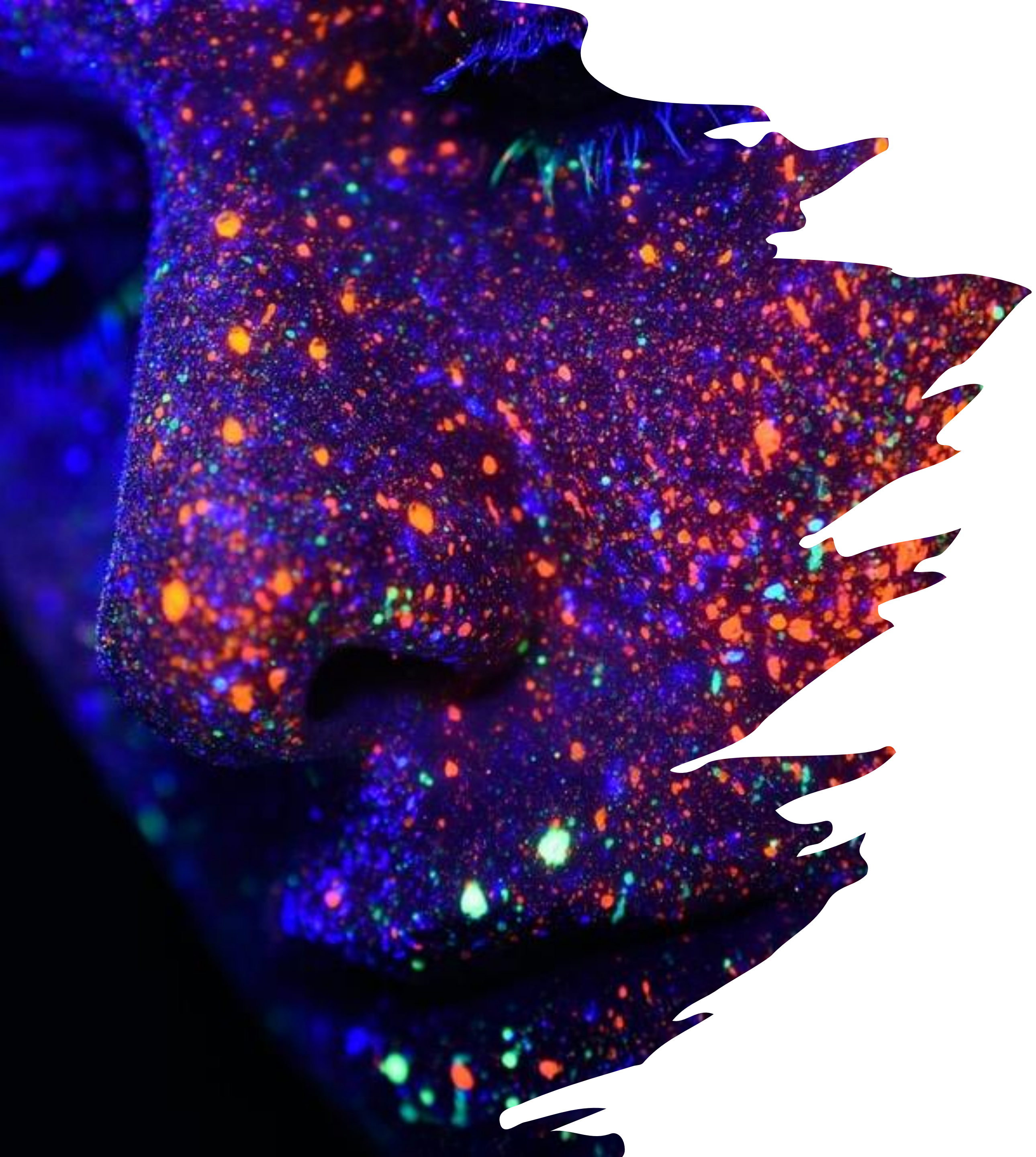
Image Encryption Process based on Chaotic Synchronization Phenomena

By S P Sharan – 108118095 – ECE A



Abstract

- This project presents an image encryption scheme, which uses a chaotic True Random Bits Generator (TRBG).
- The chaotic TRBG is based on the coexistence of two different synchronization phenomena.
  - The first one is the well-known **complete chaotic synchronization**
  - while the second one is a recently new proposed synchronization phenomenon, the **inverse p-lag synchronization**.

Overview of Project

This coexistence is observed in the case of two mutually coupled identical nonlinear circuits. The nonlinear circuit, which is used, produces **double-scroll chaotic attractor**s.

The initial conditions of the coupled system and the values of the circuit's parameters serve as **the private key** of the proposed cryptographic scheme.

This bit-sequence has then been used to **encrypt and decrypt gray-scale images**.

True Random Number Generators (TRNGs)

- Generators that produce random sequences can be classified into three types:
  - True Random Number Generators (TRNGs)
  - Pseudo-Random Number Generators (PRNGs)
  - Hybrid Random Number Generators (HRNGs)
- TRNGs take advantage of nondeterministic sources, which come from **an unpredictable natural process in a physical or hardware device** that can output a sequence of statistically independent data.

