily integrated into the scan based design flow.

## References

[1] Y. Bonhomme, T. Yoneda, H. Fujiwara, and P. Girard. An efficient scan tree design for test time reduction. In *Proc. 9th IEEE ETS*, pages 6–11, November.

[2] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *IEEE Int. Symp. on Circuits and Sys.*, pages 1929–1934, May 1989.

[3] C. D. Caniere and B. Preneel. Trivium specifications. eSTREAM submitted papers.

[4] S. Mangard, M. Aigner, and S. Dominikus. A highly regular and scalable aes hardware architecture. *IEEE Transactions of Computers*, 52(1):483–491, April 2004.

[5] D. Mukhopadhyay, S. banerjee, D. RoyChowdhury, and B. Bhattacharya. Cryptoscan: Secured scan chain architecture. In *Proc. 14th IEEE ATS*, pages 348–353, 2005.

[6] D. Mukhopadhyay and D. Roychowdhury. An efficient end to end design of rijndael cryptosystem in 0.18 $\mu$ cmos. In *Proc. 18th Int. Conf. VLSID*, pages 405–410, January 2005.

[7] G. Sengar, D. Mukhopadhyaya, and D. RoyChowdhury. Secured flipped scan chain model for crypto-architecture. *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, 26(11):2080–2084, November 2007.

[8] B. Yang, K. Wu, and R. Karri. Secure scan: A design for test architecture for crypto chips. *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, 25(10):2287–2293, October 2006.