Introdução aos Sistemas Lógicos

Cofre de Pão de Queijo

Ítalo Dell'Areti

Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte – MG – Brasil

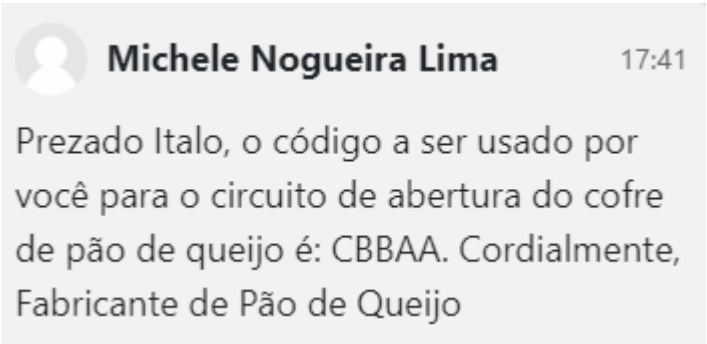
italodellareti@ufmg.br

1. Análise Geral

Este trabalho possui como foco aplicar as técnicas e conhecimentos na área de sistemas lógicos, com o objetivo de construção de um circuito sequencial para controlar o mecanismo de destravamento da porta do cofre. Podemos resumir o comportamento deste cofre da seguinte forma: Ao ser inserida a senha correta, a porta do cofre é aberta, se não a porta permanece fechada. A senha pode ser inserida após sequências incorretas de senha, sendo esses valores, A, B e C. Após a senha ser inserida corretamente, a porta do cofre deve permanecer aberta até o circuito ser resetado (botão reset).

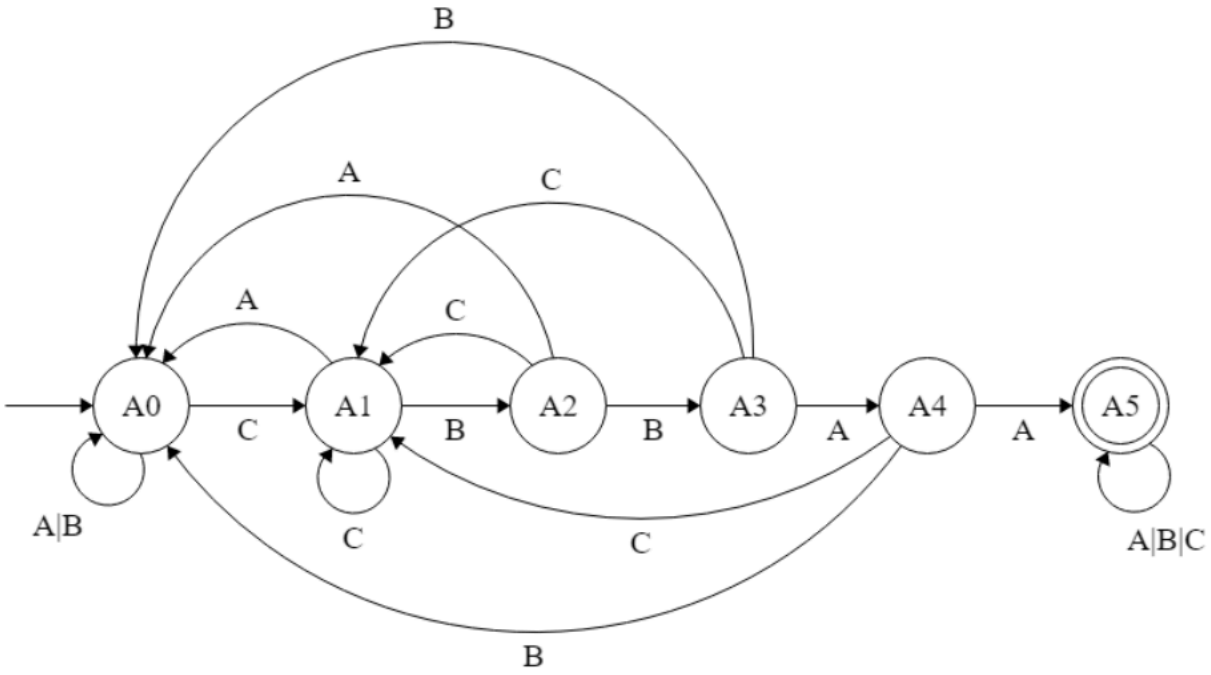
Portanto, para realizar esta implementação utilizaremos a senha fornecida para construir o circuito, pois o circuito vai se moldar de acordo com a senha fornecida. Os passos fornecidos na documentação para a construção, sendo eles os diagramas de estado, tabela de próximo estado, mapa de Karnaugh, circuito e teste em verilog serão seguidos.