Cryptography Scheme

- Idea of our method is to encrypt a gray-scale image via **a chaotic True Random Bits Generator (TRBG)**, which is based on the interaction between two **mutually coupled identical chaotic circuits**.
- According to a binary sequence generated from the chaotic generator, **the pixels of the gray-scale image XOR-ed to the predetermined keys**.

Types of Synchronization

- Complete Synchronization

The most well-known type of synchronization is the complete or full synchronization, in which the interaction between two identical coupled chaotic systems leads to **a perfect coincidence of their chaotic trajectories**

$$\boldsymbol{x_1(T) = x_2(t)\ as\ t \to \infty}$$

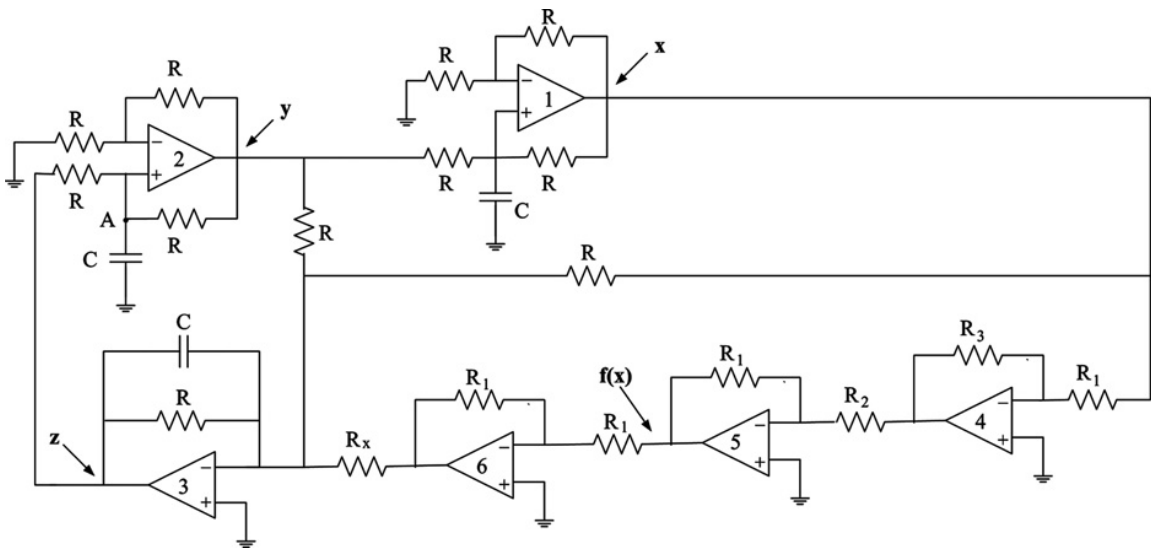
- Inverse π-lag Synchronization

When a coupled system is in a phase locked (periodic) state, depending on the coupling factor and it can be characterized by eliminating the sum of two relevant periodic signals (x_1 and x_2) with a time lag which is equal to $\frac{T}{2}$, where T is the period of the signals x_1 and x_2

$$\boldsymbol{x_1(t) = -x_2(t+\tau), \qquad where\ \tau = \frac{T}{2}}$$

The Chaotic True Random Bits Generator

- The autonomous nonlinear circuit (shown in figure 1), which has been used, can produce double-scroll chaotic attractors.
- This has the characteristic of two attractors, between which the process state will oscillate.
- The state equations of the chosen system are the following (shown in figure 2)
- Explanation
    - Op-Amp 1 – Integrator
    - Op-Amp 2 – Integrator
    - Op-Amp 3 – Adder



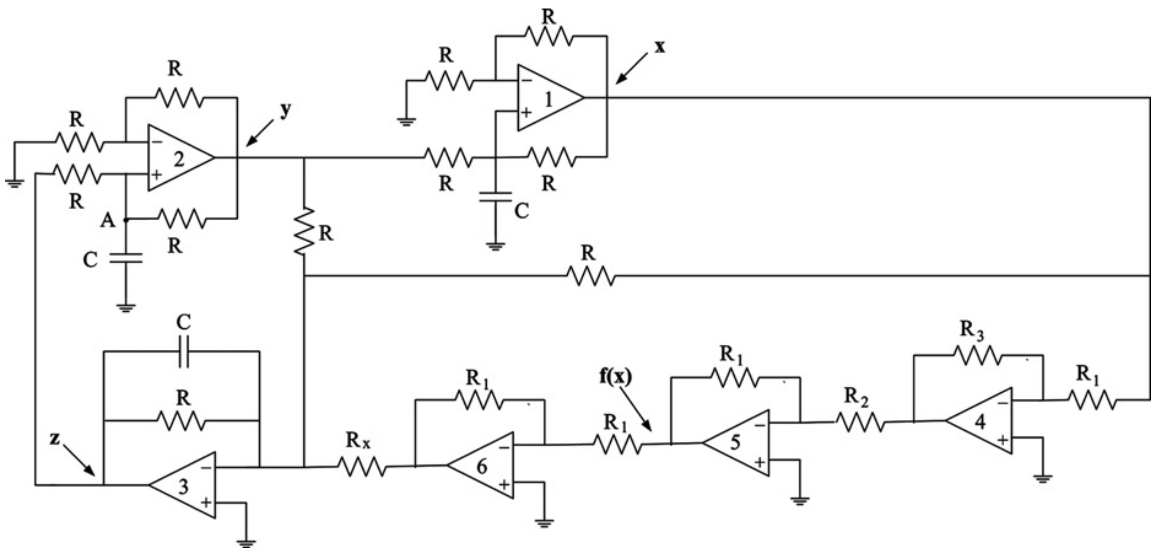
$$\begin{cases} \frac{dx}{dt} = y \\ \frac{dy}{dt} = z \\ \frac{dz}{dt} = -\alpha \cdot (x+y+z) + b \cdot f(x) \end{cases} \tag{3}$$

