Centro de Informática
Universidade Federal da Paraíba

Relatório Final

Cinema

Caio Victor Do Amaral Cunha Sarmento - 20170021332

João Da Mata De Sousa Neto - 20170020335

João Pessoa, 10 de junho de 2018

Centro de Informática
Universidade Federal da Paraíba

Cinema

Relatório elaborado para o projeto final da disciplina Circuito Lógicos II, ministrada pelo Professor Eudisley Gomes dos Anjos do Centro de Informática da Universidade Federal da Paraíba.

João Pessoa, 10 de junho de 2018

Agradecimentos

Agradecemos aos nossos colegas Bruno Passeti e João Gabriel por terem nos ajudado na implementação do código e por dividirem o FPGA com o nosso grupo, Gabriel Patrício que deu um suporte e ao monitor Thiago Wesley por ajudar com algumas dúvidas inclusive na hora da apresentação do projeto.

Resumo

Este trabalho consiste na simulação de uma máquina de venda de ingressos de cinema com a utilização do FPGA Altera DE II codificado pela linguagem de descrição de hardware (HDL) Verilog. Foram utilizados displays de 7 segmentos, botões, switches e LEDs no processo.

A seguir, será detalhada a utilização do projeto, de como o mesmo foi feito pela equipe e o desenvolvimento do código utilizado na programação da placa, desconsiderando características elétricas da mesma.

Palavras-chave: FPGA.

Lista de figuras

Lista de siglas

- HDL - Hardware Description Language (Linguagem de descrição de hardware);
- FPGA - Field Programmable Gate Array (Arranjo de portas programáveis em campo);
- IDE - Integrated Development Environment (Ambiente de desenvolvimento integrado);
- LEDs - Light Emitting Diode (Diodo emissor de luz);
- HEX[0-7] – Displays de 7 segmentos;
- KEY[3-0] – Botões do FPGA;

Sumário

1. Introdução

Para aplicações em desenvolvimentos de projetos, os FPGAs (Field Programmable Gate Array) são uma alternativa com custo relativamente baixo para simulações, tendo em vista o alto investimento que seria feito para desenvolver um chip ou um processador diretamente. Um único FPGA pode simular não apenas um processador simples, mas também outros circuitos de apoio, como o controlador de vídeo, uma interface serial e assim por diante.

Dessa forma, foi utilizado o FPGA Altera DE2 para a simulação de uma máquina de venda de ingressos de cinema com opções de escolha de dois filmes com dois horários para cada e escolha de assentos para cada sessão.

1.1. Equipe

A equipe foi composta por Caio Victor Do Amaral Cunha Sarmento e João Da Mata De Sousa Neto e ambos fizemos o projeto juntos, presencial e via Discord (Aplicativo de voz e vídeo) através do compartilhamento de tela. Não houve divisão de atividades, pois fizemos juntos o tempo todo.

2. Metodologia

Neste projeto foi utilizada a linguagem de descrição de hardware (HDL) Verilog e o software Quartus II como ambiente de desenvolvimento (IDE) para carregar os códigos compilados para o FPGA Altera DE2.

2.1 Quartus II

O software Quartus II foi utilizado em sua versão 9.1sp2 para criação e compilação do código em Verilog para o seu carregamento na FPGA Altera DE2.

