I.J. Information Technology and Computer Science, **2026, 3, 1-12**
Published Online on June 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2026.03.01

FPGA Implementation of a Fault-tolerant Encryptor Based on Reconfigurable Fredkin Gates

Taras Kyryliuk
Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58012, Ukraine
E-mail: flexx02@gmail.com
ORCID iD: https://orcid.org/0009-0004-5033-1200
*Corresponding author

Mykhailo Palahuta
Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58012, Ukraine
E-mail: palahuta.mykhailo@chnu.edu.ua
ORCID iD: https://orcid.org/0009-0003-4896-903X

Oleksii Dovhaniuk
School of Computer Science, University College Cork, Cork T12 R229, Ireland
E-mail: 122119242@umail.ucc.ie
ORCID iD: https://orcid.org/0009-0003-2247-9323

Vitaly Deibuk
Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58012, Ukraine
E-mail: v.deibuk@chnu.edu.ua
ORCID iD: https://orcid.org/0000-0002-1216-0031

Received: 17 November 2025; Revised: 19 January 2026; Accepted: 05 March 2026; Published: 08 June 2026

Abstract: As classical computation approaches its fundamental limits related to power dissipation, reversible logic, which theoretically achieves zero energy loss, is becoming a critical technology for future low-power and quantum computing. However, most research in this field remains theoretical, lacking practical, hardware-verified implementations. This paper bridges this gap by presenting the complete hardware implementation and rigorous fault-tolerance validation of a reversible encryptor based on extended Fredkin gates. First, we detail the full realization of the encryptor on an Altera Cyclone IV Field-Programmable Gate Array. This implementation is not just a simulation but a complete, interactive hardware prototype, featuring real-time data input via a standard keyboard and output to a video graphics array monitor. Second, since functional verification is insufficient for cryptographic hardware and exhaustive testing is computationally infeasible, we introduce a novel validation methodology. This core contribution utilizes a metaheuristic ant colony optimization algorithm, not for synthesis, but for the intelligent generation of an optimal and compact set of test vectors. This test set is designed to achieve maximum fault coverage for the industry-standard "stuck-at fault" model. The algorithm successfully generated a minimal test set achieving 100% coverage for the considered single stuck-at fault model. We then experimentally validated this methodology by manually injecting a stuck-at fault into the hardware description language, recompiling the faulty circuit into the device, and confirming that the metaheuristically generated test vector successfully detected the physical fault. Thus, this work demonstrates the full cycle from theory to a practically validated and reliable hardware implementation of a reversible system.

Index Terms: Reversible Logic, Ant Colony Optimization (ACO), Simulation, Fault Tolerance, Test Vector Generation, FPGA, Stuck-at Fault, Fredkin Gate, Model.

1. Introduction

The rapid development of information technologies and the continuous growth of computational demands pose new challenges to traditional irreversible computer architecture. The primary challenges include approaching the fundamental physical limits of miniaturization, known as Moore's Law, and the associated significant power consumption, which results in excessive heat dissipation. This stimulates the global scientific community to actively search for and develop

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

alternative computing paradigms, among which reversible logic and quantum computers hold key positions [1-4]. This research is focused on the intersection of these innovative fields, proposing practical solutions for the implementation and verification of reliable next-generation computing devices, particularly in the field of cryptography.

The problem of energy efficiency is directly related to Landauer's principle, which states that every irreversible logical operation results in energy dissipation. The minimum energy E_{min}, dissipated when losing one bit of information is equal to

$$E_{min} = k_B T \ln 2 \quad (1)$$

where k_B — is the Boltzmann constant, and T is the absolute temperature of the system.

In this context, reversible logic becomes one of the key technologies for the future, since, as it has been proven in [2], it allows computations without energy dissipation, which is critical for the development of low-power devices and the Internet of Things (IoT) [5-8]. Simultaneously, the rapid advancement of quantum computing presents a dual challenge. Firstly, quantum algorithms threaten to break most modern asymmetric cryptographic systems. Second, quantum computations themselves are inherently reversible. This creates an urgent need to develop new, reliable cryptographic devices based on reversible circuits that remain resilient in the era of post-quantum cryptography.

Most contemporary research in this field, including our previous works [9-20], focuses on theoretical synthesis, optimization (e.g., by quantum cost), and simulation (in Qiskit or Active-HDL). However, to confirm the practical viability of these designs, a transition from simulations to hardware implementation is necessary. Field-Programmable Gate Array (FPGA) platforms are an ideal tool for rapid prototyping, allowing for the verification of logic and the assessment of real performance of reversible circuits in a hardware environment [5,6]. A key aspect for cryptographic devices is their reliability and fault tolerance. It is insufficient to merely implement a circuit; it is necessary to prove that it functions correctly and to present a mechanism for fault detection. Traditional testing methods (e.g., exhaustive search) are inefficient for circuits with a large number of inputs. Therefore, the development of intelligent methods for generating test sequences that provide maximum fault coverage with a minimal set of tests is a relevant challenge. The use of meta-heuristic approaches, such as the ant colony optimization (ACO) algorithm, for finding optimal test vectors to detect specific faults (e.g., stuck-at-faults) in reversible circuits is a novel and promising research direction.

The main contribution of this work lies in the hardware implementation and comprehensive fault-tolerance validation of a reconfigurable encryptor based on extended Fredkin gates. Firstly, we present the complete implementation of the encryptor on an Altera Cyclone IV FPGA, including the integration of peripheral devices (a keyboard and a monitor) for interactive demonstration. Secondly, we propose a new method for test sequence generation based on the ACO algorithm, designed for the efficient detection of "stuck-at" faults in reversible circuits. The effectiveness of the proposed approach is demonstrated through experimental validation on the hardware platform with fault injection.

The article is structured as follows: Section II contains a review of related solutions. Section III details the hardware implementation of the encryptor on the FPGA. Section IV presents the fault-tolerance testing methodology based on ACO. Section V analyses the experimental validation results. Finally, Section VI summarizes the conclusions and outlines future research directions.

