Trabalho Prático

Vernam Cipher

Júlio Guerra Domingues (2022431280)

Introdução aos Sistemas Lógicos - DCC/UFMG - 2023-2

Introdução

O presente trabalho aborda a integração de conceitos de lógica combinatória, lógica sequencial e criptografia, com foco na implementação de um sistema de criptografia baseado no chamado *Vernam Cipher*. Também conhecido como *one-time pad*, é um método de criptografia que, quando corretamente implementado, é considerado inquebrável, como demonstrado por Claude Shannon em 1949. Para a sua contrução serão utilizados, dentre outros, flip-flops D, cuja implementação constitui a primeira parte do trabalho.

Método

O programa foi desenvolvido em Verilog utilizando, como proposto em sala de aula, o EDA Playground (©Copyright 2023, Doulos), disponível em: https://edaplayground.com/. Essa plataforma permite o desenvolvimento dos algoritmos e testes de forma simples, além do compartilhamento do código, evitando problemas de compatibilidade entre a máquina do aluno e dos avaliadores. Ainda seguindo as instruções, na opção 'Testbench + Design' deve ser escolhido o item 'SystemVerilog/Verilog' e na opção 'Simulators' o item 'Icarus Verilog 0.9.7'.

A máquina utilizada tem as seguintes especificações:

- Sistema Operacional: MacOS Sonoma 14.0
- Processador: ARM Apple Silicon M2
- Memória RAM: 8,00 GB

Implementações

Na primeira parte do trabalho, tivemos de implementar um flip-flop D em Verilog de duas formas: descritiva e comportamental.

Flip-flop D - Descritivo

Para a implementação descritiva, optou-se pela utilização de 4 portas NAND. Temos como entrada um clock (clk) e d, e como saída q e q negado. No arquivo design.sv temos a implementação propriamente dita e em testbench.sv os testes.

design.sv

O código começa com a implementação do flip-flop D no módulo d_flip_flop_desc:

- inputs: 'clk' (*clock*) e 'd' (entrada de dados)
- outputs: 'q' e 'notq'

A lógica do flip-flop D é implementada usando portas NAND:

- NAND1: Combina as entradas 'd' e 'clk' usando uma porta NAND para gerar um sinal intermediário ('w1').
- NAND2: Combina o sinal intermediário 'w1' e o sinal de 'clk' usando uma porta NAND para simular o comportamento de uma báscula (*latch*).
- NAND3: Gera as saídas 'q' e 'notq' usando a porta NAND para combinar 'w1' e 'notq'.
- NAND4: Gera a saída 'notq' usando a porta NAND para combinar 'q' e 'w2'.

testbench.sv

O módulo de teste é usado para verificar o comportamento do flip-flop D implementado com portas NAND:

- reg 'd', 'clk': Variáveis de entrada simuladas para 'd' e 'clk'.
- wire 'q', 'notq': Saídas simuladas para 'q' e 'notq'.

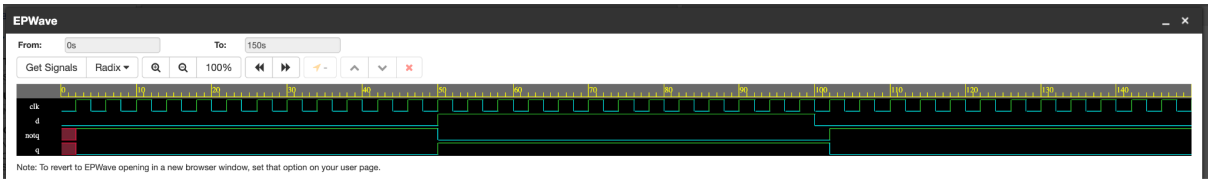
A saída da simulação é direcionada para o arquivo "dump.vcd".

As variáveis 'clk' e 'd' são inicializadas para zero. São aplicados estímulos no sinal 'd' após 50 unidades de tempo (#50), alternando entre 1 e 0. A simulação é

finalizada após 150 unidades de tempo (#50 $finish). A seção *always* é responsável por gerar o sinal de clock, alternando entre 0 e 1 a cada 2 unidades de tempo.

Diagrama de tempo

Pelo diagrama de tempo, podemos perceber que a implementação está adequada, modificando os valores de 'q' e 'notq' de acordo com 'd', quando pertinente, apenas nas bordas ascendentes do *clock*.



