Bulletin of Electrical Engineering and Informatics
Vol. 14, No. 1, February 2025, pp. 388~397
ISSN: 2302-9285, DOI: 10.11591/eei.v14i1.8270

Hardware-based efficient Mickey-128 stream cipher with unrolling factors for throughput enhancement

Raghavendra Ananth[1], Panduranga Rao Malode V.[2], Narayana Swamy Ramaiah[3]
[1]Department of Electronics and Communications, JAIN Deemed to be University, Bengaluru, India
[2]Department of Computer Science Engineering, Faculty of Engineering and Technology, JAIN Deemed to be University, Bengaluru, India
[3]Department of Computer Science Engineering, New Horizon College of Engineering (NHCE), Bengaluru, India

Article Info

Article history:

Received Feb 3, 2024
Revised Aug 17, 2024
Accepted Aug 28, 2024

Keywords:

Keystream
Mickey-128
Stream cipher
Throughput
Unrolling

ABSTRACT

The emerging trend known as "ubiquitous computation" aims to incorporate intelligent gadgets into commonplace items. The lightweight cryptographic techniques are being researched and developed to minimize the gadgets' resources and a perpetual desire to reduce production expenses. A key element of symmetric cryptography, the stream cipher has unique benefits in terms of scalability as well as performance. The Mickey-128 stream cipher is designed and implemented in this manuscript. Additionally, unrolling features are incorporated with Mickey-128 cipher to enhance the throughput. The Mickey-128 contains a 128-bit key, an initialization vector (IV), and two clocking registers (R and S) with mapping units. The finite state machine (FSM) controller initializes and controls the key, IV and RS-registers data. The proposed Mickey-128 cipher runs on an Artix-7 field programmable gate array (FPGA) at 639.1 MHz and uses less than 1% of the chip's area (Slices). For unrolling factors 8 and 16, the Mickey-128 cipher achieves a throughput of 5.12 Gbps and 10.23 Gbps, accordingly. Finally, a comparison is made between the proposed Mickey-128 cipher and the existing ciphers' better hardware efficiency and throughput.

Corresponding Author:

Raghavendra Ananth
Department of Electronics and Communications, JAIN Deemed to be University
560069 Bengaluru, Karnataka, India
Email: araghavendra.research@gmail.com

1. INTRODUCTION

The employment of stream ciphers in Wi-Fi, Bluetooth, and other wireless networks shows the significance of their structure to mobile communication. Stream ciphers' primary benefits are their simplicity of use and absence of mistake transmission. Stream ciphers can be coded into software, application specific integrated circuits (ASICs), or field programmable gate arrays (FPGAs). These require a simple encryption method, an approach for generating beginning circumstances, and in certain situations, key scheduling. A function of authenticity may also be present in certain instances of these [1], [2]. The stream cipher is a symmetric cipher that uses the exclusive-or function to integrate the key stream—a collection of securely safe bits—with the plaintext or ciphertext at each bit level [3]. Lightweight block and stream ciphers are anticipated to provide privacy and safety and function effectively in embedded systems. Since block ciphers have well-established safety features and simpler architecture, they are the most popular option in lightweight cryptography (LWC). Additionally, stream ciphers cannot be used to establish every communication technique. The beginning phase, which must occur before any exchange of information can occur, is the primary disadvantage of stream ciphers. Stream ciphers, in contrast, are appropriate for uses in which the

plaintext size is unidentified or constant, such as network streams. These include military applications in which the stream cipher is fed to machines that are anticipated to function in resentful and uneasy circumstances while being used in an encrypted way [4], [5].

Mickey is designed for hardware architectures with limited resources. Its goal is to offer an exceptional level of protection with a minimum of hardware complications. It employs erratic shift register clocking and other cutting-edge methods to compromise between preventing specific cryptanalytic assaults and requiring time and pseudo-randomness assurances [6]. The fundamental goal of a power assessment assault is to decode a security device's key by examining its power usage. This method was further developed and used in numerous articles to block ciphers and public key cryptosystems [7], [8]. It's important to note that the Estream hardware portfolio's 3 ciphers (Grain, Mickey, and Trivium) are all clearly made to handle theoretically very lengthy keystream patterns for each key/IV pair. On the other hand, the data transfer protocols employ substantially reduced packet sizes [9], [10]. To improve the expense, power, and complexity needed for tags with radio frequency identification (RFID), a working prototype mechanism known as the secure data extractor (SDE) is constructed employing the Mickey 2.0 cipher and performs an encrypted key/IV transfer [11].

The existing works of the current stream ciphers for various application usage and their performance metrics are discussed as: three estream portfolio ciphers, Mickey, Grain, and Trivium, and two stream encryption algorithms, Plantlet and Lizard, are presented by Li *et al.* [12]. Every version was developed with various technology usage in mind. Efficiency, area usage, and efficiency are among the outcome indicators for comparing these approaches and those found in the available research. The two unique condensed frameworks of the Enocoro-128v2 stream encryption algorithm are explained and debated by Pyrgas and Kitsos [13]. The datapath in the initial version is 8 bits, whereas the datapath in the subsequent version is 4 bits. The Enocoro-128v2 8-bit design's circuitry operates at 189 MHz frequency, with a speed of 302 Mbps. The enhanced cube assault is used by He *et al.* [14] to clarify the protection of stream cryptography and verified cryptography codes-for instance, the largest-round distinguisher results from the cube's quality identification approach. The logical level of the non-linear feedback shift register (NFSR) based encryption system can be estimated using the mathematical translation approach and a greedy strategy for discovering cubes. Additionally, utilizing the flag approach and the satisfiability (SAT) theory of partition assets, the work assesses the safety of multiple primitives used in cryptography versus the cube assault, potentially improving the accuracy of partition attribute transmission. Zhu *et al.* [15] describe a very area-efficient concurrent programmable symmetric cipher streaming system that combines streaming structures' benefits with unilateral cryptograph flow-state computation features. This machine features multifaceted concurrency and an ascending streaming design. It allows for concurrency between method execution and information routing by severing the connection between encryption calculation and information transfer. It has greater storage efficiency and promising potential uses than other encryption machines.