O código fornecido para a realização do circuito.



2. Passos

Passo 1: Diagrama de estados



Tendo em vista que a senha é CBBAA, a passagem do estado A0 para o A1 só ocorre quando a letra é C, se for qualquer outra letra (A e B) permanece no mesmo estado.

De A1 para A2, a mudança de estado só ocorre quando a letra B, se a letra for A o estado volta para A0 e se for C ele permanece em A1.

A mudança de estado de A2 para A3 só ocorre quando a for B, se for A ele retorna para A0 e se C volta para o estado anterior.

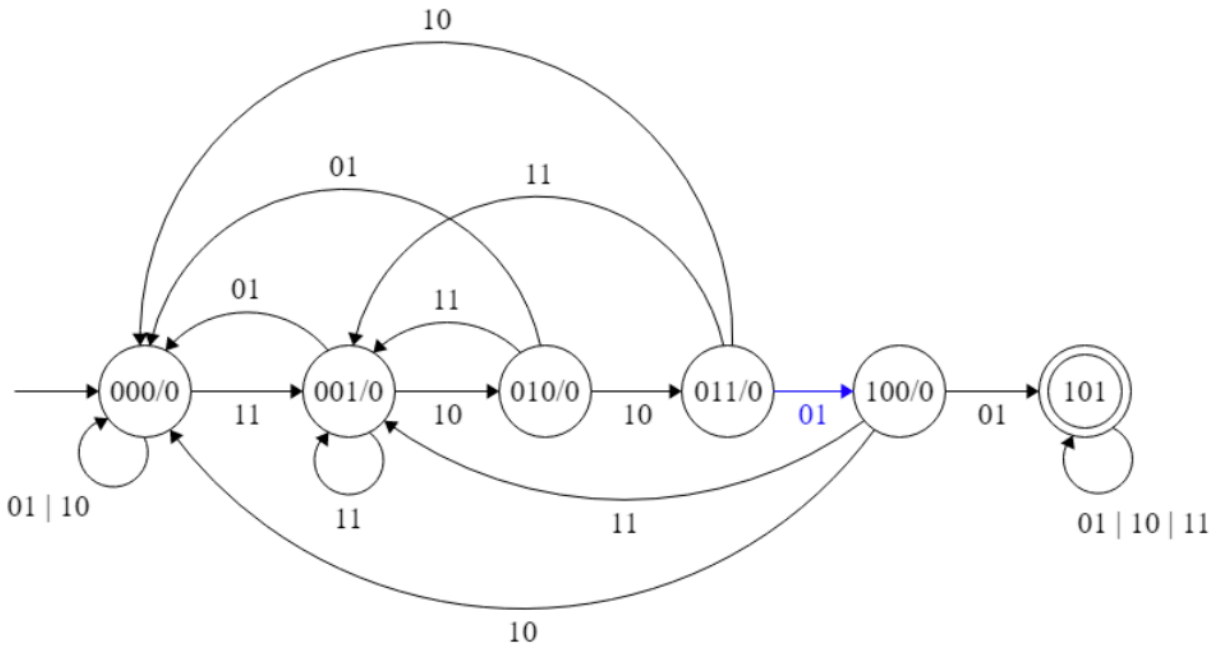
A passagem de A3 para A4 ocorrerá quando a letra da senha for A, se for B ele retorna para o estado A0 e se for C volta para o estado A1.

A alteração de A4 para A5 se for B ele retorna para o estado A0 e se for C volta para o estado A1.

Quando chega no último estado,o A5, independente da letra da senha, a porta já vai estar aberta e não mudará de estado, permanecendo em 101 até que ocorra algum reset.

