IAES Journal of Electrical and Electronics Engineering
Vol. 10, No. 2, December 2023, pp. 41~45
ISSN: 2338-7432, DOI: 10.11591/jeee.v10i2.pp41-45

Hardware architecture of the SNOW 3G stream cipher: a review of design principles, FPGA implementation, performance optimization, and security considerations

Shinta Devia[1], Anggit Pamungkas[2]
[1]Faculty of Industrial Technology, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[2]Embedded Systems and Power Electronics Research Group, Yogyakarta, Indonesia

Article Info

Article history:

Received Nov 20, 2020
Revised Sep 16, 2021
Accepted Mar 8, 2022

Keywords:

Finite state machine
FPGA
Hardware architecture
LFSR
Mobile communication security
SNOW 3G
Stream cipher

ABSTRACT

SNOW 3G is a word-oriented synchronous stream cipher adopted in mobile communication security to support confidentiality and integrity services. Its architecture combines a linear feedback shift register (LFSR) with a nonlinear finite state machine (FSM), generating a pseudorandom keystream from a secret key and initialization vector. Although the algorithm provides an established security mechanism for cellular networks, practical hardware implementation must balance throughput, area utilization, power consumption, initialization latency, and resistance to implementation attacks. This review examines the internal architecture of SNOW 3G and the principal hardware approaches reported up to 2022. Particular attention is given to LFSR and FSM organization, nonlinear substitution functions, modular arithmetic, initialization control, field-programmable gate array (FPGA) implementation, and architectural optimization. The review also considers the implications of loop unrolling, carry-lookahead adders, resource sharing, and substitution-box design for implementation efficiency. Published results indicate that substantial throughput can be achieved through carefully optimized FPGA architectures, although comparisons remain sensitive to device technology and evaluation methodology. The review identifies research priorities in reproducible benchmarking, side-channel protection, secure bitstream management, and efficient hardware integration.

This is an open access article under the CC BY-SA license.

1. INTRODUCTION

Secure mobile communication requires cryptographic mechanisms capable of protecting transmitted information while meeting strict processing and latency requirements. Stream ciphers are useful in this context because they generate pseudorandom keystream data that can be combined with message data for encryption and decryption. SNOW 3G is a word-oriented synchronous stream cipher developed for the 3GPP confidentiality and integrity algorithms used in cellular communication systems. Its standardized role and internal architecture have made it an important subject in cryptographic implementation and security research [1]–[5].

The practical implementation of SNOW 3G presents several hardware designs challenges. Its internal state requires multiple registers, nonlinear transformations, and modular arithmetic operations that must be coordinated at every clock cycle. A straightforward implementation may achieve functional correctness but still exhibit excessive critical-path delay, high logic utilization, or limited throughput. Consequently, hardware researchers have investigated optimized arithmetic units, substitution-box architectures, pipelining, and parallel processing to improve implementation efficiency [1], [2], [6]–[8].

A second concern is that implementation efficiency must not compromise cryptographic correctness or security. Modifying the internal state update, substitution functions, or initialization process can produce a faster circuit but may also create a cipher that no longer conforms to the standardized algorithm. Moreover, a

mathematically secure cipher can remain vulnerable when implemented in hardware that exposes secret-dependent power consumption, electromagnetic emissions, or configuration data. This review therefore examines SNOW 3G from both architectural and security perspectives, connecting its internal design to practical field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) implementation requirements.

2. FUNDAMENTAL ARCHITECTURE OF SNOW 3G

SNOW 3G is a synchronous stream cipher that operates on 32-bit words and uses a 128-bit secret key together with a 128-bit initialization vector. Its keystream generator consists of two principal components: a 16-stage linear feedback shift register (LFSR) and a nonlinear finite state machine (FSM) containing three 32-bit registers. The LFSR provides the linear state evolution, while the FSM introduces nonlinear operations and combines selected LFSR values to produce the output. This combination is intended to provide a balance between efficient state updating and resistance to cryptanalytic techniques [1], [4], [5].

The LFSR is responsible for maintaining and updating the main internal state of the cipher. Its feedback function uses finite-field operations and selected register values to calculate the next state word. The FSM uses nonlinear substitution operations, modular additions, and register feedback to produce intermediate values for keystream generation. The interaction between the two components is central to the cipher because the FSM output is combined with an LFSR-derived value to form the keystream word [4], [5].

