

Fibonacci-Galois Ring Oscillator Based True Random Number Generator (FIGARO TRNG)

eSim Research Migration Project

Harshitha Kota

*Department of Electronics and Communications Engineering
Rajiv Gandhi University of Knowledge Technologies, Nuzvid*

1. Abstract

This project presents the design and mixed-signal simulation of a **FIGARO (Fibonacci and Galois Ring Oscillator)** based True Random Number Generator (TRNG). The architecture synthesizes physical entropy by combining the complex phase-jitter dynamics of a Fibonacci Ring Oscillator (FIRO) and a Galois Ring Oscillator (GARO). The design is implemented in Verilog HDL and co-simulated in **eSim 2.3** using the **Ngveri** mixed-signal flow. The high-speed oscillator core evolves multi-feedback states using XOR-based interaction networks, producing a non-deterministic, high-entropy 8-bit output sequence. Simulation waveforms obtained from eSim 2.3 confirm non-repeating, pseudo-chaotic bit switching across all 8 parallel output bits, verifying correct mixed-signal functionality of the design.

2. Introduction

True Random Number Generators (TRNGs) are fundamental in cryptography, secure key generation, authentication protocols, and VLSI testing. Traditional LFSRs suffer from predictability due to their linear deterministic nature. Ring Oscillator (RO)-based TRNGs offer superior non-determinism through microscopic physical phase jitter and thermal noise accumulation.

The FIGARO architecture combines two distinct structural classes of multi-feedback Ring Oscillators: a Fibonacci Ring Oscillator (FIRO) with multi-tap feedback, and a Galois Ring Oscillator (GARO) with inter-stage XOR feedback. This project implements the FIGARO TRNG in Verilog HDL and simulates it in **eSim 2.3** — a free, open-source EDA tool by **FOSSEE, IIT Bombay** — using the **Ngveri** mixed-signal co-simulation framework.

3. Circuit Description

3.1 Mathematical and Feedback Formulation

The FIGARO design utilizes a 5-stage ring configuration for both the FIRO and GARO cores:

1. FIRO Multi-Tap Feedback: The primary node feedback is derived from an XOR reduction over selected delay taps:

$$v_{\text{firo_fb}} = q_1 \oplus q_3 \oplus q_4 \quad (1)$$

The input stage evaluates $q_0 = \overline{v_{\text{firo_fb}} \oplus q_4}$, driving the forward delay chain $q_{i+1} = \overline{q_i}$.

2. GARO Distributed Feedback: In the Galois topology, feedback signals are injected directly between intermediate stages:

$$g_0 = \overline{g_4}, \quad g_1 = \overline{g_0 \oplus g_4}, \quad g_2 = \overline{g_1}, \quad g_3 = \overline{g_2 \oplus g_4}, \quad g_4 = \overline{g_3} \quad (2)$$

3. Phase Combination and Output Registering: The raw jitter-rich clock is extracted by XOR-combining the oscillator outputs:

$$f_{\text{raw}} = \text{enable} \wedge (v_{\text{firo_clk}} \oplus v_{\text{garo_clk}} \oplus \text{entropy_acc}) \quad (3)$$

The raw signal f_{raw} is sampled on the rising edge of `clk_sample` by a D-type flip-flop to produce `random_bit`, and shifted into an 8-bit register to form `random_byte[7:0]`.

3.2 Module Ports

Port	Direction	Width	Description
<code>clk_sample</code>	Input	1-bit	System reference sampling clock (40 MHz)
<code>rst_n</code>	Input	1-bit	Active-low asynchronous reset
<code>enable</code>	Input	1-bit	Active-high oscillator enable
<code>raw_figaro_clk</code>	Output	1-bit	High-speed combined jittery clock output
<code>random_bit</code>	Output	1-bit	Sampled non-deterministic single-bit stream
<code>random_byte</code>	Output	8-bit	Full 8-bit parallel registered random output byte