2. Related Works and Background

This section presents the theoretical foundations of this work and reviews existing research in related areas. We begin with a review of key related research relevant to our goal—the synthesis of reversible circuits, their hardware implementation, and testing methodologies—and then proceed to a discussion of these findings.

2.1. Reversible Circuit Synthesis Using Meta-heuristics

The synthesis of optimal reversible circuits is an NP-hard problem, which requires effective heuristic approaches. Traditional methods, such as Binary Decision Diagrams (BDD) [9] or transformation-based approaches developed by Maslov and others [11], are effective but can have scalability issues. This led to the active development of meta-heuristic approaches. Genetic algorithms (GAs) have become widespread. In our previous work [4], we proposed an improved GA with multicomponent mutation. Other researchers have also successfully applied GAs, for example, adaptive GAs [12] and hybrid evolutionary algorithms [14], which confirms the maturity of this field [15].

Another powerful method is ant colony optimization (ACO). Studies show the effectiveness of ACO in formulating synthesis as a best-path search problem [17]. In particular, Podlaski et al. [18] demonstrated an ACO implementation in the Walsh-Hadamard Domain. In our work [19], we conducted a comparative analysis of GA and ACO specifically for synthesis tasks. Although ACO and GA are well-researched tools for the synthesis of reversible circuits, their application for generating test sequences (validation) remains a significantly less explored area.

2.2. Hardware Implementation of Reversible Circuits using FPGA

The transition from theoretical modelling to hardware implementation is a critical step in proving the practical applicability of reversible circuits. FPGA platforms are an ideal choice for this. Several research groups have already presented implementations of reversible circuits on FPGAs, particularly for cryptographic tasks. For example, works

[5,6] describe the implementation of encryption using reconfigurable reversible gates. These works confirm the feasibility of implementation and provide estimations of hardware costs. Work [20] was also focused on the synthesis of a reconfigurable Fredkin gate designed for FPGAs and verified in Active-HDL. Studies [21] also demonstrate the implementation of cryptography on reversible logic, which proves the sustained interest in this topic.

Although the implementation of reversible encryptors on FPGAs is a proven concept, most of these works focus on functional verification (proving that the circuit works), but do not offer comprehensive methodologies for fault-tolerance validation (proving that the circuit is reliable) in the hardware environment.

2.3. Testing and Fault-Tolerance of Reversible Circuits

Reliability is a key requirement for cryptography. In the field of reversible circuits, fault-tolerance research is divided into two categories: designing fault-tolerant circuits and testing them.

Design: Our previous article [6] belongs to the first category, where we proposed an encryptor with parity preservation, which is inherently fault-tolerant by design.

Testing: Testing requires clear fault models. The most common is the "stuck-at fault" (SAF) model [22]. Other recent studies [23] also propose efficient techniques for detecting and locating not only SAF but also more complex fault models (such as SMGF and SBF) in Toffoli-based circuits. For reversible circuits, this model has unique properties. While some of these properties can simplify testing (for example, a test set that covers all single stuck-at faults often detects multiple faults as well [22]), other studies [24,25] emphasize that designing for testability remains a significant challenge that introduces high operational costs.

Existing works establish SAF fault models and the theoretical foundations for testing. However, they do not offer efficient, scalable algorithms for generating optimal test sequences that would guarantee maximum fault coverage with a minimal test set size.

Our work fills this gap by combining all three directions: we take an encryptor implemented on an FPGA and apply the stuck-at fault model to it. But for generating test sequences, we use a new approach, adapting a meta-heuristic (ACO) from its traditional task of synthesis to a new task – optimizing test coverage. To our knowledge, using ACO for generating test vectors for the hardware validation of reversible encryptors is a novel contribution in this field.

2.4. Background

As mentioned, the architecture under study is based on the reversible Reconfigurable Reversible Gate 2 (RRG2*)* gate, which is a product of our previous work on GA-synthesis [4,6]. This subsection describes its fundamental components and architecture. A fundamental problem of CMOS electronics, as mentioned, is energy dissipation. Reversible logic offers a fundamental solution to this problem. It operates using bijective functions, which ensure a unique one-to-one correspondence between input and output states. Since information is not lost in such a system, computations occur without energy dissipation, which reduces entropy and, consequently, the number of errors. Such circuits are built from a set of reversible gates that have an equal number of inputs and outputs. The universal NCT library (Fig. 1) includes the gates: X(NOT), CX (CNOT), C2X(Toffoli), which implement the functions presented in (2)

$$\begin{aligned} X:&\quad out = \bar{in}\,, \\ CX:&\quad out = ctrl \oplus in\,, \\ Toffoli:&\quad out = ctrl1 \cdot ctrl2 \oplus in \end{aligned} \tag{2}$$

It is important to note that the NCT library constitutes a functionally complete basis, meaning that any more complex function can be realized, including the CSWAP (Fredkin) gate, as presented in (3) and in Fig.1, e.

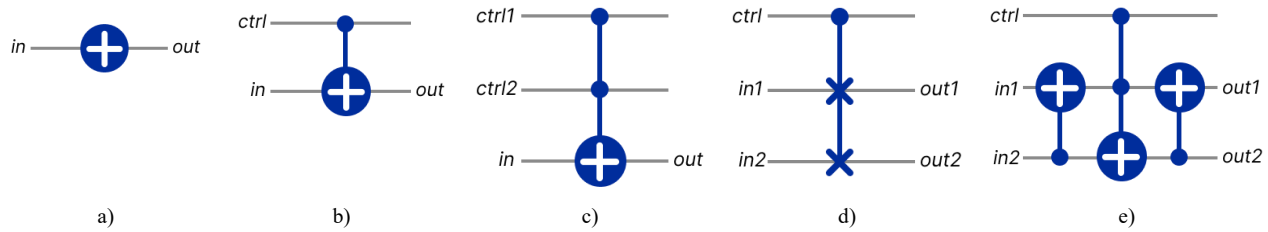


Fig. 1. The reversible logic gates: a – NOT; b – CNOT; c – Toffoli gate; d – Fredkin gate; e – NCT based implementation of Fredkin gate

$$\begin{aligned} out1 &= ctrl\ in2 \oplus \overline{ctrl}\ in1, \\ out2 &= ctrl\ in1 \oplus \overline{ctrl}\ \ in2 \end{aligned} \tag{3}$$

One of the promising application areas for reversible logic is cryptography, as the reversibility of operations is ideally suited for encryption algorithms, where the decryption process is the inverse. In our previous research [6], we proposed and analysed a series of reconfigurable reversible gates. Reconfiguration in this context means the gate's ability to change its logical function depending on control signals [4]. These control inputs act as a key, allowing the same cascade of gates

to perform different transformations. For the purposes of this work, which focuses on hardware implementation and validation, we selected the RRG2 [6]. This choice is due to its relative simplicity and elegance, making it an ideal candidate for demonstrating the full cycle of implementation and testing.

