Trabalho Prático 2

Introdução aos Sistemas Lógicos

Igor Joaquim da Silva Costa - 2021032218

1. Introdução

O problema proposto foi implementar 3 LFSR de Fibonacci e usá-los em conjunto com o Vernam Cypher para fazer a cifragem da logo da UFMG.

Para isso, foi utilizada a linguagem de descrição e simulação de hardware verilog para construir 3 tipos de registradores LFSR, um de 7 bits, outro de 11 bits e outro de 16 bits.

2. Método

2.1 Registradores e tesbench

Primeiramente, é válido dizer que todo registrador foi implementado de maneira análoga. Para a construção do componente, foi tomada a decisão de representá-lo como um array de registradores binários chamado memory, que é a estrutura responsável por guardar o estado do registrador em um período de tempo específico. A cada subida do clock, os valores de memory são atualizados com base na função Polinômio de realimentação específica. A saída também é um registrador de N bits, para facilitar a geração de gráficos.

Para se realizar o testbench, foi utilizada a lógica de ciclos. Cada registrador possui uma variável ciclos definida, que define quantos períodos de clock o circuito irá simular, conforme o necessário. A saída do registrador de 7 e 11 bits são os números já convertidos em decimal, para facilitar a plotagem dos gráficos, no caso do registrador de 16 bits, apenas é armazenado o ultimo bit.

2.2 Plotagens e bitmap

Para os gráficos e o bitmap, foi usado o matplotlib com a saída gerada pela execução do design.sv, onde cada número é a representação decimal do estado do registrador em algum ciclo do clock. Para o bitmap, foram escolhidas dimensões arbitrárias para gerar a imagem.

3. Arquivos importantes

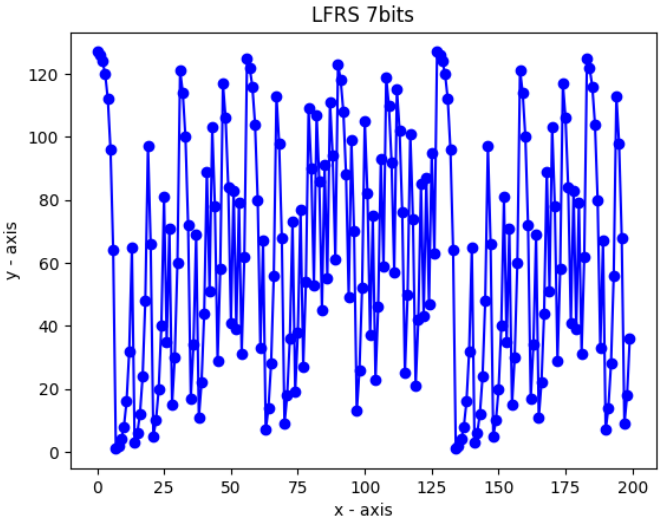
Cada registrador possui uma pasta específica com seu testbench e design salvos. Para cada pasta de registrador, existe um arquivo plot.py, responsável por gerar os respectivos gráficos.

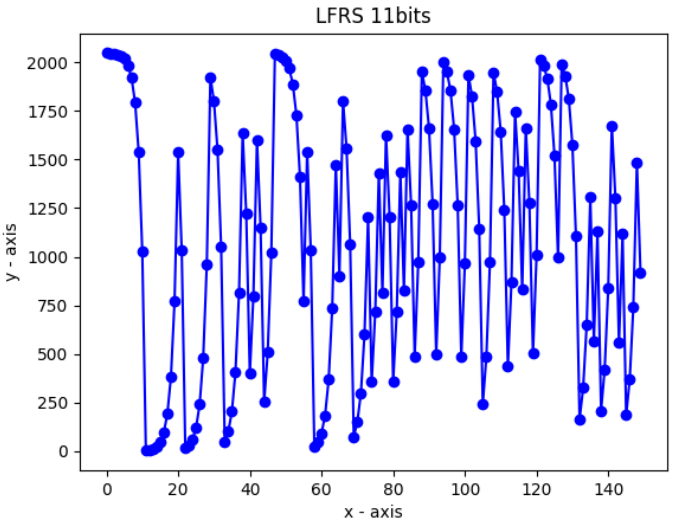
O arquivo de testbench fora de qualquer pasta corresponde ao codigo disponibilizado para realizar a cifragem da imagem, ele deve ser compilado junto ao arquivo LSFR.sv para cada respectivo registrador, a fim de criar o lfsr_output do registrador esperado. O arquivo criabits.py serve para transformar a imagem do brasão em bits e o arquivo converte_imagem.py tranforma os dados binários do lfsr_output de volta em imagem.