where α and b are the circuit parameters and are defined as follows:

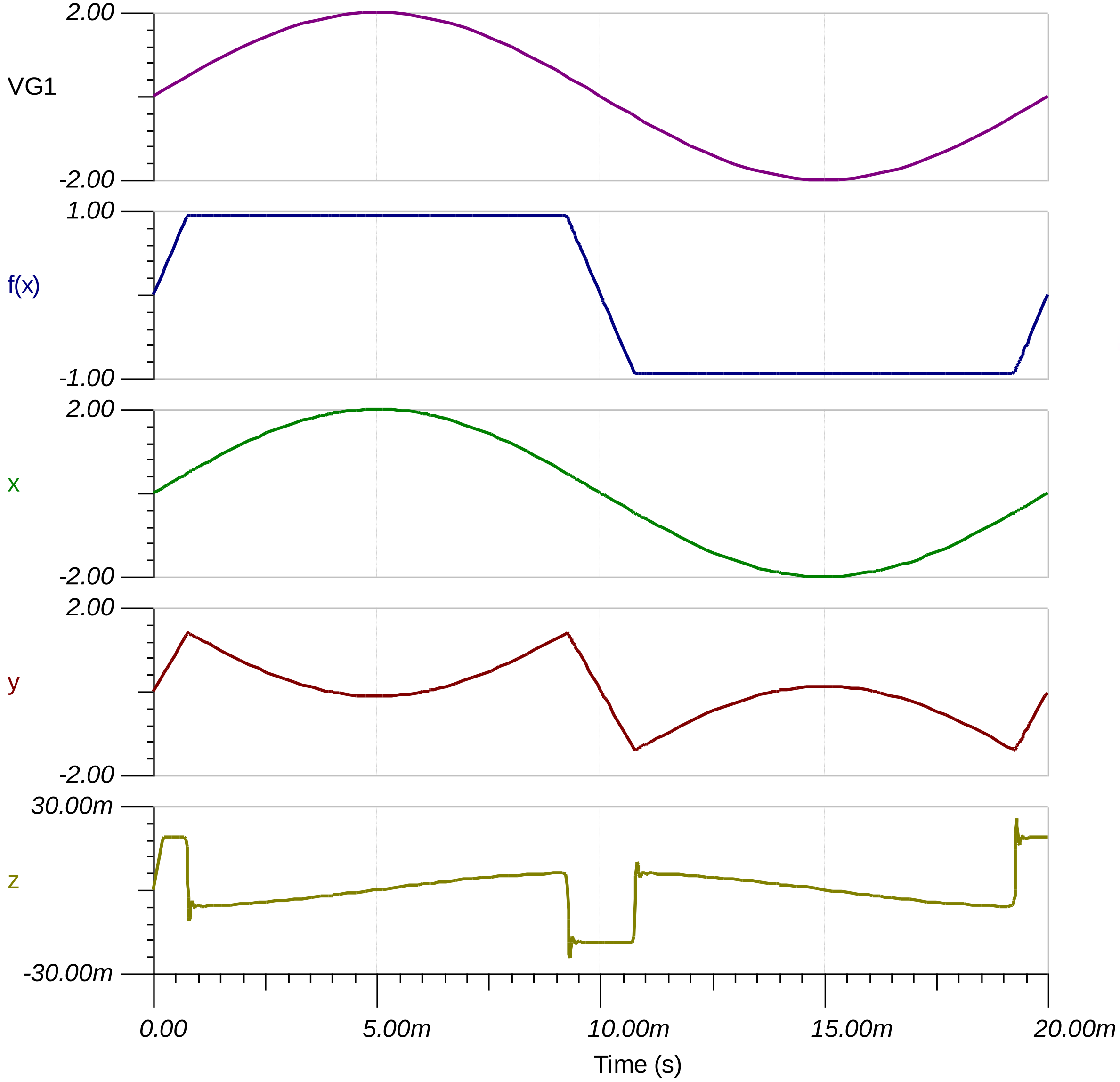
$$\alpha = (R \cdot C)^{-1} \; b = (R_X \cdot C)^{-1} \tag{4}$$

