Trabalho Prático - Introdução à Sistemas Lógicos

Marcos Daniel Souza Netto - 2022069492

December 8, 2023

1 Introdução

A atividade é dividida em duas partes: a primeira consiste em implementar um flip-flop D, e a segunda em implementar um *One Time Pad (OTP)*, cujo propósito é cifrar uma mensagem utilizando uma operação de Ou-exclusivo com uma chave de cifragem. A seguir, serão apresentados questões sobre a implementação, os resultados obtidos e os testes implementados.

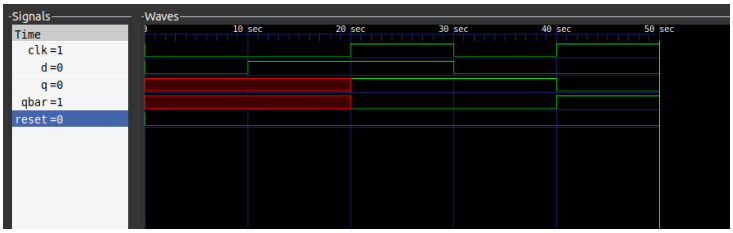
2 *Flip-Flop D*

2.1 Especificação Comportamental

A especificação comportamental do flip-flop D consiste na declaração de um *process* que é sensível à borda ascendente do sinal de clock. A cada borda ascendente do clock, o valor de D é atribuído ao valor de Q e Q recebe o sinal oposto de Q. Esse comportamento é ilustrado na Figura 1b.

```
1  module flipflop_d (
2      input       clk,
3      input       reset,
4      input       d,
5      output reg  q,
6      output      qbar
7  );
8
9      always @(posedge clk)
10     begin
11         if (reset) q <= 1'b0;
12         else q <= d;
13     end
14
15     assign qbar = ~q;
16
17 endmodule
```

(a) Implementação em Verilog



(b) Gráfico de ondas

Figure 1: Especificação comportamental do flip-flop D

2.2 Especificação Estrutural

Já a especificação estrutural do flip-flop D é detalhado os componentes utilizados na construção. Dessa forma, foi definido uma *Latch SR*, utilizada na contrução da *Latch D*, que por sua vez é utilizada na construção do *Flip-Flop D*. Para a construção do *Flip-Flop D*, utilizamos duas *Latch D*, uma como *master* e outra como *slave*. O comportamento do *Flip-Flop D* é ilustrado na Figura 2b.

```
1
2  > module latch_sr( ⋯
7  > ); ⋯
11
12   endmodule
13
14 > module latch_d( ⋯
19 > ); ⋯
33
34   endmodule
35
36 > module flipflop_d (        You, 4 days
42   );
43       wire data;
44       wire clk_bar = ~clk;
45       wire n1;
46
47       assign data = d & ~reset;
48
49       latch_d master_latch(
50           .clk(clk_bar),
51           .d(data),
52           .q(n1),
53           .qbar()
54       );
55
56       latch_d slave_latch(
57           .clk(clk),
58           .d(n1),
59           .q(q),
60           .qbar(qbar)
61       );
62
63   endmodule
```

(a) Implementação em Verilog

(b) Gráfico de ondas

Figure 2: Especificação estrutural do flip-flop D

3 Cifrador

No cifrador implementado, temos uma mensagem de entrada (messagem a ser cifrada) e uma chave de cifragem. As messagens têm um número maior de bits do que as chaves, dessa forma, tratamos a mensagem em *streams* de bits do tamanho da chave, e para cada *stream* de bits, realizamos uma operação de Ou-exclusivo com a chave. Dessa forma, concatenanos os resultados dessas operações e temos a mensagem cifrada.

Para tanto, dividimos esse módulo em três submodulos: *shifter*, *cryptor* e *cypher*.

3.1 *Shifter*

É responsável por receber a mensagem e realizar o deslocamento da mensagem de acordo com o tamanho da chave. Dessa forma, a mensagem é dividida em *streams* de bits do tamanho da chave. Que são enviados para o barramento de saída a cada borda ascendente do clock.