SNOW 3G operates through initialization and keystream-generation phases. During initialization, the secret key and initialization vector are loaded into the internal state, and the cipher performs a prescribed number of state-update cycles without releasing keystream output. This procedure allows the initial state to be mixed before the cipher begins generating output words. During normal operation, the LFSR and FSM update synchronously, producing one 32-bit keystream word per logical clock cycle in the standard architecture [1], [4].

3. HARDWARE ARCHITECTURE AND FUNCTIONAL MODULES

A hardware implementation of SNOW 3G can be divided into an input and initialization unit, LFSR datapath, FSM datapath, keystream output unit, and control logic. The input unit receives the secret key and initialization vector and distributes their words to the appropriate internal registers. The LFSR and FSM datapaths perform the algorithm's state transformations, while the controller manages initialization and output-generation phases. This modular organization supports functional verification and allows individual hardware blocks to be optimized without changing the external algorithmic behavior.

The LFSR implementation is dominated by register storage, feedback selection, and finite-field arithmetic. Its 16-word state can be mapped to flip-flops or other suitable FPGA storage resources, depending on the target device and design constraints. The feedback calculation requires exclusive OR (XOR) operations and multiplication-related transformations in the prescribed finite field. Efficient implementation of these operations can reduce logic depth and improve the maximum operating frequency [2], [3], [6].

The FSM is generally more demanding because it combines nonlinear transformations with arithmetic operations. Its three 32-bit registers participate in the state-update process, while substitution functions introduce nonlinear behavior. The two substitution transformations are based on byte-oriented nonlinear mappings and diffusion operations, making their implementation a significant factor in area and delay. Researchers have therefore investigated alternative logic organizations and optimized substitution-box structures to reduce hardware overhead [2], [3], [4].

The controller must also ensure that initialization and keystream generation follow the specified sequence. A control error during initialization may lead to an incorrect internal state and therefore an invalid keystream. The controller should prevent output generation until the initialization procedure has completed successfully. A clear separation between initialization and generation states also simplifies verification and reduces the possibility of accidental output during the setup phase.

4. FPGA AND ASIC IMPLEMENTATION STRATEGIES

FPGA implementation is attractive because it allows rapid prototyping and hardware evaluation without requiring a custom integrated-circuit fabrication process. Very high-speed integrated circuit hardware description language (VHDL) and Verilog descriptions can be synthesized into configurable logic blocks, registers, and routing resources. Researchers can then measure throughput, maximum clock frequency, logic utilization, and latency under a specified device configuration. However, FPGA results are strongly influenced by the target family, synthesis settings, timing constraints, and resource-mapping decisions [2], [6], [9]–[11].

Hulle *et al*. [2] proposed an optimized SNOW 3G architecture using a carry-lookahead adder structure and an improved substitution-box implementation. Their FPGA results reported a maximum operating frequency of approximately 254.9 MHz and a throughput of approximately 7.22 Gbps on a Xilinx Virtex-5 device. The work illustrates how arithmetic optimization and nonlinear logic redesign can improve the implementation characteristics of the cipher. Nevertheless, the reported values should be interpreted within the specific device and synthesis environment used by the authors.

Madani *et al*. [3] investigated an FPGA implementation of an enhanced SNOW 3G stream cipher using a hyperchaotic system. This work is relevant to hardware experimentation, but the introduction of an additional chaotic component means that the resulting design should not automatically be treated as a drop-in implementation of standardized SNOW 3G. Any modification to the keystream-generation process requires independent security analysis and interoperability assessment. The distinction between implementing the standard algorithm and proposing a modified cipher is therefore essential [12].

Other studies provide useful comparative information about the hardware behavior of stream ciphers. An FPGA-based performance analysis by D' and colleagues compared SNOW 3G with ZUC, Grain, MICKEY, Trivium, and E0 using throughput, area, and throughput-to-area measures [6]. Ma and Guan [9] examined how loop unrolling affects hardware efficiency in eSTREAM stream ciphers, offering useful design lessons about the relationship between parallelism and area consumption. These studies demonstrate that hardware optimization must be evaluated against both performance and resource requirements rather than clock frequency alone [13]–[16].

5. ARCHITECTURAL OPTIMIZATION TECHNIQUES

Arithmetic optimization is one of the most effective ways to reduce the critical-path delay in SNOW 3G hardware. Modular addition is used in the FSM, and its implementation can influence the maximum clock frequency of the entire datapath. Ripple-carry structures are relatively simple but may introduce long carry-propagation delays for 32-bit operations. Carry-lookahead or other parallel-prefix structures can reduce this delay, although they may require additional logic resources and routing complexity [2], [10].

