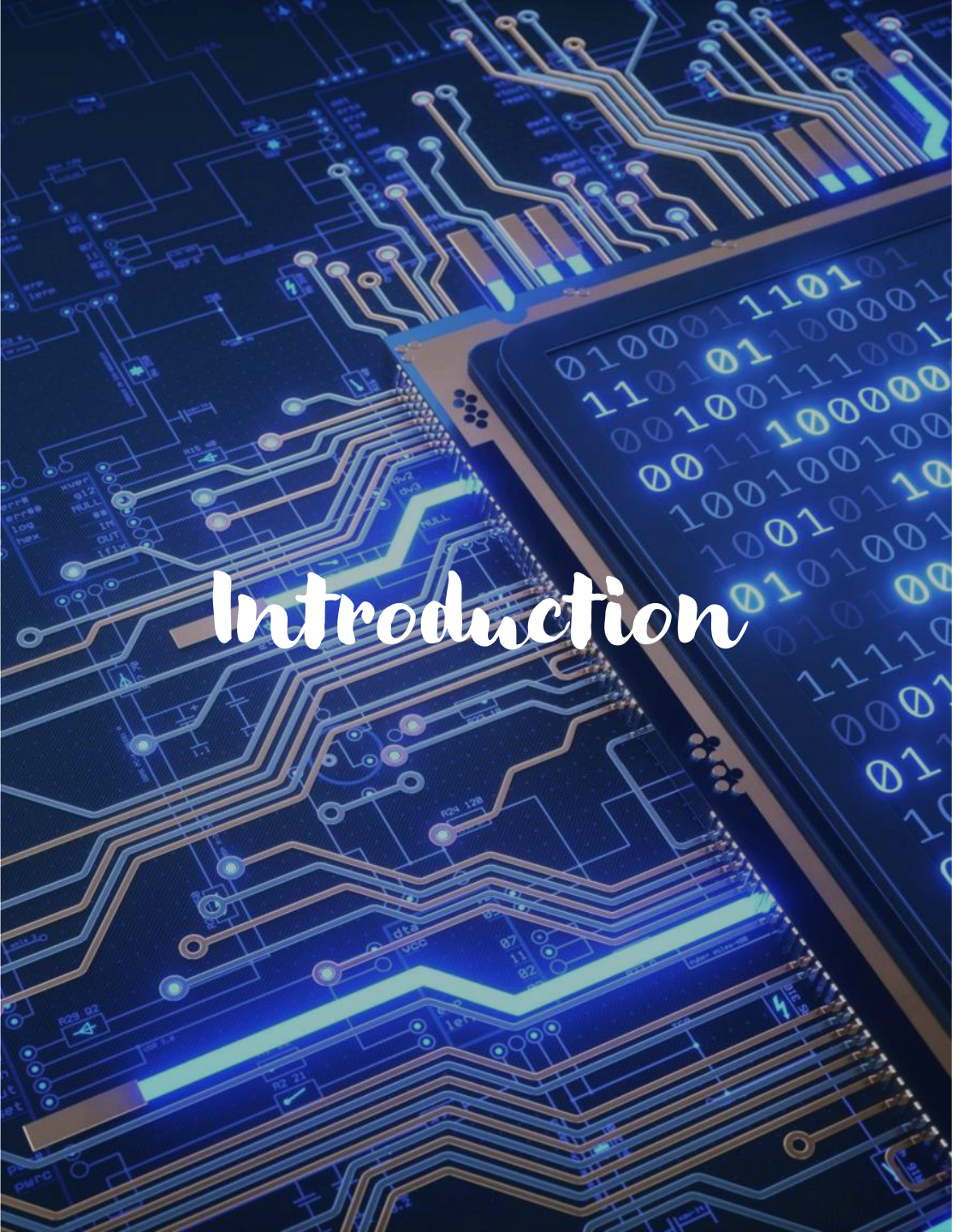
Theoretical Analysis on Henon Map

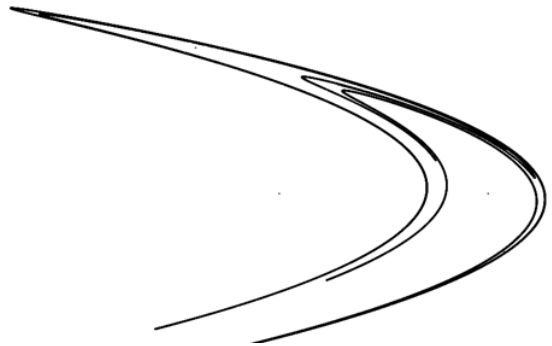
SAMARTH SUDARSHAN INAMDAR

CED18I045



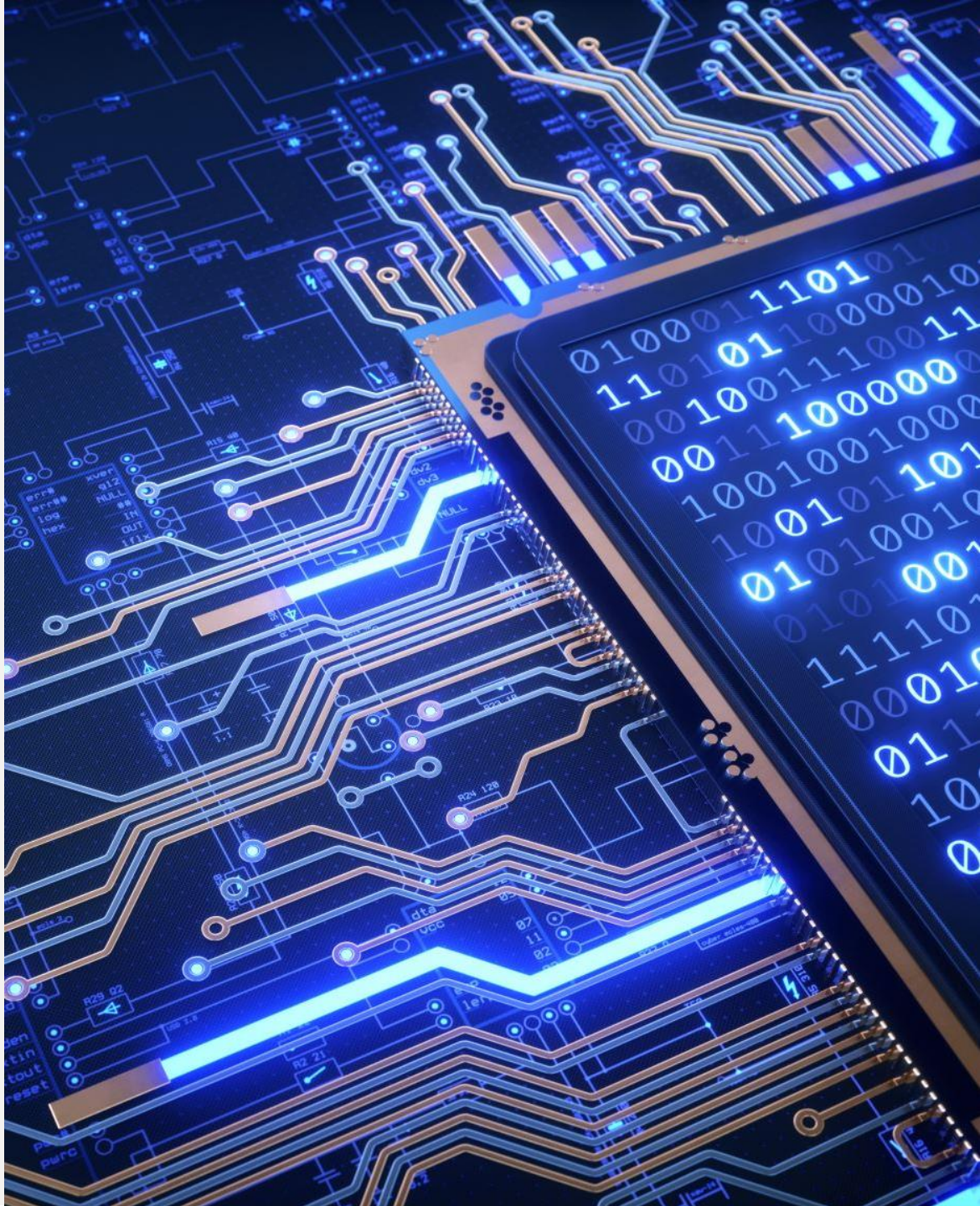
Introduction

The Henon map is an iterated discrete-time dynamical system that exhibits chaotic behavior in two-dimension. It is sometimes called Hénon-Pomeau attractor/map is a discrete-time dynamical