

Universidade Federal de São Carlos - UFSCar

|  |  |
| --- | --- |
| Joao Vitor Azevedo Marciano | 743554 |

Lorhan Sohaky de Oliveira Duda Kondo 740951

**Experimento 05 - Implementação de um circuito sequencial utilizando Verilog**

São Carlos - SP

2017

Universidade Federal de São Carlos - UFSCar

|  |  |
| --- | --- |
| Joao Vitor Azevedo Marciano | 743554 |

Lorhan Sohaky de Oliveira Duda Kondo 740951

**Experimento 05 - Implementação de um circuito sequencial utilizando Verilog**

Orientador: Fredy João Valente

Universidade Federal de São Carlos - UFSCar

Departamento de Computação

Ciência da Computação

Laboratório de Circuitos Digitais

São Carlos - SP

2017

Lista de ilustrações

[Figura 1 – Ilustração da máquina de estado do problema da garagem.](#page10) . . . . . . . 9 [Figura 2 – Resultado da compilação do código da máquina de estado do problema](#page13)

[da garagem.](#page13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

Lista de tabelas

Lista de quadros

[Quadro 1 – Lista das entradas da máquina de estado do problema da garagem.](#page9) . . 8 [Quadro 2 – Significados dos estados relacionando com os estados reais do problema](#page10)

[proposto.](#page10) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

Lista de abreviaturas e siglas

FPGA *Field Programmable Gate Array* - Arranjo de Portas Programáveis em

Campo

LED *Light Emitting Diode* - Diodo emissor de luz

Sumário

* [**RESUMO**](#page8) **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7**
* [**DESCRIÇÃO DA EXECUÇÃO DO EXPERIMENTO**](#page9) **. . . . . . . . . 8**

[**2.1Passo 1 – Desenhar a máquina de estado**](#page9) .. . . . . . . . . . . . . . . **8**

[**2.2Passo 2 - Escrever um código Verilog para a máquina de estado**](#page10) .. **9**

**2.3Passo 3 – Executar o código na FPGA e realizar simulação 14**

**2.4Passo 4 – Modificar para englobar comportamento do**

**portão e o aviso luminoso( A para Aberto e F para fechado)**................... 17

**2.5Passo 5 – Circuito extra, sua FSM e código Verilog...............................18**

* [**AVALIAÇÃO DOS RESULTADOS DO EXPERIMENTO**](#page15) **. . . . . . . 20**

**4**[**ANÁLISE CRÍTICA E DISCUSSÃO**](#page16) **. . . . . . . . . . . . . . . . . .**

**INFORMAÇÕES ADICIONAIS . . . . . . . . . . . . . . . . . . . . . . .**

7

1 Resumo

A ideia deste experimento é implementar uma máquina de estado utilizando a linguagem de descrição de *hardware* Verilog. A máquina tenta representar uma situação do mundo real de um portão de garagem.

Considere o cenário de um controle para um portão de garagem. Em um estado inicial, o portão está fechado. Caso um acionador externo seja selecionado, o portão abre. Caso o portão esteja aberto e o acionador externo seja selecionado, o portão fecha. O portão nunca abre e fecha ao mesmo tempo. O trilho no qual o portão se desloca é equipado com dois sensores que indicam quando o portão está completamente aberto e quando está completamente fechado. O motor não deve tentar abrir o portão quando esse estiver aberto e nem deve fechá-lo quando este já estiver fechado.

Para maior segurança dos usuários, o motor está equipado com um aviso luminoso que deve ser acesso quando o portão se desloca.

Deve-se assumir que não é possível parar o portão enquanto ele estiver abrindo ou fechando, mas é possível que o usuário aperte o acionador externo enquanto o portão estiver se deslocando. Nesse caso, se o portão estiver abrindo, ele deve passar a fechar e vice-versa.

Para solucionar tal problema e facilitar sua resolução, dividiu-se o processo em quatro passos:

1. Desenhar a máquina de estado para o cenário em questão;
2. Escrever um código Verilog para a máquina de estado no passo anterior;
3. Executar o código na *Field Programmable Gate Array* - Arranjo de Portas Progra-máveis em Campo (FPGA) e simulação.
4. Modificar código para usar a placa do Kit DE1 para simular o acionador externo, o comportamento do portão e o aviso luminoso. Sinalizar no display a letra A enquanto o portão estiver abrindo e a letra F enquanto o portão estiver fechando. Apresentar o código, o resultado da compilação e fotos que ilustrem o uso do hardware.
5. Além disso, escolheu-se uma máquina de estado qualquer para estudar como implementá-la em Verilog. ( Apresentado na seção “Informações Adicionais”).