Substitution-box optimization is another important design consideration because nonlinear transformations can consume a substantial portion of the combinational logic. A direct table-based implementation may be mapped to FPGA lookup tables or memory resources, depending on the synthesis environment. An alternative logic-based implementation can reduce memory usage but may increase combinational depth or routing demand. The preferred structure should therefore be selected through technology-specific synthesis and measurement rather than by assuming that one S-box architecture is universally optimal [2], [10], [11].

Loop unrolling and parallel processing can increase the number of keystream words generated per clock cycle. However, unrolling duplicates or expands portions of the state-update logic and may increase area, routing congestion, and critical-path delay. Ma and Guan [9] showed that the efficiency gains of unrolling in stream ciphers are not necessarily proportional to the unrolling factor, because physical implementation effects can alter theoretical expectations. For SNOW 3G, the complexity of the state-update function makes aggressive unrolling a particularly important trade-off to evaluate.

Table 1 summarizes representative research relevant to SNOW 3G hardware architecture and implementation. It distinguishes direct SNOW 3G implementation studies from comparative stream-cipher hardware research and security studies. This distinction prevents results from other stream ciphers or modified SNOW variants from being interpreted as direct performance evidence for the standardized algorithm. The table also identifies the main architectural lessons that can be drawn from each research group.

Table 1. Representative literature relevant to SNOW 3G architecture and hardware implementation

Reference	Research focus	Architectural relevance
Hulle *et al*. [2]	Optimized SNOW 3G FPGA architecture	Carry-lookahead arithmetic and S-box optimization
Madani *et al*. [3]	Enhanced SNOW 3G FPGA implementation	Modified architecture and hardware evaluation
Bahadori *et al*. [4]	FPGA implementation of SNOW-family 256-bit ciphers	Comparative hardware lessons for the SNOW family
Nandi *et al*. [5]	SNOW 1.0, SNOW 2.0, and SNOW 3G review	Cipher structure and implementation considerations
FPGA stream-cipher comparison [6]	Comparative FPGA evaluation	Throughput and area benchmarking
Ma and Guan [9]	Hardware unrolling of eSTREAM ciphers	Parallelism and area-efficiency trade-offs
Yang *et al.* [10]	Differential attacks on reduced-round SNOW 3G	Security implications of internal nonlinear functions
Dridi *et al*. [11]	Vectorized linear approximations	FSM-related cryptanalytic evaluation
Moraitis and Dubrova [17]	Integral distinguishers	Security evaluation of reduced-round variants
Ordóñez *et al*. [18]	FPGA bitstream modification attack	Hardware configuration and key protection

Table 1 demonstrates that the available literature addresses three connected dimensions: functional architecture, hardware efficiency, and implementation security. Direct hardware studies provide information about arithmetic and S-box optimization, while comparative studies help explain the effects of unrolling and resource allocation [2], [3], [6], [9]. Cryptanalytic studies are also relevant because they examine the internal transformations that the hardware must preserve [10], [11], [17]. Finally, FPGA bitstream research shows that protecting the configuration and deployment environment is as important as implementing the cipher correctly [18].

6. SECURITY AND IMPLEMENTATION CONSIDERATIONS

The security of SNOW 3G should be assessed separately from the efficiency of its hardware implementation. Cryptanalysis of reduced-round variants does not automatically imply that the full standardized cipher is practically broken. Ma and Guan investigated differential attacks on reduced-round SNOW 3G, while Yang *et al*. [10] examined vectorized linear approximations of its FSM, Dridi *et al*. [11]. Moraitis and Dubrova [17] further studied practical integral distinguishers for reduced-round SNOW 3G, illustrating the continuing importance of evaluating the cipher's internal structure.

Physical implementation introduces additional risks that are not captured by conventional mathematical cryptanalysis. FPGA configuration bitstreams may reveal design information or permit modifications if they are not adequately protected. Moraitis and Dubrova demonstrated a bitstream-modification attack against an unprotected FPGA implementation of SNOW 3G, showing that hardware configuration security can directly affect key confidentiality [18]. Side-channel leakage, fault injection, and insecure key-loading procedures should therefore be considered during system design [19], [20].