system. It is one of the most studied examples of dynamical systems that exhibit chaotic behavior.



What is Henon Map?

The Henon map presents a simple two-dimensional invertible iterated map with quadratic nonlinearity and chaotic solutions called strange attractor. Strange attractors are a link between the chaos and the fractals.



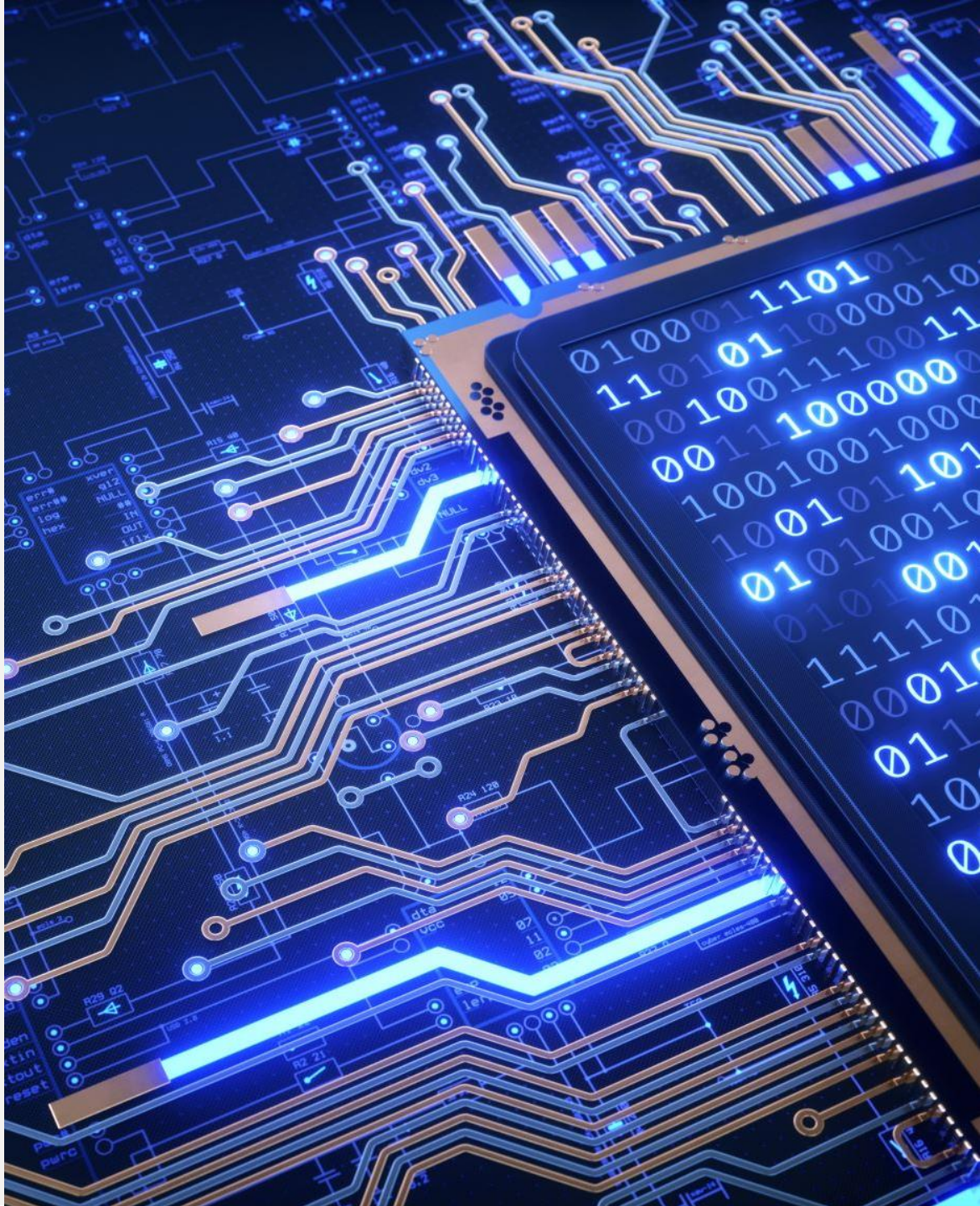
What is Henon Map?