2 Descrição da execução do experimento

2.1 Passo 1 – Desenhar a máquina de estado

Com o problema em questão, tem-se em mente que trata-se de um portão de garagem que move-se horizontalmente, ou seja, da esquerda para direita e vise e versa. Assim, para a elaboração da máquina[1](#page9), considerou-se as entradas conforme o [Quadro 1](#page9).

Quadro 1 – Lista das entradas da máquina de estado do problema da garagem.

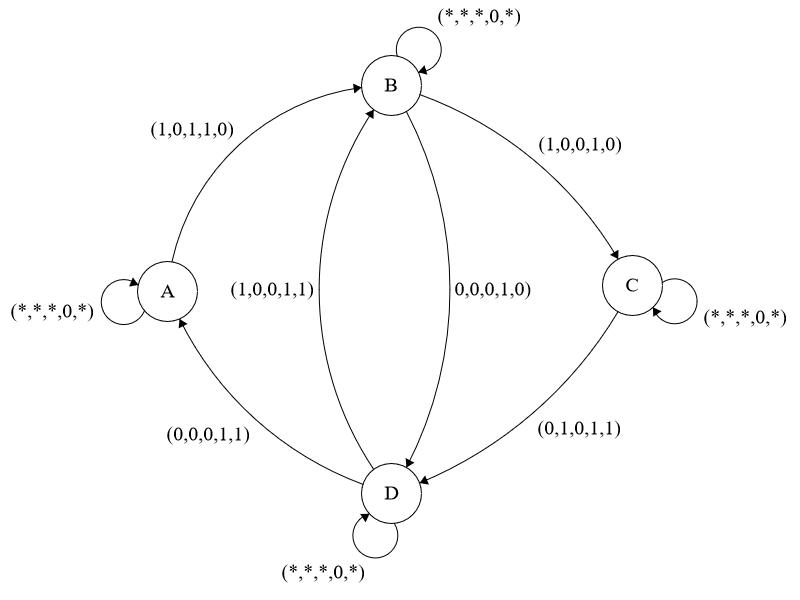
|  |  |  |
| --- | --- | --- |
| **Entrada** | **Valor lógico 0** | **Valor lógico 1** |
| Botao (b) | Abrir | Fechar |
| Aberto (a) | Não aberto | Totalmente aberto |
| Fechado (f) | Não fechado | Totalmente fechado |
| Motor (m) | Motor desligado | Motor ligado |
| Sentido (s) | Esquerda | Direita |
|  | Fonte: Próprio | Autor |

Com as entradas descritas no [Quadro 1](#page9) elaborou-se a máquina conforme a [Figura 1](#page10).

* Preferiu-se a utilização de uma máquina de Moore por haver conhecimento prévio deste tipo de máquina.

|  |  |
| --- | --- |
| *Capítulo 2. Descrição da execução do experimento* | 9 |
|  |  |

Figura 1 – Ilustração da máquina de estado do problema da garagem.



Nota: A entrada está está no formato ( b, a, f, m, s ). Os significados dos estados estão no [Quadro 2](#page10).

Quadro 2 – Significados dos estados relacionando com os estados reais do problema pro-posto.

|  |  |
| --- | --- |
| **Estado** | **Significado** |
| A | Totalmente fechado |
| B | Abrindo |
| C | Totalmente aberto |
| D | Fechando |

Fonte: Próprio Autor

2.2 Passo 2 - Escrever um código Verilog para a máquina de estado

Com a máquina de estado pronto, criou-se a código em Verilog, que no estado "totalmente aberto" apresenta A *display*, no estado "totalmente fechado" apresenta F no *display*, no estado "abrindo" acende um *Light Emitting Diode* - Diodo emissor de luz (LED)verde e no estado "fechando" acente um LED vermelho. Nos estados "abrindo" e "fechando"

|  |  |
| --- | --- |
| *Capítulo 2. Descrição da execução do experimento* | 10 |
|  |  |

o *display* apresenta 0. O Verilog encontra-se no [Código 2.1](#page11) e o resultado da compilação na

[Figura 2](#page13).

module inicial ( botao , aberto , fechado , motor , sentido , ledVerde , ledVermelho , display , clock ) ;

input botao , aberto , fechado , motor , sentido , clock ;

output ledVerde , l e d V e r m e l h o ;

output [6:0] display ;

|  |  |  |  |
| --- | --- | --- | --- |
| reg | [1:0] | estado ; |  |
| reg | [4:0] | entrada ; |  |
| reg | [6:0] | t m p D i s p l a y ; |  |
| reg tmpLedVerde , t m p L e d V e r m e l h o ; | | | |
| p ar am et er | | Fechado = 2 | ’b00 , Abrindo = 2 ’b01 , Aberto = 2 ’b10 , Fechando = |
| 2 ’b11; | |  |  |

initial estado = Fechado ;

always @( posedge clock ) begin

entrada [4] = botao ;

entrada [3] = aberto ;

entrada [2] = fechado ;

entrada [1] = motor ;

entrada [0] = sentido ;

case ( estado )