A secure implementation should protect secret keys throughout loading, storage, processing, and output handling. Designers should consider constant-time or leakage-aware logic where appropriate, secure configuration mechanisms, fault detection, and access control for debugging interfaces. These countermeasures should be evaluated without altering the standardized keystream-generation behavior. Functional conformance testing and cryptographic test vectors remain necessary to verify that optimization has not introduced implementation errors [7], [20].

7. PERFORMANCE EVALUATION AND RESEARCH GAPS

A comprehensive evaluation of SNOW 3G hardware should include throughput, maximum frequency, area utilization, power consumption, initialization latency, and energy per processed bit. Throughput alone is insufficient because a high-speed architecture may require excessive logic resources or power. Comparisons should also identify whether the reported design implements standard SNOW 3G or a modified variant. This information is necessary to ensure that performance claims are technically comparable and cryptographically meaningful.

The literature still lacks fully standardized benchmarking conditions across different FPGA and ASIC implementations. Results obtained from different devices, synthesis tools, clock constraints, and implementation strategies cannot be compared directly without normalization. Researchers should report the target device, logic utilization, register count, memory usage, maximum frequency, throughput calculation, and power-measurement methodology. Releasing synthesizable source code and verification vectors would further improve reproducibility and independent evaluation [6], [9], [21].

Future research should focus on hardware architectures that combine high throughput with low energy consumption and strong implementation protection. Resource sharing may reduce area in constrained devices, while carefully selected parallelism can improve throughput in high-speed communication equipment. Side-channel countermeasures and secure FPGA configuration should be evaluated as part of the complete architecture rather than added after performance optimization. Such a system-level approach would provide a more realistic basis for deployment in communication equipment.

8. CONCLUSION

SNOW 3G is based on a structured combination of a 16-stage LFSR and a nonlinear FSM, providing a word-oriented keystream generator for mobile communication security. Its hardware implementation requires coordinated optimization of state storage, finite-field feedback, modular arithmetic, nonlinear substitution functions, and initialization control. Published FPGA studies demonstrate that carry-lookahead arithmetic and optimized S-box structures can improve implementation performance, while comparative stream-cipher research highlights the trade-offs associated with unrolling and parallel processing. However, hardware efficiency must not be achieved by modifying standardized algorithmic

behavior without a separate security and conformance assessment. The practical quality of a SNOW 3G implementation depends on more than its maximum throughput. Area, power, initialization latency, side-channel leakage, fault resistance, and configuration security all contribute to the reliability of the final hardware. Future studies should adopt reproducible benchmarking procedures and evaluate cryptographic correctness alongside physical implementation characteristics. These efforts can support efficient and secure SNOW 3G hardware for communication systems and embedded security applications.

REFERENCES

[1] 3GPP, "Specification of the 3GPP confidentiality and integrity algorithms UEA2 & UIA2; Document 5: Design and evaluation report," no. 35.919, 2006, [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/35919.htm.

[2] N. B. Hulle, B. Prathiba, S. R. Khope, K. Anuradha, Y. Borole, and D. Kotambkar, "Optimized architecture for SNOW 3G," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 1, pp. 545–557, 2021, doi: 10.11591/ijece.v11i1.pp545-557.

[3] M. Madani, I. Benkhaddra, C. Tanougast, S. Chitroub, and L. Sieler, "FPGA implementation of an enhanced SNOW-3G stream cipher based on a hyperchaotic system," *2017 4th International Conference on Control, Decision and Information Technologies, CoDIT 2017*, vol. 2017-January, pp. 1168–1173, 2017, doi: 10.1109/CoDIT.2017.8102758.

[4] M. Bahadori, K. Järvinen, and V. Niemi, "FPGA implementations of 256-bit SNOW stream ciphers for postquantum mobile security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 11, pp. 1943–1954, 2021, doi: 10.1109/TVLSI.2021.3108430.

[5] S. Nandi, S. Krishnaswamy, and P. Mitra, "Recent results on some word oriented stream ciphers: SNOW 1.0, SNOW 2.0 and SNOW 3G," *Information Security and Privacy in the Digital World - Some Selected Topics*, 2023, doi: 10.5772/intechopen.105848.

[6] P. Kitsos, N. Sklavos, G. Provelengios, and A. N. Skodras, "FPGA-based performance analysis of stream ciphers ZUC, Snow3g, Grain V1, Mickey V2, Trivium and E0," *Microprocessors and Microsystems*, vol. 37, no. 2, pp. 235–245, 2013, doi: 10.1016/j.micpro.2012.09.007.