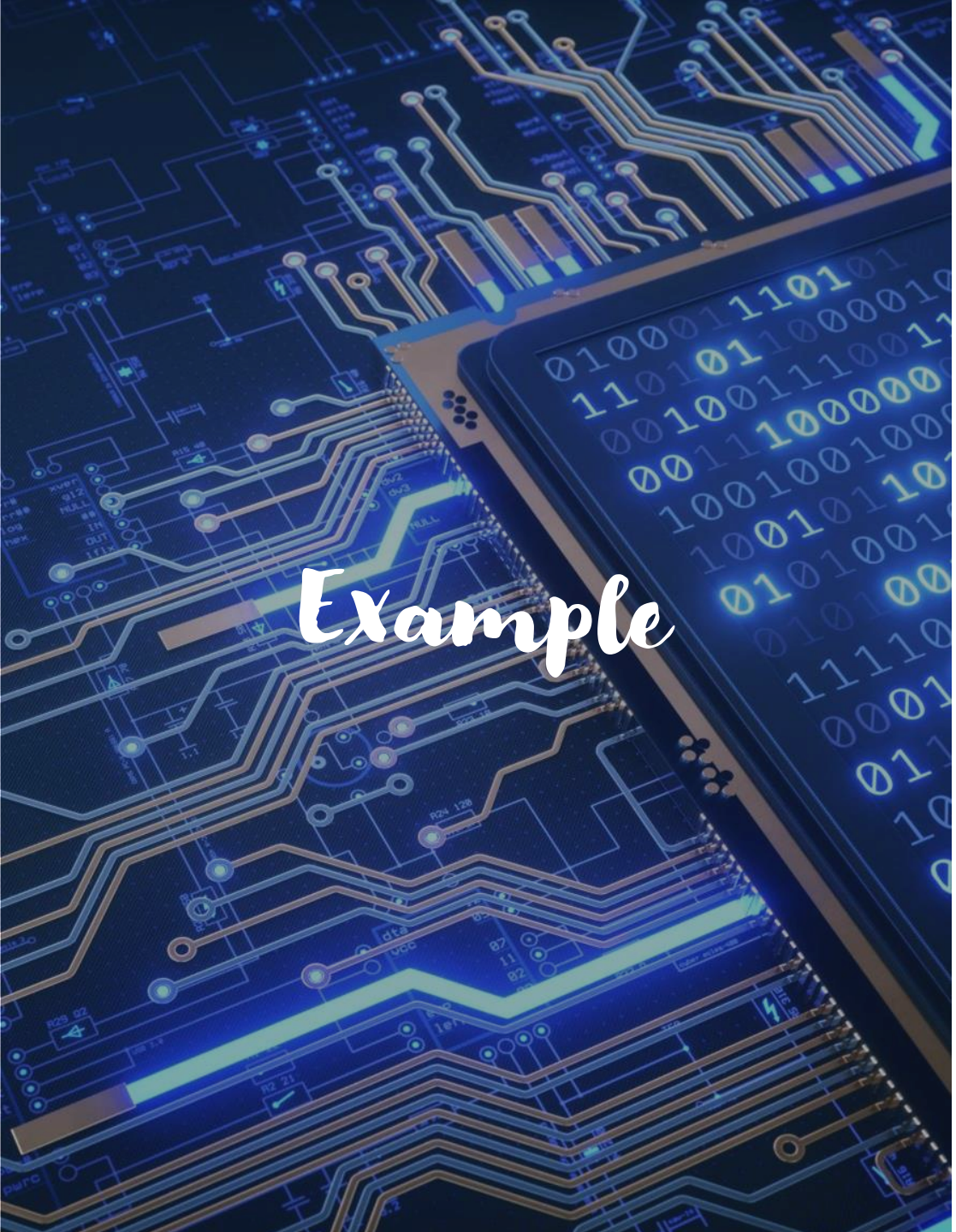
The Henon map takes a point (x_n, y_n) in the plane and maps it to a new point. The below equation helps us to find the next point based on the previous point.

$$\begin{cases} x_{n+1} = 1 - ax_n^2 + y_n \\ y_{n+1} = bx_n . \end{cases}$$

The Henon map may also be deconstructed into a one-dimensional map, defined similarly to the Fibonacci Sequence.