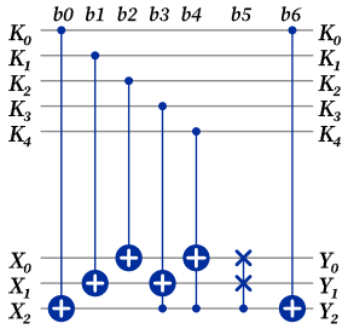


Fig. 2. Schematic diagram of the RRG2 encryptor block [[6]]

The RRG2 gate circuit, presented in Fig. 2, implements the transformation of a data block $\{X_0, X_1, X_2\}$ under the control of a 5-bit key $\{K_0, K_1, K_2, K_3, K_4\}$ and is a cascade of the basic CNOT, Toffoli, and Fredkin gates, described above. For processing 8-bit ASCII characters, we use the *ascii_codec* architecture, where three RRG2 blocks operate in parallel, as shown in Fig. 3.

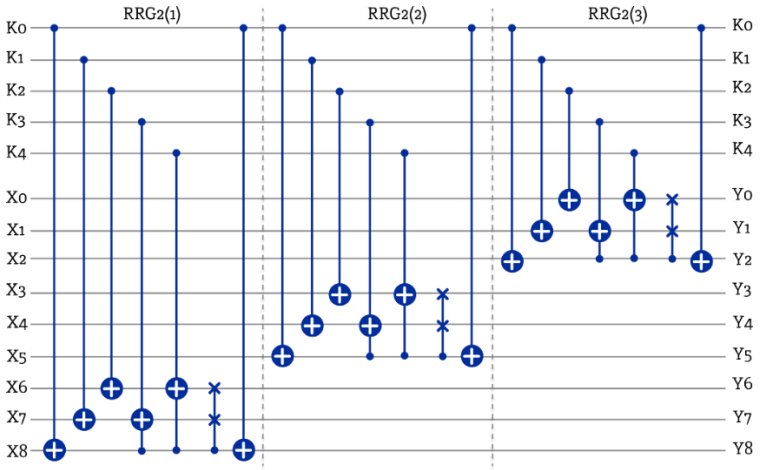


Fig. 3. Schematic diagram of the RRG2 encryptor block

3. Hardware Implementation of the Encryptor

This section details the process of migrating the theoretical model of the RRG2 encryptor, proposed in [6] and presented in Fig. 2, from the logical simulation level to a physical FPGA hardware platform. It describes the modular Verilog architecture used for the implementation and the integration of peripheral devices (a PS/2 keyboard and a VGA monitor) to create an interactive hardware prototype.

3.1. Hardware Platform and Development Environment

A development board equipped with an Altera Cyclone IV FPGA (EP4CE6E22C8N) was chosen as the target platform. This choice is justified by a sufficient number of logic elements (LEs) to implement both the encryptor circuit itself and the necessary peripheral controllers. The entire process of design, synthesis, Place & Route, and generation of the programming file (*.sof*) was performed in the Altera Prime development environment. The project (*keyboard_text.qsf*) was configured to operate with a 50 MHz clock frequency (*clk_50mhz*).

3.2. Verilog Project Architecture

The project has a modular structure, where the top level (*keyboard_text.v*) integrates three main functional blocks: an input controller, an encryption core which consists of three RRG2 blocks connected in parallel, and an output controller, the block diagram of which is presented in Fig. 4.

- Input Controller (PS/2 Keyboard) The *ps2_keyboard.v* module implements a low-level interface for operating the keyboard via the PS/2 protocol. It performs synchronization of the input signals (*ps2_clk, ps2_data*),

processes the serial data stream, filters scan-codes, and converts them into 8-bit ASCII codes (*ascii_code*).

- Output Controller (VGA Display) The display system consists of three modules: *vga_controller.v* (generates 640x480 @ 60Hz signals), *font_rom_8x16.v* (contains the 8x16 character bitmaps), and *keyboard_display.v* (acts as the "video card," contains the text buffer, and manages the cursor).
- Encryption Core (Reversible Codec) The *ascii_codec.v* module acts as a wrapper for the encryption core. It accepts an 8-bit ASCII code, a 5-bit key, and an *encode_mode* signal from a switch. Depending on the mode, *ascii_codec* directs the data either to the forward encryption path (the RRG2.v modules) or to the reverse path (RRG2_Inverse.v modules). Since the function our circuit computes is not an involution, meaning it is not its own inverse $enc(enc(x)) \neq x$, we need a circuit inversion mechanism, which was realized by mirroring the RRG2 as a separate implementation – RRG2_Inverse.

