International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-3, February 2020

# Compact Reconfigurable Architecture for Sosemanuk Stream Cipher

**Nagnath B. Hulle, Prathiba B, Sarika R Khope**

***Abstract*: Sosemanuk is word oriented synchronous stream cipher capable to produce 32 bit ciphertext. It uses variable key from 128 bit to 256 bit and publically known Initialization Vector (IV) of 128 bit. Sosemanuk is one of the finalists in Profile 1 of the eSTREAM Portfolio. This cipher targets to avoid structural properties of SNOW2.0 to improve its efficiency by reducing the internal state size. It also uses reduced round Serpent24 block cipher to provide secure and efficient key loading process. This paper presents compact architecture for Sosemanuk stream cipher. The proposed architecture uses compact S-box architecture and compact modulo adders designed using CLA. The proposed compact S-box minimizes resources utilized without affecting performance. Proposed modulo adder architecture minimizes resources used as compared to conventional CLA implementation. The algorithm was designed by using VHDL language with CAD tool Xilinx ISE design suite 13.2 and implemented on Xilinx Virtex XC5VFX100E FPGA device. The proposed architecture achieved throughput of 4.281 Gbps at clock frequency of 133.788 MHz***

***Keywords: Compact, FPGA, modulo adders, Stream Cipher, Sosemanuk.***

## I. INTRODUCTION

New European Schemes for Signatures, Integrity and Encryption (NESSIE) project was started in the year 2000 to identify new secure cryptographic algorithms [1]. During the security algorithm selection process only block ciphers were selected in its final portfolio. Attacks published on stream ciphers during this project were not practical and big question rose on security of stream ciphers. The failure result of all 6 stream ciphers in NESSIE project evolved eSTREAM project coordinated by European Network of Excellence in Cryptology (ECRYPT). The project was started in year 2004 to inspire new effective hardware as well as software based stream cipher designs. This project was not a competition for deciding standards as in case of DES and AES, but the main focus of the project is to find auspicious stream ciphers. After rigorous security testing against various attacks, seven stream cipher algorithms were announced in in final of eSTREAM project [2].

Revised Manuscript Received on February 05, 2020.
* Correspondence Author
**Nagnath Bhagwat Hulle,** Associate. Prof., G. H. Raisoni Institute of Engineering & Technology, Pune, India. E-mail: nagnath.hulle@raisoni.net
**Prathiba B,** Asst. Prof., G. H. Raisoni Institute of Engineering & Technology, Pune, India. E-mail: balireddyprathibha@gmail.com
**Sarika R Khope,** Asst. Prof., G. H. Raisoni Institute of Engineering & Technology, Pune, India. E-mail: Sarika.gabhane@raisoni.net



Sosemanuk is one of the finalists in Profile 1 (software) ciphers selected for the eSTREAM Portfolio [2]. Looking at current design platforms such as System on Chips (SoCs), it is difficult to differentiate between hardware and software part of the system. These trends are increasing demand for hardware implementation of software based stream ciphers. Considering SOC design aspects proposed work considers hardware implementation of software based stream cipher Sosemanuk [3].

This cipher is 32 bit word oriented synchronous stream cipher designed by Come Berbain et al. [4] to use best properties of block and stream ciphers. It provides good security by using combined effect of SNOW2.0 [5] stream cipher and Serpent block cipher [6]. Two versions of reduced round Serpent block ciphers are used in the design as Serpent1 and Serpent24. Serpent1 is single round Serpent algorithm used at output stage during encryption and Serpent24 is 24 round Serpent algorithm used during initialization phase of Sosemanuk. This cipher uses variable key length between 128 bits to 256 bits [4].

Reconfigurable devices such as FPGAs are a highly attractive option for a hardware implementation as they provide the flexibility of dynamic system evolution as well as the ability to easily implement a wide range of functions/algorithms. It appears to be especially relevant to focus on high-throughput implementations for FPGA-based encryption [7].

The paper is organized in following sections. Section II describes the specifications and working of Sosemanuk stream cipher. Section III presents proposed architecture for Sosemanuk stream cipher. The VLSI implementation, its results are discussed in Section IV and Section V concludes the proposed work.

## II. SPECIFICATIONS AND WORKING OF SOSEMANUK STREAM CIPHER

Sosemanuk stream cipher consists of main three modules, SNOW 2.0, Serpent24 and Serpent1 as shown in Figure 1.