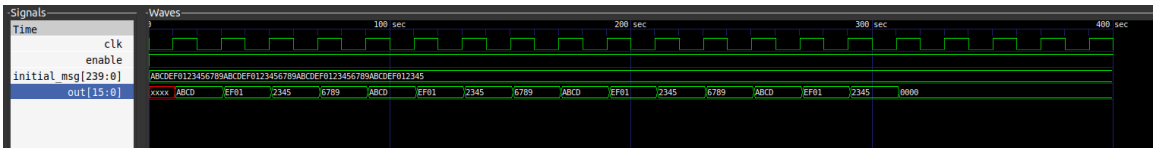


Figure 3: Diagrama de ondas do *shifter*

3.2 *Cryptor*

É responsável por receber a chave e realizar a operação de Ou-exclusivo com a mensagem (Ou a *stream* da mensagem neste caso). Essa operação é realizada de forma contínua, ou seja, a cada modificação da mesagem ou chave, o resultado é atualizado.

```
neg@negIdeapad:~/Documents/dev/isl/cryptor$ vvp cryptor.out
VCD info: dumpfile cryptor_wave.vcd opened for output.

msg=0000000000000000
key=1111111111111111
out=1111111111111111


msg=1111111111111111
key=1111111111111111
out=0000000000000000


msg=1010101010101010
key=0101010101010101
out=1111111111111111


msg=0101010101010101
key=0101010101010101
out=0000000000000000


msg=1111111111111111
key=0101010101010101
out=1010101010101010
```

Figure 4: Cifragem de algumas mensagens

3.3 *Cypher*

É responsável por instanciar os módulos *shifter* e *cryptor*, e realizar a conexão entre eles. Também realiza a concatenação dos resultados do *cryptor* e envia para o barramento de saída quando a mensagem está completa.

```
neg@negIdeapad:~/Documents/dev/isl/cypher_main$ vvp cypher.out
VCD info: dumpfile cypher_wave.vcd opened for output.

ENCODING:
msg =           Hello World! A secret message!
key =           #
ciphertext =    IFmOnVLsOe!b!Pd@sFulFrP`Dd


DECODING:
ciphertext =    IFmOnVLsOe!b!Pd@sFulFrP`Dd
key =           #
msg =           Hello World! A secret message!
```

Figure 5: Cifragem e decifragem de uma mensagem completa

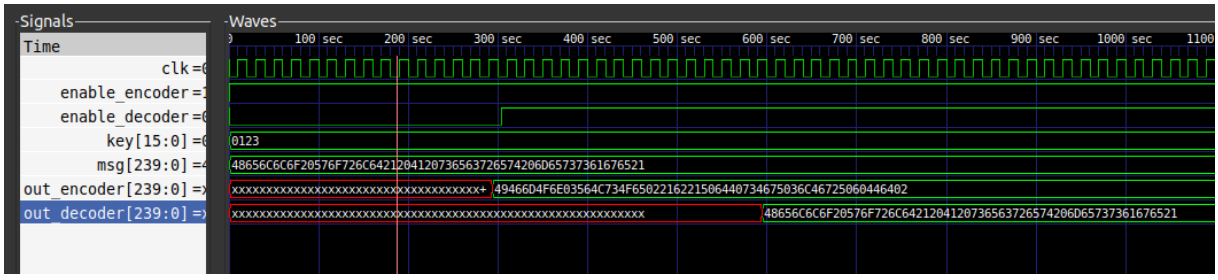


Figure 6: Gráficos de ondas do *cypher*

4 Testes

Cada módulo implementado possui um arquivo de testbench, dessa forma garantimos que cada módulo está funcionando corretamente.

Arquivos de teste:

- *Flip-Flop D*

  Comportamental: *dflipflop_behavioral_tb.v*

  Estrutural: *dflipflop_structural_tb.v*

- *Shifter*: *shifter_tb.v*
- *Cryptor*: *cryptor_tb.v*
- *Cypher*: *cypher_tb.v*

5 Como rodar

5.1 EDA Playground

Os módulos podem ser testados no *EDA Playground* [1]. Para isso, basta copiar o código do módulo desejado e o código do arquivo de teste correspondente, e colar no ambiente. Escolhendo o compilador **Icarus Verilog 0.9.7** e marcando a opção **Open EPWave after run**, basta clicar em rodar.

Alternativemente, pode-se utilizar dos seguintes links para rodar os testes:

- *Flip-Flop D*

  Comportamental: `https://edaplayground.com/x/7S_N`

  Estrutural: `https://edaplayground.com/x/PdMx`

- *Shifter*: `https://edaplayground.com/x/YkX5`
- *Cryptor*: `https://edaplayground.com/x/BhA2`
- *Cypher*: `https://edaplayground.com/x/ZJJT`

5.2 Localmente

Para rodar localmente, é necessário ter o *iverilog* instalado. Após isso, basta rodar o comando:

```
$ iverilog <arquivo principal> <arquivo de teste> -o <nome do executavel>
```

Por exemplo, para rodar o teste do *Flip-Flop D* comportamental, basta rodar o comando:

```
$ iverilog dflipflop_behavioral.sv dflipflop_behavioral_tb.v -o dflipflop_wave.out
```

Para executar o executável gerado, basta rodar o comando :

```
$ vvp <nome do executavel>
```

E para a visualização do gráfico de ondas, é necessário ter algum software tipo *GTKWave* e assim basta rodar o comando:

```
$ gtkwave <nome do arquivo de ondas>
```

[1] `https://www.edaplayground.com/`