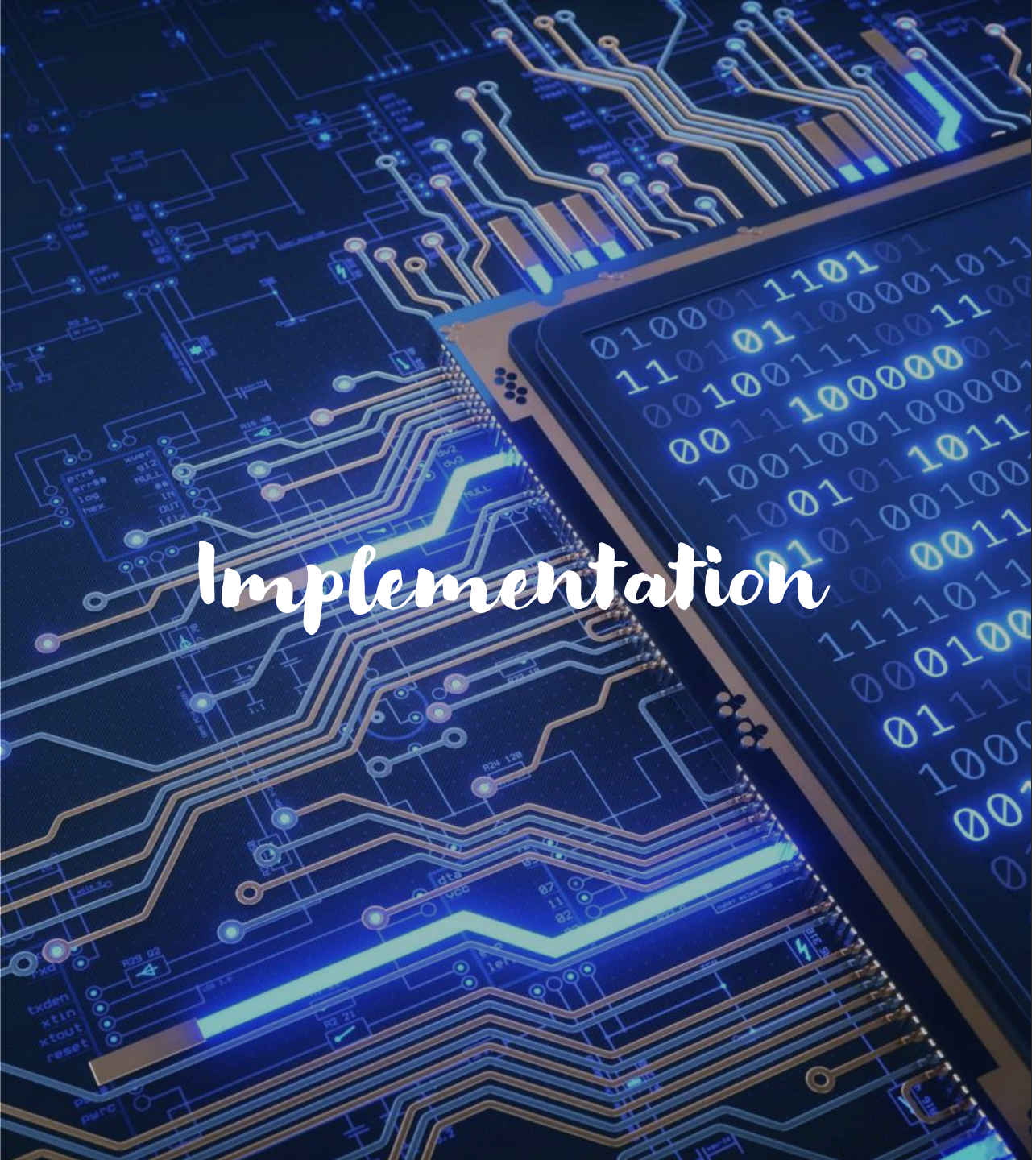
$$x_{n+1} = 1 - ax_n^2 + bx_{n-1}$$





Classical Henon map have values of a = 1.4 and b = 0.3

The Henon map does not have a strange attractor for all values of the parameters a and b. For example, by keeping b fixed at 0.3 the bifurcation diagram shows that for a = 1.25 the Henon map has a stable periodic orbit as an attractor.