### A. SNOW 2.0

Sosemanuk [4] uses SNOW 2.0 [5] by modifying in some aspects for better security and improved performance. Linear Feedback Shift Register (LFSR) used in SNOW 2.0 consists of 16 stages each of 32 bit providing 512 bits of internal states. The aim of Sosemanuk is to provide 128 bit security,

so shorter LFSR will do the task. LFSR with length of 8 will be vulnerable to guess and determine attack. Internal states must be ≥ 384 bits to avoid above attack. Sufficient level of security is achieved by using LFSR with 10 stages and two 32 bit stages are used from Finite State Machine (FSM), as $R_1$ and $R_2$ as shown in Figure 2. This modification in the SNOW 2.0 provides requisite security with less hardware [4].

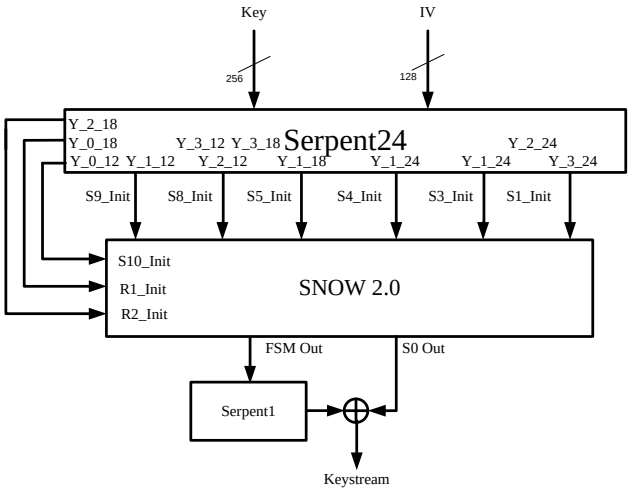


**Figure 1. Sosemanuk Block Diagram**

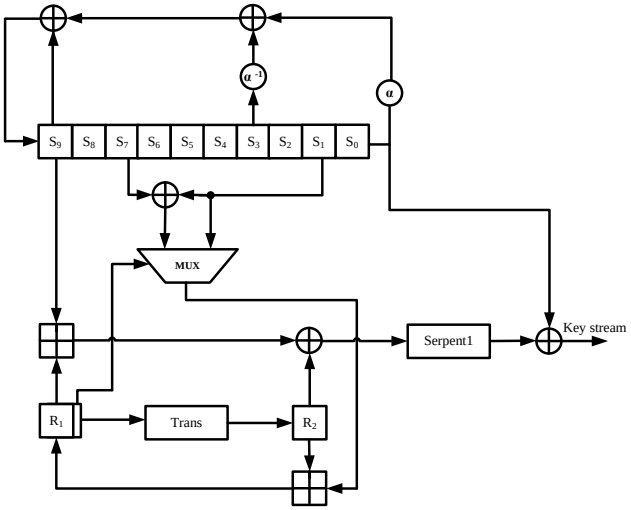


**Figure 2. Sosemanuk architecture**

The feedback polynomial used must be as light as possible to have improved performance. Sosemanuk uses feedback polynomial given by equation (1) and it is lighter than feedback polynomial used 2.0 [4].

$$\pi(x) = \alpha x^{10} + \alpha^{-1} x^{7} + x + 1 \in F_{2^{32}}[X] \quad (1)$$

This polynomial is primitive polynomial having maximal sequence of $(2^{320}-1)$.

Finite State Machine (FSM) of Sosemanuk takes input from $S_9$, $S_7$, $S_1$, $R_{1t-1}$ and $R_{2t-1}$ and delivers output $FSM_t$ for t > 0 as given by equation (2).

$$FSMt = (R_{2t} \oplus [S_9 \boxplus R_{1t}] \quad (2)$$

Where,

$R1_t = [R2_{t-1} \boxplus ((S_1) \ \ or \ \ (S_1 \oplus S_7))]$, selection of $S_1$ or $(S_1 \ \ or \ \ S_1 \oplus S_7)$ is decided by is decided by select line (LSB ($R_{1t-1}$)) of the multiplexer.

$R_{2t-1} = \text{Trans}(R_{1t-1})$

$\text{Trans}(R_{1t-1}) = (M * R_{1t-1}) \bmod 2^{32} <<< 7$

$\oplus$ is bitwise XOR operation

M is 0x54655307 hexadecimal constant value.