Estado	Codificação
A0	000
A1	001
A2	010
A3	011
A4	100
A5	101

Entrada	Codificação
	00
A	01
B	10
C	11



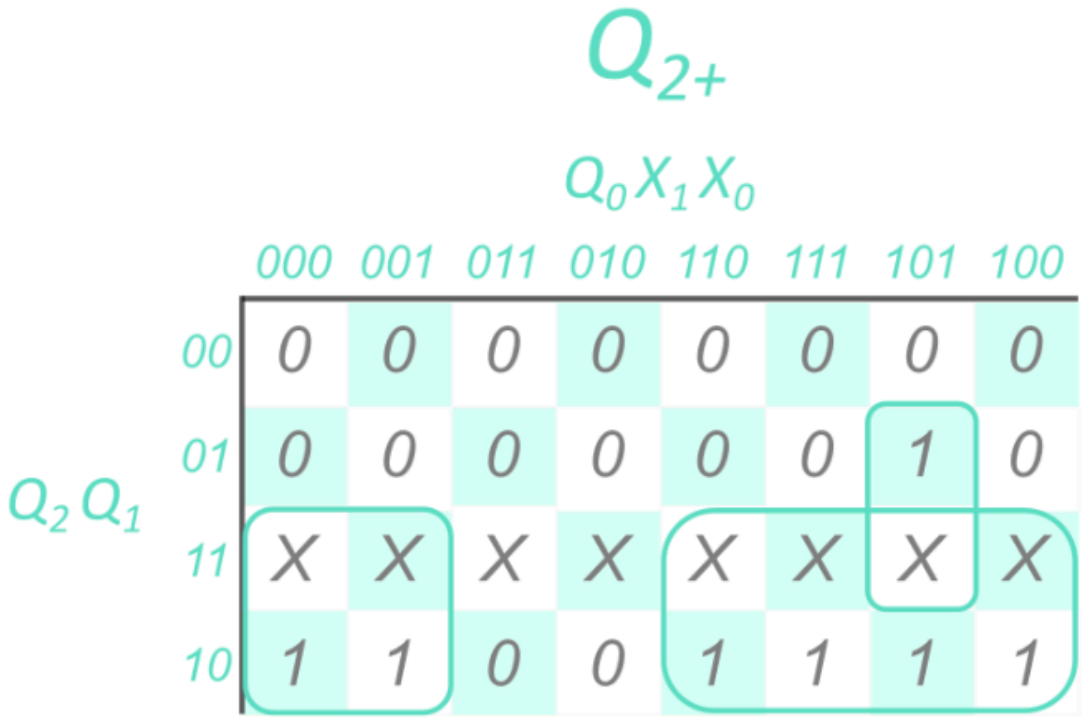
Após a codificação dos estados e das entradas, agora podemos substituir as letras da senha para números que serão utilizados nas tabelas de próximo estado que são apresentadas no próximo passo. A partir deste novo mapa de máquina de estados finitos, poderemos saber os valores que vão fazer com que o circuito passe por todos os estados, até abrir a porta do cofre. Sendo esta sequência:

11 -> 10 -> 10 -> 01 ->01

Passo 2: Tabela de próximo estado

	ESTADO ATUAL			ENTRADAS		PRÓXIMO ESTADO			
ESTADO	Q2	Q1	Q0	X1	X0	Q2+	Q1+	Q0+	Índices
	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1
	0	0	0	1	0	0	0	0	2
A0	0	0	0	1	1	0	0	1	3
	0	0	1	0	0	0	0	1	4
	0	0	1	0	1	0	0	0	5
	0	0	1	1	0	0	1	0	6
A1	0	0	1	1	1	0	0	1	7
	0	1	0	0	0	0	1	0	8
	0	1	0	0	1	0	0	0	9
	0	1	0	1	0	0	1	1	10
A2	0	1	0	1	1	0	0	1	11
	0	1	1	0	0	0	1	1	12
	0	1	1	0	1	1	0	0	13
	0	1	1	1	0	0	0	0	14
A3	0	1	1	1	1	0	0	1	15
	1	0	0	0	0	1	0	0	16
	1	0	0	0	1	1	0	1	17
	1	0	0	1	0	0	0	0	18
A4	1	0	0	1	1	0	0	1	19
	1	0	1	0	0	1	0	1	20
	1	0	1	0	1	1	0	1	21
	1	0	1	1	0	1	0	1	22
A5	1	0	1	1	1	1	0	1	23
	1	1	0	0	0	X	X	X	24
	1	1	0	0	1	X	X	X	25
	1	1	0	1	0	X	X	X	26
	1	1	0	1	1	X	X	X	27
	1	1	1	0	0	X	X	X	28
	1	1	1	0	1	X	X	X	29
	1	1	1	1	0	X	X	X	30
	1	1	1	1	1	X	X	X	31