Fechado : begin

t m p D i s p l a y = 7 ’ b0001110 ;

t m p L e d V e r d e = 0;

t m p L e d V e r m e l h o = 0;

if ( entrada == 5 ’ b10110 ) *// botao = 1 & aberto = 0* *& fechado = 1 & motor = 1 & sentido = 0*

estado = Abrindo ;

end

Abrindo : begin

t m p D i s p l a y = 7 ’ b1000000 ;

t m p L e d V e r d e = 1;

t m p L e d V e r m e l h o = 0;

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | if ( | entrada | == | 5 ’ b10010 | ) *// botao = 1* | *&* | *aberto* | *=* | *0* |
| *&* | *fechado* | *=* | *0* | *&& motor = 1 & sentido = 0* | | | | |  |  |  |  |
|  |  |  |  | estado = | | Aberto ; | |  |  |  |  |  |
|  |  |  |  | if ( | entrada | == | 5 ’ b00010 | ) *// botao = 0* | *&* | *aberto* | *=* | *0* |
| *&* | *fechado* | *=* | *0* | *& motor* | *= 1 &* | *sentido == 0* | | |  |  |  |  |

|  |  |
| --- | --- |
| *Capítulo 2. Descrição da execução do experimento* | 11 |
|  |  |

estado = Fechando ;

end

Aberto : begin

t m p D i s p l a y = 7 ’ b0001000 ;

t m p L e d V e r d e = 0;

t m p L e d V e r m e l h o = 0;

if ( entrada == 5 ’ b01011 ) *// botao = 0 & aberto = 1* *& fechado = 0 & motor = 1 & sentido = 1*

estado = Fechando ;

end

Fechando : begin

t m p D i s p l a y = 7 ’ b1000000 ;

t m p L e d V e r d e = 0;

t m p L e d V e r m e l h o = 1;

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | if ( entrada | == 5 ’ b10011 | ) | *//* | *botao* | *=* | *1* | *&* | *aberto* | *=* | *0* |
| *& fechado = 0* | *& motor = 1 & sentido = 1* | |  |  |  |  |  |  |  |  |  |
|  | estado = | Abrindo ; |  |  |  |  |  |  |  |  |  |
|  | if ( entrada | == 5 ’ b00011 | ) | *//* | *botao* | *=* | *0* | *&* | *aberto* | *=* | *0* |
| *& fechado = 0* | *& motor = 1 & sentido = 1* | |  |  |  |  |  |  |  |  |  |
|  | estado = | Fechado ; |  |  |  |  |  |  |  |  |  |
|  | end |  |  |  |  |  |  |  |  |  |  |

default : estado = Fechado ;

endcase

end

assign display = t m p D i s p l a y ;

assign ledVerde = t m p L e d V e r d e ;

assign l e d V e r m e l h o = t m p L e d V e r m e l h o ;

e nd mo du le

module maquina ( SW , LEDG , LEDR , HEX0 , CLK ) ; input [4:0] SW;

input CLK;

output [0:0] LEDG , LEDR ;

output [6:0] HEX0 ;

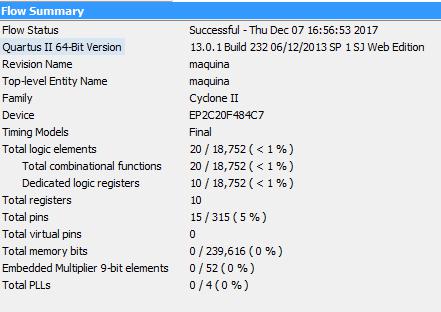
inicial a( SW [4] , SW [3] , SW [2] , SW [1] , SW [0] , LEDG [0] , LEDR [0] , HEX0 , CLK) ;

e nd mo du le

Código 2.1 – Código da máquina de estado do problema da garagem.

|  |  |
| --- | --- |
| *Capítulo 2. Descrição da execução do experimento* | 12 |
|  |  |

Figura 2 – Resultado da compilação do código da máquina de estado do problema da garagem.