$\boxplus$ integer addition over GF($2^{32}$)

<<< 7 is bitwise rotation of 32 bit value by 7 bit

## B. Serpent1

Serpent1 is single round Serpent block cipher without key addition and linear transformation. This algorithm takes four 32 bit inputs in bitslice mode and S-Box used in this algorithm is S2 [2, 4].

## C. Serpent24

Serpent24 is 24 round block cipher used for initializing LFSR, R1 and R2. This block cipher provides secure 384 bits internal states to Sosemanuk stream cipher. Architecture of Serpent24 is as shown in Figure 3 and this architecture provides initial values to all internal states Sosemanuk [2, 4].

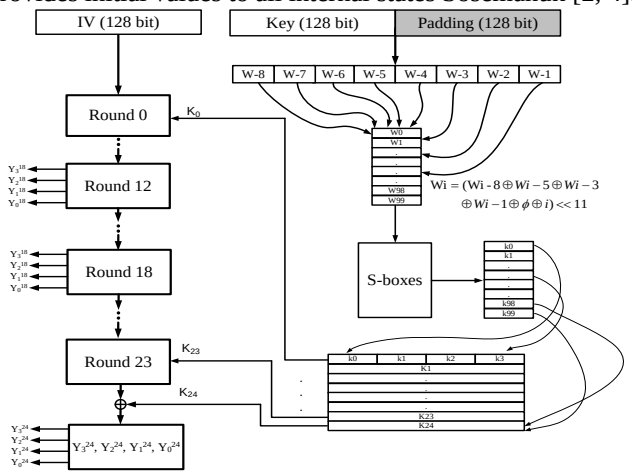


**Figure 3. Serpent24 architecture**

Working of Sosemanuk stream cipher is divided into two phases, initialization phase and working phase.

### 1. Initialization phase

During this phase 128 bit IV and 256 bit key are applied to the Serpent24 as shown in Figure 3. Serpent24 architecture is used for secure internal state generation. Complete working of Serpent is divided into two parts, round key generation and round operations.

Round key generation part takes 256 bit key and divides it into eight initial words. Using initial eight words on equation (3), 100 sub words are produced.

$$Wi = (Wi-8 \oplus Wi-5 \oplus Wi-3 \oplus Wi-1 \oplus \phi \oplus i) <<< 11 \quad (3)$$

Where, Φ is the fractional part of the golden ratio $(\sqrt{5}+1)/2$ or 0x9e3779b9 in hexadecimal form.

These 100 sub words are applied over S-boxes [6] to produce 100 sub keys. These keys are concatenated to produce 25 round keys.

During round operations each round of serpent24 is as shown in Figure 4 and performs three tasks, round key addition, S-box implementation and linear transformation. Working of first 24 rounds is same but last round does not uses S-box and linear transformation.

First 24 rounds use respective round key to perform XOR operation with input data at each round. The result of XOR operation is passed to S-boxes. The selection of S-box from S0 to S7 for each round is defined in [6] and these S-boxes are implemented as lookup tables. The last step in each round is linear transformation as shown in Figure 5. This operation maximizes the avalanche effect.