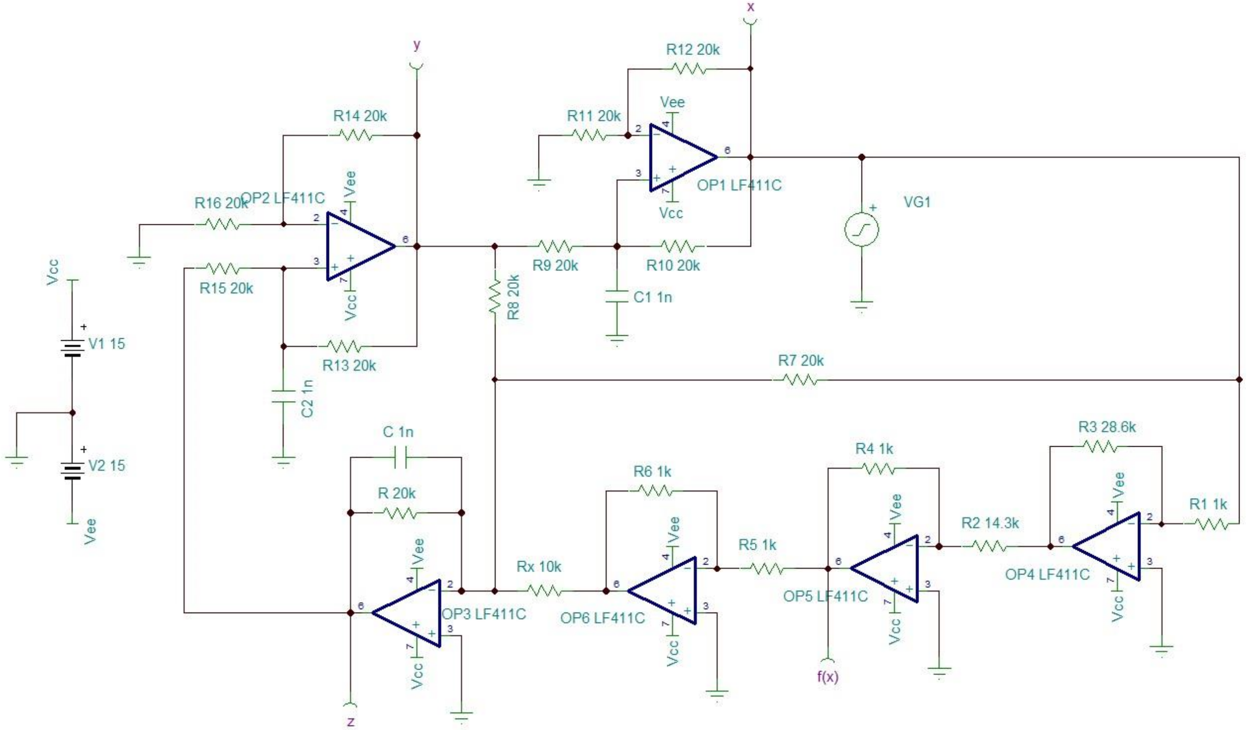
Saturation Function

- The function f(x) in system's equation is a saturation function which represents the voltage at the output of the operational amplifier numbered as "5" and is defined by the following expression
- So, the function f(x) is implemented in such a way that the saturation plateaus are ± 1 and the slope of the intermediate linear region is $k = \frac{R_3}{R_2}$.



$$f(x) = \begin{cases} 1 & \text{if} \quad x > \frac{R_2}{R_3} \cdot 1\ \mathrm{V} \\ \frac{R_3}{R_2} \cdot x & \text{if} \quad -\frac{R_2}{R_3} \cdot 1\ \mathrm{V} \le x \le \frac{R_2}{R_3} \cdot 1\ \mathrm{V} \\ -1 & \text{if} \quad x < -\frac{R_2}{R_3} \cdot 1\ \mathrm{V} \end{cases}$$

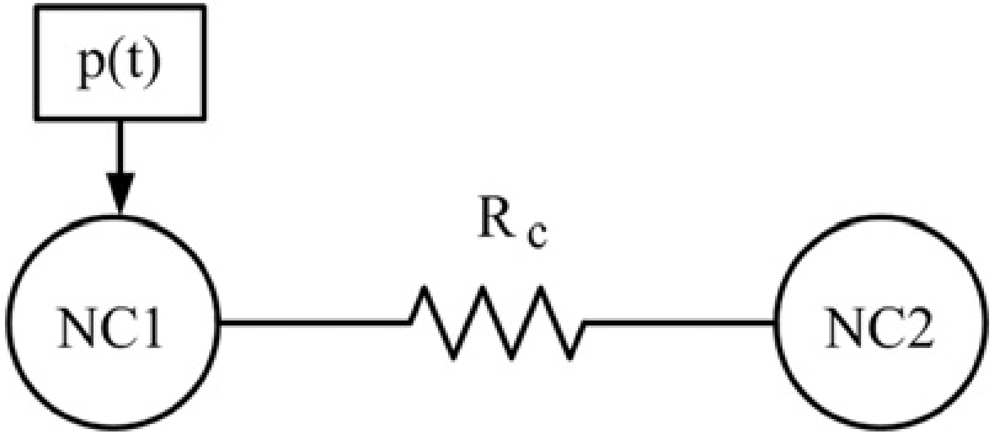




$$f(x) = \begin{cases} 1 & \text{if} \quad x > \frac{R_2}{R_3} \cdot 1\ \text{V} \\ \frac{R_3}{R_2} \cdot x & \text{if} \quad -\frac{R_2}{R_3} \cdot 1\ \text{V} \leq x \leq \frac{R_2}{R_3} \cdot 1\ \text{V} \\ -1 & \text{if} \quad x < -\frac{R_2}{R_3} \cdot 1\ \text{V} \end{cases}$$