Para a realização dos testes, criou-se um *test bench* conforme o [Código 2.2](#page13).

module m a q u i n a \_ T B ;

integer a;

wire ledVerde , l e d V e r m e l h o ;

wire [6:0] HEX ;

reg [4:0] val;

reg clock ;

inicial i( a[4] , a[3] , a[2] , a[1] , a[0] , ledVerde , ledVermelho , HEX , clock ) ;

*// Al te ra nd o clock*

always begin

clock <= 0;

#25;

clock <= 1;

#25;

end

initial begin

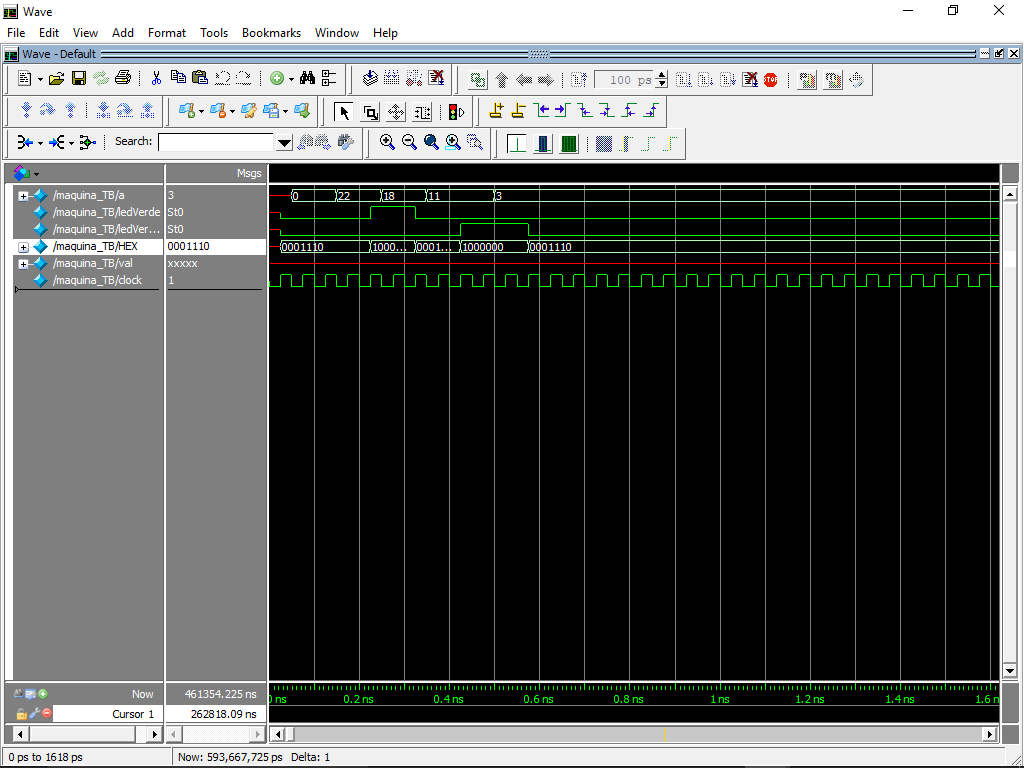
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Capítulo 2. Descrição da execução do experimento* | | | | | |  |  |  |  |  |  |  |  |  |  | 13 |  |
|  |  | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $display ( "\ tSistema Porta \n" ) ; | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $display ( " | | Estado | | | Entrada | LG | | | LR | | | HEX " ) ; | |  |  |  |  |  |  |  |
|  | $display ( " - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -" ) ; | | | | | | | | | |  |  |  |  |  |  |  |
|  | *// Estado* | *Fechado* | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| #50 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | a =0;#100 | *// a = 00000* | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $write ( | " | Fechado | | | %x%x%x%x%x | | | %x | | |  | %x |" , a[4] , | | | a[3] , | a[2] , | | a |  |
|  | [1] , a[0] , ledVerde , l e d V e r m e l h o ) ; | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | if ( HEX | == | 7 ’ b0001110 | | ) $display ( " | | F" |  | ) ; | if ( HEX | == |  | 7 ’ b1000000 | | ) |  |  |
|  | $display | (" | O" ) ; | if ( HEX == 7 ’ b0001000 | | | | | ) | $display ( | | " | A" ) ; |  |  |  |  |
|  | *// Estado* | *Abrindo* | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | a =22;#100 | | *// a =* | *10110* | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $write ( | " | Abrindo | | | %x%x%x%x%x | | | %x | | |  | %x |" , a[4] , | | | a[3] , | a[2] , | | a |  |
|  | [1] , a[0] , ledVerde , l e d V e r m e l h o ) ; | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | if ( HEX | == | 7 ’ b0001110 | | ) $display ( " | | F" |  | ) ; | if ( HEX | == |  | 7 ’ b1000000 | | ) |  |  |
|  | $display | (" | O" ) ; if ( | | HEX == 7 ’ b0001000 | | |  | ) | $display ( | " |  | A" ) ; |  |  |  |  |
|  | *// Estado* | *Aberto* | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | a =18;#100 | | *// a =* | *10010* | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $write ( | " | Aberto | | | %x%x%x%x%x | | | %x | | |  | %x |" , a[4] , | | | a[3] , | a[2] , | | a |  |
|  | [1] , a[0] , ledVerde , l e d V e r m e l h o ) ; | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | if ( HEX | == | 7 ’ b0001110 | | ) $display ( " | | F" |  | ) ; | if ( HEX | == |  | 7 ’ b1000000 | | ) |  |  |
|  | $display | (" | O" ) ; if ( | | HEX == 7 ’ b0001000 | | |  | ) | $display ( | " |  | A" ) ; |  |  |  |  |
|  | *// Estado* | *Fechando* | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | a =11;#100 | | *// a =* | *01011* | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $write ( | " Fechando | | | | %x%x%x%x%x | | | %x | | |  | %x |" , a[4] , | | | a[3] , | a[2] , | | a |  |
|  | [1] , a[0] , ledVerde , l e d V e r m e l h o ) ; | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | if ( HEX | == | 7 ’ b0001110 | | ) $display ( " | | F" |  | ) ; | if ( HEX | == |  | 7 ’ b1000000 | | ) |  |  |
|  | $display | (" | O" ) ; if ( | | HEX == 7 ’ b0001000 | | |  | ) | $display ( | " |  | A" ) ; |  |  |  |  |
|  | *// Estado* | *Fechado* | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| #50 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | a =3;#100 | *// a = 00011* | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $write ( | " | Fechado | | | %x%x%x%x%x | | | %x | | |  | %x |" , a[4] , | | | a[3] , | a[2] , | | a |  |
|  | [1] , a[0] , ledVerde , l e d V e r m e l h o ) ; | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | if ( HEX | == | 7 ’ b0001110 | | ) $display ( " | | F" |  | ) ; | if ( HEX | == |  | 7 ’ b1000000 | | ) |  |  |
|  | $display | (" | O" ) ; | if ( HEX == 7 ’ b0001000 | | | | | ) | $display ( | | " | A" ) ; |  |  |  |  |
|  | end |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | e nd mo du le |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