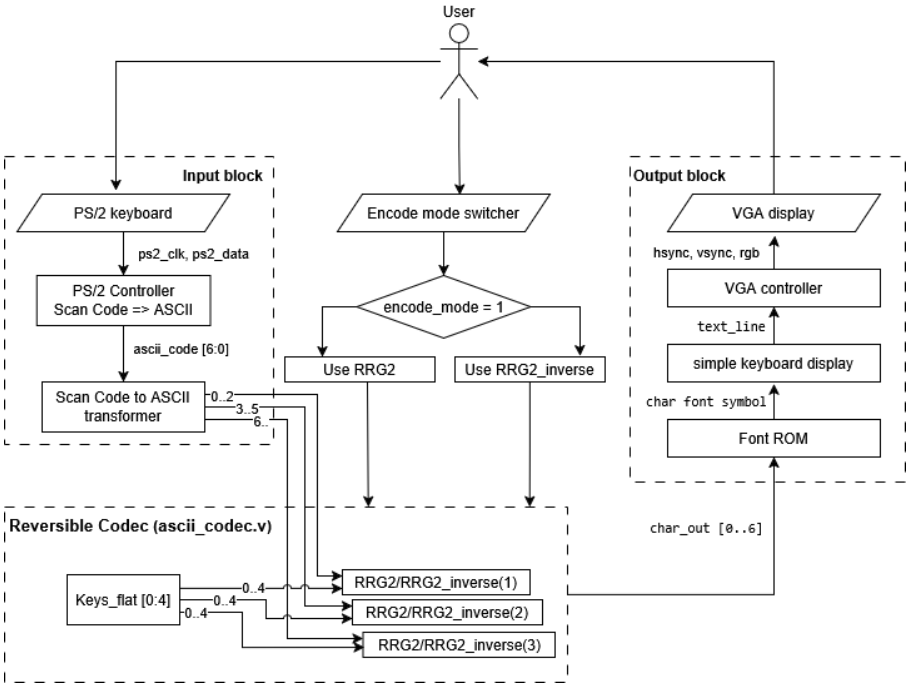


Fig. 4. System block diagram of the hardware prototype

3.3. Implementation of Basic Reversible Gates

The basis of the entire encryption logic is the basic reversible CNOT, Toffoli, and Fredkin gates, implemented as separate Verilog modules presented in Fig. 5. They are a direct hardware description of the quantum gates from the NCT library and the extended Fredkin gates [4].

```
module CX (
    input wire ctrl,
    input wire in,
    output wire out
);
    assign out = ctrl
      ? ~in
      : in;
endmodule
```

a)

```
module C2X (
    input wire ctrl0,
    input wire ctrl1,
    input wire in,
    output wire out
);
    assign out = (ctrl0 & ctrl1)
      ? ~in
      : in;
endmodule
```

b)

```
module CSWAP (
    input wire ctrl,
    input wire in1,
    input wire in2,
    output wire out1,
    output wire out2
);
    wire ctrl_out;
    assign ctrl_out = ctrl;
    assign out1 = ctrl ? in2 : in1;
    assign out2 = ctrl ? in1 : in2;
endmodule
```

c)

Fig. 5. Verilog implementation of NCT based gates: a – CNOT, b – Toffoli, c – Fredkin

3.4. RRG2 Encryptor Core Module Code

The RRG2.v encryptor module, presented in Fig. 6, is a cascade of the basic gates described in 3.3. This structure directly implements the optimized encryption circuit proposed in [6]. The Verilog code for *RRG2.v* demonstrates how the CNOT, Toffoli, and Fredkin gates are connected sequentially to perform the transformation of a 3-bit data block *X* under the control of a 5-bit key *K*. A *wire_array* is used to transfer signals between the stages.

```
module RRG2(
  input  wire [4:0] K,
  input  wire [2:0] X,
  output wire [2:0] Y
);
  wire [2:0] wire_array [6:0];
  //b0
  assign wire_array[0][0] = X[0]; assign wire_array[0][1] = X[1];
    CX b0 (.ctrl(K[0]), .in(X[2]), .out(wire_array[0][2]));
  //b1
  assign wire_array[1][0] = wire_array[0][0]; assign wire_array[1][2] = wire_array[0][2];
    CX b1 (.ctrl(K[1]), .in(wire_array[0][1]), .out(wire_array[1][1]));
  //b2
    assign wire_array[2][1] = wire_array[1][1]; assign wire_array[2][2] = wire_array[1][2];
  CX b2 (.ctrl(K[2]), .in(wire_array[1][0]), .out(wire_array[2][0]));
  //b3
  assign wire_array[3][0] = wire_array[2][0]; assign wire_array[3][2] = wire_array[2][2];
  C2X b3 (.ctrl0(K[3]), .ctrl1(wire_array[2][2]), .in(wire_array[2][1]), .out(wire_array[3][1]));
  //b4
  assign wire_array[4][1] = wire_array[3][1]; assign wire_array[4][2] = wire_array[3][2];
    C2X b4 (.ctrl0(K[4]), .ctrl1(wire_array[3][2]), .in(wire_array[3][0]), .out(wire_array[4][0]));
  //b5
    assign wire_array[5][2] = wire_array[4][2];
  CSWAP b5 (.ctrl(wire_array[4][2]), .in1(wire_array[4][0]), .in2(wire_array[4][1]),
        .out1(wire_array[5][0]), .out2(wire_array[5][1]));
  //b6
  assign wire_array[6][0] = wire_array[5][0]; assign wire_array[6][1] = wire_array[5][1];
    CX b6 (.ctrl(K[0]), .in(wire_array[5][2]), .out(wire_array[6][2]));
  //output
  assign Y[0] = wire_array[6][0]; assign Y[1] = wire_array[6][1]; assign Y[2] = wire_array[6][2];
endmodule
```

Fig. 6. Verilog implementation of RRG2

RRG2_Inverse.v is a similar module that implements the same circuit in reverse order, which ensures correct decryption.

3.5. Hardware Implementation Details and Reproducibility

The system was mapped to the Altera Cyclone IV EP4CE6E22C8 FPGA. The design utilizes a 50 MHz global clock with a Synopsys Design Constraints file defining a 20 ns period. To ensure signal integrity, all I/O ports follow the 3.3-V LVTTL standard as detailed in Table 1.

Table 1. FPGA pin assignments for the reversible encryptor prototype

Signal Name	FPGA Pin	Direction	Description
clk_50mhz	PIN_23	Input	Global system clock (50 MHz)
rst_n	PIN_24	Input	Asynchronous reset (Active Low)
encode_switch	PIN_91	Input	Mode selector (Encrypt/Decrypt)
hsync	PIN_101	Output	VGA horizontal synchronization signals
vsync	PIN_103	Output	VGA vertical synchronization signals
blue	PIN_104	Output	VGA blue channels
green	PIN_105	Output	VGA green channels
red	PIN_106	Output	VGA 1red channels
ps2_clk	PIN_119	Input	Keyboard clock signal
ps2_data	PIN_120	Input	Keyboard serial data

A 3-stage synchronization pipeline was implemented for the PS/2 interface, and a 25.175 MHz pixel clock for VGA was derived via a hardware frequency divider.