Passo 3: Mapas de Karnaugh



$$Q_{2+} = Q_1Q_0X_1'X_0 + Q_2X_1' + Q_2Q_0$$

Q_{1+}

Q_2Q_1 \ $Q_0X_1X_0$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	0	0	0
01	1	0	0	1	0	0	0	1
11	X	X	X	X	X	X	X	X
10	0	0	0	0	0	0	0	0

$Q_{1+} = Q_2'Q_1'Q_0X_1X_0' + Q_1Q_0'X_0' + Q_1X_1'X_0'$

Q_{0+}

Q_2Q_1 \ $Q_0X_1X_0$	000	001	011	010	110	111	101	100
00	0	0	1	0	0	1	0	1
01	0	0	1	1	0	1	0	1
11	X	X	X	X	X	X	X	X
10	0	1	1	0	1	1	1	1

$Q_{0+} = X_1X_0 + Q_0X_1'X_0' + Q_1Q_0'X_1 + Q_2X_0 + Q_2Q_0$

Posteriormente realizar os mapas de karnaugh de 5 variáveis (os dois bits de entrada e os três bits de estado anterior) e agrupar os conjuntos de 1's , podemos pegar os mintermos para realizarmos o circuito e pegar o resultado da saída e colocar na entrada Q do respectivo flip-flop D. As funções são :

```
Q2+= Q1Q0X1'X0 + Q2X1' + Q2Q0

Q1+= Q2'Q1'Q0X1X0' + Q1Q0'X0' + Q1X1'X0'

Q0+= X1X0 + Q0X1'X0' + Q1Q0'X1 + Q2X0 + Q2Q0
```

Passo 4: Circuito e Testes

4.1 - Circuito

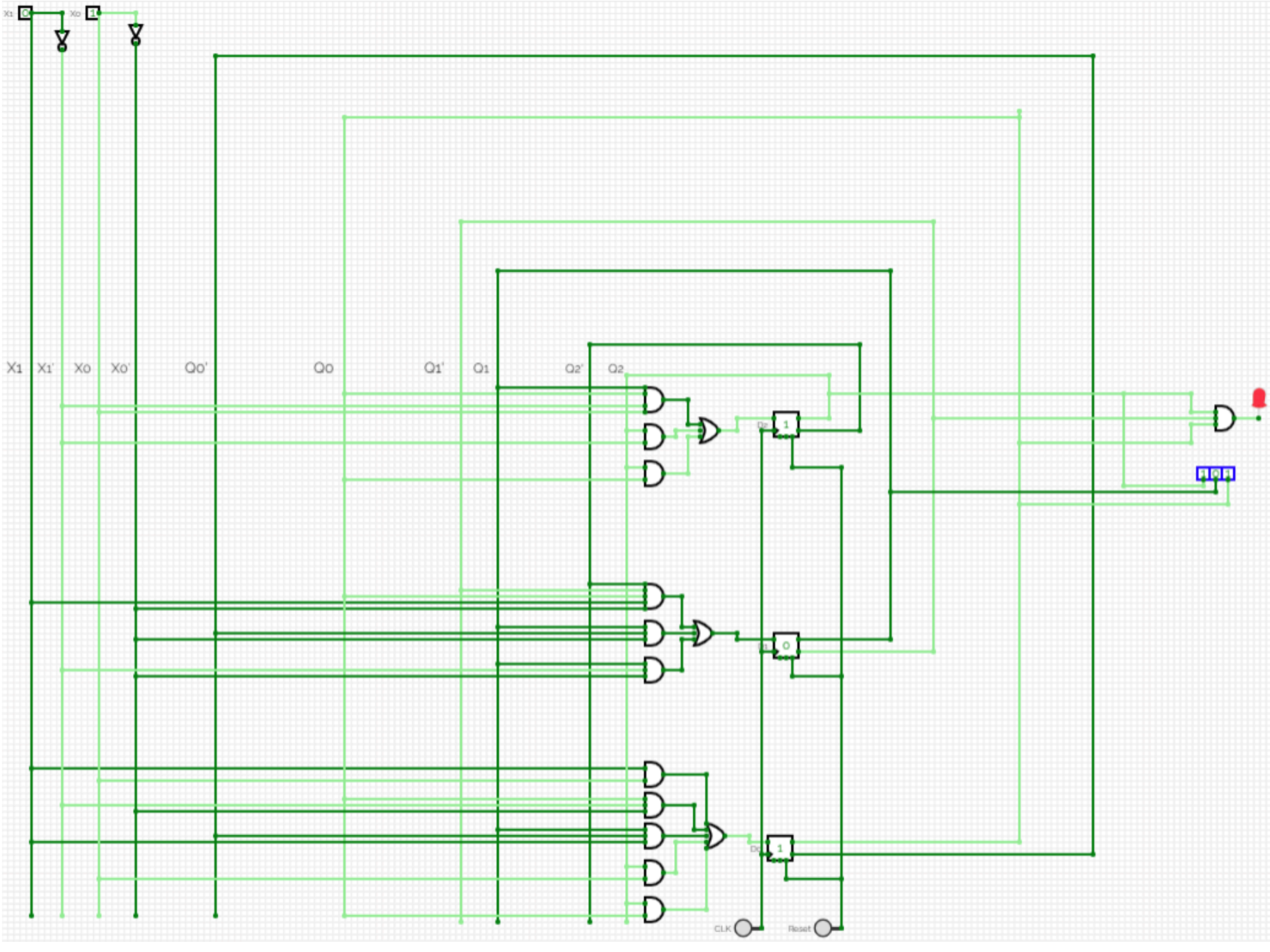
Utilizando a ferramenta CircuitVerse, foi construído o circuito respeitando todos os requisitos necessários