[7] R. Ananth, P. R. M. V. Rao, and N. S. Ramaiah, "An efficient Grain-80 stream cipher with unrolling features to enhance the throughput on hardware platform," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 218–226, 2024, doi: 10.11591/ijeecs.v33.i1.pp218-226.

[8] F. Alharbi, M. K. Hameed, A. Chowdhury, A. Khalid, A. Chattopadhyay, and I. T. Javed, "Analysis of area-efficiency vs. Unrolling for eSTREAM hardware portfolio stream ciphers," *Electronics (Switzerland)*, vol. 9, no. 11, pp. 1–17, 2020, doi: 10.3390/electronics9111935.

[9] S. Ma and J. Guan, "Differential attacks on reduced-round SNOW 3G and SNOW 3G⊕," *IET Information Security*, vol. 14, no. 5, pp. 587–594, 2020, doi: 10.1049/iet-ifs.2019.0478.

[10] J. Yang, T. Johansson, and A. Maximov, "Vectorized linear approximations for attacks on SNOW 3G," *IACR Transactions on Symmetric Cryptology*, vol. 2019, no. 4, pp. 249–271, 2019, doi: 10.13154/tosc.v2019.i4.249-271.

[11] F. Dridi, S. El Assad, W. El Hadj Youssef, M. Machhout, and R. Lozi, "The design and fpga-based implementation of a stream cipher based on a secure chaotic generator," *Applied Sciences (Switzerland)*, vol. 11, no. 2, pp. 1–19, 2021, doi: 10.3390/app11020625.

[12] P. Kitsos, N. Sklavos, and A. N. Skodras, "An FPGA implementation of the ZUC stream cipher," *Proceedings - 2011 14th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, DSD 2011*, pp. 814–817, 2011, doi: 10.1109/DSD.2011.109.

[13] P. Kitsos and O. Koufopavlou, "An FPGA-based implementation of the pomaranch stream cipher," 2007, doi: 10.4108/icst.mobimedia2007.1679.

[14] S. Y. A. M. Fauzi, M. Othman, F. M. M. Shuib, and K. Seman, "Hardware implementation of modified A5/1 stream cipher," *International Journal of Computer Theory and Engineering*, vol. 9, no. 5, pp. 385–389, 2017, doi: 10.7763/ijcte.2017.v9.1171.

[15] M. Ubaidullah Bokhar, S. Alam, and S. Hamid Hasan, "A detailed analysis of grain family of stream ciphers," *International Journal of Computer Network and Information Security*, vol. 6, no. 6, pp. 34–40, 2014, doi: 10.5815/ijcnis.2014.06.05.

[16] J. Hoki, K. Sakamoto, K. Minematsu, and T. Isobe, "Practical integral distinguishers on SNOW 3G and KCipher-2," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E104.A, no. 11, pp. 1603–1611, 2021, doi: 10.1587/transfun.2020EAP1102.

[17] M. Moraitis and E. Dubrova, "Bitstream modification attack on SNOW 3G," *Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition, DATE 2020*, pp. 1275–1278, 2020, doi: 10.23919/DATE48585.2020.9116222.

[18] F. E. Potestad-Ordóñez, M. Valencia-Barrero, C. Baena-Oliva, P. Parra-Fernández, and C. J. Jiménez-Fernández, "Breaking trivium stream cipher implemented in asic using experimental attacks and DFA," *Sensors (Switzerland)*, vol. 20, no. 23, pp. 1–19, 2020, doi: 10.3390/s20236909.

[19] F. E. Potestad-Ordonez, E. Tena-Sanchez, J. M. Mora-Gutierrez, M. Valencia-Barrero, and C. J. Jimenez-Fernandez, "Trivium stream cipher countermeasures against fault injection attacks and DFA," *IEEE Access*, vol. 9, pp. 168444–168454, 2021, doi: 10.1109/ACCESS.2021.3136609.

[20] A. Khalid *et al.*, "RunStream: A high-level rapid prototyping framework for stream ciphers," *ACM Transactions on Embedded Computing Systems*, vol. 15, no. 3, 2016, doi: 10.1145/2891412.

[21] S. A. Qassir, M. T. Gaata, and A. T. Sadiq, "Modern and lightweight component-based symmetric cipher algorithms," *ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY*, vol. 10, no. 2, pp. 152–168, Dec. 2022, doi: 10.14500/aro.11007.