Implementation

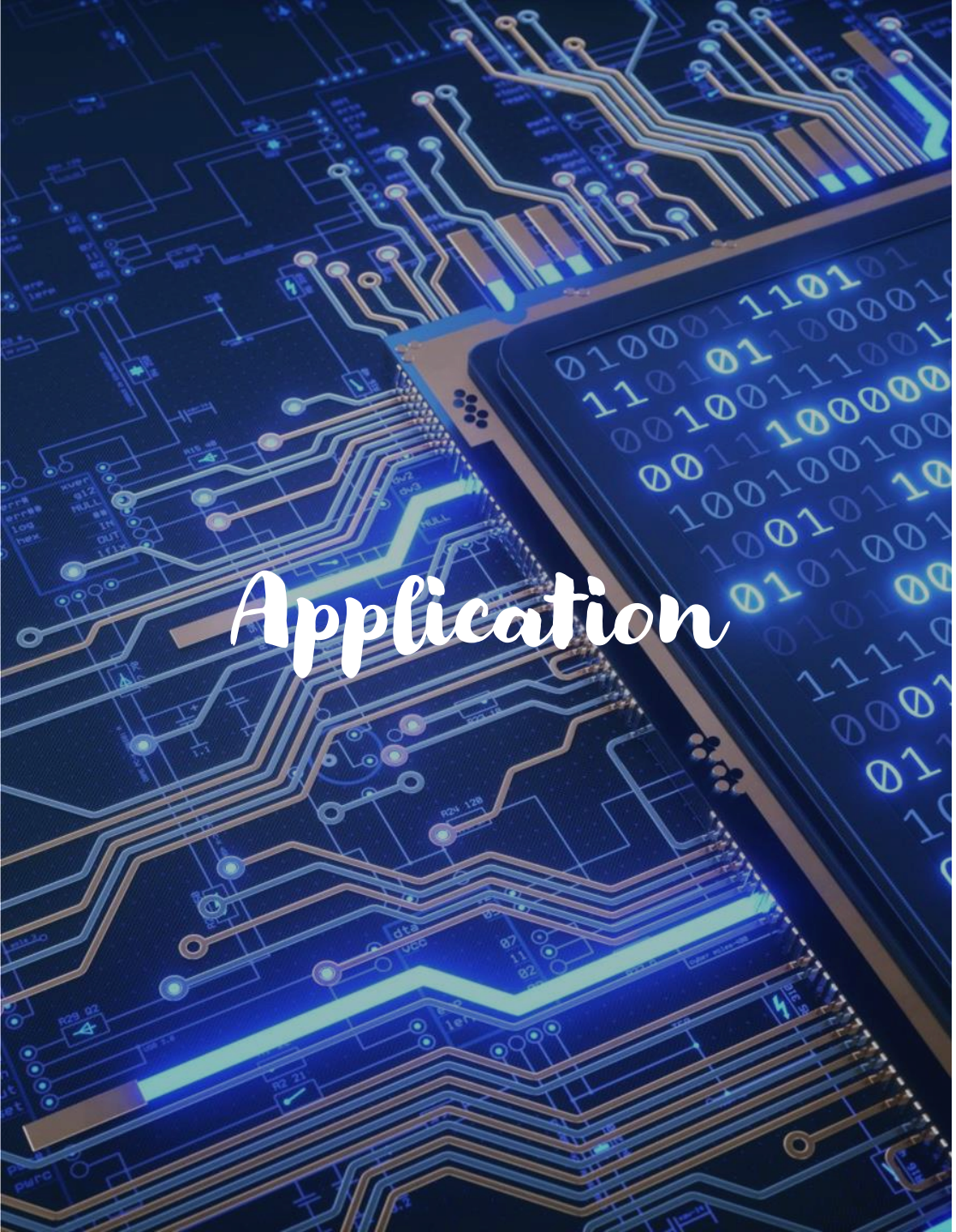
Consider some starting point in the cartesian plane namely x_0 , y_0 and plot them.

Consider some values for constants a = 1.4, b = 0.3 to generate classical henon map

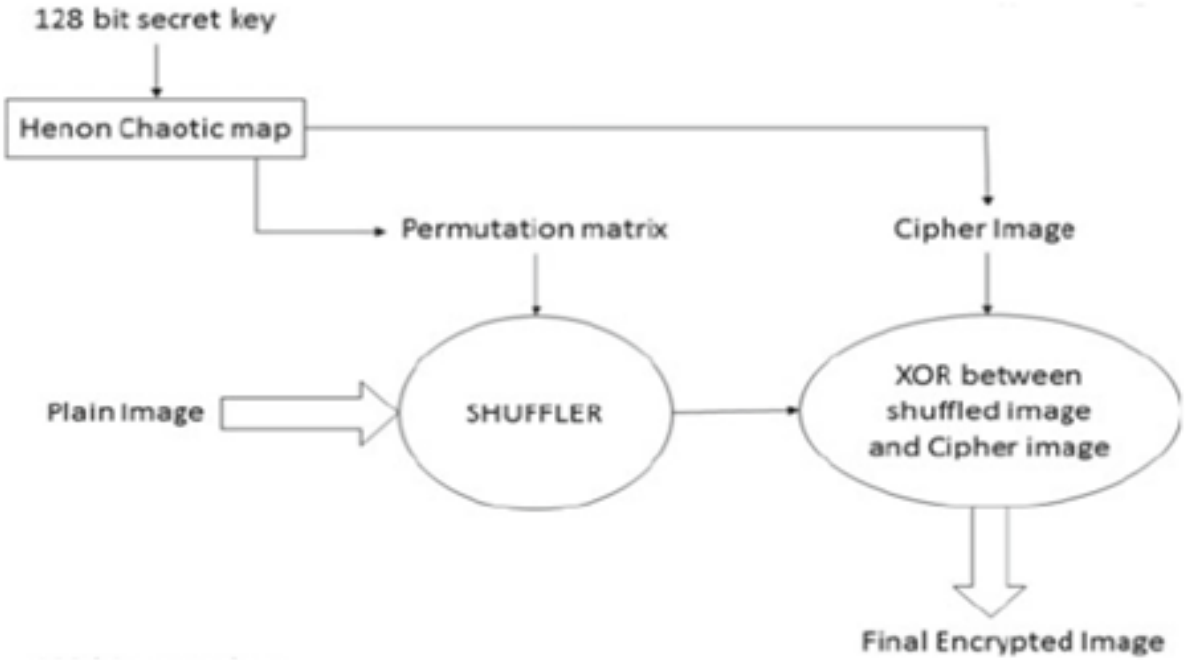
Use the equations

$x_{n+1} = 1 - ax_n^2 + y_n$,

$y_{n+1} = bx_n$, to find the next points and plot them on the cartesian plane.



