UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Universidade Federal de Minas Gerais

Turma: Ciência da Computação **Prof.:** Gilberto

Nome: Rubia Alice Moreira de Souza

Questões Verilog

Belo Horizonte
2023

1. Atividade: Gerar uma OTP

Implementação em código Verilog disponível em:

- Código EDA Workbanch: One Time Pad
- Simulação Digital JS: Digital JS - Simulação One Time Pad

O One Time Pad foi criado utilizando esta cifra gerada no site Random.org: 01011110_01000011_00101111_01111100.

Os resultados obtidos foram os seguintes:

Mensagem	**Binário**	**Texto**
Original	01100111011000010110110001101111	Galo
Cifrada	00111001001000100100001100010011	“9"Cnull”
Decifrada	01100111011000010110110001101111	Galo

2. Atividade: Geração de sequência pseudo-aleatória

2.1. Parte 1: Código Verilog

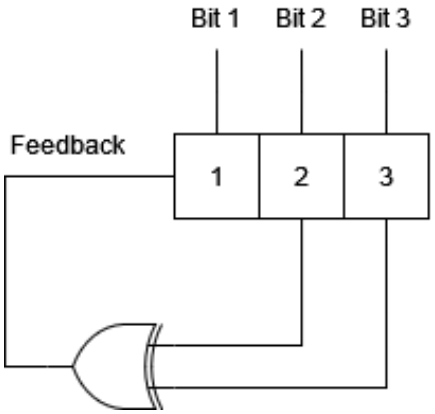
Implementação do código Verilog disponíveis em:

Quantidade Bits LFSR	Simulações	
	EDA Workbanch	Digital JS
3bits	Linear Feedback Shift Register - 3 bits	Digital JS - Simulação LFSR 3 Bits
4bits	Linear Feedback Shift Register - 4 bits	Digital JS - Simulação LFSR 4 Bits
8bits	Linear Feedback Shift Register - 8 bits	Digital JS - Simulação LFSR 8 Bits

2.2. Parte 2: Gráficos dos Períodos de Repetição das Sequências

2.2.1. Polinômio: $X^3 + X^2 + 1$

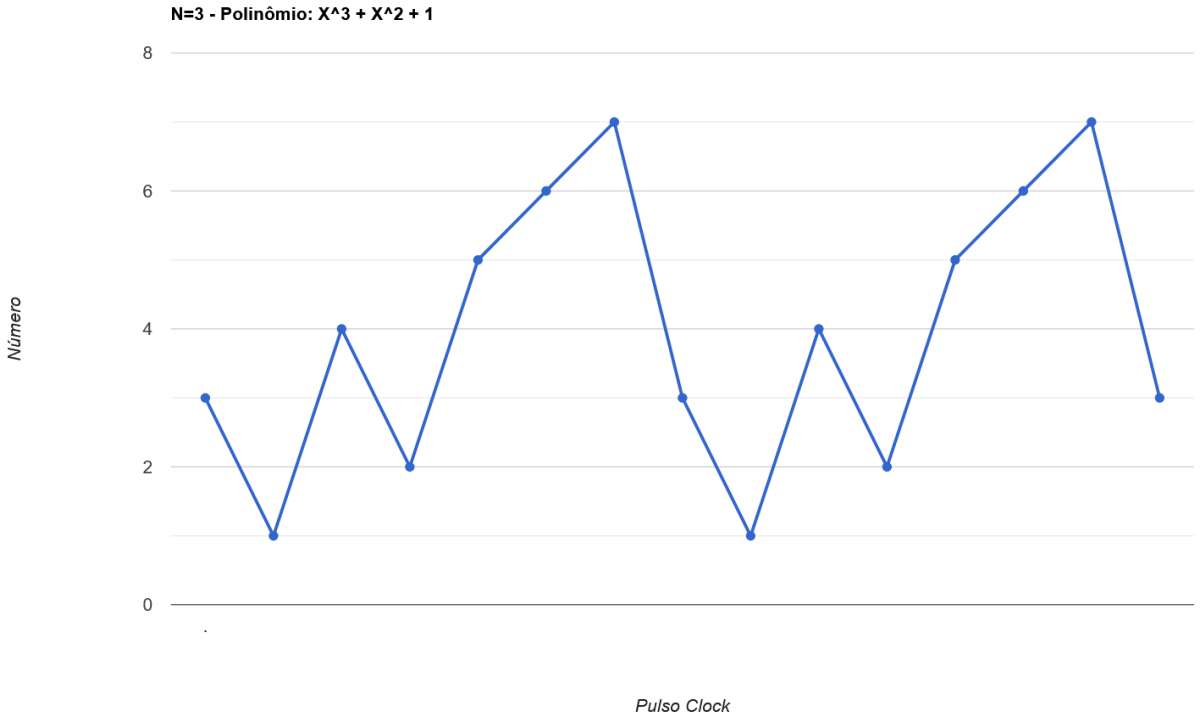
Para o polinômio X^3+X^2+1, temos o circuito da seguinte forma:



011
001
100
010
101
110
111
011
001
100
010

Ciclo com 7 iterações

Nele, é possível notar um ciclo a cada 7 valores (pontos), no gráfico abaixo:



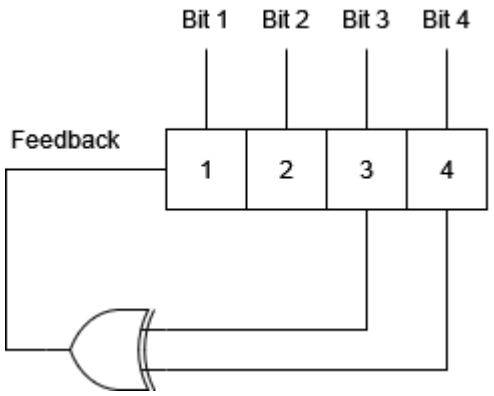
Neste caso, ele segue a seguinte sequência 3 → 1 → 4 → 2 → 5 → 6 → 7

Índice	1	2	3	4	5	6	7

Binário	011	001	100	010	101	110	111
Decimal	3	1	4	2	5	6	7

2.2.2. Polinômio: $X^4 + X^3 + 1$

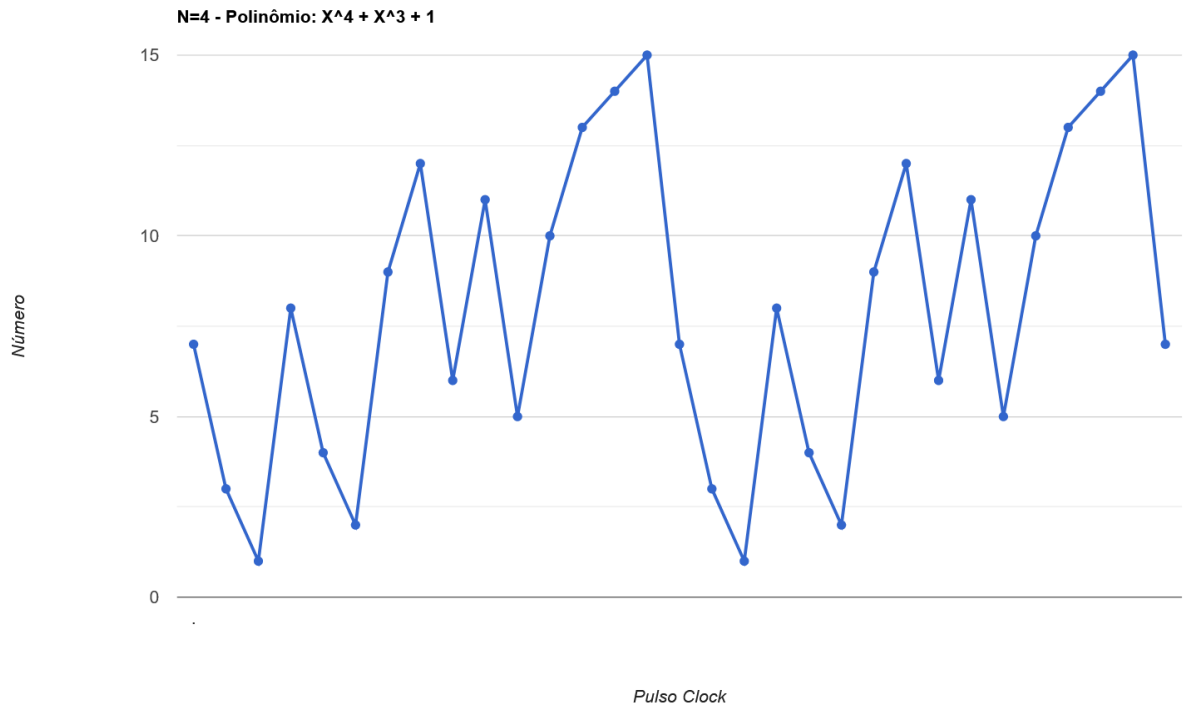
Já para o polinômio $X^4 + X^3 + 1$, temos:



0111	
0011	1010
0001	1101
1000	1110
0100	1111
0010	0111
1001	0011
1100	0001
0110	1000
1011	0100
0101	0010

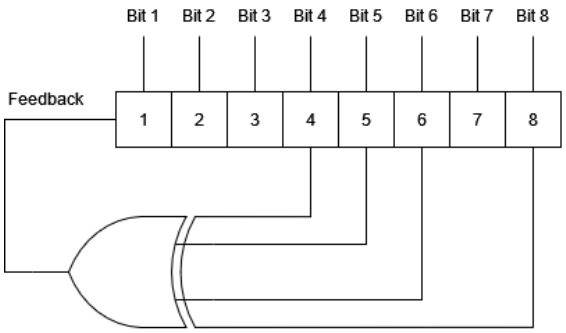
Ciclo com 15 valores diferentes

Ele segue uma sequência com 15 valores:



2.2.3. Polinômio: $X^8 + X^6 + X^5 + X^4 + 1$

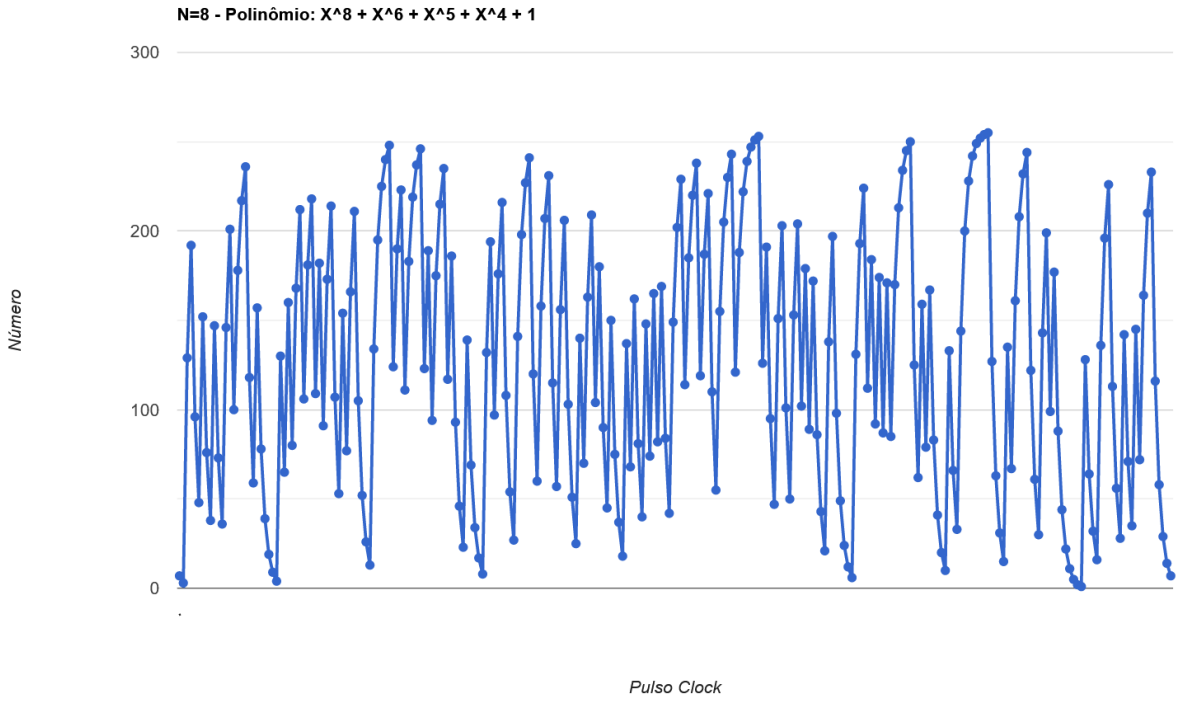
Por fim, temos o polinômio $X^8 + X^6 + X^5 + X^4 + 1$:



00000111	…
00000011	01110001
10000001	00111000
11000000	00011100
01100000	10001110
00110000	01000111
10011000	00100011
01001100	10010001
00100110	01001000
10010011	10100100
01001001	11010010
00100100	11101001
10010010	01110100
10001000	00111010
11000100	00011101
11100010	00001110
…	00000111