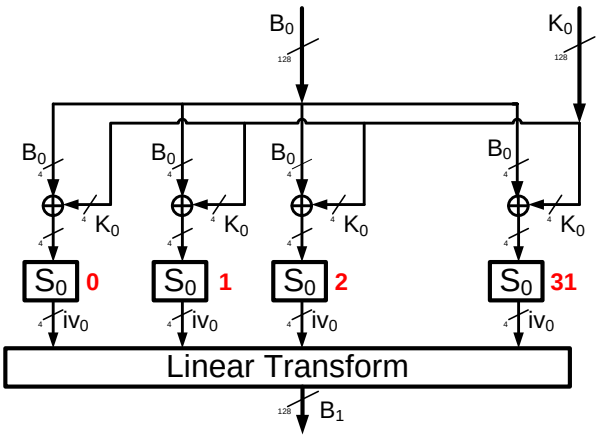


**Figure 4. Round structure for Serpent**

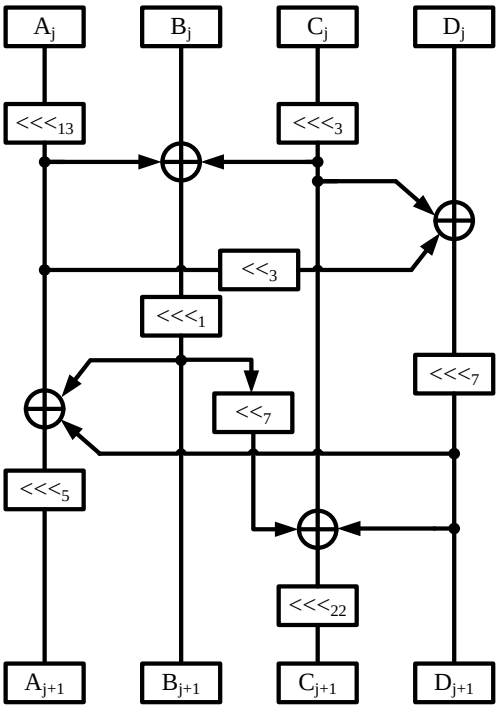


**Figure 5. Linear transform**

The outputs of Round12, Round18 and Round24 are used to initialize the LFSR, state register R1 and R2 as shown by equation (4).

$$\text{Round 12 output} = (Y_3^{12}, Y_2^{12}, Y_1^{12}, Y_0^{12})$$

$$\text{Round 18 output} = (Y_3^{18}, Y_2^{18}, Y_1^{18}, Y_0^{18})$$

$$\text{Round 24 output} = (Y_3^{24}, Y_2^{24}, Y_1^{24}, Y_0^{24})$$

$$(S_7, S_8, S_9, S_{10}) = (Y_3^{12}, Y_2^{12}, Y_1^{12}, Y_0^{12})$$

$$(S_5, S_6) = (Y_1^{18}, Y_3^{18})$$

$$(S_1, S_2, S_3, S_4) = (Y_3^{24}, Y_2^{24}, Y_1^{24}, Y_0^{24})$$

$$R_{10} = Y_0^{18}$$

$$R_{20} = Y_2^{18}$$

(4)

**2. Keystream generation**

$$(S_{9update}, S_9, ..., S_2, S_1) \rightarrow (S_9, S_8, ..., S_1, S_0)$$

(5)

Where, $S_{9update} = S_9 \oplus (\alpha^{-1} S_3 \oplus \alpha S_0)$

During each clock cycle FSM output is updated by equation (2) and applied at the input of the Serpent1. Finally output of Serpent is XORed with $S_0$ to produce key stream.

## III. PROPOSED SOSEMANUK ARCHITECTURE

Proposed architecture of Sosemanuk stream cipher is shown in Figure 6. This implementation targets speed improvement of modulo adders in FSM, S-box area minimization in serpent24 and Serpent1. Speed of modulo adders is improved by using compact Carry Lookahead Adder (CLA) [8]. Similarly, S-box area is minimized by using novel multiport ROM architecture for lookup table implementation.

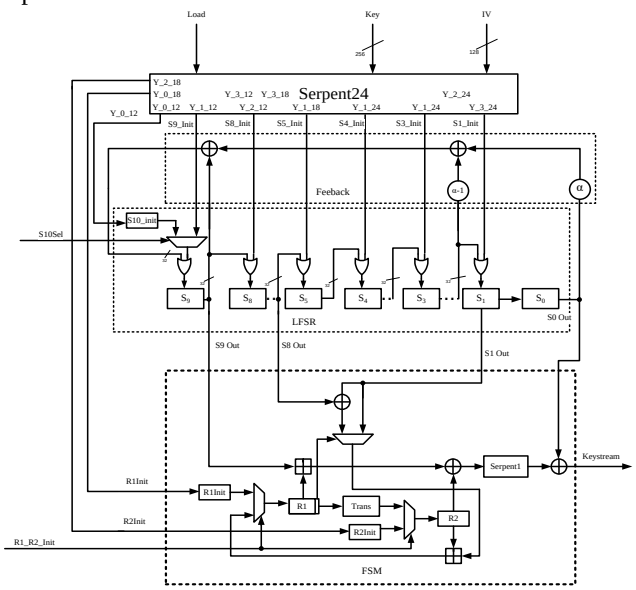


**Figure 6. Sosemanuk proposed architecture**

As shown in above figure the architecture provides initial state loading facility in the hardware. When load terminal = 1, allows loading 12 initial values into 10 LFSR stages and two state registers R1 & R2. Initial loading and normal working facility is provided by using multiplexers with select terminals R1_R2_Init, S10Sel and OR gates. During initial loading OR gate receives data from Serpent24 rounds and other terminal values are zero. When normal working starts data received from Serpent24 is zero. Selecting proper select line of multiplexer initialization and normal working are differentiated.