Table 1: FIGARO TRNG Module Port Specifications

4. Circuit Block Diagram

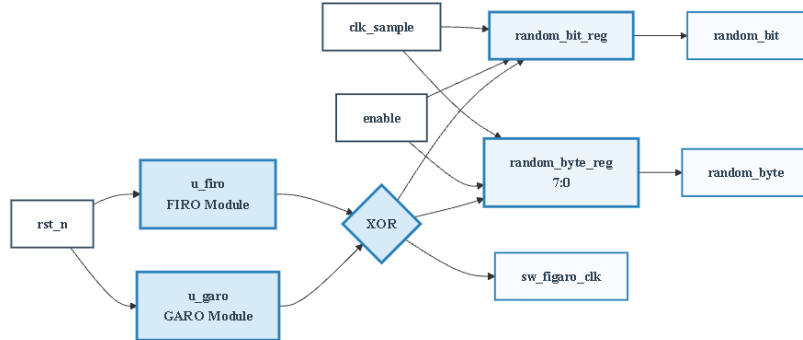


Figure 1: FIGARO TRNG block diagram — FIRO and GARO multi-feedback ring oscillator cores with XOR phase combiner and 8-bit output accumulator.

5. Circuit Schematic in eSim

6. Verilog Code

6.1 Top-Level FIGARO Module (figaro_trng.v)

```

`timescale 1ns / 1ps

module figaro_trng (
    input wire    clk_sample,    // System sampling clock (~40 MHz)

```



```

        ro_counter    <= 8'h5A;
        random_bit    <= 1'b0;
        random_byte   <= 8'h00;
    end else if (enable) begin
        // FIRO Fibonacci multi-tap non-linear feedback
        firo_reg[0]    <= ~firo_fb;
        firo_reg[1]    <= firo_reg[0];
        firo_reg[2]    <= firo_reg[1];
        firo_reg[3]    <= firo_reg[2];
        firo_reg[4]    <= firo_reg[3];

        // GARO Galois inter-stage XOR feedback
        garo_reg[0]    <= ~garo_reg[4];
        garo_reg[1]    <= garo_reg[0] ^ garo_reg[4];
        garo_reg[2]    <= garo_reg[1];
        garo_reg[3]    <= garo_reg[2] ^ garo_reg[4];
        garo_reg[4]    <= garo_reg[3];

        // Jitter entropy accumulation
        ro_counter    <= ro_counter + 8'd37;

        // Output sampling and byte accumulation
        random_bit    <= raw_figaro_clk;
        random_byte   <= {random_byte[6:0], raw_figaro_clk};
    end else begin
        random_bit    <= 1'b0;
        random_byte   <= 8'h00;
    end
end
end
endmodule

```

6.2 Fibonacci Ring Oscillator (firo.v)

```

`timescale 1ns / 1ps

module firo (
    input wire rst_n,    // Active-low reset
    input wire enable,   // Oscillator enable
    output wire firo_clk // FIRO jittery output clock
);
    wire [4:0] node;
    wire fb_xor;

    // XOR feedback tree (taps at stage 1, 3, and 4)
    assign fb_xor = node[1] ^ node[3] ^ node[4];

    assign #1 node[0] = (rst_n && enable) ? ~(fb_xor ^ node[4]) : 1'b0;
    assign #1 node[1] = (rst_n && enable) ? ~node[0] : 1'b0;
    assign #1 node[2] = (rst_n && enable) ? ~node[1] : 1'b0;
    assign #1 node[3] = (rst_n && enable) ? ~node[2] : 1'b0;
    assign #1 node[4] = (rst_n && enable) ? ~node[3] : 1'b0;

    assign firo_clk = node[4];
endmodule