Mutually coupled nonlinear circuits

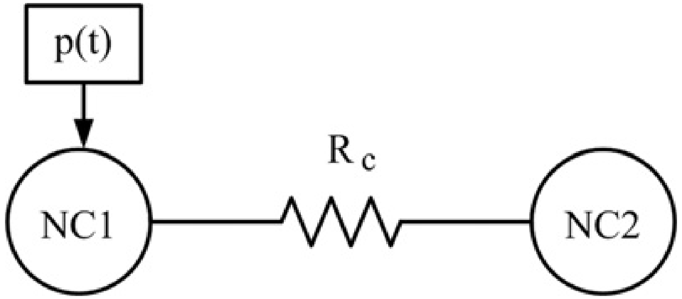
- As it is previously mentioned, the proposed TRBG uses a system of two mutually coupled identical double-scroll chaotic circuits of this type.
- For this reason, the coupling of the identical nonlinear circuits is achieved via a linear resistor RC connected between the nodes A of each circuit.
- The first three equations of system (6) describe the first of the two coupled identical nonlinear circuits (NC1), while the other three describe the second one (NC2).
- The coupling coefficient is $\xi = \frac{R}{R_C}$ and it is present in the equations of both circuits, since the coupling between them is bidirectional.

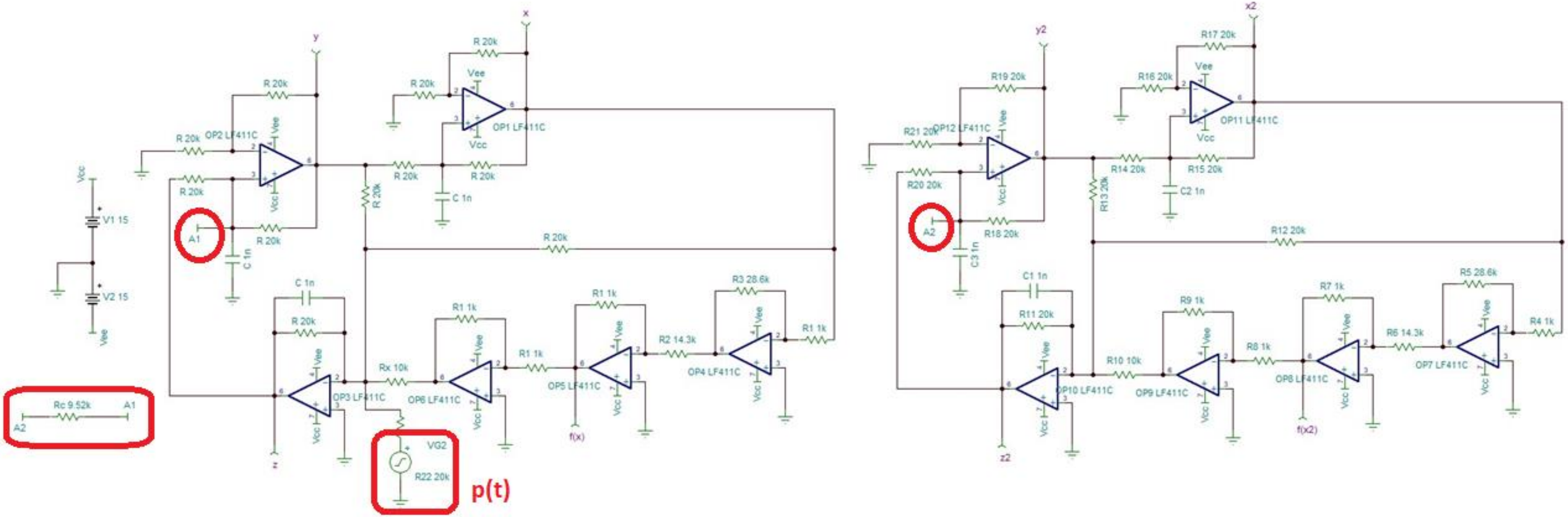


So, the state equations, describing the coupled system, are

$$\begin{cases} \frac{dx_1}{dt} = y_1 \\ \frac{dy_1}{dt} = z_1 + \xi \cdot (y_2 - y_1) \\ \frac{dz_1}{dt} = -\alpha \cdot (x_1 + y_1 + z_1) + b \cdot f(x_1) - p(t) \\ \frac{dx_2}{dt} = y_2 \\ \frac{dy_2}{dt} = z_2 + \xi \cdot (y_1 - y_2) \\ \frac{dz_2}{dt} = -\alpha \cdot (x_2 + y_2 + z_2) + b \cdot f(x_2) \end{cases} \tag{6}$$