4. ACO-Based Test Set Generation Methodology

Ensuring the reliability and correct functioning of reversible circuits is a critically important stage in their design and implementation. Given the exponential growth of the input vector space with an increasing number of lines (N) in the circuit (a space of 2^N), the use of exhaustive search methods for generating test sets becomes computationally infeasible. This is particularly relevant for large circuits from the RevLib library [26] that we are investigating, which can have $N \geq 10$ and a gate count exceeding 10^4. In this regard, we propose and apply a heuristic approach based on ACO to find an optimal test set of minimal length that provides the maximum fault coverage (FC).

4.1. Modelling and Injection of Stuck-at Faults

For verification purposes, we focus on the classical and well-studied SAF [22] fault model, adapted to the architecture of reversible circuits. According to this model, a line in the circuit can be permanently fixed at a logical level '0' (stuck-at-0, s-a-0) or at a logical level '1' (stuck-at-1, s-a-1).

Potential fault locations include:

- The input lines of the circuit.

- Internal lines connecting the output of one reversible gate to the input of the next.

To analyse the complete fault set, we generate a list of all possible SAF faults occurring at each of the specified positions. If the circuit has *L* lines (where $L \approx N+G \cdot N$, *G* is the number of gates), the total number of elementary SAF faults is *2L* (for s-a-0 and s-a-1).

Fault Simulation. The verification process requires simulating the operation of the faulty circuit (RC_f). For each fault *f*, its structural model is created by modifying the original reversible circuit *RC*. The modification consists of forcibly fixing the value of the corresponding line at the specified location. A test vector V_{test} is applied in parallel to the input of both the reference circuit *RC* and the faulty circuit RC_f. The fault *f* is considered as detected by this vector if (4) is true

$$Output(RC, V_{test}) \neq Output(RC_f, V_{test}) \tag{4}$$

In our software implementation, written in the Go language, this process is initiated by calling the *stuckAt.Induce(rc)* structural fault injection procedure, which creates a complete set of faulty circuits for subsequent simulation.

4.2. Problem Formalization as an ACO Task

The generation of an optimal test set $T = \{V_1, V_2, \ldots, V_K\}$ is considered as a shortest path search problem in a state space, where each state corresponds to a certain fault coverage.

State Space and Transitions.

- Node (State): Each node in the graph represents the current set of detected faults $F_{cov} \subset F_{total}$, where F_{total} is the set of all SAF faults. The initial node corresponds to empty coverage $F_{cov}=\emptyset$
- Edge (Transition): A transition from node S_i to node S_{i+1} corresponds to the selection and addition of a new input vector V_{new} to the test set *T*. The state S_{i+1} is determined by the set of faults detected by the set $T \cup \{V_{new}\}$.

Heuristic Search. In ACO, ants sequentially select input vectors, constructing a path (the test set) with the goal of maximizing coverage. The selection of the next vector V_{new} is based on the pheromone trail represented as (5)

$$P(V_{new}) \sim (\tau(V_{new}))^{\alpha} \cdot (\eta(V_{new}))^{\beta} \tag{5}$$

where τ is the accumulated experience of heuristic information, η is (visibility), and α and β – are weighting coefficients that determine the relative importance of pheromone and visibility.

Definition of visibility η. The heuristic information $\eta(V_{new})$ is the driving force of the search (6). For the test generation task, η is defined as the number of new faults from the set F_{rem} (faults that remained undetected in state S_i) that are detected by the chosen vector V_{new}

$$\eta(V_{new}) = |F_{det}(V_{new}) \cap F_{rem}| \tag{6}$$

Vectors with a higher η (those that cover more undetected faults) have a higher probability of being chosen by an ant.

Fitness Function. The optimality criterion for the found test set *T* (path) is the fitness function *F*, which combines two opposing objective metrics:

- Maximization of FC: $C = \frac{|F_{cov}|}{|F_{total}|}$.
- Minimization of the test set size: C $|T| \rightarrow$100% with minimal $|T|$.

Pheromone Update. After an iteration is completed (when all ants have built their test sets), the pheromones on the edges are updated:

- Evaporation: Evaporation ρ is applied to all edges, modelling forgetting.
- Deposition: Ants that found the best set (with the highest *F*) deposit pheromone on the edges they traversed. In this way, paths that lead to high FC with a small number of test vectors are reinforced.

This approach allows us to efficiently find sub-optimal test sets. For example, we were able to generate a single input vector that already provides 50% fault coverage for urf1 from the RevLib library, a standard benchmark used for evaluating scalability of large reversible circuits which has 12 lines and approximately 12000 gates.

5. Results and Discussion

This section presents the results of the experimental approbation of the hardware-implemented device. The goal of

the experiments is to verify the functional correctness and analyse the efficiency of FPGA hardware resource utilization, and to validate the effectiveness of the proposed ACO-based test set generation methodology for fault detection.

5.1. Functional Verification and Hardware Resource Analysis

The hardware prototype, serving as a functional emulation of the reversible logic on a CMOS-based FPGA, demonstrated full functional correctness. Testing the «end-to-end» path (PS/2 > ascii_codec > VGA) confirmed that encryption and decryption operate according to [6]. The implementation quality was evaluated using the Intel Quartus Prime Timing Analyzer and Power Analyzer Tool on the Altera Cyclone IV (EP4CE6E22C8N). The resulting performance metrics, resource utilization, and power profile are summarized in Table 2.