An impartial, external investigation of Grain-128 authenticated encryption with associated data (AEAD) resistance to malfunction assaults is presented by Salam *et al.* [16]. They investigate whether three distinct divergent failure attack approaches can be used against Grain-128AEAD. Grain-128AEAD can be restored to its starting condition by any of those assaults. A method for modelling less strict accuracy, like medium and insufficient control that are more useful than exact authority, is also introduced in this work. To enhance the conventional Zu Chongzhi (ZUC) algorithm, Guang *et al.* [17] provide an efficient lightweight ZUC (LZUC) encryption that utilises a chaotic structure. An additional mechanism is created in the following manner to include chaos into the compact ZUC. The study used data entropy assessment on the results of its key streams and National Institute of Standards and Technology (NIST) statistical evaluation to confirm the LZUC cipher's functionality. The common assaults on the method's resilience have been addressed in the study. A few of the most significant power evaluation assaults on modern hardware stream encryption algorithms, including Mickey, Grain, and Trivium, are explained by Silva *et al.* [18]. The study concludes that, albeit employing conventional techniques, new strategies like artificial intelligence were employed to carry out the assaults more successfully. Chen *et al.* [19] present a robust restricted encryption approach to increase the resilience of video security. The suggested strategy is coordinated with slices. As a result, it facilitates virtual instantaneous communication and usually decodes if a packet is lost.

Xiao *et al.* [20] describe strategies to optimise backup and light interruption techniques, create a graphics processing unit-central processing unit (GPU-CPU) transmission engine with continuous information intake utilising this technique, and install an advanced encryption system (AES) and ShāngMì4 (SM4) technique on compute unified device architecture (CUDA) using a Python context. The overall pipeline encrypting throughput for SM4 and AES is 1,001 Mbps and 210 Mbps, respectively, with an additional time overhead of 19% and 4%. A unique picture encryption constructed using two chaos maps with a rectangle serving as a scrambler to improve the speed of computation is explained in [21]. They have generated arbitrary numbers using two chaos maps of Hénon and an individually regular chaos map. Additionally, the work achieved a data entropy of 7.9975 and a computing time of 0.5156 seconds. The

enhanced Chacha20 (EC-20) stream cipher, which uses addition-rotation-XOR (ARX) with six rotating constants, is presented by Kebande *et al.* [22]. To evaluate the effectiveness of EC-20 versus the Chacha20 cipher, the investigation will thoroughly assess the efficiency and consider additional parameters like speed and delays. The ciphertext-only attack against the geostationary equatorial orbit (GEO)-mobile radio interface-2 (GMR-2) system is described by Lee *et al.* [23]. The work uses an innovative technique called pre-filtration to enhance the prior inverted assault. The work repeatedly predicts its contents to retrieve the current session key and launches an assault using the current plaintext.

This article designs a reliable high throughput stream cipher with unrolling features based on the Mickey-128 algorithm. The following highlights the contribution of the suggested Mickey-128 cipher: the proposed Mickey-128 cipher uses the register balancing optimization technique to get around the frequency and throughput limitations of the traditional Mickey-128 cipher. In the proposed Mickey-128 cipher, unrolling features are employed as one of the optimization strategies in place of pipelining to increase throughput while requiring little additional resources. The simulation findings are confirmed and validated concerning the theoretical values of the original Mickey-128 cipher. The proposed cipher outperforms the current Mickey-128 cipher techniques regarding performance metrics, including frequency, throughput, and hardware efficiency. The manuscript's organization is as follows: the proposed Mickey-128 stream cipher hardware architecture is described in detail with unrolling features or factors in section 2. The proposed work results are realized with tabulation and performance comparison in section 3. Lastly, it concludes the overall work with improvement with a futuristic scope in section 4.

2. PROPOSED MICKEY-128 STREAM CIPHER

The Mickey-128 stream cipher is also known as the mutual irregular clocking keystream generator, which offers better security with lower complexity and is suitable for low-constrained environments. The Mickey-128 version-1 was initially developed by Babbage and Dodd [24]. The Mickey-128 generates random clocking using Shift registers with suitable operations to maintain randomness over specific attacks. The Mickey-128 supports Key (K) size of 128-bit and 128-bit initialization vector (IV). The Mickey-128 cipher uses two register sets (R and S), and each register is 128-bit length. The R acts as a linear register, and S acts as a non-linear register and is represented as R_0 ... R_{127} and S_0 ... $S_{127,}$ respectively. The hardware architecture of Mickey-128 stream cipher is illustrated in Figure 1. It has two clock registers (CLK-R and CLK-S), a mapping unit and a finite state machine (FSM) controller. The FSM controller initializes key and IV and Key stream generation. The clock RS generator uses two clock registers, providing a pseudorandom feature based on clocking. The RS mapping unit provides the synchronization mechanism between two clock registers. The RS mapping unit provides the updated output (R and S) are input (feedback) to the clock RS generator. The typical flow of the Mickey-128 stream cipher is represented in Algorithm 1.