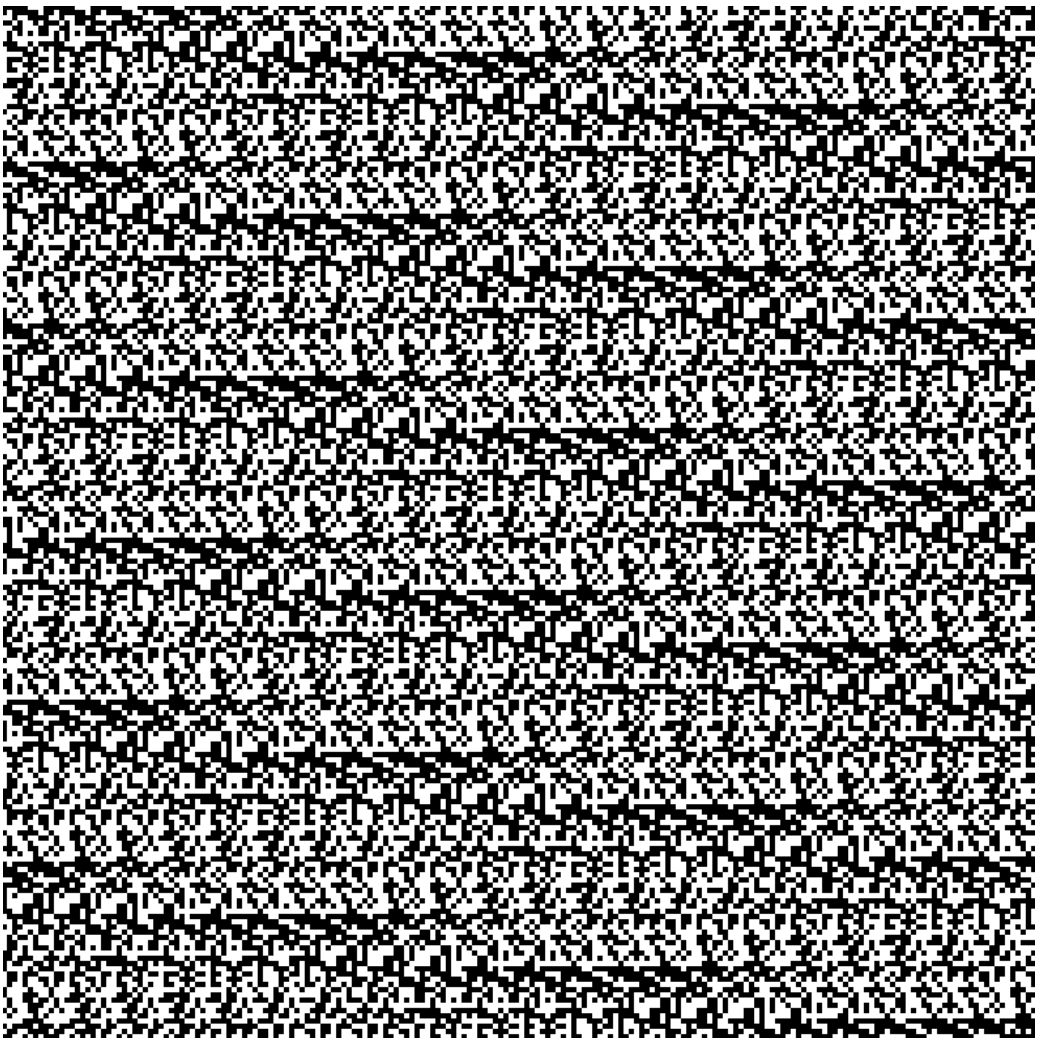
Ciclo com 255 valores diferentes

Ele possui um ciclo de 255 valores. Neste gráfico, eu representei um ciclo apenas do polinômio, para que seja visível a variação nele:



2.3. Imagem Gerada

A imagem foi gerada utilizando o seguinte código em C++: Basico Verilog. Ela foi criada em um arquivo PGM (portable graymap format) com 39.200 pixels. Ou seja, possui 4900 iterações do LFSR com 8 bits.