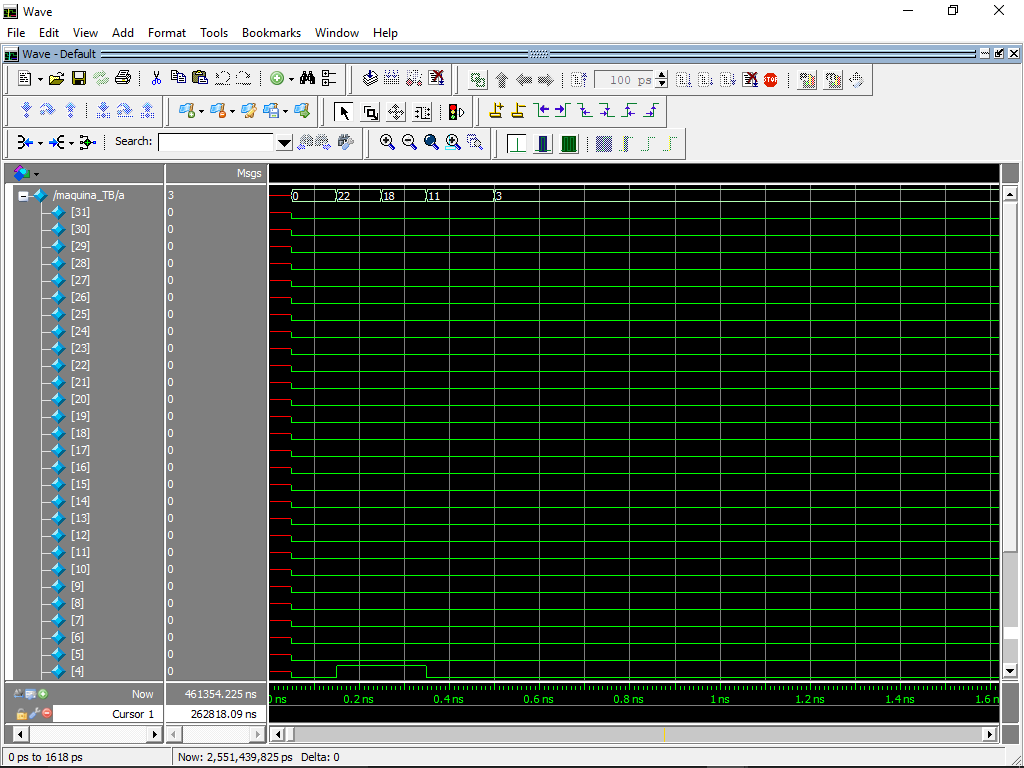
Código 2.2 – Código da máquina de estado do problema da garagem.