Table 2. Comprehensive implementation quality and timing summary

Metric Category	Parameter	Value	Constraint
Timing	Max Frequency (F_{max})	220.85 MHz	50.00 MHz
	Setup slack	+15.47 ns	> 0 ns
	Hold slack	+0.45 ns	> 0 ns
	Data Latency	1 clock cycle	Real-time
Power	Total Thermal Power	79.47 mW	Low-power profile
	Core Dynamic Power	5.26 mW	Energy-efficient
Resources	Total logic elements (LE)	1,424 / 6,272 (23%)	23% Utilization
	Total registers	750	Optimized Pipeline

The data demonstrates high implementation efficiency. Logic utilization (LE) is 23%, with a significant portion of these elements (750) used as registers, primarily for the text buffer and VGA logic. It is important to note that the compiler report showed 0% utilization of dedicated memory blocks (M9K) and PLLs. This means that the font ROM (*font_rom_8x16.v*) was implemented as distributed ROM directly in the logic elements, and the frequency division for VGA was achieved with a logical divider. The low resource consumption (23% LE, 32% pins) confirms that the complete prototype with I/O is practically viable and scalable even on an entry-level FPGA.

5.2. Fault-tolerance Validation and Hardware Fault Injection

The second goal of the experiments is the validation of the ACO methodology. We applied our simulator to the complete fault dictionary of the *ascii_codec* circuit, which contains 1444 potential SAF faults. The ACO algorithm generated a compact test set. As shown in Table 3, the algorithm found optimal ways to achieve 92% FC with only 2 input words and 100% FC, meaning the detection of all possible faults, with only 3 test vectors.

Table 3. Key test vectors generated by ACO (Key = 0x155)

№	Input (ASCII, binary)	FC	SAFs Covered
1	'k', [11011010]	100%	1444
	‘4’, [11011010]		
	‘0’, [11101001]		
2	'C', [01000011]	92%	1332
	‘?’, [10011111]		

The total runtime of the algorithm to find these solutions was less than one second. The ACO parameters used for the generation are presented in Table 4.

Table 4. ACO algorithm parameters

Parameter	Used value
Num. of ants	50
Num. of iterations	20
α	3
β	1
ρ	0.6
Q	10

where α defines the pheromone importance weight, β defines the visibility importance weight, ρ defines the pheromone evaporation rate, and Q defines the pheromone deposit strength. The starting point for the search is always at the food

node as the algorithm performs a reverse search.

For the hardware verification of the tests, we conducted a fault injection experiment. We sequentially injected stuck-at-0 faults at each possible position using the following Verilog program:

```
11    // CX b0 (.ctrl(K[0]), .in(X[2]), .out(wire_array[0][2]));
12    CX b0 (.ctrl(1'B1'), .in(X[2]), .out(wire_array[0][2])); //FAULT INJECTED: SA1
```

Fig. 7. Fault injection program

After that, we proceeded with injecting stuck-at-1 faults with a similar program. As a result, we were able to detect all of the injected faults using the generated vector set 1 (from Table 4).

To demonstrate the scalability of the proposed validation approach, we applied our algorithm to the large reversible circuit from the RevLib library - urf1. The ACO parameters used were the same (Table 4). As a result, we obtained an input vector “011101111011” that provides 50% fault coverage for the circuit.

5.3. Discussion and Comparative Analysis

The results obtained in sections 5.1 and 5.2 should be considered not only independently but also in the context of the broader research presented in section 2. A direct quantitative comparison of the hardware resources presented in Table 2 with works such as [7,8] is complicated. Firstly, authors often use different FPGA families (Altera Cyclone 3 vs. Xilinx Spartan), which makes a direct comparison of logic elements (LEs) and lookup tables (LUTs) incorrect. Secondly, it is not always clear whether the authors are reporting resources spent only on the encryption core or on the full project with infrastructure. However, the main difference in our work lies not in minor savings of logic elements, but in the fundamental goal of the research.

- Transition from Verification to Validation: Most existing works, including [7,8], focus on functional verification. Their main goal is to prove that the proposed circuit works, i.e., that it correctly encrypts and decrypts data. This is undoubtedly an important step, which we also perform in section 5.1. However, our work takes the next logical step: from functional verification to comprehensive fault-tolerance validation. We ask a different, more complex question: "How can we guarantee that this circuit is reliable, and how can we efficiently detect physical faults when they occur?"
- Novelty of the testing methodology: While existing works [7,8,20,21] only demonstrate functionality, we propose a complete validation cycle. The key contribution is not the FPGA implementation itself (although it is efficient, at 23% LE), but the application of the ACO methodology (Section IV) to solve the practical problem of testing. The results from Table 3, where we achieve 100% FC for 1444 potential faults using only three test vectors, demonstrate the power of this approach.
- Hardware Confirmation: The final experiment with fault injection (Fig. 7) is decisive. We do not just theoretically claim that our ACO-generated test set works. We experimentally prove it by physically programming the "faulty" circuit into the FPGA and showing that our generated vector ('4') successfully catches this error on real hardware (output 'X' instead of 'x'). To our knowledge, such a complete cycle (from GA-synthesis of the core to ACO-validation and hardware fault injection) is presented for the first time in the context of reversible encryptors.

Thus, our work shifts the focus from the question "Can this be implemented?", to which works [7,8] have already given a positive answer, to the question "How can we trust this implementation and guarantee its reliability?".

The proposed design is positioned as a functional hardware primitive. Its measured low dynamic power profile (5.26 mW) suggests potential inherent resistance to side-channel attacks, a hypothesis that will be further explored in future cryptographic evaluations.

6. Conclusions

This paper presents a complete development and validation cycle for a hardware encryptor based on reversible circuits: from the theoretical GA-synthesis of the model to its hardware implementation and full ACO-validation of its fault tolerance. The research successfully achieved two key objectives. First, a complete hardware implementation of the reconfigurable reversible encryptor [6] was presented and tested on the Altera Cyclone IV FPGA platform. The analysis of synthesis results (Table 2) showed the high efficiency of the approach: the complete "end-to-end" prototype, including PS/2 and VGA controllers, occupied only 23% of the logic elements of an entry-level FPGA, confirming the practical viability and scalability of the implementation. Second, and the key contribution of this work, a new methodology for fault-tolerance testing was proposed and validated. As shown in Section 5.3, our work takes a step forward from functional verification, where most similar works [7,8] stop, to comprehensive fault-tolerance validation. Our ACO simulator generated a minimal test set (only 3 vectors) to achieve 100% FC for 1444 potential SAF faults (Table 3). The effectiveness of this set was proven experimentally: by manually injecting a fault into the Verilog code (Fig. 7), we confirmed that the ACO-generated test vector successfully detected the hardware fault on the real device.