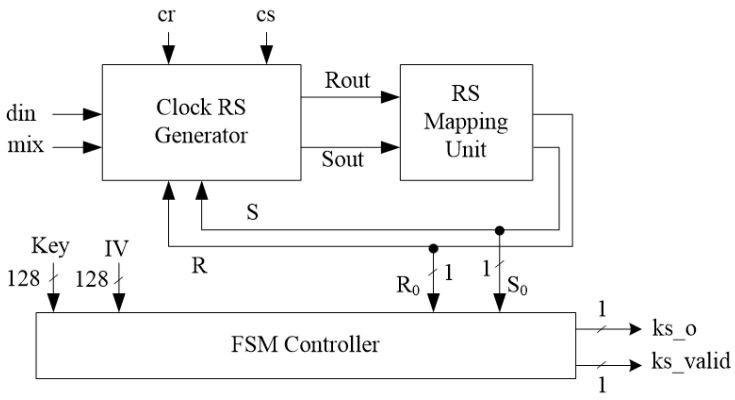

Figure 1. Hardware architecture of Mickey-128 cipher

The hardware architecture of the clock RS generator is illustrated in Figure 2. The R and S registers are updated using register balancing approach, which optimizes the throughput and efficiency of the cipher. It has an RS input generator, R-clocking, and S-clocking register units (CLK-R and CLK-S). The 1-bit input signal (din) and mixing (mix) signal are used in the RS input generator, which generates the actual input R

and S to the RS-clocking units. When the mix signal is active, then the Updated R (rin) or S (sin) value, else actual R or S value, is considered for the key generation process.

Algorithm 1. Mickey-128 stream cipher
Input: 128 − bit $Key(K), IV$;
Output: ks_o *and* ks_valid ;
Define: Control Inputs (din, mix) are generated using FSM;

- ***Clock RS Generator***:
  1. Control Bit generation cr and cs for RS using Eq. (1) ;
  2. Feedback Bit Generation F_r and F_s for RTAPs using Eq. (2) ;
  3. ***RS Input*** : ***if*** (mix = 0) ***then*** $rin_n = din$ ***else*** $din \oplus S_{64}$; $sin_n = din;\ for\ 1 \le n \le 127;$
  4. ***R clocking***:
     - $Rout_0 = 0\ and\ Rout_n = rin_{n-1}$; $for\ 1 \le n \le 127;$
     - ***if*** $(n \in RTAPS)$ ***then*** $Rout_n = Rout_n \oplus F_r$; $for\ 0 \le n \le 127;$ `// R-Tapping`
     - ***if*** (cb = 0) ***then*** $Rout_n = rin$ ***else*** $Rout_n \oplus rin;\ for\ 0 \le n \le 127;$ `// R-Generation`
  5. ***S clocking***:
     - $SC_0 = 0;\ SC_{127} = SC_{126};$ `// S-Component`
     - $SC_n = sin_{n-1} \oplus \big((sin_n \oplus CMP0_n)(sin_{n+1} \oplus CMP1_n)\big);\ for\ 1 \le n \le 126;$
     - ***if*** (cb = 0) ***then*** `// S-Generation`
       $Sout_n = SC_n \oplus (FB0_n.F_s);\ for\ 0 \le n \le 127;$
       else
       $Sout_n = SC_n \oplus (FB1_n.F_s);\ for\ 0 \le n \le 127;$
- ***RS Mapping***:
  $R = Rout_n ; S = Sout_n ;\ for\ 0 \le n \le 127;$
- ***FSM Controller***:
  $(ks\_o, ks\_valid) = FSM(Key, IV, R, S)$

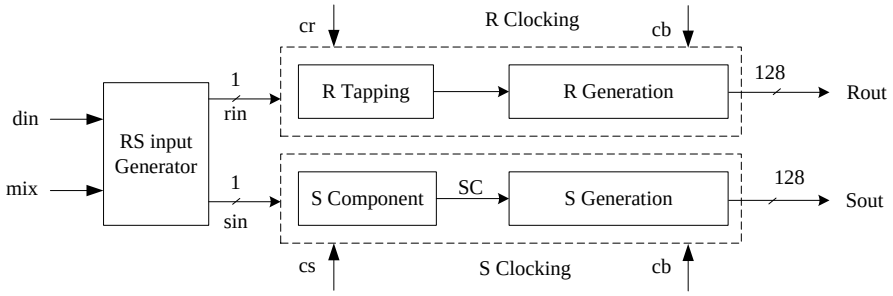

Figure 2. Hardware architecture of clock RS generator

The R-clocking unit performs the R- tapping (RTAPS) followed by R generation operations. The control bits (cr and cs) for RS are generated using (1), and the feedback bit (F_r) for RTAPS and feedback bit (F_s) for S-generation are obtained using (2). So, define a range of R's feedback tap placements using (3). Each placement of the RTAPS performs the XOR operation with a feedback bit (F_r). The R-generation unit considers only the updated R-tap output as the final R-clocking output (Rout) only when the control bit (cb) is active.

$$cr = S_{43} \oplus R_{85}\,; cs = S_{85} \oplus R_{42}\,; \tag{1}$$

$$F_r = R_{127} \oplus din\,; F_s = S_{127} \oplus din; \tag{2}$$

$$RTAPS = \{0,1,2,3,6,7,9,11,12,13,14,15,23,26,27,28,31,32,35,37,40,43,47,48,52,54,55,61,62,63,64,69,70,71,73,74,76,77,78,81,82,83,84,85,87,89,90,94,95,96,100,102,104,106,107,108,110,114,115,120,121,124,125,126\}; \tag{3}$$