2.2 FPGA Altera DE2

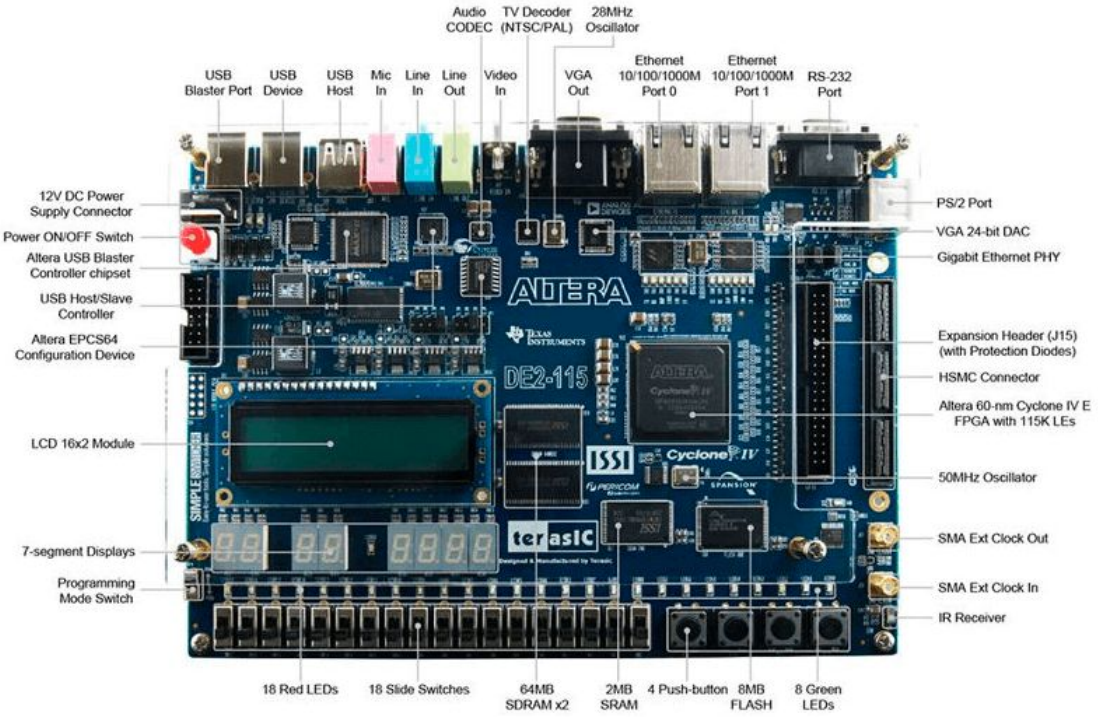

Figura 1 - FPGA Altera DE2

Esta é a placa FPGA Altera DE II que foi utilizada no projeto.

3. Descrição do Projeto

Foram utilizados os displays de 7 segmentos HEX7 e HEX6 para mostrar o filme escolhido com um "F" e "1" para o primeiro filme ou "2" para o segundo, seguido dos HEX5 e HEX4 para mostrar o horário escolhido com um "H" e "1" para o primeiro horário ou "2" para o segundo e os HEX3, HEX2, HEX1, HEX0 para indicar a hora escolhida (14:00 ou 16:00). A escolha do filme é feita por meio dos botões KEY0 para o primeiro e KEY1 para o segundo, seguido dos KEY2 e KEY3 para seleção dos horários de forma análoga ao filme.

Logo após a escolha do horário, os 18 LEDs acima dos switches acendem indicando os assentos disponíveis da sessão. Para selecionar os locais desejados, utiliza-se de 1 a 18 switches e depois o(s) confirma(m) por meio do botão KEY3 e o(s) LED(s) correspondentes se apagam. Depois da seleção, o programa supostamente deveria entrar na parte do pagamento, porém não foi possível implementar corretamente pela pouca disponibilidade de tempo do FPGA, visto que o mesmo teve que ser compartilhado com outro grupo e o problema não foi identificado a tempo.

3.1 Código

```
1  module Cinema(input [17:0]SW, input clk, output reg [17:0]LEDR , input [3:0]KEY, output [7:0]HEX0,
2                output [7:0]HEX4, output [7:0]HEX5, output [7:0]HEX6, output [7:0]HEX7, output [7:0]HEX1, output [7:0]HEX2, output [7:0]HEX3);
3
4   integer filmeEscolhido = 0; //flag que indica que o filme já foi escolhido e passa para a escolha do horário;
5   integer filme = 0; //variável que indica qual filme foi escolhido, 1 ou 2;
6   integer horarioEscolhido = 0; //flag que indica que o horário ja foi escolhido e passa para a escolha dos assentos;
7   integer horario = 0; //variável que indica qual horário foi escolhido, 1 ou 2;
8   integer assentoEscolhido = 0; //flag que indica que o(s) assento(s) foram escolhidos;
9   reg [17:0] assento11 = 18'b000000000000000000; //array que guarda os assentos da sessão do filme 1 horário 1;
10  reg [17:0] assento12 =   18'b000000000000000000; //array que guarda os assentos da sessão do filme 1 horário 2;
11  reg [17:0] assento21 = 18'b000000000000000000; //array que guarda os assentos da sessão do filme 2 horário 1;
12  reg [17:0] assento22 = 18'b000000000000000000; //array que guarda os assentos da sessão do filme 2 horário 2;
13  integer valorInserido = 0; //flag que indica que o valor foi inserido (se tivesse funcionando o estado de pagamento);
14  integer valor = 0; //variável que guarda o valor que o usuário deve pagar;
15  integer pago = 0; //flag que indica se já foi pago o valor;
16  integer quantidadeDeIngresso = 0; //variável que guarda a quantidade de ingressos que o usuário escolheu;
17  integer trocoFeito = 0; //flag que indica se o troco foi dado ao usuário;
18  integer troco = 0; //variável que indica quanto de troco deve ser dado;
19  reg [7:0]F = 4'hF; //array que armazena o "F" de filme no display;
20  reg [7:0]H = 4'hC; //array que armazena o "H" de horário no display;
21  reg [7:0]dFilme = 4'h0; //array que armazena qual filme deve ser indicado no display, 1 ou 2;
22  reg [7:0]dHorario = 4'h0; //array que armazena qual horário deve ser indicado no display, 1 ou 2;
23  reg [7:0]display1 = 4'h0; //array do primeiro display que indica a hora no display, 14:00 ou 16:00;
24  reg [7:0]display2 = 4'h0; //array do segundo display que indica a hora no display, 14:00 ou 16:00;
25  reg [7:0]display3 = 4'h0; //array do terceiro display que indica a hora no display, 14:00 ou 16:00;
26  reg [7:0]display4 = 4'h0; //array do quarto display que indica a hora no display, 14:00 ou 16:00;
27  integer bouncing = 0; //tentativa de uso de flag para parar o bouncing;
28  integer bouncing2 = 0; //tentativa de uso de flag para parar o bouncing;
29
30
```