Thus, this work successfully bridges the gap between theoretical synthesis and practical, validated implementation. We have not only proven that an RRG2-based encryptor can be efficiently implemented in hardware, but have also demonstrated that meta-heuristic algorithms are a powerful tool for ensuring the reliability of such systems.

Future research directions will include expanding the testing methodology to more complex "bridging faults" models, applying ACO for test generation for larger circuits from the RevLib library [26], and exploring built-in self-testing.

Future research directions will include expanding the testing methodology to more complex "bridging faults" models and applying ACO for test generation for larger circuits from the RevLib library [26]. A key focus will be a systematic comparative analysis of ACO with other established meta-heuristic and traditional methods, such as Genetic Algorithms and greedy search approaches, to evaluate performance in terms of runtime, scalability, and vector set minimality. Additionally, we plan to explore the implementation of built-in self-testing structures directly within the reversible hardware architecture.

All the Declarations and Statements

Author Contributions Statement

Taras Kyryliuk: Conceptualization, Methodology, and Hardware Implementation: Proposed research ideas, constructed the overall framework of the reversible encryptor, and executed the complete FPGA implementation.
Mykhailo Palahuta: Software Implementation and Validation: Handled the development of the ACO-based software simulator, implemented the test vector generation model, and performed hardware fault injection testing.
Oleksii Dovhaniuk: Visualization, Validation, and Statistical Analysis: Performed in-depth analysis of experimental results, prepared performance tables and charts, and ensured the robustness of the fault-coverage evaluation.
Vitaly Deibuk: Supervision, Formal Analysis, and Review: Supervised project execution, performed in-depth theoretical analysis of the reversible logic circuits, and guided the research methodology.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The authors declare no conflicts of interest.

Funding Declaration

The authors received no financial support for the research, authorship, and/or publication of this article.

Data Availability Statement

The datasets and simulation files generated during the current study are available from the corresponding author on reasonable request.

Ethical Declarations

This research did not involve human subjects or animals. Therefore, no statements of ethical approval are required.

Acknowledgments

We are grateful to the editorial board and the reviewers for their constructive evaluation of our work. Their valuable recommendations played a crucial role in refining our methodology and improving the reliability of the presented results.

Declaration of Generative AI in Scholarly Writing

During the preparation of this work, the authors used Generative AI (Large Language Models) for the sole purpose of improving the English language, grammar, and stylistic consistency of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the accuracy and integrity of the final version of the article. No original scientific content, data, or technical results were generated by artificial intelligence.

Abbreviations

The following abbreviations are used in this manuscript:

ACO - Ant Colony Optimization
ASCII - American Standard Code for Information Interchange
BDD - Binary Decision Diagrams
CMOS - Complementary Metal-Oxide-Semiconductor
CNOT - Controlled-NOT (gate)
FC - Fault Coverage
FPGA - Field-Programmable Gate Array

GA - Genetic Algorithm
HDL - Hardware Description Language
IoT - Internet of Things
LE - Logic Element
NCT - NOT, CNOT, Toffoli (gate library)
PS/2 - Personal System/2 (keyboard interface)
RRG2 - Reconfigurable Reversible Gate 2
SAF - Stuck-At Fault
VGA - Video Graphics Array

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Research & Development*, Vol. 5, No. 3, pp. 183-191, 1961.

[2] A. Bérut, A. Petrosyan, and S. Ciliberto, "Information and thermodynamics: experimental verification of Landauer's erasure principle," *Journal of Statistical Mechanics: Theory and Experiment*, Vol. 2015, No. 6, pp. P06015, 2015.

[3] Z. Hu, V. Deibuk, "Design of ternary reversible/quantum sequential elements," *Journal of Thermoelectricity*, No. 1, pp. 5 – 17, 2018.

[4] V. Deibuk, O. Dovhaniuk, and T. Kyryliuk, "The extended Fredkin gates with reconfiguration," *Lecture Notes on Data Engineering and Communications Technologies*, Vol. 181, pp. 95-105, 2023. DOI: 10.1007/978-3-031-36118-0_9.

[5] G. Thakur, S. Keshri and K. Chaudhari, "Design of Energy Efficient 16-Bit Reversible ALU for Low Power IoT Applications," *2025 12th Int. Conf. on Computing for Sustainable Global Development (INDIACom)*, Delhi, India, 2025, pp. 1-6. DOI: 10.23919/INDIACom66777.2025.11115804.

[6] O. Dovhaniuk, T. Kyryliuk, and V. Deibuk, "Reversible Fault-Tolerant Encryption Using Extended Fredkin Gate With Reconfiguration," *Advances in Transdisciplinary Engineering*, Vol. 65, pp. 69 – 76. 2025. DOI: 10.3233/ATDE250113

[7] K. Rajesh and G. Umamaheswara Reddy, "FPGA implementation of encryption and decryption of a message using optimized reconfigurable reversible gate," *International Journal of Recent Technology and Engineering* (IJRTE), Vol. 8, No. 2, pp. 1654-1658, 2019.

[8] M. Bryk, K. Gracki, P. Kerntopf, M. Pawłowski, and A. Skorupski, "Encryption Using Reconfigurable Reversible Logic Gate and Its Simulation in FPGAs," in Proc. 23rd Int. Conf. "Mixed Design of Integrated Circuits and Systems" (MIXDES 2016), Łódź, Poland, 2016, pp. 203-206.

[9] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic," *International Journal of Applied Metaheuristic Computing* (IJAMC), Vol. 1, No. 4, pp. 25–41, 2010.

[10] Y. Kimura, S. Li, H. Sato, M. Fujita and R. Wille, "Improving Decision Diagram-Based Quantum Circuit Simulation Using Static Variable Ordering and Multinode Ring Communication," *IEEE Transactions on Quantum Engineering*, Vol. 7, 2500315, 2026. https://doi.org/10.1109/TQE.2026.3654543.