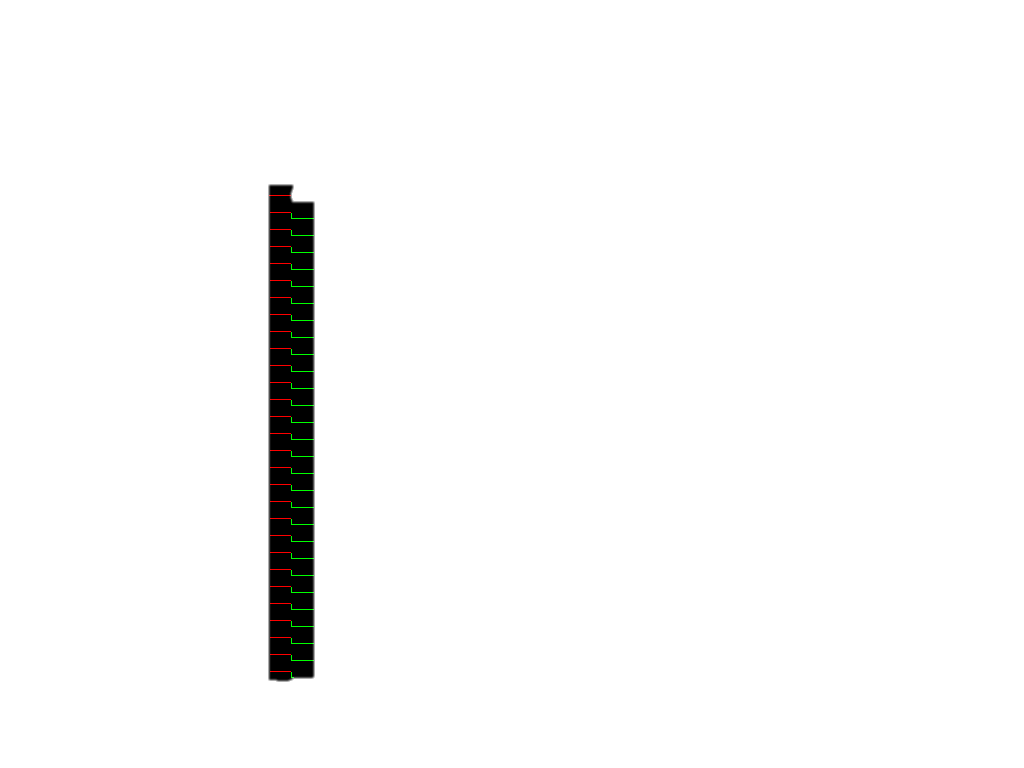
14

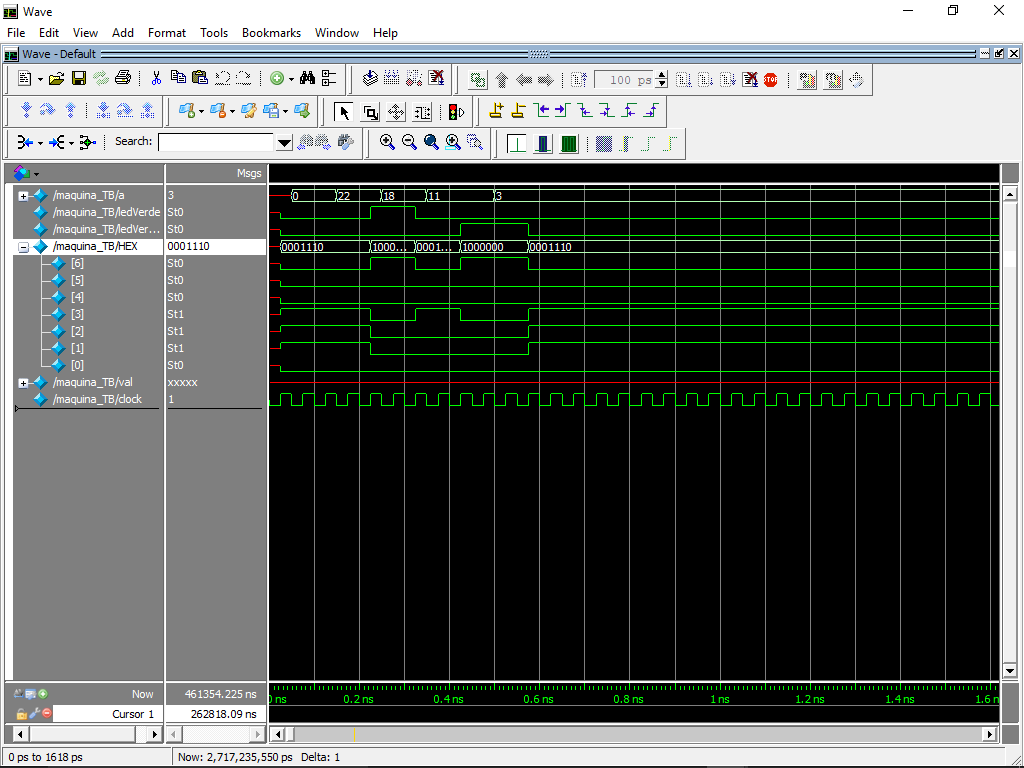
3 Avaliação dos resultados do experimento

