IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555
Vol. 3, No.2, April 2013

A Low Power Multi-Bit Flip-Flop Design for Counter Measure Circuit in Cryptographic Applications

Ashna v r
ME VLSI Design
Sri Ramakrishna Engineering College
Coimbatore, India

Dr.M Jagadeeswari
ME VLSI Design
Sri Ramakrishna Engineering College
Coimbatore, India

Abstract–**The clock power is the major dynamic power source VLSI circuits. The multi bit flip-flop technique is one of the techniques used to reduce the clock power. The power reduction is achieved through the merging of flip-flops based on certain timing constraints. A combination table which can store the flip-flops that can be merged is introduced in the proposed work. The Differential Power Analysis (DPA) is a big threat to crypto chips since it can efficiently disclose the secret key. Using the self generated true random sequence based counter measure circuit the differential power attack can be reduced. The multi-bit flip-flop technique is introduced in the counter measure circuit to reduce the power as well as area. According to the experimental results it is found that the flip-flops after merging reduces the dynamic power about 27.27% and the total power about 12.59%. It is also found that the total gate count is reduced from 7709 to 7389.**

Keywords-** **merging,combination table; multi-bit flip-flop; differential power analysis;LFSR;counter measure circuit

I. INTRODUCTION

To optimize the power consumption, many low-power design techniques have been introduced, such as clock gating [3][4], power gating[5] creating multi-supply-voltage designs, dynamic voltage/frequency scaling, and minimizing clock network. Among these techniques, minimizing clock network is very important in reducing power consumption of an SoC. Recent studies have proposed various approaches to minimize clock network, including buffer sizing[6], placement optimization of registers[7] and applying multi-bit flip flops (MBFFs)[9].

Multi-bit flip-flop is an effective power-saving implementation methodology by merging single-bit flip-flops in the design. The multi-bit flip-flops can reduce clock dynamic power and the total flip-flop area effectively.

Usually the clocks are made up of inverter-based clock buffers. In the case of multi-bit flip-flop technique the clock power is reduced by replacing the available flip-flops with the multi-bit flip-flops. To achieve this no of clock buffers have to reduce. To reduce the number of clock buffer the concept of sharing of the clock buffers by several flip-flops can be used. Fig.1 [2] shows the two 1-bit flip-flops, if it is replaced by a 2- bit flip-flop. In this circuit the individual clock is replaced by a single clock buffer. So the two 1-bit flip-flops can share the same clock.

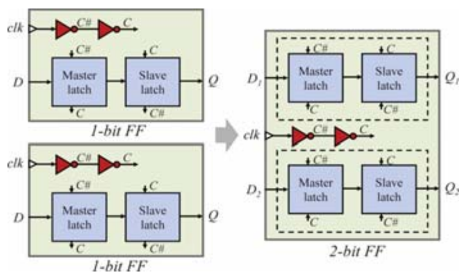


Figure 1 Merging of 2 1-bit flip-flop

The concept of merged flip-flop technique is introduced in the counter measure circuit which is used to avoid the DPA attack in cryptographic circuits. This DPA attack is a serious problem in the cryptographic circuit since it can disclose the secret key efficiently. There are several methods present to resist the DPA attack. One of the methods is data masking method [13]. In this method the data present in the cryptographic circuit is randomized. The data is changed by the internally generated random mask before encryption or decryption. So in order to recover the actual data the same mask should be used. Usually the power consumption of cryptographic circuits will be independent of the predicted power consumption. The power consumption is balanced by sense amplify based logic or Wave Dynamically Differential Logic (WDDL)[14]. In some method the power supply and the cryptographic circuits are isolated using the switching capacitors. But these methods have the disadvantage of increasing hardware cost. A ring-oscillator-based DPA counter measure circuit can overcome these disadvantages. The random bytes from the pseudo random number generator can have the same value after the system is reset.

This problem can be avoided by using true random number generator. The proposed true random
number generator can avoid the DPA attack and it can also self generate the true random number.