Código 1/11

```
always@(posedge clk) begin
    if(!filmeEscolhido && bouncing == 0) begin //início da escolha do filme;
        troco = 0;
        if(KEY == 4'b1110) begin //precionar o botao 4 da esquerda para a direita para escolher o filme 1;
            filme = 1; //confirmação de que filme é;
            filmeEscolhido = 1; //ativação da flag;
            dFilme = 4'h1; //variável do display;
            bouncing = 1; //tentativa de parar o bouncing;
        end
        if(KEY == 4'b1101) begin //precionar o botão 3 da esquerda para a direita para escolher o filme 2;
            filme = 2; //confirmação de que filme é;
            filmeEscolhido = 1; //ativação da flag;
            dFilme = 4'h2; //variável do display;
            //bouncing2 = 1;
        end
    end
    if(!horarioEscolhido && filmeEscolhido == 1) begin //início da escolha do horário;
        if(KEY == 4'b1011) begin //precionar o botao 2 da esquerda para a direita para escolher o horário 1;
            horario = 1; //confirmação de que horário é;
            horarioEscolhido = 1; //ativação da flag;
            dHorario = 4'h1; //variável do display para mostrar que é o horário 1
            display1 = 4'h1; //variável do display para mostrar a hora (1);
            display2 = 4'h4; //variável do display para mostrar a hora (4);
            display3 = 4'h0; //variável do display para mostrar a hora (0);
            display4 = 4'h0; //variável do display para mostrar a hora (0);
        end
```

Código 2/11

```
        if(KEY == 4'b0111) begin //precionar o botao 1 da esquerda para a direita para escolher o horário 2;
            horario = 2; //confirmação de que horário é;
            horarioEscolhido = 1; //ativação da flag;
            dHorario = 4'h2; //variável do display para mostrar que é o horário 2
            display1 = 4'h1; //variável do display para mostrar a hora (1);
            display2 = 4'h6; //variável do display para mostrar a hora (6);
            display3 = 4'h0; //variável do display para mostrar a hora (0);
            display4 = 4'h0; //variável do display para mostrar a hora (0);
        end
    end
    quantidadeDeIngresso=0; //zera a quantidade de ingressos escolhidos;
    if(!assentoEscolhido && horarioEscolhido == 1) begin //início da escolha dos assentos;
        integer i; //variável do for;
        LEDR = 18'b111111111111111111; //acender todos os leds;
        for (i = 0; i<18; i = i + 1) begin
            if(SW[i] == 1) quantidadeDeIngresso = quantidadeDeIngresso + 1; //calcular quantas cadeiras foram escolhidas;
        end
        valor = quantidadeDeIngresso * 15; //calcular o preço do ingresso;
        if(KEY == 4'b0011) begin //pressionar os dois primeiros botões para confirmar os assentos;
            assentoEscolhido <= 1; //confirmar o que o assento foi escolhido para não entrar mais no if de escolher os assentos;
            if(filme == 1 && horario == 1) assento11 <= assento11 + SW; //guardar quais assentos foram escolhidos para o filme 1 horário 1
            if(filme == 1 && horario == 2) assento12 <= assento12 + SW; //guardar quais assentos foram escolhidos para o filme 1 horário 2
            if(filme == 2 && horario == 1) assento21 <= assento21 + SW; //guardar quais assentos foram escolhidos para o filme 2 horário 1
            if(filme == 2 && horario == 2) assento22 <= assento22 + SW; //guardar quais assentos foram escolhidos para o filme 2 horário 2
        end
    end

    if(filme == 1 && horario == 1)begin //apagar os leds dos respectivos assentos escolhidos
        LEDR = ~assento11;
        bouncing = 0;
    end
```