The S-clocking operation contains S-component and S-generation units. The S-component receives the 'sin' input from the RS-input generator and performs XOR operations with 127-bit CMP0 and CMP1

coefficients. The coefficients CMP0 and CMP1 are defined using (4). The S-generation receives the S-component output (SC) and generates the S-clocking output (Sout) using feedback bit (F_s) and 128-bit feedback coefficients (FB0 and FB1). The feedback coefficients FB0 and FB1 are defined using as (4) and (5):

$$CMP0 = 7D27B5DD5548324F2307004F464DFAF6;$$
$$CMP1 = 467C4C5F0C9E36BF83E1801F51638333; \quad (4)$$

$$FB0 = F5F83C2344C5F47081B2A76689D22A2B;$$
$$FB1 = \text{D5EE2FD909319E0E6D1859F6E77ED236}; \quad (5)$$

The RS-mapping unit receives the updated Rout and Sout from the RS-clocking unit. It is stored in a temporary register for one clock cycle to maintain RS-clocking synchronization. The stored values are considered as final R and S output. These 128-bit R and S values are again feedback to the clock RS-Generator unit. The FSM controller receives the least significant bit (LSB) of R and S values like R_0 and S_0, respectively. The FSM controller has two states (idle and run) and runs based on a counting mechanism. In an idle state, the input (din) and control signals (mix) are reset to zero by setting the counter to one (cnt=1). The IV value is allocated as input for the first 128 clock cycles in the run state. Next, the Key (K) values are allocated as input (din) for the subsequent 128 clock cycles. The Key and IV values are updated for the subsequent 128 clock cycles till the keystream valid (ks_valid) signal becomes high. So, the final keystream output (ks_o) is generated after 384 clock cycles.

2.1. Mickey-128 cipher with unrolling features

The architecture of the Mickey-128 cipher with unrolling features for N-times (N x) is illustrated in Figure 3. The architecture contains a clock RS generator and RS mapping Unit with N-times and FSM controller. Where 'N' is an unrolling factor. This work uses a Mickey-128 cipher with N=1, 2, 4, 8, and 16. The RS mapping unit outputs (R and S) are performed repeatedly to accomplish the unrolling procedure. The Mickey-128 cipher output is anticipated to be one bit per clock since the clock RS generator and mapping outputs (R and S) are updated per clock cycle. With more resources (chip area), the Mickey-128 cipher can operate faster or with higher throughput. The architecture only repeats (N-1) times the clock RS generator and RS mapping units using the same FSM controller based on unrolling factors (N).

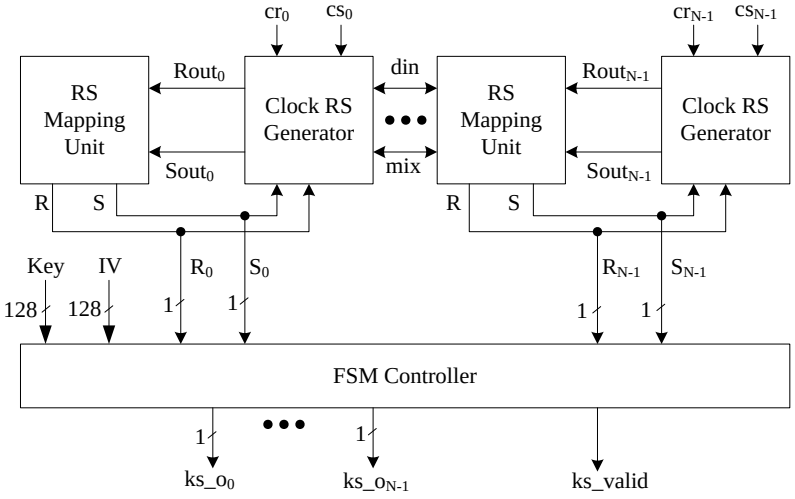

Figure 3. The architecture of the Mickey-128 cipher with unrolling features

In the Mickey-128 cipher, the procedure continues as the unrolling factor (N) rises. For each "N" unrolling factor in the architecture, the control signals (cr and cs) and intermediate outputs (Rout and Sout) are updated N-1 times. Keystream outputs such as ks_o_0 ... ks_o_{N-1} are provided by the FSM controller per the unrolling factor (N). The 256 CC is divided by an unrolling factor (N) to reduce the key initialization (Key and IV) procedure. Therefore, this strategy is suitable for increasing throughput.

3. RESULTS AND DISCUSSION

The results of the Mickey-128 cipher with unrolling features are discussed in this section. The design modules are synthesized on the Xilinx environment using Verilog-HDL and implemented on Artix-7 FPGA. The simulation results are carried out using modelsim simulator from mentor graphics tool. The synthesis results concerning resource utilization are generated after place and route operation for Mickey-128 cipher. The performance metrics like latency, throughput (Mbps) and hardware efficiency are generated with the help of simulation and synthesis results. Lastly, the Mickey-128 cipher performance metrics are compared with existing Mickey-128 ciphers with improvements. The Mickey-128 cipher simulation results are illustrated in Figure 4. The global clock (clk) is activated with a lower reset (rst) to initiate the Mickey-128 operation and define the 128-bit key and IV values. The Mickey-128 cipher takes 385 clock cycles (CC) to generate the output of a 128-bit key stream (key_output). The first 128-CC is for IV, and the next 128-CC is for key initialization. The key-stream valid signal (ks_valid) is activated after the initialization (IV and key) operation. The last 128-CC is used for the actual keystream generation process by activating the done signal (ks_done) signal.