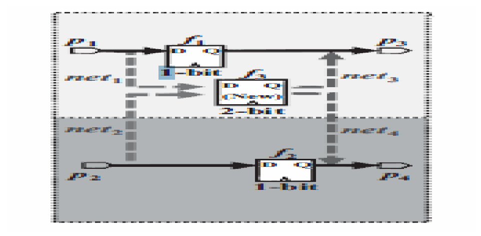


Figure 2 Problems faced during merging of flip-flop

II.RELATED WORK

Y.-T. Chang et al [2] used the multi-bit flip-flop technique in the post-placement stage to reduce power consumption. In this work a graph based approach is used to reduce the clock power. Each node in the graph represents a flip-flop. Two flipflops which satisfy the timing and capacity constraint can be replaced by an edge which is built between the corresponding nodes. The replacement of the flip-flop is done using m-clique in the graph. The flip-flops corresponding to the nodes in an m-clique can be replaced by a m-bit flip flop. The branch-and-bound and backtracking algorithm is made use to find the m-cliques in a graph. In this work there is a possibility that one flip-flop may belong to more number of m-bit flip-flop. A greedy heuristic algorithm is used to find the maximum independent set of cliques, in that each node belongs to only one clique. In this method there is a chance of getting a multi-bit flip-flop which has a bit width outside the required library. So finding out this undesired bit width may lead to the wastage of time.

III. PROPOSED METHOD

In the earlier method the time is wasted by finding the impossible combination. To avoid that find out the flip-flops that can be replaceable by using the multi-bit flip-flop. But as the number of flip-flop increase the complexity will increase. In order to avoid that certain approaches are used. In the proposed method first prepare a combination table which can store the flip-flops that can be merged. If the library considers three kinds of flip-flops which are 1-bit, 2-bit and 3-bit, then there is no need to consider the combination of 1-bit and 3-bit, because the library does not provide a 4-bit flip-flop. A cell library with lots of flip-flops is considered, merge as many flip-flops as possible to reduce the power consumption. The bit width of the replaced flip-flops must be equal to the sum of bit width of individual flip-flops. The replacement may cause some changes in the routing length of the nets and it will change the timing of some paths. To ensure the legalized replacement certain constraints must be satisfied.

A. Building the Combination Table

Algorithm1- Build Combination Table

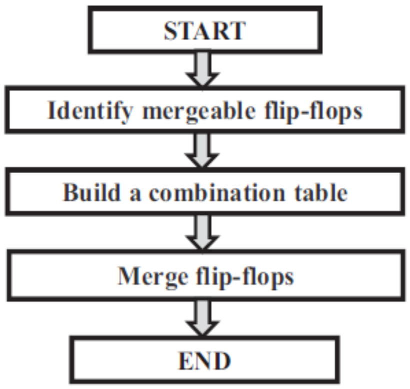


Figure 3. Flow chart of the proposed algorithm

```
1. T = InitializationCombinationTable(L);
2. InsertPseudoType(L);
3. SortByBitNumber (L);
4. for each ni in T do
5. InsertChildrens (ni, NULL, NULL);
6. index = 0;
7. whileindex != size(T) do
8. range_first = index;
9. range_second = size(T);
10. index = size(T);
11. for each $n_i$ in T
12. for j = 1 to range_first do TypeVerify($n_i$, $n_j$, T);
[Figure.3]

13. for j = i to range_second do TypeVerify($n_i$, $n_j$, T);
14 .T = DuplicateCombinationDelete(T);
15. T = UnusedCombinationDelete(T);
```

```
InsertPseudoType(L):
1. for i = ($b_{min}$+1) to ($b_{max}$-1)
2. if (L does not contain a type whose bit width is equal to i )
3. insert a pseudo type typej with bit width i to L;
                    [Figure 4]
InsertChildrens(n, n1, n2):
1. n.left_child ← n1;
2. n.right_child ← n2;

TypeVerify(n1, n2, T):
1. bsum = b(n1) + b(n2);
2. if (L contains a type whose bit width is bsum)
3. insert a new combination n whose bit width bsum to T;
4. InsertChildrens( n , n1, n2);
```