Também geramos uma imagem de um LFSR com 4 bits e 3 bits. Ambos com 4900 iterações:

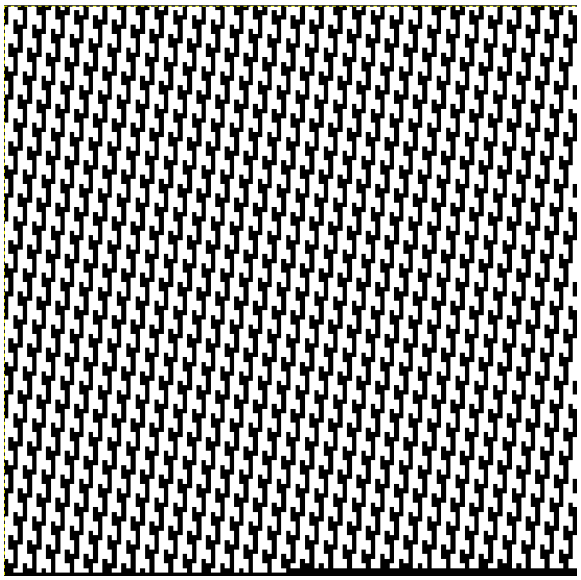


Imagem gerada a partir de um LFSR com 3 bits

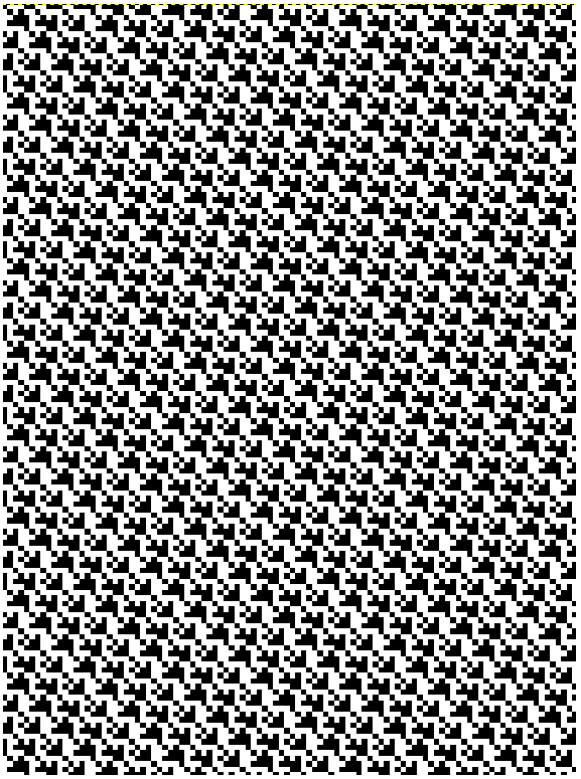


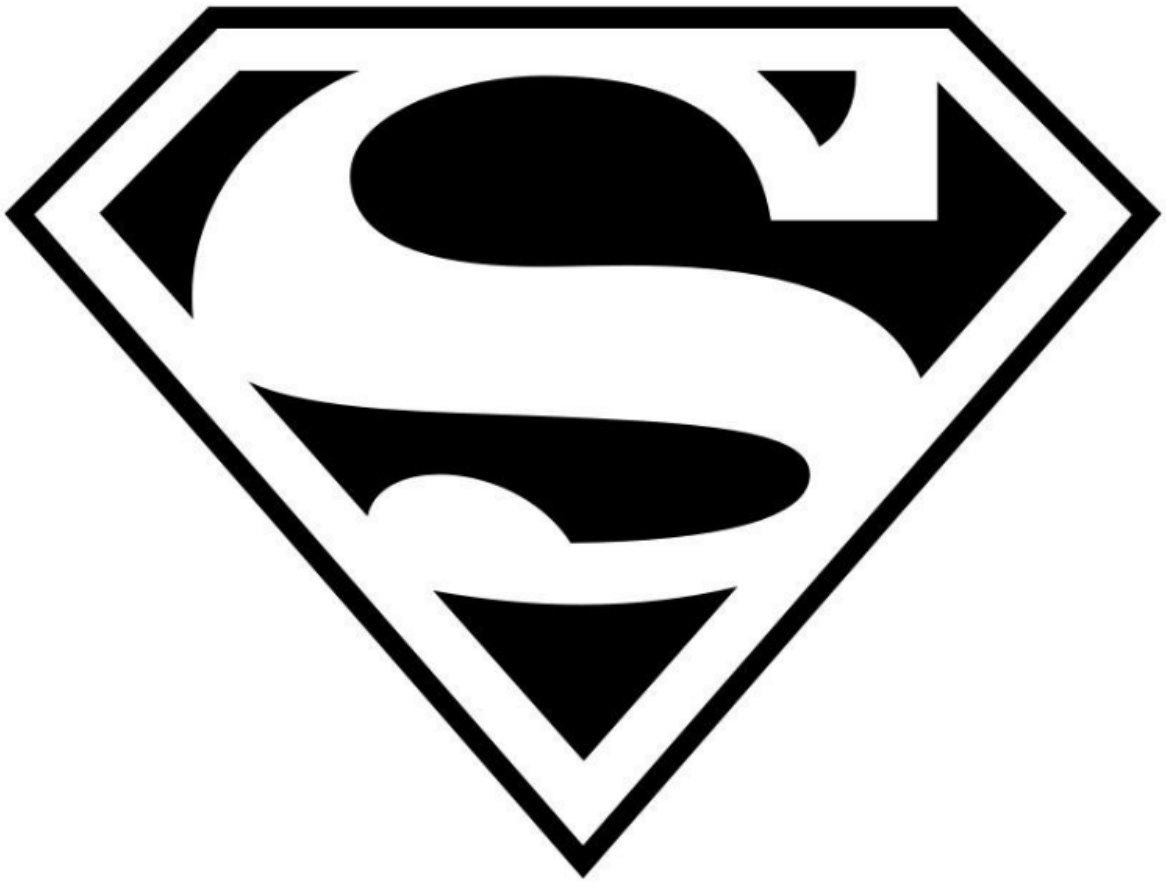
Imagem gerada a partir de um LFSR com 4 bits