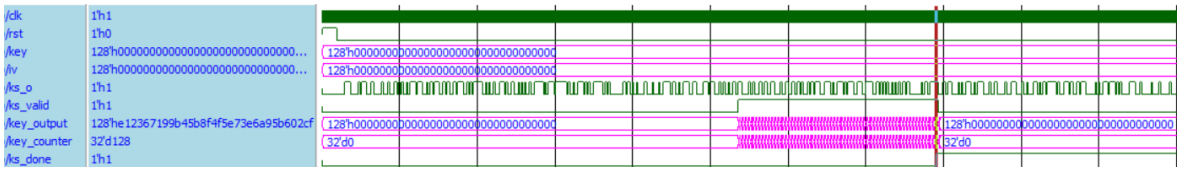

Figure 4. Simulation results of Mickey-128 stream cipher

The design summary of the Mickey-128 stream cipher is tabulated in Table 1. The Mickey-128 cipher uses slices of 204, lookup tables (LUTs) of 268 and operates at 639.1 MHz frequency by consuming the total power of 88 mW on Artix-7 FPGA. The Mickey-128 cipher uses only 1 CC as latency to perform either encryption or decryption after keystream generation. The throughput calculation depends on the Latency and frequency of the design module. The Mickey-128 cipher works with a throughput of 639.1 Mbps with an efficiency of 3.133 Mbps/Slice on Artix-7 FPGA.

Table 1. Design utilization of Mickey-128 cipher on Artix-7 FPGA

Resources	Utilization
Slices	204
LUTs	268
Max. frequency (MHz)	639.1
Total power (mW)	88
Key generation (CC)	385
Latency (CC)	1
Throughput (Mbps)	639.1

The performance summary of the Mickey-128 cipher with unrolling factors is tabulated in Table 2. The unrolling factors are used to speed up the system. The Mickey-128 cipher with 2X speed uses the slices of 402, by consuming 94 mW of power with a throughput of 1.279 Gbps and an efficiency of 3.17 Mbps/Slice. Similarly, the Mickey-128 cipher with 4X speed uses the slices of 804, consuming 106 mW of power with a throughput of 2.557 Gbps and an efficiency of 3.18 Mbps/Slice.

Table 2. Performance summary of Mickey-128 cipher with unrolling factors

Resources	Unrolling factors				
	1X	2X	4X	8X	16X
Slices	204	402	804	1608	3218
LUTs	268	537	1073	2146	4290
Max. frequency (MHz)	639.1	639.1	639.1	639.1	639.1
Total power (mW)	88	94	106	132	184
Throughput (Gbps)	0.639	1.279	2.557	5.113	10.226
Efficiency (Mbps/Slice)	3.133	3.17	3.18	3.18	3.18

The Mickey-128 cipher with 8X speed uses slices 1,608, consuming 132 mW of power with a throughput of 5.113 Gbps and an efficiency of 3.18 Mbps/Slice. Lastly, the Mickey-128 cipher with 16X speed uses slices 3218, consuming 184 mW of power with a throughput of 10.226 Gbps and an efficiency of 3.18 Mbps/Slice on Artix-7 FPGA. The chip area utilization of Mickey-128 cipher with unrolling factors is represented in Figure 5. The Mickey-128 with 1X and 2X uses <1% chip area (Slices and LUTs), with 4X speed uses 1%, 8X uses <2%, and 16X utilizes <3% of chip area on Artix-7 FPGA.

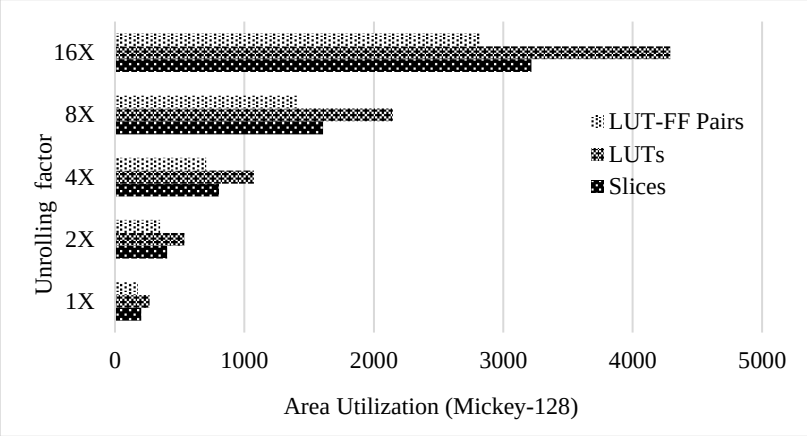

Figure 5. Chip area utilization of Mickey-128 cipher with unrolling factors

The throughput realization of Mickey-128 cipher at various FPGAs is illustrated in Figure 6. The FPGAs like low-resource Spartan-6, low-cost Artix-7, and high-volume Virtex-7 are considered throughput realization purposes. The Mickey-128 cipher operates at 327 MHz, 639 MHZ, and 852 MHz on Spartan-6, Artix-7, and Virtex-7 FPGA, respectively. As the unrolling factor increases, the throughput also increases. The Mickey-128 cipher obtains a throughput of 5.23 Gbps, 10.23 Gbps, and 13.64 Gbps on Spartan-6, Artix-7, and Virtex-7 FPGA. The throughput value is varied based on the FPGA device and unrolling factors. The highest throughput is obtained at the high-end device (Virtex-7) at a maximum unrolling factor (16X).