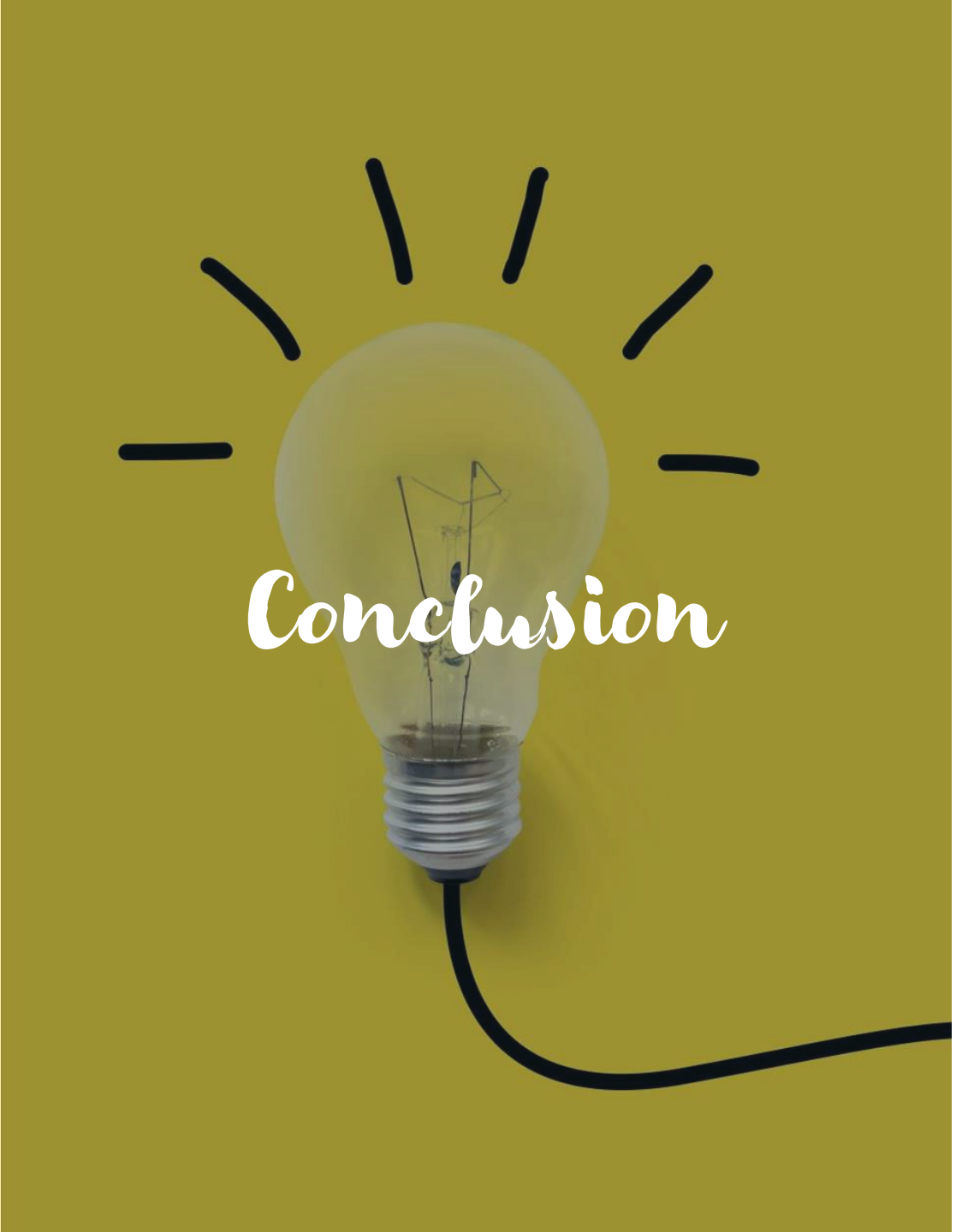
Henon map can be used for encrypting and decrypting images as shown in the flow chart.



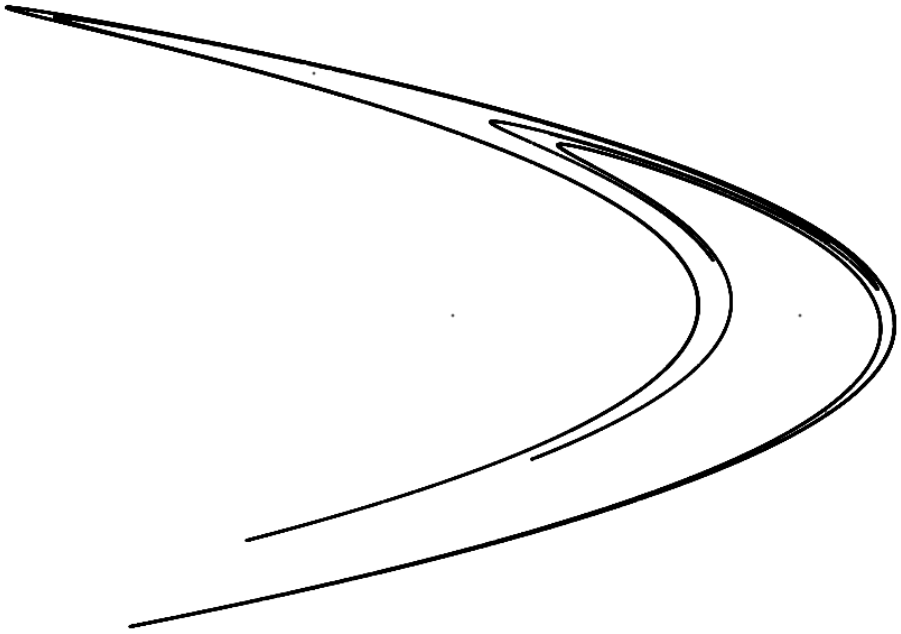
The critical part of the program is shown in the above image.