2.4. Referências

Referência utilizada para gerar os polinômios com quantidade máxima de valores por ciclo foi a “Tabela 3 - Taps for Maximum-Length LFSR Counters”. Disponível neste link: <https://docs.xilinx.com/v/u/en-US/xapp052>.

3. Atividade: Cifragem de Imagens

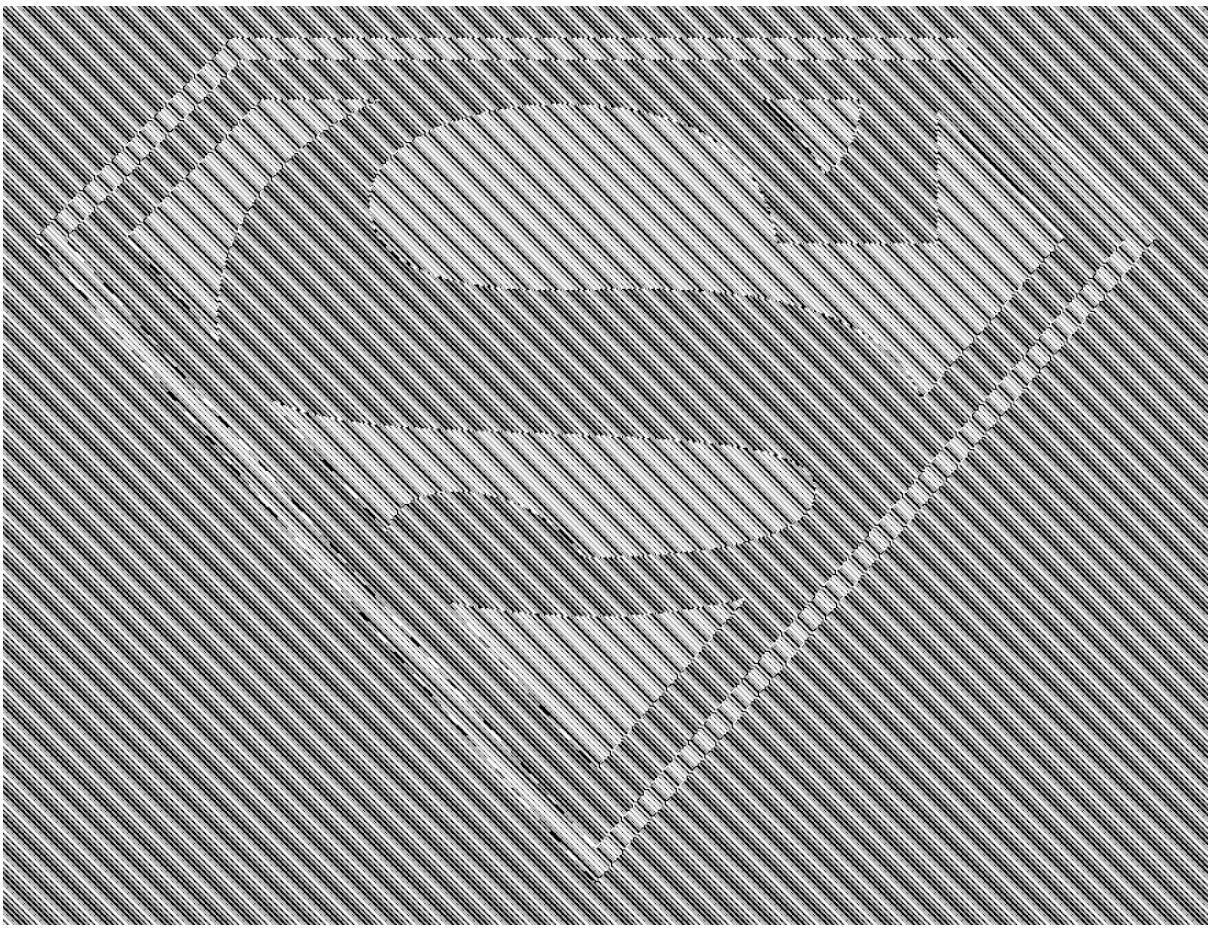
A imagem original que será cifrada, foi aquela fornecida no enunciado do trabalho:



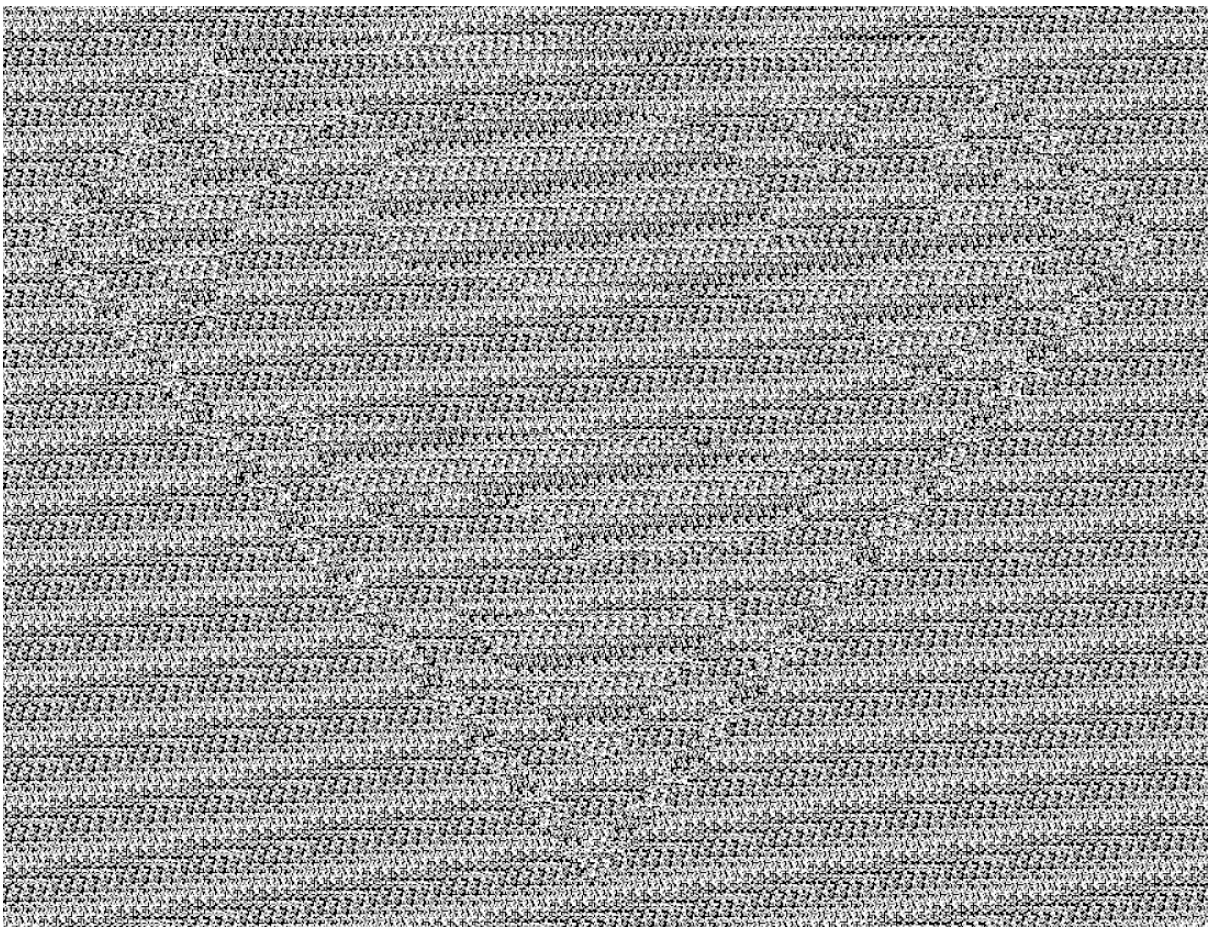
Inicialmente, criei um programa em C++ para convertê-la para PGM e junto com um arquivo binário, em que os valores maiores que 200 eram marcados como 1. Já os outros como 0.