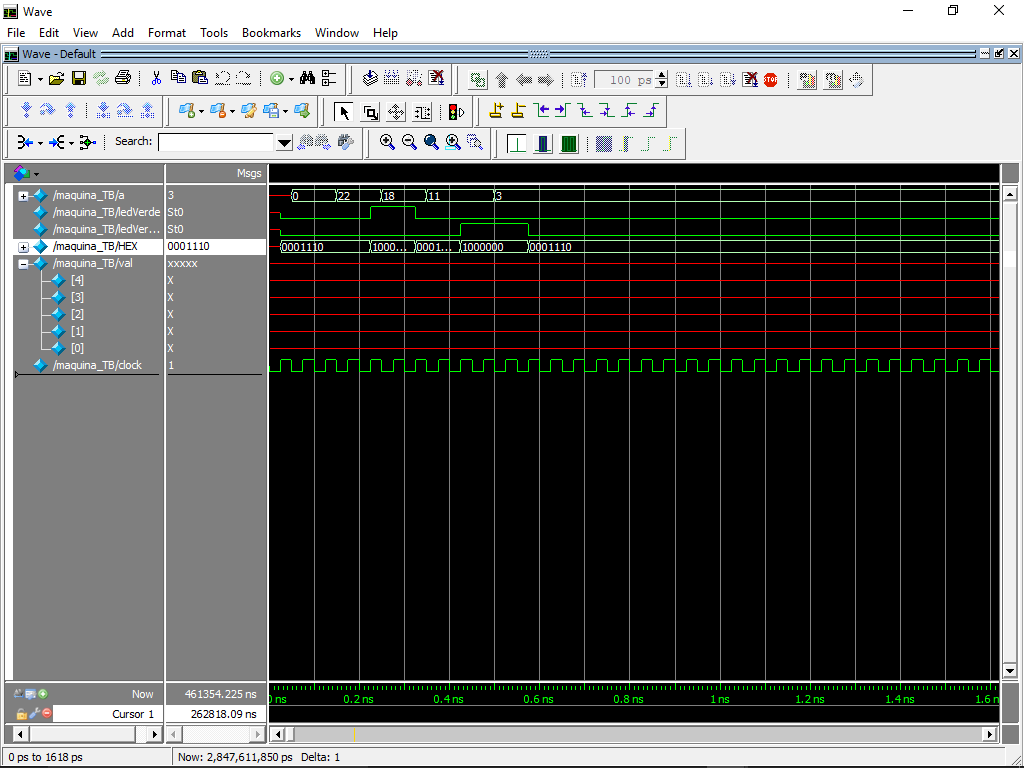
Resultados da Simulação RTL











Com base nas “Waves” da simulação RTL, conclui-se que o circuito programado em

Verilog corresponde ao requisitado na máquina de estados, migrando entre os estados “A”, “B”, “C” e “D” conforme as entradas são alteradas.

Resultado rodando na placa

Não foi possível incluir fotos da placa rodando o circuito, devido à impossibilidade de login na máquina do laboratório.

Porém conforme mencionado, o circuito acende “A” no display de 7 segmentos quando o portão está aberto, e “F”, quando fechado. As entradas definidas na máquina de estados se demonstraram corretas neste teste prático, também, apesar da impossibilidade das fotos.

4 Análise crítica e discussão

O [Código 2.1](#page11) engloba todos as entradas, todos os estados e suas saídas, assim, se fosse um circuito sequencial, englobaria o circuito de excitação, circuito de memória e o circuito de saída.

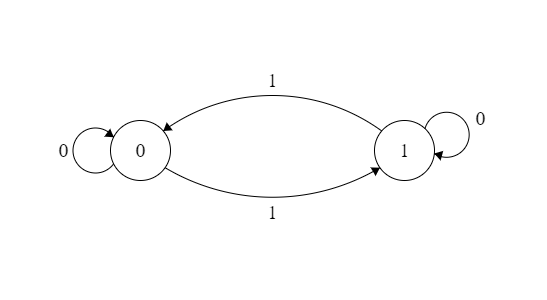
Os resultados obtidos representam o comportamento ideal esperado, como definido no projeto. Das dificuldades encontradas, destacam-se as dificuldades em controlar as mudanças de estados, já que as entradas eram “pegas” automaticamente pela placa da Altera, podendo pular 2 estados de uma vez dependendo da entrada, e assim desorientar e dificultar a vida dos observadores.