This part of the serial where we are passing the coordinates of the previous point to the current i.e., coord[i] = eq(coord[i].y) is a challenge to implement when using the multiple threads as this would require to remove the dependency of the next point with the previous point. Hence running multiple threads without resolving this dependency is a challenge.

```
for (int i = 1; i < n; i++)
{
    coord[i] = eq(coord[i - 1]);
    glBegin(GL_POINTS);
    glColor3f(1, 1, 1);
    glVertex2f(coord[i].x, coord[i].y);
    glEnd();
    glFlush();
    glutSwapBuffers();
}
```



Looking at the implementation of the program and the implementation of the algorithm, the parallelization fraction is very low as for every one point in the plane we need the previous point and hence there is dependency which clearly shows how less parallel this program is. So, using some parallelization techniques and making the program run faster is what we will learn as we proceed in the course further.



Henon Map

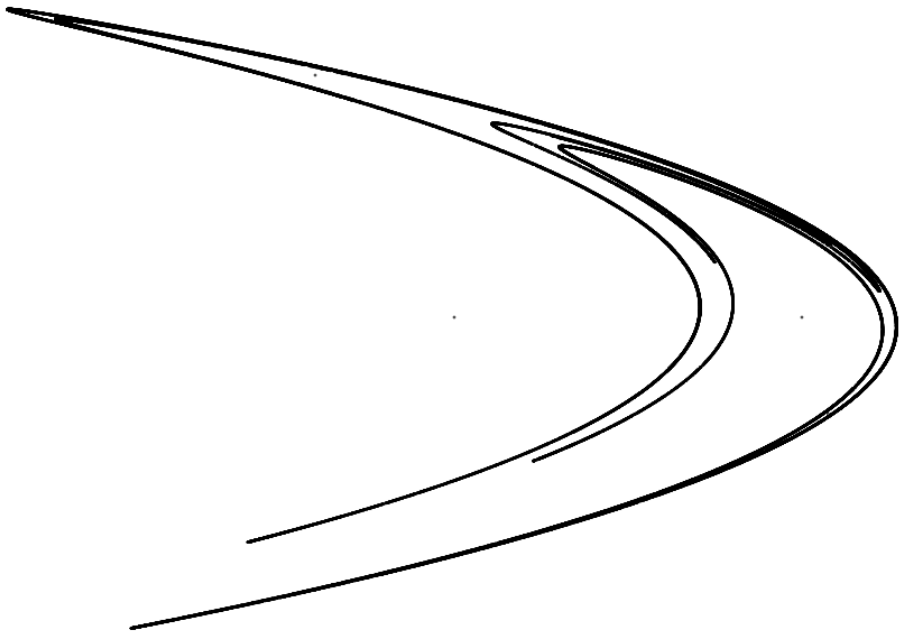
A THEORETICAL ANALYSIS

Samarth Sudarshan Inamdar (CED18I045) | High Performance Computing
August 2021

Department of Computer Science and Engineering,
Indian Institute of Information Technology Design and Manufacturing Kancheepuram, Chennai

Table of Contents

Introduction



The Henon map is an iterated discrete-time dynamical system that exhibits chaotic behavior in two-dimension. It is sometimes called Hénon-Pomeau attractor/map is a discrete-time dynamical system. It is one of the most studied examples of dynamical systems that exhibit chaotic behavior.

What is Henon Map?

The Henon map presents a simple two-dimensional invertible iterated map with quadratic nonlinearity and chaotic solutions called strange attractor. Strange attractors are a link between the chaos and the fractals.

The Henon map takes a point (x_n, y_n) in the plane and maps it to a new point. The below equation helps us to find the next point based on the previous point.

$$\begin{cases} x_{n+1} = 1 - ax_n^2 + y_n \\ y_{n+1} = bx_n. \end{cases}$$

The Henon map may also be deconstructed into a one-dimensional map, defined similarly to the Fibonacci Sequence.

$$x_{n+1} = 1 - ax_n^2 + bx_{n-1}$$

What is a Dynamical System?

A dynamical system is all about the evolution of something over time. To create a dynamical system, we simply need to decide

- what is the "something" that will evolve over time
- what is the rule that specifies how that something evolves with time.

In this way, a dynamical system is simply a model describing the temporal evolution of a system.

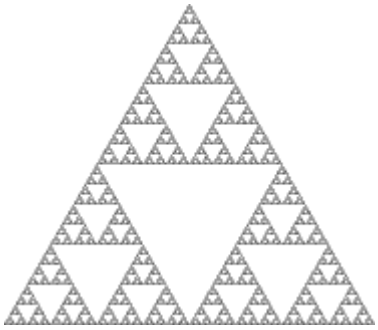
What is an Attractor?

In the mathematical field of dynamical systems, an attractor is a set of states toward which a system tends to evolve, for a wide variety of starting conditions of the system.

What are Strange Attractors?

An attractor is called strange if it has a fractal structure. Strange attractors are unique from other phase-space attractors in that one does not know exactly where on the attractor the system will be. Two points on the attractor that are near each other at one time will be arbitrarily far apart at later times. Two points on the attractor that are near each other at one time will be arbitrarily far apart at later times.