- Três entradas para A, B e C (Possui)
- Um botão para redefinir o estado para 000 (Reset)
- Um relógio (CLK)
- Três flip-flops do tipo D (D2,D1 e D0)
- Três saídas conectadas aos flip-flops (Possui)
- Uma luz LED (Possui)

O teste como o circuito funcionando foi realizado e gravado, podendo ser acessado por estes links.

Google Drive:
https://drive.google.com/file/d/1WGVqZ6JgP3s7dJiVf6QWPISd-ne1I-LU/view?usp=sharing

Dropbox:
https://www.dropbox.com/s/ezbcu88e0vi1y3v/Circuito%20Teste%20Final%20.mp4?dl=0


X1
X1'
X0
X0'
Q0'
Q0
Q1'
Q1
Q2'
Q2
CLK
Reset

4.2 - Código em Verilog

design.sv

```sv
// Code your design here
module Main(X1, X0,reset,clk,led);
    // Componentes exigidos
    output led;
    input clk,reset;
    input X0,X1;
    wire Q2M,Q1M,Q0M;
    wire Q2,Q1,Q0;

  always @ (posedge clk or posedge reset)
  begin
    // referentes aos valores da tabela
    // Q2M = Q2+
    // Q1M = Q1+
    // Q0M = Q0+

    $display("Estado Atual do circuito: %b %b %b",Q2M,Q1M,Q0M);
    $display("Saída do circuito: %b",led);
  end

  DflipFlop D2 (Q2, clk,Q2M, reset);
  DflipFlop D1 (Q1, clk,Q1M, reset);
  DflipFlop D0 (Q0, clk,Q0M, reset);

  // Saídas dos mapas de karnaugh
  assign Q2M = (Q1 & Q0 & ~X1 & X0) | (Q2 & ~X1) | (Q2 & Q0);
  assign Q1M = (~Q2 & ~Q1 & Q0 & X1 & ~X0) | (Q1 & ~Q0 & ~X0) | (Q1 & ~X1 & ~X0);
  assign Q0M = (X1 & X0) | (Q0 & ~X1 & ~X0) | (Q1 & ~Q0 & X1) | (Q2 & X0) | (Q2 & Q0);
  assign led = (Q2 & ~Q1 & Q0);

endmodule

module DflipFlop(q, clk, d, reset);

    output reg q;
    input clk, reset;
    input d;
  always @ (posedge clk or posedge reset)
    if (reset) begin
        q <= 'b0;
    end
    else begin
        q<= d;
    end
endmodule
```