[11] D. Maslov, G. W. Dueck, and D. M. Miller, "Synthesis of Fredkin–Toffoli reversible networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 13, No. 6, pp. 765-769, 2005.

[12] T. N. Sasamal, A. K. Singh, and A. Mohan, "Reversible Logic Circuit Synthesis and Optimization using Adaptive Genetic Algorithm," *Procedia Computer Science*, Vol. 70, pp. 407-413, 2015.

[13] D. Volpe, G. Orlandi, and G. Turvani, "Improving the Solving of Optimization Problems: A Comprehensive Review of Quantum Approaches," *Quantum Reports,* Vol. 7, No. 1: 3, 2025. https://doi.org/10.3390/quantum7010003.

[14] X. Wang et al., "Hybrid Encoding Evolutionary Algorithm for Reversible Logic Synthesis," *Journal of Physics: Conference Series*, Vol. 2504, 012040, 2023.

[15] J. Han, X. Zhang, and X. Wang, "Application Research of Evolutionary Algorithm in Synthesis of Reversible Logic Circuits," *Journal of Physics: Conference Series*, Vol. 1237, 022083, 2019.

[16] A. Rajput, R. Kumawat, A. Srinivasulu, *et al.* Robust and compact reversible logic gate for low-power and high-performance computing. *Sci Rep* Vol. 16, 154, 2026. https://doi.org/10.1038/s41598-025-29369-5.

[17] M. Li, Y. Zheng, M. S. Hsiao, and C. Huang, "Reversible logic synthesis through ant colony optimization," in Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden, Germany, 2010, pp. 307-310. DOI: 10.1109/DATE.2010.5457190.

[18] K. Podlaski, "Ant Colony Optimization Implementation for Reversible Synthesis in Walsh-Hadamard Domain," Lecture Notes in Computer Science, vol. 12141, pp. 230-243, 2020. DOI: 10.1007/978-3-030-50426-7_18.

[19] T. Kyryliuk, M. Palahuta, and V. Deibuk, "Using artificial intelligence methods for the optimal synthesis of reversible networks," Radioelectronic and Computer Systems, no. 4, pp. 112-122, 2024. DOI: 10.32620/reks.2024.4.10.

[20] O. Dovhaniuk, and V. Deibuk "Synthesis and Implementation of Reconfigurable Reversible Generalized Fredkin Gate," in Proc. 2021 IEEE XIIth Int. Conf. on Electronics and Information Technologies (ELIT), 2021, pp. 165-169.

[21] B. Murali Krishna, K. C. Sri Kavya, P. V. S. Sai Kumar, K. Karthik, and Y. Siva Nagababu, "FPGA implementation of image cryptology using reversible logic gates," *International Journal of Advanced Trends in Computer Science and Engineering,* Vol. 9, No. 3, pp. 2522-2526, 2020.

[22] A. Patel, I. L. Markov, and J. P. Hayes, "Fault testing for reversible circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 23, No. 8, pp. 1220-1230, 2004.

[23] D. Kheirandish, M. Haghparast, M. Reshadi, and M. Hosseinzadeh, "Efficient techniques for fault detection and location of multiple controlled Toffoli-based reversible circuit," Quantum Information Processing, vol. 20, no. 11, art. 370, 2021. DOI: 10.1007/s11128-021-03292-w.

[24] L. Burgholzer, R. Wille, "Exploiting Reversible Computing for Verification: Potential, Possible Paths, and Consequences",

Proceedings of the 28th Asia and South Pacific Design Automation Conference (ASPDAC '23), pp. 429–435, 2023.

[25] H. M. Gaur, A. K. Singh, and U. Ghanekar, "Design for Stuck-at Fault Testability in Toffoli–Fredkin Reversible Circuits", *National Academy Science Letters,* Vol. 44, pp. 215–220, 2021.

[26] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An Online Resource for Reversible Functions and Reversible Circuits," in *Proc. 38th Int. Symp. on Multiple-Valued Logic (ISMVL)*, Dallas, TX, USA, 2008, pp. 220-225, 2008.

Authors' Profiles



Taras Kyryliuk Taras Kyryliuk earned his master's degree in computer engineering from Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine, in 2012.

He has been a professional Software Engineer since 2012. He is currently a .NET Developer at additiv, specializing in enterprise software development. He is also a PhD student in computer science at Yuriy Fedkovych Chernivtsi National University. His current and previous research interests include reversible logic synthesis, quantum computing, fault-tolerance models, and the application of metaheuristic algorithms (such as Ant Colony Optimization and Genetic Algorithms) for hardware validation on FPGAs.



Mykhailo Palahuta earned his master's degree in computer engineering from Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine, in 2023.

He has been a professional Software Engineer since 2019, currently working at SoftServe. He is also a PhD student at the Software Engineering Department of Yuriy Fedkovych Chernivtsi National University. His primary research interests include reversible logic synthesis, quantum computing, fault-tolerance models, and the application of metaheuristic algorithms (such as Ant Colony Optimization and Genetic Algorithms) for hardware validation on FPGAs.



Oleksii Dovhaniuk earned his bachelor's and master's degrees in computer engineering and IoT technologies from Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine, in 2021. He also completed advanced studies in Germany and Austria.

In 2022, he began his PhD studies. He currently develops medical resource scheduling applications at University College Cork, Cork, Ireland. His research interests include genetic algorithms, meta-heuristics, artificial intelligence, and reversible computing.



Vitaly Deibuk is a Professor at the Computer Systems and Networks Department of Yuriy Fedkovych Chernivtsi National University, Ukraine.

He is actively engaged in academic teaching and advanced research. His primary areas of scientific interest include quantum computing, reversible circuit synthesis, metaheuristic optimization, and artificial intelligence.

How to cite this paper

Taras Kyryliuk, Mykhailo Palahuta, Oleksii Dovhaniuk, Vitaly Deibuk, "FPGA Implementation of a Fault-tolerant Encryptor Based on Reconfigurable Fredkin Gates", International Journal of Information Technology and Computer Science(IJITCS), Vol.18, No.3, pp.1-12, 2026. DOI:10.5815/ijitcs.2026.03.01