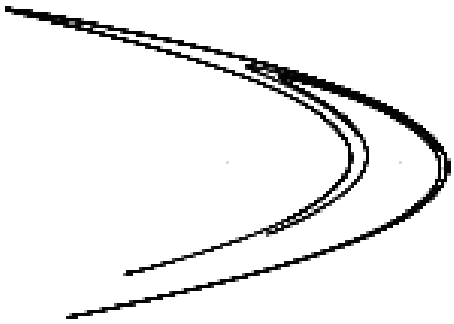
What are Fractals?



A fractal is a never-ending pattern. Fractals are infinitely complex patterns that are self-similar across different scales. They are created by repeating a simple process over and over in an ongoing feedback loop. Driven by recursion, fractals are images of dynamic systems – the pictures of Chaos. Fractal patterns are extremely familiar, since nature is full of fractals. For instance: trees, rivers, coastlines, mountains, clouds, seashells, hurricanes, etc.

Examples of Henon Map

Classical Henon map have values of a = 1.4 and b = 0.3



The Hénon map does not have a strange attractor for all values of the parameters a and b. For example, by keeping b fixed at 0.3 the bifurcation diagram shows that for a = 1.25 the Hénon map has a stable periodic orbit as an attractor.

Implementation

Consider some starting point in the cartesian plane namely x_o, y_o and plot them.

Consider some values for constants a = 1.4, b = 0.3 to generate classical henon map

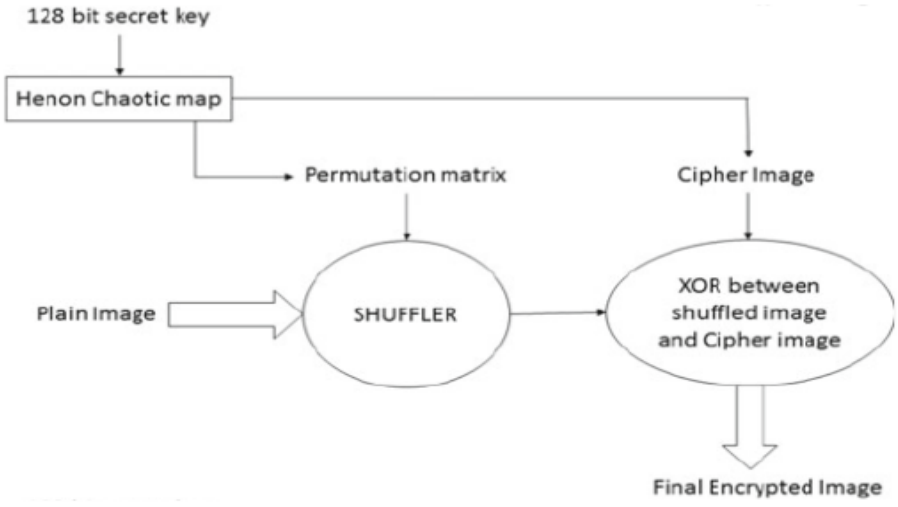
Use the equations

$x_{n+1} = 1 - ax_n^2 + y_n$,

$y_{n+1} = bx_n$, to find the next points and plot them on the cartesian plane.

Applications

Henon map can be used for encrypting and decrypting images as shown in the flow chart below.



Drawbacks

```
for (int i = 1; i < n; i++)
{
    coord[i] = eq(coord[i - 1]);
    glBegin(GL_POINTS);
    glColor3f(1, 1, 1);
    glVertex2f(coord[i].x, coord[i].y);
    glEnd();
    glFlush();
    glutSwapBuffers();
}
```

The critical part of the program is shown in the above image.

This part of the serial where we are passing the coordinates of the previous point to the current i.e., coord[i] = eq(coord[i]. y) is a challenge to implement when using the multiple threads as this would require to remove the dependency of the next point with the previous point. Hence running multiple threads without resolving this dependency is a challenge.

Code Balancing

Current Processor: Intel® Core™ i7-8750H Processor, 6 cores, 2.20 GHz

No. of flops supported by the system for double precision

= No of cores * Clock frequency * Double precision calculations

= 6 * 2.2 GHz * 4 flops/sec = 52.8 Gflops/sec

Each loop contains 5 floating point arithmetic operations.

For input size of 100000000,

No. of floating-point operations = 5 * 100000000 = 0.5 Gflops

The processor can handle a higher input size within 1 second.

- No. of words used & modified
  = 8 (6 reads, 2 writes) * 2 (Double datatype uses 8 bytes / 2 words) * 100000000 = 1.6 G words

- Code balance (B_C)
  = No. of words used & modified / No. of flops in the code
  = 1.6 G/ 0.5 Gflops = 3.2

- Computational intensity = 1 / Code balance (BC)
  = 1 / 3.2 = 0.3125

- Machine balance (B_M)
  = Memory bandwidth (B_{max}) / No. of flops of the system (P_{max})
  = 2.667 GHz / 52.8 Gflops = 0.05051

Software

• C/C++, programming language

• OpenMP, API for shared-memory parallel programming

• MPI, High performance Message Passing library

• Cuda C/C++, API for utilizing CUDA-enabled GPU for computation

• Qt, GUI library for displaying the output graph animations

Conclusion

Looking at the implementation of the program and the implementation of the algorithm, the parallelization fraction is very low as for every one point in the plane we need the previous point and hence there is dependency which clearly shows how less parallel this program is. So, using some parallelization techniques and making the program run faster is what we will learn as we proceed in the course further.

References

Wikipedia | Henon Map

Wikipedia |Dynamical System

Wikipedia | Attractor

Dynamical Properties of the Hénon Mapping