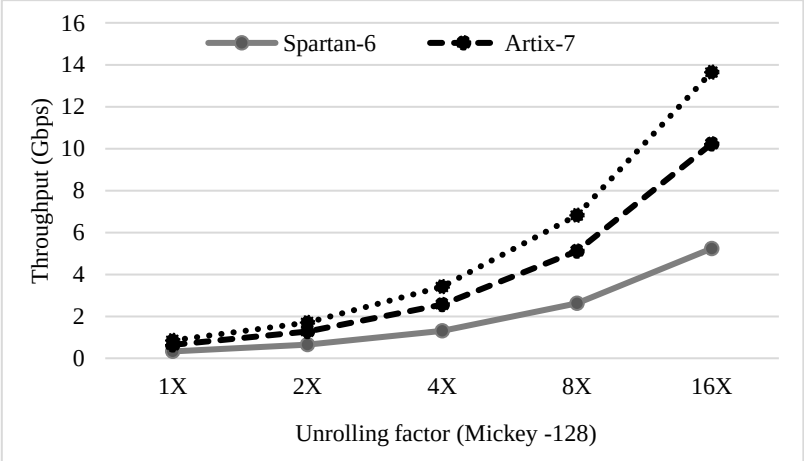

Figure 6. Throughput realization of Mickey-128 cipher at various FPGAs

The resource comparison of proposed Mickey-128 with existing Mickey-128 ciphers is tabulated in Table 3. The FPGA device type, chip area (Slices), latency, throughput usage, and hardware efficiency are considered for performance comparison. The Mickey stream cipher with a 128-bit key variant is considered for comparative analysis. The Mickey-128 cipher with hardware implementation is constructed on Virtex-2

FPGA [25]. The proposed Mickey-128 cipher provides a better throughput of 48% and efficiency of 37.2% than Mickey-128 [25]. The Target hardware-based ESTREAM cipher-Mickey-128 is implemented on Spartan-3 FPGA [26]. The proposed Mickey-128 cipher provides a better throughput of 31.8% and efficiency of 21.1% than Mickey-128 [26]. The hardware-based Mickey-128 is implemented on Virtex-2 FPGA [27]. The proposed Mickey-128 cipher provides a better throughput of 38.8% and efficiency of 34.7% than Mickey-128 [27]. The FPGA-based Mickey-128 is implemented on Spartan-3 FPGA [28]. The proposed Mickey-128 cipher provides a better throughput of 23.5% and efficiency of 21.7% than Mickey-128 [28]. The hardware profile-based stream cipher Mickey-128 is implemented on Stratix-3 FPGA [29]. The proposed Mickey-128 cipher offers a better chip area of 65.9%, better throughput of 48% and efficiency of 81% than Mickey-128 [29]. The hardware-based stream cipher Mickey-128 is implemented on Spartan-6 FPGA [30]. The proposed Mickey-128 cipher offers a better chip area of 35.9%, better throughput of 2.2% and efficiency of 37.1% than Mickey-128 [30]. The graphical representation of throughput and efficiency with existing Mickey-128 ciphers is illustrated in Figure 7. The Mickey-Cipher [25]-[28] offers better chip area (slices) than the proposed cipher but lags with throughput and efficiency. Overall, the proposed Mickey-128 cipher offers significant Throughput and hardware efficiency than existing Mickey-128 ciphers [25]-[30].

Table 3. Resource comparison of proposed Mickey-128 with existing Mickey-128 ciphers

Ref.	FPGA	Area (Slices)	Latency (CC)	Throughput (Mbps)	Efficiency (Mbps/Slices)
[25]	Virtex-2	167	1	170	1.011
[26]	Spartan-3	176	1	223	1.27
[27]	Virtex-2	190	1	200	1.05
[28]	Spartan-3	198	1	250	1.26
[29]	Stratix-3	596	1	170	0.286
[30]	Spartan-6	317	1	320	1.01
This work	Spartan-6	203	1	327	1.61