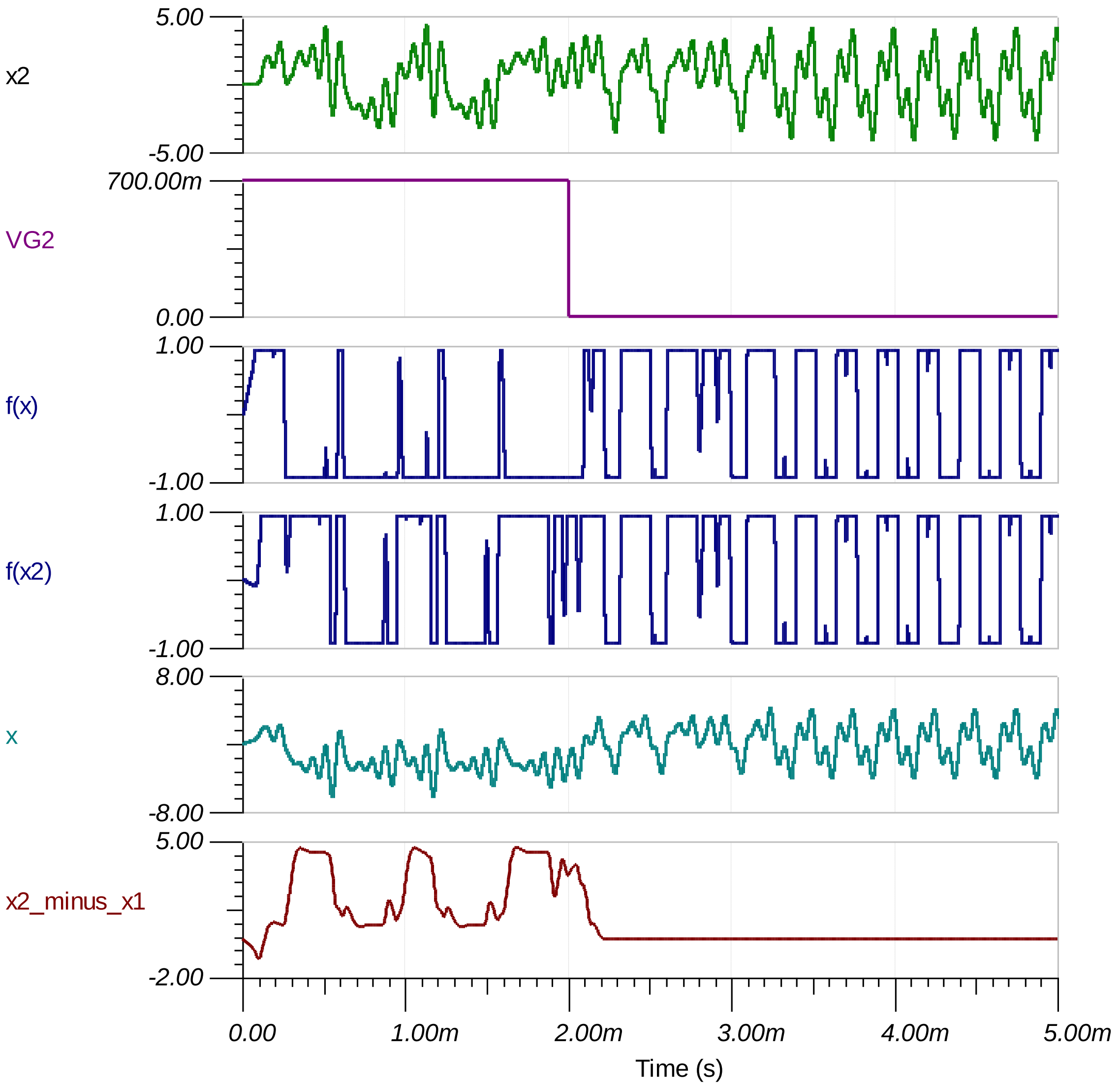
Mutually coupled nonlinear circuits





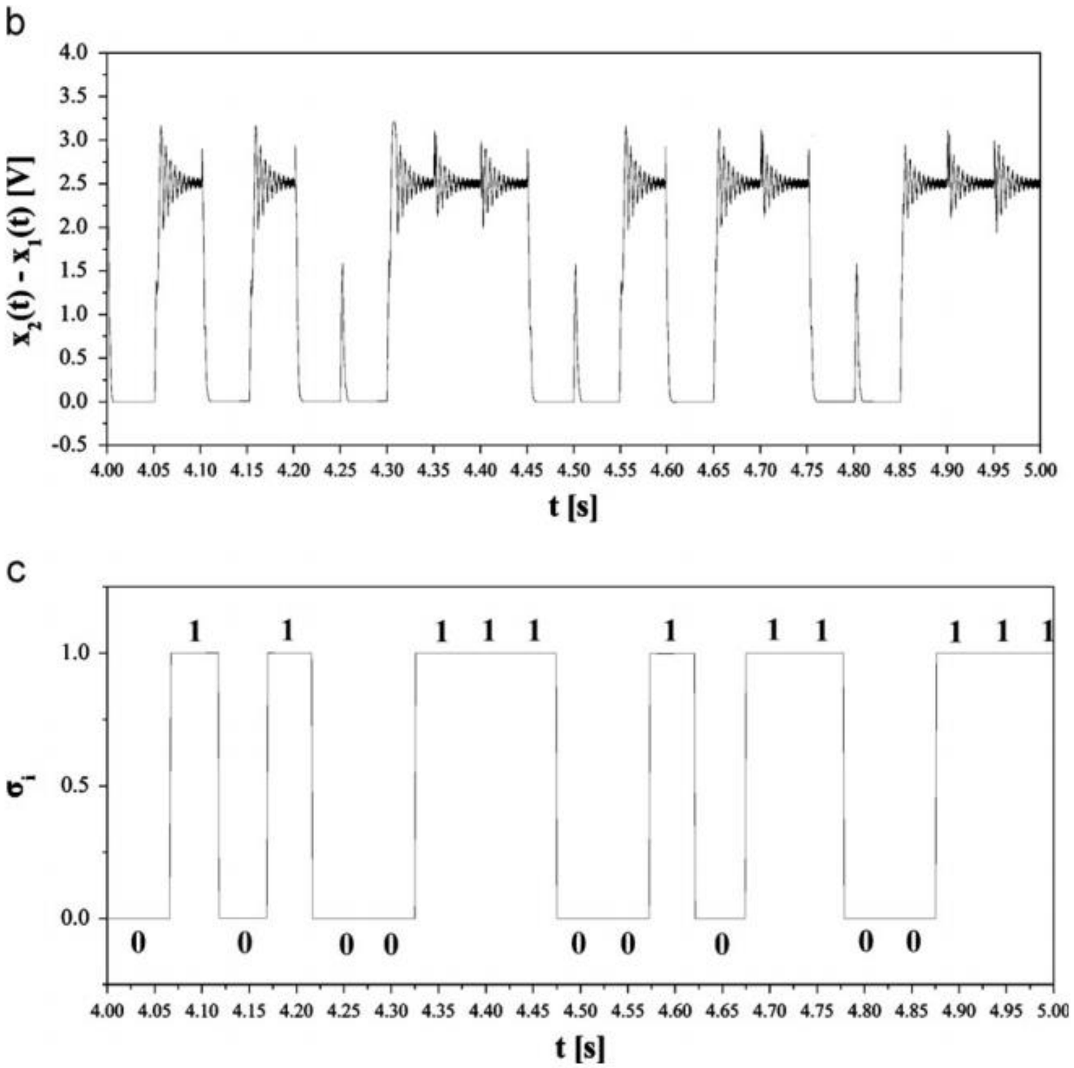
The chaotic true random bits generator scheme

- p(t) is an external source which produces pulses that are necessary, as a perturbation, for changing the initial conditions of the system and therefore the synchronization state of the coupled system (inverse p-lag or complete synchronization).
- In detail, this source produces a pulse train of amplitude 0.7 V having a duty cycle of 4%. Thus, the pulse duration is 2 ms, while the period of the pulse train is 50 ms.