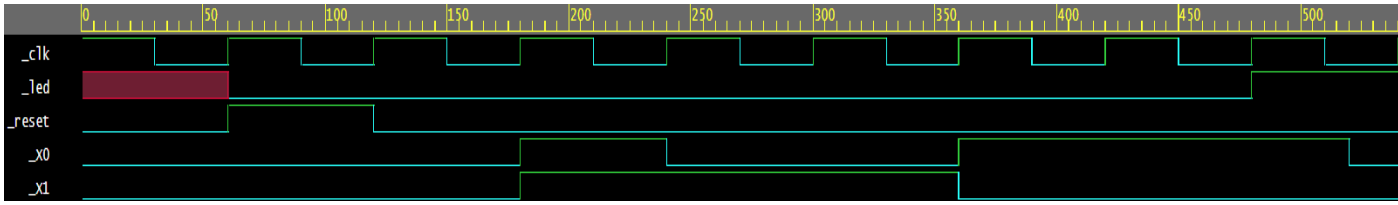
O código em verilog foi implementado de acordo com as estruturas necessárias exigidas e a lógica de linguagem de descrição de hardware. A implantação do flip-flop D, as entradas e saídas exigidas, e a lógica de funcionamento do circuito, de acordo com as entradas da senha específica, vão funcionar perfeitamente com os testes preparados para verificar a consistência do circuito.

4.3 - Testes

testebench.sv

SV/Verilog Testbench

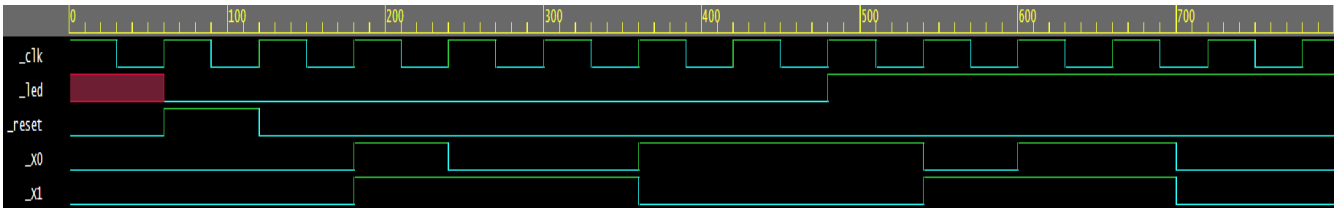
```
module test;
  reg _X0,_X1,_reset,_clk;
  wire _led;

  Main test (.X0(_X0), .X1(_X1),.reset(_reset),.clk(_clk), .led(_led));

  always begin
    _clk = 1;
    #30;
    _clk = 0;
    #30;
  end
  initial begin
    $dumpfile ("dump.vcd");
    $dumpvars (1);
  end

  initial
    begin
    _reset = 0;
    _X1 = 0;
    _X0 = 0;
    #60 _reset = 1;
    #60 _reset = 0;
      //C
    #60 _X1 = 1; _X0 = 1;
      //B
    #60 _X1 = 1; _X0 = 0;
      //B
    #60 _X1 = 1; _X0 = 0;
      //A
    #60 _X1 = 0; _X0 = 1;
      //A
    #60 _X1 = 0; _X0 = 1;

    #100 _X1 = 0 ; _X0 = 0;
    #100 _X1 = 0; _X0 = 0;
    $finish;

  end

endmodule
```



O primeiro teste tem como função verificar se o funcionamento do circuito está sendo correto, inserindo a senha certa e abrindo a porta do cofre. Podemos verificar através do diagrama de tempo que o led (_led) que precisa ser ativo passa de estado de 0 para 1, comprovando o funcionamento do código.