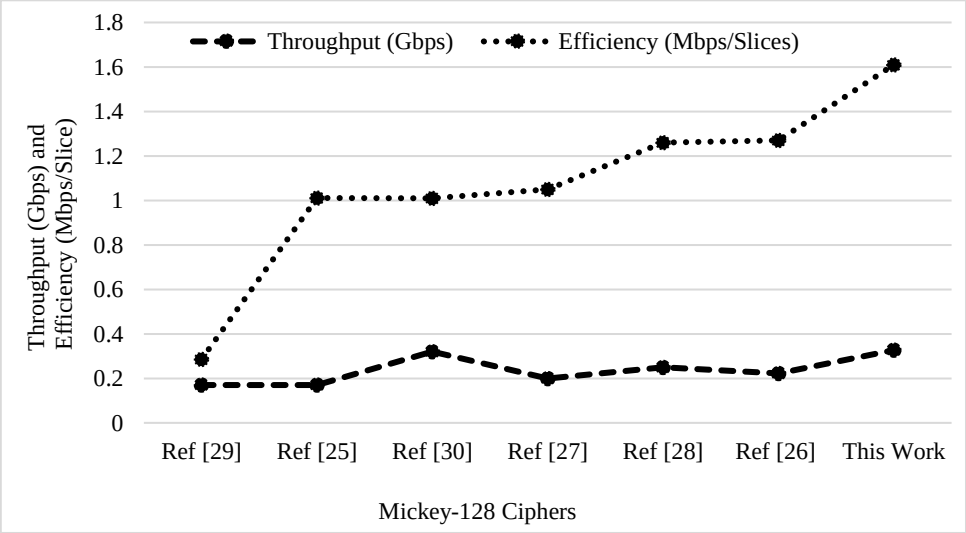

Figure 7. Performance comparison of Mickey-128 ciphers

4. CONCLUSION

This manuscript describes the design and implementation of the Mickey-128 stream cipher with unrolling features on the hardware environment. A mapping unit, an FSM controller, and two clock registers (R and S) are used in the construction of the Mickey-128 cipher. The Mickey-128 cipher uses the unrolling function as an optimization technique to increase system throughput while requiring minimal additional resources. The Mickey-128 cipher's simulation outcomes are confirmed and validated using theoretical parameters. The suggested Mickey-128 cipher runs at 639.1 MHz on an Artix-7 FPGA with a power consumption of 88 mW and a chip area of less than 1% (slices and LUTs). Employing an unrolling factor of 16, the Mickey-128 cipher runs at 10.23 Gbps with an efficiency of 3.18 Mbps/Slice. The efficiency, throughput, and frequency metrics of the proposed Mickey-128 cipher are compared to those of other Mickey-128 ciphers. Security analysis about the anticipated Mickey-128 cipher's assaults will be assessed in the future.

ACKNOWLEDGEMENTS

The authors acknowledge the support and research facilities provided by the Department of Electronics and Communication Engineering, JAIN Deemed to be University, Bangalore, India.

REFERENCES

[1] T. Good, W. Chelton, and M. Benaissa, "Review of stream cipher candidates from a low resource hardware perspective," *SASC 2006 - Stream Ciphers Revisited*, vol. 125, pp. 125–148, 2006.

[2] M. Kocheta, N. Sujatha, K. Sivakanya, R. Srikanth, S. Shetty, and P. V. A. Mohan, "A review of some recent stream ciphers," in *2013 International Conference on Circuits, Controls and Communications, CCUBE 2013*, IEEE, Dec. 2013, pp. 1–6, doi: 10.1109/CCUBE.2013.6718558.

[3] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and Y. Papaefstathiou, "A survey of lightweight stream ciphers for embedded systems," *Security and Communication Networks*, vol. 9, no. 10, pp. 1226–1246, Jul. 2016, doi: 10.1002/sec.1399.

[4] L. Jiao, Y. Hao, and D. Feng, "Stream cipher designs: a review," *Science China Information Sciences*, vol. 63, no. 3, p. 131101, Mar. 2020, doi: 10.1007/s11432-018-9929-x.

[5] O. Nariezhnii, E. Eremin, V. Frolenko, K. Chernov, T. Kuznetsova, and I. Chepurko, "Research of statistical properties of stream symmetric ciphers," in *2018 International Scientific-Practical Conference on Problems of Infocommunications Science and Technology, PIC S and T 2018 - Proceedings*, IEEE, Oct. 2018, pp. 696–700, doi: 10.1109/INFOCOMMST.2018.8632138.

[6] M. Li, J. Xu, Z. Dai, X. Yang, and H. Qu, "Research and implementation of parallel and reconfigurable MICKEY algorithm," in *ASICON 2009 - Proceedings 2009 8th IEEE International Conference on ASIC*, IEEE, Oct. 2009, pp. 831–834, doi: 10.1109/ASICON.2009.5351567.

[7] J. Liu, D. Gu, and Z. Guo, "Correlation power analysis against stream cipher MICKEY v2," in *Proceedings - 2010 International Conference on Computational Intelligence and Security, CIS 2010*, IEEE, Dec. 2010, pp. 320–324, doi: 10.1109/CIS.2010.75.

[8] A. Chakraborty and D. Mukhopadhyay, "A practical template attack on MICKEY-128 2.0 using PSO generated IVs and LS-SVM," in *Proceedings of the IEEE International Conference on VLSI Design*, IEEE, Jan. 2016, pp. 529–534, doi: 10.1109/VLSID.2016.66.

[9] M. Hamann, M. Krause, and W. Meier, "LIZARD – a lightweight stream cipher for power-constrained devices," *IACR Transactions on Symmetric Cryptology*, pp. 45–79, Mar. 2017, doi: 10.46586/tosc.v2017.i1.45-79.

[10] L. Ding, "Improved related-cipher attack on Salsa20 stream cipher," *IEEE Access*, vol. 7, pp. 30197–30202, 2019, doi: 10.1109/ACCESS.2019.2892647.

[11] A. Alamer, B. Soh, A. H. Alahmadi, and D. E. Brumbaugh, "Prototype device with lightweight protocol for secure RFID communication without reliable connectivity," *IEEE Access*, vol. 7, pp. 168337–168356, 2019, doi: 10.1109/ACCESS.2019.2954413.

[12] B. Li, M. Liu, and D. Lin, "FPGA implementations of Grain v1, Mickey 2.0, Trivium, Lizard and Plantlet," *Microprocessors and Microsystems*, vol. 78, p. 103210, Oct. 2020, doi: 10.1016/j.micpro.2020.103210.

[13] L. Pyrgas and P. Kitsos, "Compact hardware architectures of enocoro-128v2 stream cipher for constrained embedded devices," *Electronics (Switzerland)*, vol. 9, no. 9, pp. 1–14, Sep. 2020, doi: 10.3390/electronics9091505.

[14] Y. He, G. Wang, W. Li, and Y. Ren, "Improved cube attacks on some authenticated encryption ciphers and stream ciphers in the internet of things," *IEEE Access*, vol. 8, pp. 20920–20930, 2020, doi: 10.1109/ACCESS.2020.2967070.

[15] Y. Zhu *et al.*, "Area-efficient parallel reconfigurable stream processor for symmetric cryptograph," *IEEE Access*, vol. 9, pp. 28377–28392, 2021, doi: 10.1109/ACCESS.2021.3057866.