Bit-Quantization and De-Skewing

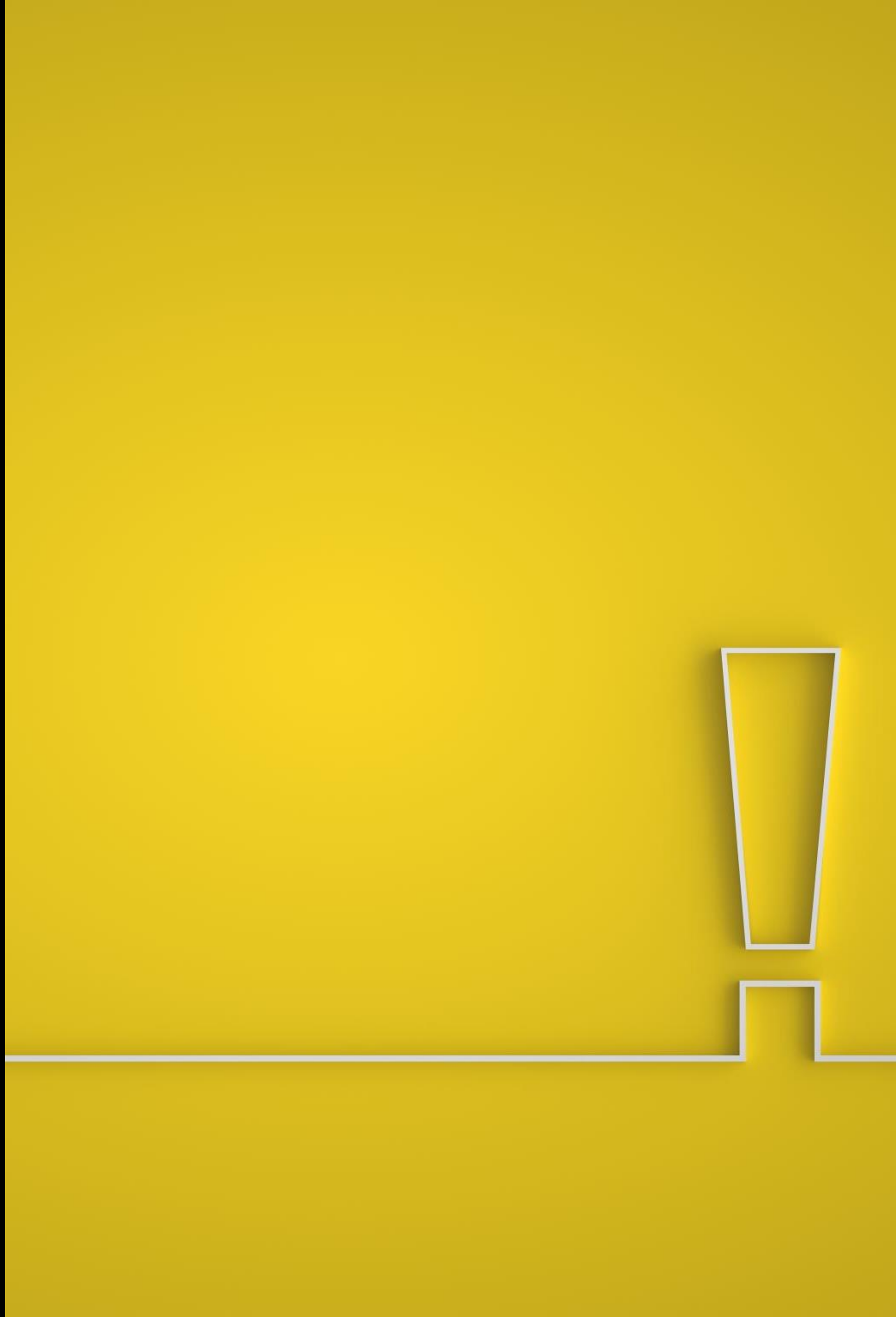
- In the second block of the proposed TRBG, the two different levels of the output signal $[x_2(t) – x_1(t)]$ are quantized to ‘‘0’’ and ‘‘1’’ according to the following equation



$$\sigma_i = \begin{cases} 0 & \text{if } x_2(t) - x_1(t) < 1 \text{ V} \\ 1 & \text{if } x_2(t) - x_1(t) > 1 \text{ V} \end{cases}$$

De-Skewing Techniques

- It is known that a natural source of random bits may not give unbiased bits as direct output.
- We must extract unbiased bits from a defective generator with unknown bias. These are called de-skewing techniques.
- Done by converting the bit pair ‘‘01’’ into an output ‘‘0’’, converting ‘‘10’’ into an output ‘‘1’’, while the pairs ‘‘11’’ and ‘‘00’’ are discarded.
- Decreases throughput because of generating approximately 1 bit from 4 bit.



Encrypting the Image

Step 1:

- The scheme finds the pixel size MxN of the image, where M and N represent row and column of the image.
- The pixels are arranged by order from left to right and top to bottom. Then an image data set, in which each element is the decimal gray-scale value of the pixel (0–255), is produced.
- Finally each decimal value is converted to a binary equivalent number and in the end a one-dimensional matrix B is produced.

Step 2:

- The matrix A which is a binary sequence produced by the chaotic TRBG, and the above-mentioned matrix B produces a third one-dimensional matrix C by using the XOR function: $C = A \oplus \mathrm{B}$.

Step 3:

- The produced in the previous step matrix C is converted to the encrypted image by the inverse process of step 1.

And follow the Inverse Operation for decryption

Implementation

- The was made in Tina TI and simulations run over night as an estimated of 131072* data points are required.
- The output obtained was exported to a txt file and parsed in python for processing.
- Python Modules used are: Numpy (array handling), Open-CV (picture handling)

```
def quantizer(values):
    return np.floor(values)


def unpack_bits_to_uint8(bits):
    output = []
    for i in range(0, len(bits), 8):
        bit_8 = bits[i:i + 8]
        output.append(int("".join(str(x) for x in bit_8), 2))
    return np.array(output)


def deskewer(bits):
    deskewed = []
    for i in range(0, len(bits), 2):
        if bits[i] == 0 and bits[i + 1] == 1:
            deskewed.append(0)
        if bits[i] == 1 and bits[i + 1] == 0:
            deskewed.append(1)
    return np.array(deskewed)

flattened_array = input_img.flatten()

# Random Stream is loaded from txt file after exporting via TinaTI
random_bits = quantizer(random_stream)
deskewed_bits = deskewer(random_bits)
deskewed_uint8 = unpack_bits_to_uint8(deskewed_bits)

encrypted_stream = np.bitwise_xor(flattened_array, deskewed_uint8)
encrypted_img = np.resize(encrypted_stream, (64, 64))
encrypted_img = encrypted_img.astype(np.uint8)

decrypted_stream = np.bitwise_xor(encrypted_stream, deskewed_uint8)
decrypted_img = np.resize(decrypted_stream, (64, 64))
decrypted_img = decrypted_img.astype(np.uint8)
```

* → (64x64 image) × (8-bit representation) × (4-bit overhead)

Final Outputs

Input Image	Encrypted Image	Decrypted Image