The hardware consuming and less utilized part of the Sosemanuk is Serpent24 block cipher. This block cipher consists of 8 S-boxes S0 to S7. These S-boxes are used in round operations and round key generation process. Each S-box implementation consumes 8 bytes of memory.

Round operation uses one of the S-box 32 times and consuming area of (32 x 8) 256 bytes. Serpent24 consists of 24 rounds and consuming area of (24 x 256) 6144 bytes.

Similarly round key generation process generates 100 intermediate round keys from
intermediate words. Each intermediate round key is of 32 bit and uses 8 S-boxes for its generation. The total (8 x 100) 800 S-boxes are required for 100 intermediate round keys. This process requires (800 x 8) 6400 bytes of memory for S-box implementation.

The total memory required for S-box implementation in Serpent24 is (6144 + 6400) 12544 bytes. Memory required for S-box implementation is reduced by using multiport ROM architecture as shown in Figure 6. This Multiport ROM architecture is capable of reading different memory with 32 addresses simultaneously.

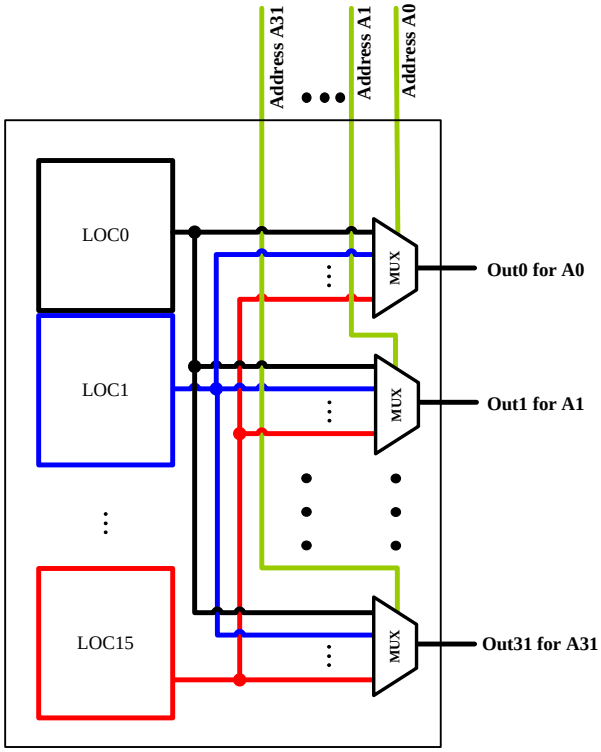


**Figure 7. S-box architecture using multiport ROM**

Proposed multiport ROM architecture replaces each S-box by 16:1 multiplexer. This architecture does not affect performance of complete architecture. Use of modulo adders designed by using CLA improves speed of Sosemanuk.

## IV. RESULT AND DISCUSSION

The proposed work is implemented on FPGA platform and coded by VHDL language with CAD tool Xilinx ISE design suite 13.2. Comparison of memory required for Serpent24 implementation in original form and with proposed multiport ROM architecture gives the amount of memory saved in designing of S-boxes.

With multiport architecture memory required for round operations is (24 x 8) 192 bytes. Memory required for round key generation with this concept is (100 x 8) 800 bytes. The total memory required using this concept is (192 + 800) 992 bytes.

Using proposed multiport architecture saving of (12544 - 992) 11552 bytes is possible without getting initial delay. The multiport architecture costs extra area of 16:1 multiplexer for each new port.

**Table 1. Comparison of proposed, [3] and [9] architectures**

| Parameter | Proposed | [3] | [9] |
|---|---|---|---|
| Memory in KB | 0.992 | 12.544 | 0.264 |
| Hardware used in K LUTs | 8.448 | 20.59 | 3.763 |
| Throughput in Gbps | 4.281 | 32.32 | 3.200 |
| Key length used | 256 | 256 | 32 |
| IV length used | 128 | 128 | 32 |

* Conversion of gate equivalent to LUTs is taken as 1:5

Architecture [9] uses less memory as compared to proposed architecture for S-box implementation and 100 iterations are performed is sequential order. This introduces initial delay in the algorithm.

Use of compact modulo CLA architecture improves speed of modulo addition in FSM. Compact modulo CLA architectures minimizes hardware of conventional CLA.