[16] I. Salam, T. H. Ooi, L. Xue, W. C. Yau, J. Pieprzyk, and R. C. W. Phan, "Random differential fault attacks on the lightweight authenticated encryption stream cipher grain-128AEAD," *IEEE Access*, vol. 9, pp. 72568–72586, 2021, doi: 10.1109/ACCESS.2021.3078845.

[17] Y. Guang *et al.*, "Chaos-Based lightweight cryptographic algorithm design and fpga implementation," *Entropy*, vol. 24, no. 11, p. 1610, Nov. 2022, doi: 10.3390/e24111610.

[18] R. D. Silva, I. Navaratna, M. Kumarasiri, J. Alawatugoda, and C. C. Wen, "On power analysis attacks against hardware stream ciphers," *International Journal of Information and Computer Security*, vol. 17, no. 1–2, pp. 21–35, 2022, doi: 10.1504/IJICS.2022.121289.

[19] C. Chen, X. Wang, G. Liu, and G. Huang, "A robust selective encryption scheme for H.265/HEVC video," *IEEE Access*, vol. 11, pp. 17252–17264, 2023, doi: 10.1109/ACCESS.2022.3210132.

[20] C. Xiao, G. Zhao, L. Zhang, and D. Ding, "A controllable pipeline framework of block ciphers on GPU for streaming data," *IEEE Access*, vol. 11, pp. 93980–93993, 2023, doi: 10.1109/ACCESS.2023.3310401.

[21] M. M. Hazzazi, N. Iqbal, and A. Ikram, "Digital images security technique using hénon and piecewise linear chaotic maps," *IEEE Access*, vol. 11, pp. 106299–106314, 2023, doi: 10.1109/ACCESS.2023.3320031.

[22] V. R. Kebande, "Extended-Chacha20 stream cipher with enhanced quarter round function," *IEEE Access*, vol. 11, pp. 114220–114237, 2023, doi: 10.1109/ACCESS.2023.3324612.

[23] D. Lee, J. Kim, D. Hong, J. Sung, and S. Hong, "A practical ciphertext-only attack on GMR-2 system," *IEEE Access*, vol. 11, pp. 44519–44530, 2023, doi: 10.1109/ACCESS.2023.3271994.

[24] S. Babbage and M. Dodd, "The MICKEY stream ciphers," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4986 LNCS, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 191–209, doi: 10.1007/978-3-540-68351-3_15.

[25] P. Kitsos, "On the hardware implementation of the MICKEY-128 stream cipher," *Cryptology ePrint Archive*, 2005.

[26] D. Hwang, M. Chaney, S. Karanam, N. Ton, and K. Gaj, "Comparison of {FPGA}-targeted hardware implementations of {eSTREAM} stream cipher candidates," in *State of the Art of Stream Ciphers Workshop, {SASC} 2008, Lausanne, Switzerland*, 2008, pp. 151–162.

[27] B. Philippe, K. Kalach, F. -X. Standaert, and J. -J. Quisquater, "FPGA implementations of eSTREAM phase-2 focus candidates with hardware profile," in *State of the Art of Stream Ciphers Workshop (SASC 2007)*, eSTREAM, ECRYPT Stream Cipher Project, Report, vol. 24, 2007.

[28] 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, Mar. 2013, doi: 10.1016/j.micpro.2012.09.007.

[29] J. M. Marmolejo-Tejada, V. Trujillo-Olaya and J. Velasco-Medina, "Hardware implementation of Grain-128, Mickey-128, Decim-128 and Trivium," 2010 IEEE ANDESCON, Bogota, Colombia, 2010, pp. 1-6, doi: 10.1109/ANDESCON.2010.5632901.
[30] 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, Nov. 2020, doi: 10.3390/electronics9111935.

BIOGRAPHIES OF AUTHORS

Raghavendra Ananth hold a bachelor's degree in electronics and communication engineering from HMSIT College, Tumkur and Master's degree in VLSI design and embedded systems from REVA ITM College, Bangalore, Karnatka. He is a research scholar in the Department of Electronics and Communications, JAIN Deemed to be University, Bangalore. His areas of research include IoT security, VLSI Frontend design, and FPGA. He can be contacted at email: araghavendra.research@gmail.com.

Panduranga Rao Malode V. obtained his Ph.D. from the National Institute of Technology Karnataka, Mangalore, India. He has completed a master of technology in computer science and a bachelor of engineering in electronics and communication. He works as a Professor in Jain (Deemed to be University) Bengaluru, India. His research interests are in the field of real-time and embedded systems on Linux platforms. He has published various research papers in journals and conferences across India. He visited JAPAN in 2008 for the IEEE international conference in Okinawa Island. He has authored two reference books on Linux Internals. He is a Life member of the Indian Society for Technical Education and IAENG. In the past three years, he has published 12 Indian patents. Here, three patents are stepping towards award/grant status. He can be contacted at email: r.panduranga@jainuniversity.ac.in.

Narayana Swamy Ramaiah received a Ph.D. from PRIST University, Tamil Nadu, in the year 2016, M.Tech. in the year 2004 from Visvesvaraya Technological University, Karnataka and a B.E in the year 2002 from Bangalore University, Karnataka. He has over 16 years of experience in teaching, research and industry. He has published over 45 papers in peer-reviewed journals. His research areas include IoT (agriculture), blockchain, ai and machine learning, and cloud computing. Currenjtly, he works as a Professor and HoD, Department of Computer Science Engineering, New Horizon College of Engineering, Bangalore. He can be contacted at email: narayanaswamy.ramaiah@gmail.com.