Fig.3[1] shows the design flow of the proposed algorithm.To transform the coordinate system equations (1) and(2) are used. The location of the point in original system is denoted by (x, y), and the new coordinate is denoted by (x’, y’). Let x’’ and y’’ denote the transformed locations.

$$x' = \frac{x+y}{\sqrt{2}} \Rightarrow x'' = \sqrt{2}.x' = x+y \qquad (1)$$

$$y' = (-x+y)/\sqrt{2} \Rightarrow y'' = \sqrt{2}.y' = -x+y \qquad (2)$$

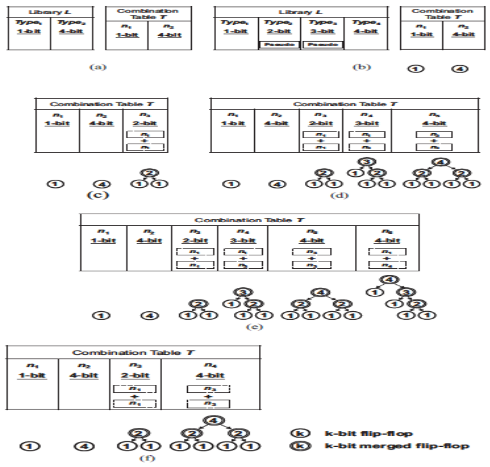


Figure 4. Example of building the combination table. (a)The library and combination table (b) Insertion of pseudo type (c)combine two n1s and form n2. (d)From n1 and n3 n5 is obtained. (e)From n1 and n4 n6 is obtained(f) Final combination table.

The flip-flops can be replaced by new flip-flops if and only if the new flip-flops are present in the library. The combination table keeps the record of all the flip-flops that can be replaced by new flip-flops. The flip-flops are replaced gradually according to the order of the combinations of flip-flops on the table. While doing the replacement only one combination of flip-flops must be considered each time.

To prepare the combination table the concept of pseudo type is used. It is shown in Algorithm 1. A binary tree is used to represent one combination of simplicity. Each node in the tree denotes one type of flip-flop in the given library. The types of flip-flops denoted by the leaves constitute the type of flip-flop in the root.

In the case of each node the bit width of the flip-flop equals to the sum of bit width of left and right child. The combination is denoted by n_i and $b(n_i)$ denotes its bit width. In the beginning of the algorithm the combination n_i is initialized for all flip-flops in the given library. To represent each code the concept of pseudo type is included. The pseudo type includes those flip-flop combinations that are not provided by the library.

The function **InsertPseudoType** is used to create pseudo type as shown in Algorithm 1. Let b_{max} and b_{min} the maximum and minimum bit width of flip-flops in library. The function **InsertPseudoType** is used to insert the pseudo types that can have bit widths more than b_{min} and lesser than b_{max}. The pseudo types are the one which is not provided by the library. After that all the combinations are sorted out in the ascending order. Finally try to combine each combination to get a new one. To check the feasibility of the combination the function **TypeVerify** is used. If the combination is feasible it is added to the table.

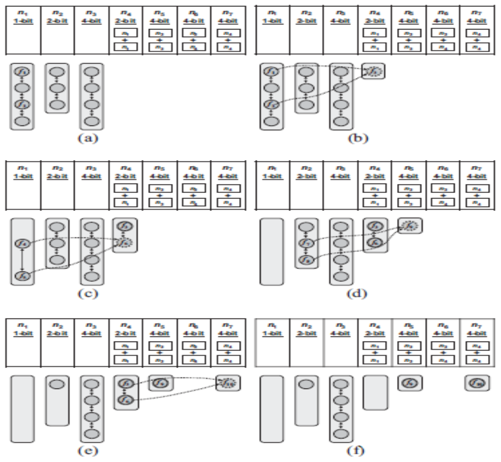


Figure 5. Example for replacemnet of flip-flops. (a) flip-flops before merging(b)f1 and f2 replaced by f3 (c) f4 and f5 replaced by f6 (d) f7 and f8 replaced by f9(e)f3 and f6 replaced f10(f) flip-flops after merging.