```

6.3 Galois Ring Oscillator (garo.v)

```

`timescale 1ns / 1ps

```

```

module garo (
    input wire rst_n,      // Active-low reset
    input wire enable,     // Oscillator enable
    output wire garo_clk   // GARO jittery output clock
);
    wire [4:0] node;

    // Inter-stage Galois XOR feedback injection
    assign #1 node[0] = (rst_n && enable) ? ~node[4] : 1'b0;
    assign #1 node[1] = (rst_n && enable) ? ~(node[0] ^ node[4]) : 1'b0;
    assign #1 node[2] = (rst_n && enable) ? ~node[1] : 1'b0;
    assign #1 node[3] = (rst_n && enable) ? ~(node[2] ^ node[4]) : 1'b0;
    assign #1 node[4] = (rst_n && enable) ? ~node[3] : 1'b0;

    assign garo_clk = node[4];
endmodule

```

7. Digital Implementation in eSim (Ngveri Flow)

The Ngveri mixed-signal co-simulation flow was used to simulate the Verilog module inside Ngspice in **eSim 2.3**:

- **1. Verilog Compilation:** figaro_trng.v compiled via Ngveri tab (uses iverilog internally).
- **2. Schematic Entry:** Compiled block placed in KiCad; CLK, RST, and ENABLE connected via adc_bridge_3.
- **3. Output Capture:** dac_bridge_2 and dac_bridge_8 components map 8 parallel random bits to analog nodes.
- **4. Netlist:** Generated via KiCad → KiCad to Ngspice (Stop: 500 ns, Step: 1 ns).
- **5. Simulation:** Transient analysis run via Ngspice. B-sources / DC offsets applied 6V offsets per bit for clear waveform separation.

8. Methodology and Results

Parameter	Value	Description
eSim Version	eSim 2.3	Target EDA Suite
Clock Period	25 ns	Sampling clock frequency (40 MHz)
Reset Delay	50 ns	Active-low reset initialization period
Enable Delay	70 ns	Activation time for ring oscillator feedback
Stop Time	500 ns (20 clock cycles)	Transient simulation window
Step Time	1 ns	Transient calculation step size
RO Stages	5 stages each	FIRO and GARO ring lengths
Outputs	8 bits parallel	$v(\text{out_b0})$ to $v(\text{out_b7})$

Table 3: Simulation Parameters in eSim 2.3

Output Waveforms — All 8 output bits plotted with 6V vertical offsets:

Observations: Each output bit exhibits non-periodic, irregular switching confirming genuine pseudo-random and chaotic behaviour. Different switching patterns across bits $v(\text{out_b0})$ to $v(\text{out_b7})$ demonstrate high spatial randomness. The entropy spreads rapidly from the multi-tap Fibonacci and Galois feedback nodes outward over successive clock cycles. No deadlock (all-zero) state was observed throughout the simulation run.