Houve relativa facilidade para a idealização e desenvolvimento da máquina de estados, conteúdo já estudado previamente.

5 Análise crítica e discussão

Circuito extra, sua FSM e código Verilog

O código a seguir representa um circuito combinatorio simples, onde estados “A” e “B” são representados por LED’s verde e vermelho, e o pressionar de um botão gera a transição entre os estados.



module inicial( botao, ledVerde, ledVermelho, clock);

input botao, clock;

output ledVerde, ledVermelho;

reg tmpLedVerde, tmpLedVermelho;

parameter A = 1'b0, B = 1'b1;

reg estado = A;

always @( posedge clock) begin

case( estado )

A: begin

tmpLedVerde = 1;

tmpLedVermelho = 0;

if( botao )

estado = B;

end

B: begin

tmpLedVerde = 0;

tmpLedVermelho = 1;

if( botao )

estado = A;

end

default: estado = A;

endcase

end

assign ledVerde = tmpLedVerde;

assign ledVermelho = tmpLedVermelho;

endmodule

module circuitoAdicional ( KEY, LEDG, LEDR, CLK );

input [0:0] KEY;

input CLK;

output [0:0] LEDG, LEDR;

inicial i( KEY[0], LEDG, LEDR, CLK);

endmodule

* Arquivo Test-Bench

module circuitoAdicional\_TB;

wire ledVerde, ledVermelho;

reg clock, botao;

inicial i( botao, ledVerde, ledVermelho, clock);

//Alterando clock

always begin

clock <= 0;

#25;

clock <= 1;

#25;

end

initial begin

$display("\tSistema Porta\n");

$display("Estado | Entrada | LG | LR");

$display("--------------------------");

//Estado A

#50

botao = 0;#100

$display( " A | %x | %x | %x ", botao, ledVerde, ledVermelho);

//Estado B

botao = 1;#100

$display( " B | %x | %x | %x ", botao, ledVerde, ledVermelho);

//Estado A

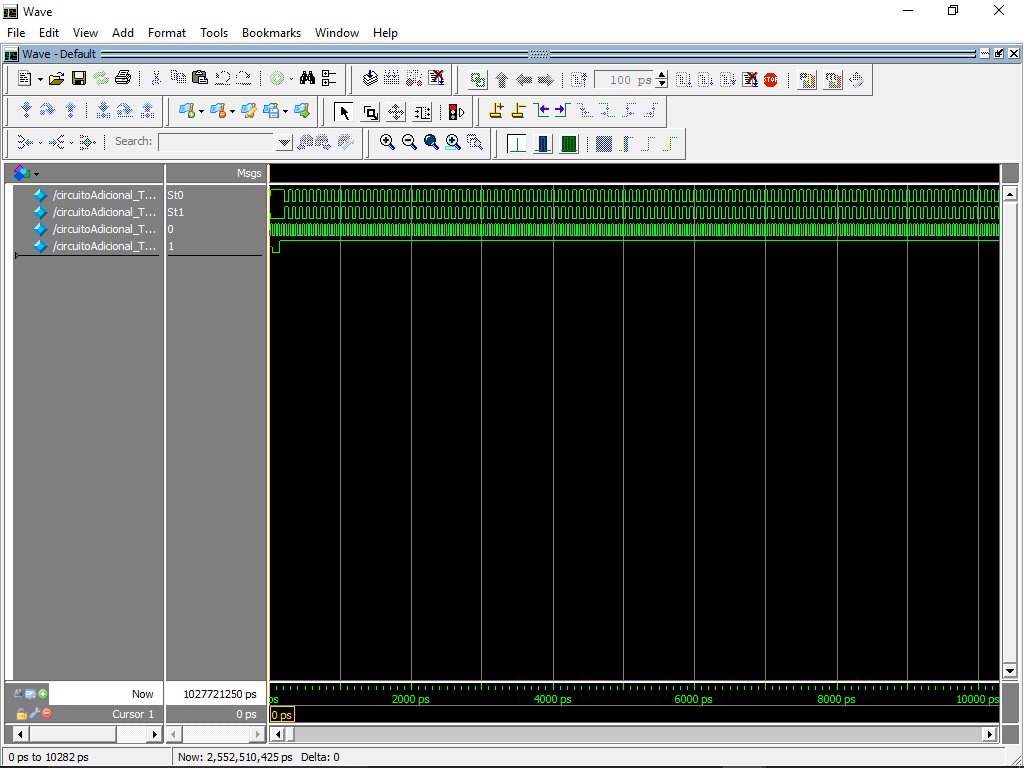
botao = 1;#150

$display( " A | %x | %x | %x ", botao, ledVerde, ledVermelho);

end

endmodule

Simulação RTL



Novamente o circuito se comporta como esperado, aceitando 1 como condição para mudanças entre os estados A e B.