In **TypeVerify** the bits of the combinations are added and stored in the b_{sum}. This b_{sum} is further added to the remaining combinations. After completing all these procedures there may be some unused or duplicate combinations. This unused and duplicate combination is deleted. In order to delete that two functions **DuplicatecombinationDelete** and **UnusedCombinationDelete** are used.

Suppose the library only provides the two types of flip-flops with bit widths 1 and 4 as in Fig.4a, and then initialize the two combinations n1 and n2 to represent the two flip-flops. The function **InsertPseudoType** is used next to check the flip-flop type with bit width between 1 and 4 exist or not. After that the flip-flop with bit widths 2 and 3 are added to the table. For each combination in the table a binary tree with 0-level is built. The root of the binary tree denotes the combination. From the Fig.4[1] by combining the two 1-bit flip-flop a new combination n3 is obtained. The n4 is obtained by combining n1 and n3, and n5 by combining two n3s. Finally n6 is obtained by combining n1 and n4. All the possible combinations are shown in Fig. 4(e). In these combinations n5 and n6 are duplicate, because both representing the same condition. So n6 can be deleted to speed up the program. In this n4 is an unused combination so it can be eliminated.

In the case of combination table all the combination of the flip-flops must be entered. i.e. both the flip-flops which are present in the table as well as the combination which are not present. But inserting the not present flip-flops consumes more time so only some types of flip-flops are inserted. To insert a flip-flop combination n_i whose type is typej, only those flip-flops whose bit widths are (b(type j)/2) and (b(type j)– b(type j)/2) should exist. Algorithm 2 shows the enhanced procedure to insert flip-flops of pseudo types. In the case of *typej* the function **PseudoTypeVerifyInsertion** checks the flip-flops whose bit widths are [b(type j)/2] and add to the library if it is not exist in it.

B. Merge Flip-Flops

Algorithm 2- Insert Pseudo Types

```
InsertPseudoType(L):
1. for eachtypej in L do
2. PseudoTypeVerifyInsertion( typej, L) ;
PseudoTypeVerifyInsertion( typej, L):
1. if (mod (b(typej) /2) == 0)
2. b1 = [b(typej)/2], b2 = [b(typej)/2];
[Figure.5]

3. else
4.b1 =[b(typej)/2], b2 = b(typej) –[b(typej)/2];
5.for i = 1 to 2
6. if ((bi > bmin) &&
(L does not contain a type whose bit width is equal to bi))
7.insert a pseudo type typej with bit width bi to L;
8.PseudoTypeVerifyInsertion(typej, L);
```

Algorithm3- Merge all flip-flops in sub region r

```
1. for each combination n in a combination table
2. $l_{right}$ <- the list for the combination of the right-child n;
[Figure.5]

3. $l_{left}$<- the list for the combination of the left-child of n;
4. for each flip-flop fi in the list $l_{left}$
5. $c_{best}$<-œ
6. for each flip-flop fi in the list $l_{right}$
7. if(fi and fj can be merged)then
8. Compute cost c;
9. ifc< $c_{best}$ then $c_{best}$= c, $f_{best}$=fl
10. end
11. end
12. Add flip-flop f' merged from fl and $f_{best}$ to combination n
13. Remove fl from $l_{left}$ ;
14. Remove the flip-flop recorded by $f_{best}$ from $l_{right}$
15. end
```

Algorithm 3 shows the procedure to get a new flip-flop corresponding to the combination n. The binary tree helps to find out the combinations associated with the left and right child of the root. Hence, the flip-flops in the lists, named l_{left} and l_{right}, linked below the combinations of its left child and its right child are checked. Then, for each flip-flop in l_{left}, the best flip-flop f_{best} in l_{right}, which is the flip-flop that can be merged with the smallest cost recorded in C_{best}, is picked. Finally, add a new flip-flop in the list of the combination and remove the picked flip-flops.