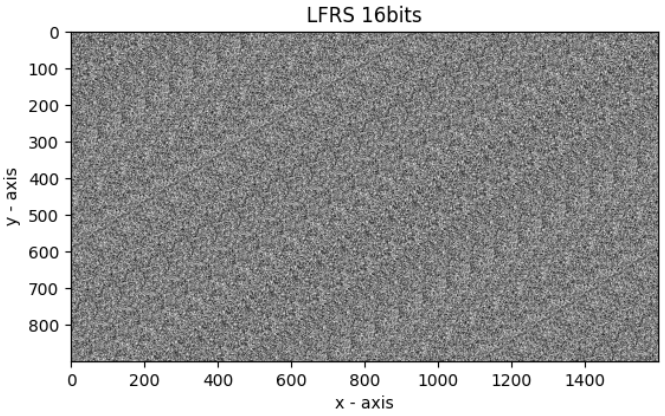
Existe um comando no Makefile chamado “all”, que cria todas as imagens cifradas para cada registrador implementado.

4.Imagens

4.1 Plotagem de ciclos




LFRS 11bits
y - axis
x - axis
0
250
500
750
1000
1250
1500
1750
2000
0
20
40
60
80
100
120
140

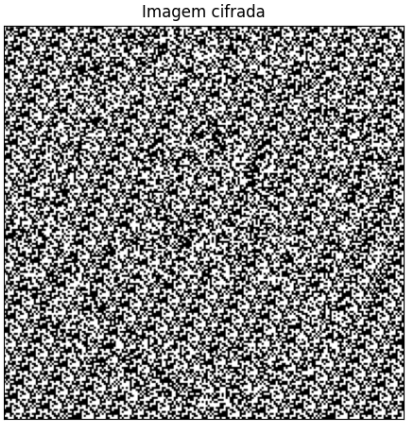

LFRS 16bits
y - axis
x - axis
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200
1400

4.2 Imagens Cifradas

Cifragem de imagem - LFSR (Período: 7)

Imagem original

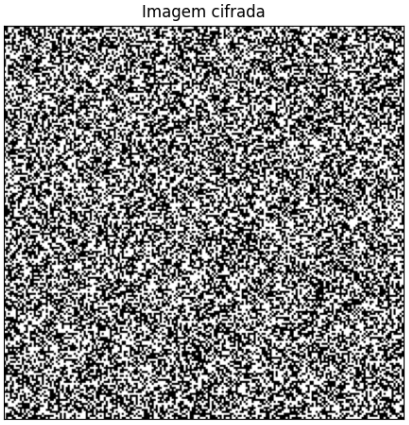
Imagem cifrada



Cifragem de imagem - LFSR (Período: 11)

Imagem original

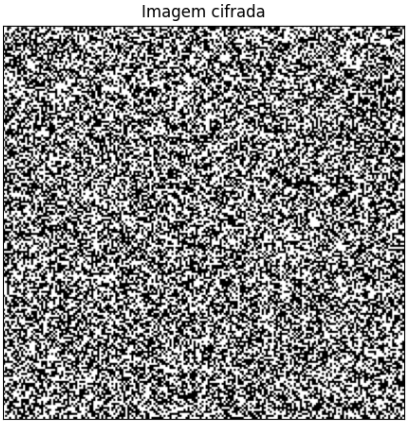
Imagem cifrada



Cifragem de imagem - LFSR (Período: 16)

Imagem original

Imagem cifrada



5. Conclusão

Durante o trabalho, foram tomadas diversas decisões que serviram para experimentar como é simular um sistema em hardware, tanto na parte de implementação, usando o Verilog, quanto na parte de debugação. Além disso, tornou-se sucinto o poder que técnicas de cifragem possuem, principalmente quando combinadas com técnicas que usam geração de dados aleatórios como chave, como é o caso dos LFSR de período médio e longo.

Como visto nos gráficos, a implementação dos LFSR ocorreu com êxito, garantidas as propriedades de pseudo-aleatoriedade e ciclicidade que os registradores de cifragem necessitam.

Por fim, conclui-se que o trabalho serviu como um exemplo de como os conteúdos aprendidos na disciplina podem ser úteis na vida técnica de um profissional de ciência da computação, uma vez que implementar circuitos específicos para algum fim em um projeto pode ser mais eficiente que implementar a mesma funcionalidade em software, como é o caso do Vernam Cypher.