testbench2.sv

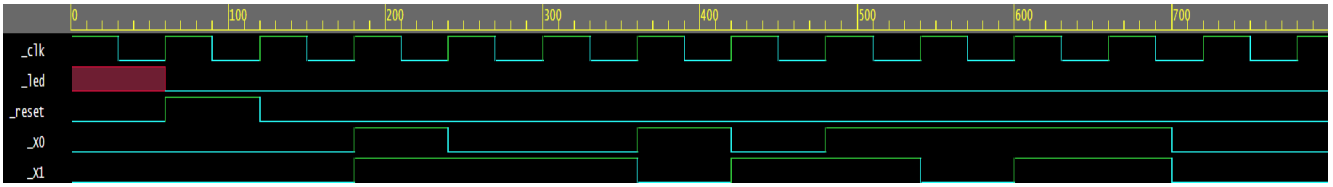
```
module test;
  reg _X0,_X1,_reset,_clk;
  wire _led;

  Main test (.X0(_X0), .X1(_X1),.reset(_reset),.clk(_clk), .led(_led));

  always begin
    _clk = 1;
    #30;
    _clk = 0;
    #30;
  end
  initial begin
    $dumpfile ("dump.vcd");
    $dumpvars (1);
  end

  initial
    begin
    _reset = 0;
    _X1 = 0;
    _X0 = 0;
    #60 _reset = 1;
    #60 _reset = 0;
      //C
    #60 _X1 = 1; _X0 = 1;
      //B
    #60 _X1 = 1; _X0 = 0;
      //B
    #60 _X1 = 1; _X0 = 0;
      //A
    #60 _X1 = 0; _X0 = 1;
      //A
    #60 _X1 = 0; _X0 = 1;
      // Nao vao mudar o estado do circuito
     #60 _X1 = 0; _X0 = 1;
     #60 _X1 = 1; _X0 = 0;
       #60 _X1 = 1; _X0 = 1;

    #100 _X1 = 0 ; _X0 = 0;
    #100 _X1 = 0; _X0 = 0;
    $finish;

  end
endmodule
```



No segundo teste é verificado se, mesmo depois de ter chegado no estado final (101), que é quando o cofre é aberto, com novos inputs o estado do circuito deve permanecer no mesmo estado. O circuito permanece no mesmo estado final.

testbench3.sv

```sv
module test;
  reg _X0,_X1,_reset,_clk;
  wire _led;

  Main test (.X0(_X0), .X1(_X1),.reset(_reset),.clk(_clk), .led(_led));

  always begin
    _clk = 1;
    #30;
    _clk = 0;
    #30;
  end
  initial begin
    $dumpfile ("dump.vcd");
    $dumpvars (1);
  end

  initial
    begin
    _reset = 0;
    _X1 = 0;
    _X0 = 0;
    #60 _reset = 1;
    #60 _reset = 0;
      //C
    #60 _X1 = 1; _X0 = 1;
      //B
    #60 _X1 = 1; _X0 = 0;
      //B
    #60 _X1 = 1; _X0 = 0;
      //A
    #60 _X1 = 0; _X0 = 1;

      // Nao vai chegar no estado final
     #60 _X1 = 1; _X0 = 0;
       #60 _X1 = 1; _X0 = 1;
     #60 _X1 = 0; _X0 = 1;
       #60 _X1 = 1; _X0 = 1;

    #100 _X1 = 0 ; _X0 = 0;
    #100 _X1 = 0; _X0 = 0;
    $finish;

  end
endmodule
```



O teste é feito pensando em colocar vários inputs e verificar se em algum estado vai ativar o led, mesmo com a senha errada. Como podemos verificar,em momento algum o led fica aceso, com base no diagrama de tempo.

Para testar todos os casos basta utilizar a ferramenta EDA Playground e inserir os códigos que estão juntamente com este arquivo, design.sv sendo o circuito e os testbenches sendo as entradas para realizar os testes

3. Conclusão

Após o término do trabalho, depois de praticar vários conceitos abordados durante o curso, como tabela verdade, o comportamento de máquinas de estado finito,mapa de karnaugh, simplificação das equações dos mínimo e máximo termos, a construção dos circuitos baseados nas funções e a implementação dessa lógica em sistemas lógicos, podemos inferir que neste trabalho, podemos revisar e reforçar estes conteúdo, de forma prática.

Além disso, uma nova ferramenta foi nos apresentada, o verilog, que auxilia a realizar o comportamento do circuito de forma virtual e realizar vários testes, mesmo sem necessitar de ter implementado fisicamente o circuito, e isso é uma grande vantagem, de forma que não precisamos perder dinheiro realizando os testes e ganhando tempo, pois realizar vários testes apenas editando algumas linhas de código. O circuito se comporta conforme o planejado e os testes realizados no EDA Playground foram de acordo com o esperado.

Referências

Circuito projetado :

https://circuitverse.org/users/137724/projects/tp-isl-ed724e72-84bd-4997-943f-a2f51bd9b4cf

Projeto EDA verilog:

https://www.edaplayground.com/x/CTGB

Sites utilizados:

https://www.edaplayground.com/

https://circuitverse.org/