In Fig. 5(a), f1 and f2 are chosen because their combination gains the smallest cost. Thus, add a new node f3 in the list below n4, and then delete f1 and f2 from their original list. Similarly, f4 and f5 are combined to obtain a new flip-flop f6, and the result is shown in Fig.5(c). After all flip-flops in the combinations of 1-level trees (n4 and n5) are obtained as shown in Fig.5(d), start to form the flip-flops in the combinations of 2-level trees (n6, and n7). In Fig. 5(e), there exist some flip-flops in the lists below n2 and n4,

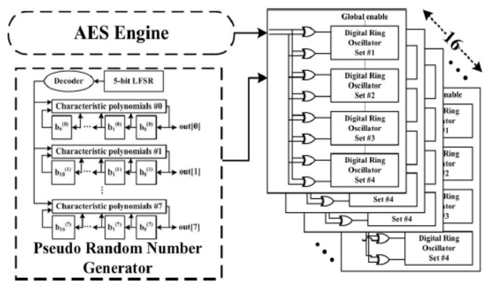

AES Engine
Global enable
Digital Ring Oscillator Set #1
Digital Ring Oscillator Set #2
Digital Ring Oscillator Set #3
Digital Ring Oscillator Set #4
Set #4
Set #4
16
Decoder
5-bit LFSR
Characteristic polynomials #0
out[0]
Characteristic polynomials #1
out[1]
Characteristic polynomials #7
out[7]
Pseudo Random Number Generator

Figure.6 Pseudo random-based counter measure circuit

and we will merge them to get flip-flops in n6 and n7, respectively. Suppose there is no overlap region between the couple of flip flops in n2 and n4. It fails to form a 4-bit flip-flop in n6. Since the 2-bit flip-flops f3 and f6 are mergeable, combine them to obtain a 4-bit flip-flop f10 in n7.This method of merging of flip-flop is introduced in counter measure circuits which are used to avoid the DPA attack in the cryptographic applications.

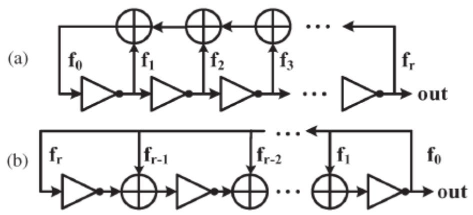
C. DPA Attack

The DPA attack utilizes the statistical analysis to calculate the correlation between the leaked power information and the predicted power consumption. DPA attack can successfully conduct in a noisy environment because of the statistical analysis. It is possible to disclose the secret key of a cryptographic circuit from the correlation index of the analysis result. N different patterns are there for en-/decryption and it also records the power trace of these patterns. These N power traces, which consist of T sample points, are firstly arranged as an N-by-T measured power array for further processing. Also these N patterns are also applied to a power prediction model to obtain predicted power values. These power values are arranged as an N-by-K predicted power array, where N is the number of patterns, and K is the number of all possible key hypotheses.

The measured and the predicted power arrays are taken, then disclose the secret key by the statistical analysis. Each column of the predicted power array is used to find a correlation index with every column of the measured power array. If the key hypothesis matches the secret key used by the cryptographic circuit, the correlation index would be higher than that of other key hypotheses. The correlation index can be obtained by statistical analysis using difference on-means or correlation.