Código

O código completo pode ser verificado e rodado no link do EDA Playground: https://edaplayground.com/x/YCq4

Flip-flop D - Comportamental

Na implementação do Flip-flop D de forma comportamental, por outro lado, o enfoque é a funcionalidade do circuito. A implementação é mais direta e os testes são parecidos.

design.sv

O módulo d_flip_flop implementa um flip-flop D, com a entrada de dados 'd', entrada de clock 'clk', e saída 'q'. O flip-flop D é sensível à borda ascendente do clock (*posedge*) e atualiza a saída 'q' com o valor da entrada 'd'.

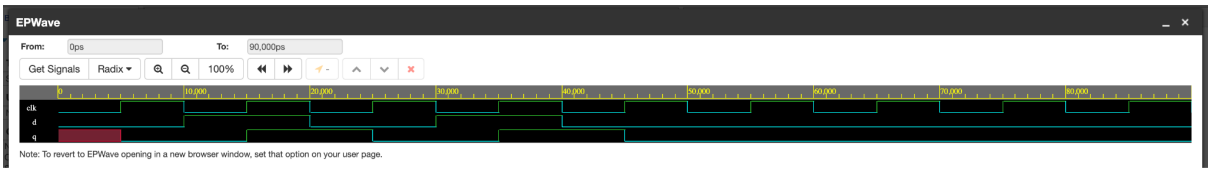
testbench.sv

Assim como no banco de testes do Flip-flop descritivo, temos aqui testes para verificar o comportamento do flip-flop D implementado.

- reg 'clk': Variável de entrada simulada para o sinal de clock (clk).
- reg 'd': Variável de entrada simulada para o sinal de dados (d).
- wire 'q': Saída simulada para o sinal 'q'.

Diagrama de tempo

Pelo diagrama de tempo, podemos identificar que o comportamento do Flip-flop D segue conforme o esperado:



Código

O código completo da implementação do flip-flop D comportamental pode ser verificado e rodado em:

https://edaplayground.com/x/Jtn9

Stream Cipher and Decipher

A segunda questão do trabalho consiste na implementação de um sistema que criptografa e decifra mensagens.

design.sv

O módulo 'streamcipher' realiza operações básicas de cifragem e decifragem de dados utilizando um algoritmo simples de cifra de fluxo. Para tal, recebe sinais de clock, reset, controle e um dado de entrada. Com base nos sinais de controle, o módulo pode carregar novos dados, habilitar a saída de dados, e realizar operações de deslocamento nos bits. A lógica de deslocamento e carregamento é controlada pelos sinais 'loadControl' e 'shiftControl'. O módulo gera uma saída de 8 bits ('dataOutput') e indicações de estado ('dataOutputEnabled', 'dataShifted').
O módulo utiliza parâmetros locais (MAX_BIT, MIN_BIT, DEFAULT_REG_VALUE) para definir valores padrão e limites. As operações lógicas são realizadas dentro de um bloco *always* que é acionado na borda ascendente do *clock*.

testbench.sv

O arquivo de testes verifica o funcionamento de 'streamcipher', cifrando e decifrando a mensagem de exemplo ("ISL UFMG"). Para isso, são inicializadas variáveis que armazenam a mensagem original ('originalMessage'), a mensagem cifrada ('encryptedMessage'), o *clock* ('clk'), o *reset* ('reset'), os sinais de controle ('controlSignal1' e 'controlSignal2'), e a chave ('key'). As tarefas 'setKey', 'encryptMessage', 'decryptMessage' e 'displayResults' são definidas para modularizar o processo de teste.
O módulo está organizado da seguinte maneira:

- Configuração Inicial: Define o *reset* e configura o *clock*.
- Configuração da Chave: Utiliza a tarefa 'setKey' para configurar a chave de cifragem/decifragem.
- Cifragem da Mensagem: A tarefa 'encryptMessage' cifra a mensagem original.
- Decifragem da Mensagem: A tarefa 'decryptMessage' decifra a mensagem para verificar a integridade da cifragem.
- Exibição dos Resultados: A tarefa 'displayResults' exibe a mensagem cifrada e a mensagem original após a decifragem.

Código

O código completo da implementação do Stream Cipher / Decipher pode ser verificado e rodado em:

https://edaplayground.com/x/i9V4

Bibliografia

1. Harris, D., & Harris, S. (2007). Digital Design and Computer Architecture (2nd Edition). Morgan Kaufmann.

2. Shannon, C. E. (1949). Communication Theory of Secrecy Systems. Bell System Technical Journal, 28(4), 656-715.