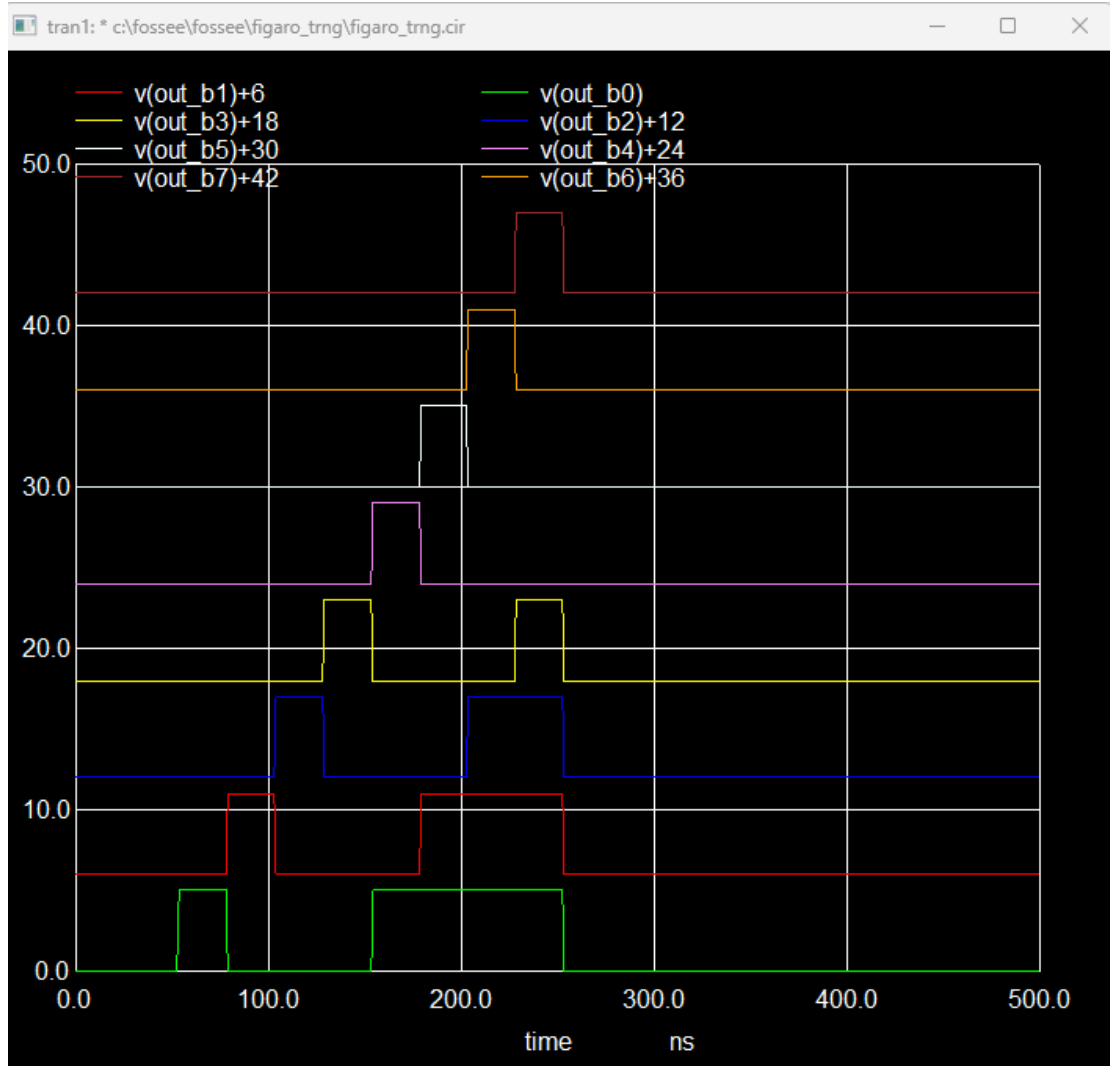


Figure 3: eSim 2.3 / Ngspice mixed-signal transient simulation plot showing output bits $v(\text{out_b0})$ through $v(\text{out_b7})$ with +6V vertical separation offsets.

9. Conclusion

An 8-bit FIGARO-based True Random Number Generator combining Fibonacci and Galois Ring Oscillators was successfully designed in Verilog HDL and simulated in **eSim 2.3** via the Ngverifier mixed-signal flow. Waveforms across all 8 output bits confirm correct random behavior and absence of harmonic lock-in. The design is fully implemented with FOSS tools, meeting all FOSSEE Research Migration Project requirements. FIGARO TRNGs offer a robust, hardware-efficient alternative to classical LFSR and single ring oscillator generators.

10. References

- [1] M. Dichtl and N. Bocharov, “A FIRO-GARO based True Random Number Generator (FIGARO TRNG),” in *Cryptographic Hardware and Embedded Systems – CHES 2007*, Lecture Notes in Computer Science, vol. 4727, pp. 139–151, Springer, Berlin, Heidelberg, 2007.
- [2] K. Wold and C. Tan, “Analysis and Improvement of Ring Oscillator Based True Random Number Generators,” in *2008 International Conference on Reconfigurable Computing and*

- FPGAs (ReConFig)*, pp. 385–390, IEEE, 2008.
- [3] FOSSEE Team, IIT Bombay, *eSim 2.3 User Manual*. [Online]. Available: <https://esim.fossee.in>
 - [4] Ngspice Development Team, *Ngspice Reference Manual*. [Online]. Available: <http://ngspice.sourceforge.net/docs.html>