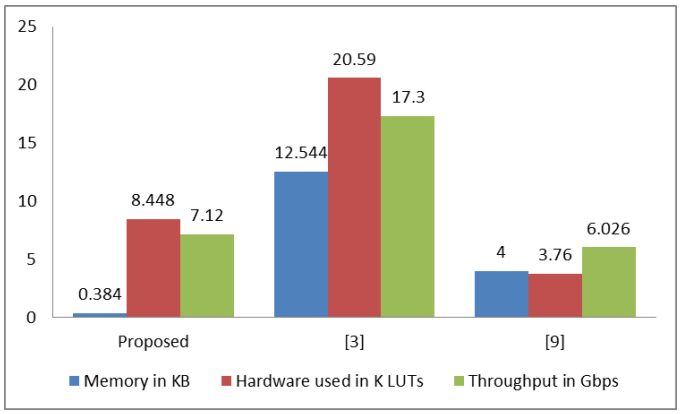


**Figure 8. Comparisons of proposed, [3] and [9] architectures**

## V. CONCLUSION

This paper presents compact architecture for Sosemanuk stream cipher. Proposed work uses compact modulo CLA architecture to minimize delay in FSM. This work uses 992 bytes of memory for S-box implementations without affecting performance of Serpent24 block cipher. Sosemanuk achieved throughput of 4.281 Gbps at a clock frequency of 133.788 MHz, where as other architectures [3] and [9] achieved throughput better than proposed architecture. Proposed architecture uses less hardware and consumes less energy as compared to other [3] and [9] architectures.

## FUTURE SCOPE

Resolving the fan-out issue of memory locations of S-boxes only 8 (S0 to S7) S-boxes are sufficient to implement all S-boxes required in Serpent block cipher.

## REFERENCES

1. cosic.esat.kuleuven.be, 'nessie', 2011. [Online]. Available: https://www.cosic.esat.kuleuven.be/nessie/. [Accessed: 23- May- 2015].
2. Ecrypt.eu.org, 'The eSTREAM Project', 2011. [Online]. Available: http://www.ecrypt.eu.org/stream/finallist.html. [Accessed: 07- Jun- 2015].
3. G. Paul and A. Chattopadhyay, "Three Snakes in One Hole : A 67 Gbps Flexible Hardware for SOSEMANUK with Optional Serpent and SNOW 2.0 Modes*", in IACR Cryptology ePrint Archive, 2013, pp. 1–19.
4. Berbain, Côme, et al. "Sosemanuk, a fast software-oriented stream cipher" New Stream Cipher Designs, Springer Berlin Heidelberg, 2008, pp. 98-118.
5. Ekdahl, Patrik, and Thomas Johansson. "A new version of the stream cipher SNOW" Selected Areas in Cryptography, Springer Berlin Heidelberg, 2003.
6. Anderson, Ross, Eli Biham, and Lars Knudsen. "Serpent: A proposal for the advanced encryption standard." NIST AES Proposal 174, 1998.
7. A. Elbirt and C. Paar, "An FPGA Implementation and Performance Evaluation of the Serpent Block Cipher," in International symposium on Field programmable gate arrays, 2000, pp. 33–40.
8. N. B. Hulle, Dr. R. D. Kharadkar, and Dr. A. Y. Deshmukh, "The Novel Architecture for Carry Lookahead Adder," Technical Journal of The Institution of Engineers (India), Pune Local Centre, vol. 36, no. 1, pp. 84–88, 2012.
9. T. Good and M. Benaissa, "Hardware results for selected stream cipher candidates," in Hardware results for selected stream cipher candidates, 2007, pp. 1–14.

## AUTHORS PROFILE

**Dr. Nagnath B. Hulle** is Associate Professor at G. H. Raisoni Institute of Engineering and Technology, Pune, Maharashtra, India.
He has teaching experience of 20 years. He guides UG and PG students. His area of working is VLSI & Wireless Network Security. He published papers in National & International Journals and conferences.
Mr. N. B. Hulle is life member of ISTE and IETE.

**B. Prathiba,** currently working as an Asst.Prof. in the department of E&TC, G.H.Raisoni Institute of Engineering and Technology, Pune. She has published the articles in various international publications in the area of Wireless Sensor Networks, VLSI. Her interested areas are Wireless Sensor Networks, Microcontrollers and VLSI. Mrs Prathiba is life member of ISTE and IETE.

**Sarika Khope**, currently working as an Asst.Prof. in the department of E&TC, G.H.Raisoni Institute of Engineering and Technology, Pune. She has published the articles in various international publications in the area of VLSI,. Digital Systems. Her interested areas are Machine Learning and VLSI.