D. DPA Countermeasure Circuit

To resist the DPA attack, both the pseudo and the true random-based DPA countermeasure circuits are presented. The disadvantage of pseudo random-based generator is avoided by the true random-based counter measure circuit



.Figure 7. (a)FiRO (b) GaRO

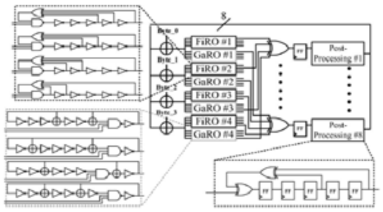


Figure 8.proposed true random-based countermeasure circuit

E.Pseudo Random-Based Dpa Countermeasure Circuit

Usually to break the dependency between the measured power traces and the predicted power values the DPA countermeasure circuits are used. The proposed DPA countermeasure circuit consists of 16 identical sub circuits as shown in fig 7. Each sub circuit, which is composed of 12 digitally controlled ring oscillators. As shown in Fig. 7, the random number generator is designed based on linear feedback shift registers (LFSRs) with dynamic feedback configuration to make the random sequence more unpredictable [12]. Each sub circuit is controlled by a data byte of the encryption data block and the random byte from the pseudo random number generator.

F. True Random-Based Dpa Countermeasure Circuit

To overcome the disadvantage of pseudo random-based architecture, a true random sequence for the DPA countermeasure circuit is required. The true random generators are analog circuits with much higher power consumption, Goli proposed a digital method to generate random data by using ring oscillators in Fibonacci and Galois configurations [13]. The Fibonacci and the Galois ring oscillator consist of a series of inverters connected with feedback polynomial as shown in Fig7[13]. The proposed DPA countermeasure circuit that can generate a true random sequence by itself is shown in Fig 8.

The proposed DPA countermeasure circuit consists of four Fibonacci ring oscillator sets (FiRO), four Galois ring oscillator sets (GaRO),and eight postprocessing circuits. The FiRO and GaRO are composed of four Fibonacci and Galois ring oscillators, respectively. The proposed architecture incorporates a true random number generator into the DPA countermeasure circuit to resist the DPA attack and the reset problem The combination of two FiROs and two GaROs is used as the random source to generate one random sequence.

In order to generate eight independent random bits for each data byte, a total of 32 ring oscillators (including Fibonacci Galois ring oscillators) are required in the DPA countermeasurecircuit. Flip-flops are using for the post processing. These eight random bits are XORed with data bytes from the cryptographic circuit. The post processing circuits are composed of LFSRs with different initial seeds. The post processing circuit is used to remove the bias of the random source. In each post processing circuit, the feedback value is XORed with that from the random source. the post processing circuit can start from a deterministic state even after the system is reset.

The proposed DPA countermeasure circuit is taken and prepared a combination table for the LFSRs present in the circuit. In order to prepare the combination table the algorithm which is proposed earlier is used. From the prepared combination table taken out the flip- flops that can be merged and carried out the merging of flip-flop.

IV. EXPERIMENTAL RESULTS

This section shows the experimental results. The experiment is carried out in the proposed DPA countermeasure circuit. The experimental results obtained are shown in table I and table II. Usually the true random-based circuit results an overhead in power but by using the proposed method it is found that 27.27% of dynamic power and 12.59% of total power is reduced, it is also found that the total gate count is reduced from 7709 to 7389 compared to the design without using the merging concept. The

TABLE I. POWER ANALYSIS

	Power	**Before Merging**	**After Merging**	**Percentage Reduction in power**
Proposed Counter-measure circuit	Dynam-ic Power	66mW	48mW	27.27%
	Total Power	143mW	125mW	12.59%

TABLE II. AREA ANALYSIS

		Before Merging	**After Merging**
Proposed Counterm-easure circuit	Total Gate Counts	7709	7389

experiment is carried out using Xilinx ISE Design suite 12.3 keeping Spartan 3E-XC3S500 as the target device.