Então, fiz um programa que utiliza uma cifra dos polinômios pseudo-aleatórios da questão anterior e aplicar um XOR aos bits da imagem original. A leitura do polinômio foi feita de forma circular para que fosse necessário armazenar apenas uma sequência.

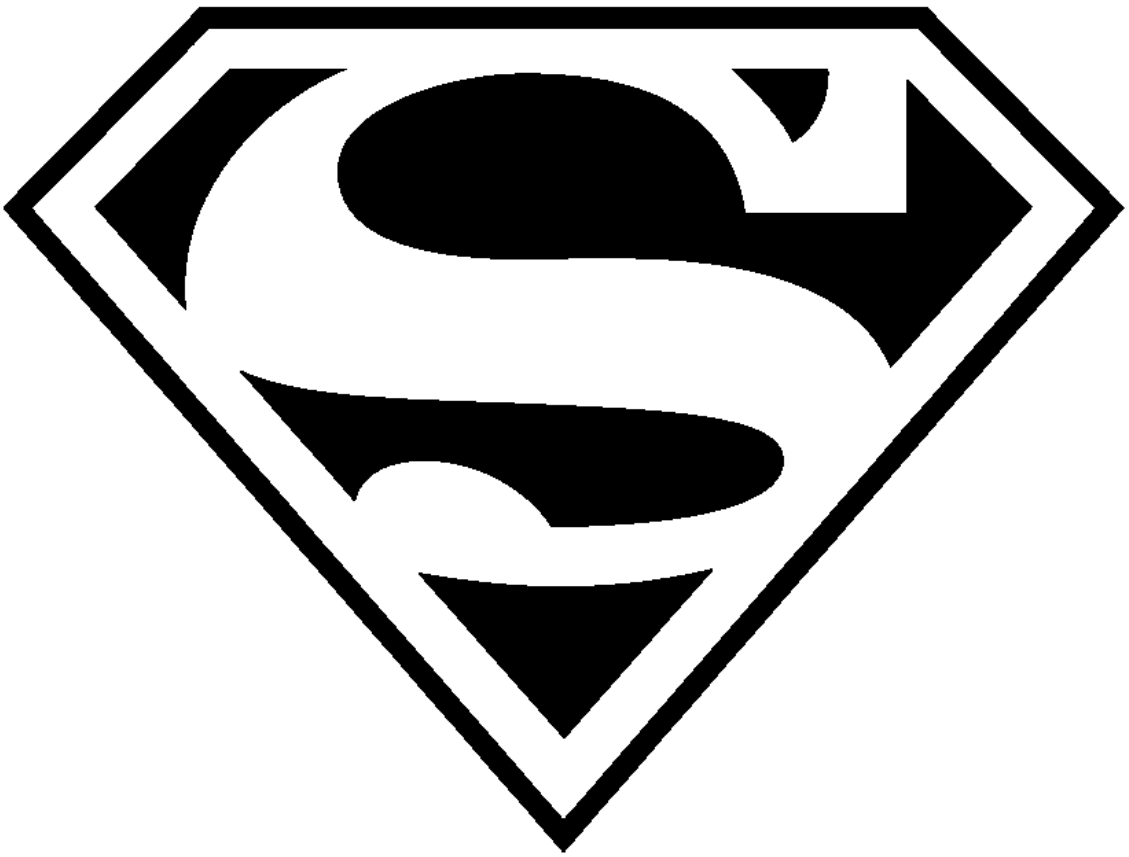
Utilizando a sequência pseudo aleatória de 3 bits do LFSR, obtive esta imagem ao cifrar a original:



Ainda é possível notar o símbolo original nela. No entanto, ao utilizar a cifra de 8 bits, o padrão fica menos reconhecível na imagem, que ficou da seguinte forma:



Ao reverter a imagem para o tipo original, obtivemos o seguinte resultado:



Todos os códigos para manipulação de imagens estão disponíveis no link: Rubia Souza - BasicoVerilog/Manipulacao-Imagens.

4. Referências:

AMD. Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators. Disponível em: <https://docs.xilinx.com/v/u/en-US/xapp052>. Acesso em: 07/06/2023.

Gilberto Medeiros Ribeiro. Slides virtuais da disciplina de Introdução aos Sistemas Lógicos. Disponibilizado via moodle. Departamento de Ciência da Computação. Universidade Federal de Minas Gerais. Belo Horizonte.