V. CONCLUSION

The proposed algorithm is used to merge the flip-flops for power reduction, in cryptographic circuit. The procedure includes the preparation of combination table, which records the relationships among the flip-flop types. The concept of pseudo type is introduced to help to enumerate all possible combinations in the combination table. This concept of merging of flip-flop is introduced in the DPA countermeasure circuit. After analysis it is found that the dynamic power is reduced to 27.27% and the total power is reduced to 12.59%. The total gate count is reduced from 7709 to 7389 compared to the circuit without merging.

VI. ACKNOWLEDGMENT

The authors also thank the Principal and Management of Sri Ramakrishna Engineering College, Coimbatore for providing excellent computing facilities.

REFERENCES

[1] Ya-Ting Shyu, Jai-Ming Lin, Chun-Po Huang, Cheng-Wu Lin, Ying-Zu Lin, and Soon-Jyh Chang,. 2012Member, IEEE" Effective and Efficient Approach for Power Reduction by Using Multi-Bit Flip-Flops"(will be published).

[2] Chang, Y.-T. Hsu, C.-C. Lin, M. P.-H. Tsai, Y.-W. and Chen, S.-F.2010,"Post-placement power optimization with multi bit flip- flops," in Proc.IEEE/ACM Int. Conf. Comput.-Aided Des., pp. 218–223.

[3] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *IEEE Trans. CircuitsSyst. I*, vol. 47, no. 3, pp. 415–420, Mar. 2000.

[4] H. Mahmoodi, V. Tirumalashetty, M. Cooke, and K. Roy, Ultra low power clocking scheme using energy recovery and clock gating," *IEEE* Trans. Very Large Scale Integr. Syst., vol. 17, no. 1, pp. 33– 44, Jan.2009

[5] D.-S. Chiou, S.-H. Chen, S.-C. Chang, and C. Yeh, "Timing driven power gating," in *Proc. ACM/IEEE Des. Autom. Conf.*, Sep. 2006, pp.121–124.

[6] K. Wang and M. Marek-Sadowska, "Buffer sizing for clock power minimization subject to general skew constraints," in *ProcCM/IEEE Des. Autom.Conf.*, Jul. 2004, pp. 159–164.

[7] L. Huang, Y. Cai, Q. Zhou, X. Hong, J. Hu, and Y. Lu, "Clock network minimization methodology based on incremental placement," in *Proc.IEEE/ACM Asia South Pacific Des. Autom.Conf.*, Jan. 2005, pp. 99– 102. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science,1989.

[8] Y. Lu, C. N. Sze, X. Hong, Q. Zhou, Y. Cai, L. Huang, and J. Hu, "Navigating registers in placement for clock network minimization," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2005.

[9] R. Pokala, R. Feretich, and R. McGuffin, "Physical synthesis for performance optimization," in *Proc. IEEE Int. ASIC Conf. Exhibit.*, Sep.1992, pp. 34–37.

[10] Y. Kretchmer, "Using multibit register inference to save area and power," *EE Times Asia*, May 24, 2001.

[11] G. Magklis, M. L. Scott, G. Semeraro, D. H Albonesi, and S. Dropsho, "Profile-based dynamic voltage and frequency scaling for a multiple clock domain microprocessor," in *Proc. ACM Int. Symp. Comput Architecture*, 2003, pp. 14–27–37.

[12] A True Random-Based Differential Power Analysis Countermeasure Circuit for an AES Engine Po- Chun Liu, Hsie-Chia Chang, *Member,IEEE, and Chen-Yi Lee, Member, IEEE,Feb 2012*

[13] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proc. 28th Eur. Solid-State Circuits Conf.*, Sep. 2002, pp. 403–406.

[14] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, Feb. 2004, vol. 1, pp. 246–251.

AUTHORS PROFILE

Ashna V R pursuing M.E(Master of Engineering) in VLSI Design from Sri Raakrishna Engineering College. She has received her B.Tech in Electronics and Communication Engineering from Cochin University of Science and Technology.

Her areas of interests include low power vlsi design and testing of vlsi circuits.