Código 3/11

```
88      if(filme == 1 && horario == 2)begin //apagar os leds dos respectivos assentos escolhidos
89          LEDR = ~assento12;
90          bouncing = 0;
91      end
92      if(filme == 2 && horario == 1)begin //apagar os leds dos respectivos assentos escolhidos
93          LEDR = ~assento21;
94          //bouncing2 = 0;
95      end
96      if(filme == 2 && horario == 2)begin //apagar os leds dos respectivos assentos escolhidos
97          LEDR = ~assento22;
98          //bouncing2 = 0;
99      end
100
101     if (!pago && assentoEscolhido == 1) begin //if que inicia a parte do pagamento;
102         if(KEY == 4'b1110) begin //para inserir 5 reais;
103             valorInserido = valorInserido + 5;
104         end
105         if(KEY == 4'b1101) begin //para inserir 10 reais;
106             valorInserido = valorInserido + 10;
107         end
108         if(KEY == 4'b1011) begin //para inserir 20 reais;
109             valorInserido = valorInserido + 20;
110         end
111         if(KEY == 4'b0111) begin //para inserir 50 reais;
112             valorInserido = valorInserido + 50;
113         end
114         if(valor == valorInserido) begin //conferir se o valor inserido é igual ao que se deve pagar;
115             pago = 1; //ativação da flag;
116         end
```

Código 4/11

Código 5/11

```
117
118         if (valor < valorInserido && !trocoFeito) begin //se o valor inserido for maior do que se deve pagar deverá ter troco
119             troco <= valorInserido - valor; //cálculo do troco;
120             pago <= 1; //ativação da flag;
121             trocoFeito <= 1; //ativação da flag;
122         end
123     end
124
125     if(pago == 1) begin //zerando todas as variáveis para começar o programa novamente;
126         filmeEscolhido <= 0;
127         horarioEscolhido <= 0;
128         horario <= 0;
129         assentoEscolhido <= 0;
130         valorInserido <= 0;
131         valor <= 0;
132         pago <= 0;
133         quantidadeDeIngresso <= 0;
134         trocoFeito <= 0;
135         dinheiroFeito <= 0;
136         pago <= 0;
137     end
138 end
139
```

```
140  reg [7:0] SevenSeg; // array auxiliar;
141      always @(*) //display que mostra "F";
142      case(F) //variável que vai mudar o display;
143          4'h0: SevenSeg = 8'b11111100;
144          4'h1: SevenSeg = 8'b01100000;
145          4'h2: SevenSeg = 8'b11011010;
146          4'h3: SevenSeg = 8'b11110010;
147          4'h4: SevenSeg = 8'b01100110;
148          4'h5: SevenSeg = 8'b10110110;
149          4'h6: SevenSeg = 8'b10111110;
150          4'h7: SevenSeg = 8'b11100000;
151          4'h8: SevenSeg = 8'b11111110;
152          4'h9: SevenSeg = 8'b11110110;
153          4'hF: SevenSeg = 8'b10001110;
154          default: SevenSeg = 8'b00000000;
155      endcase
156      assign {HEX7[0], HEX7[1], HEX7[2], HEX7[3], HEX7[4], HEX7[5], HEX7[6], HEX7[7]} = ~SevenSeg;
157
```

Código 6/11

```
158      reg [7:0] SevenSeg1;
159      always @(*) //display que mostra "H";
160      case(H)
161          4'h0: SevenSeg1 = 8'b11111100;
162          4'h1: SevenSeg1 = 8'b01100000;
163          4'h2: SevenSeg1 = 8'b11011010;
164          4'h3: SevenSeg1 = 8'b11110010;
165          4'h4: SevenSeg1 = 8'b01100110;
166          4'h5: SevenSeg1 = 8'b10110110;
167          4'h6: SevenSeg1 = 8'b10111110;
168          4'h7: SevenSeg1 = 8'b11100000;
169          4'h8: SevenSeg1 = 8'b11111110;
170          4'h9: SevenSeg1 = 8'b11110110;
171          4'hC: SevenSeg1 = 8'b01101110;
172          default: SevenSeg1 = 8'b00000000;
173      endcase
174      assign {HEX5[0], HEX5[1], HEX5[2], HEX5[3], HEX5[4], HEX5[5], HEX5[6], HEX5[7]} = ~SevenSeg1;
175
```

Código 7/11

```
176      reg [7:0] SevenSeg2;
177      always @(*) //display que mostra qual filme vai ser, "1" ou "2";
178      case(dFilme)
179          4'h0: SevenSeg2 = 8'b11111100;
180          4'h1: SevenSeg2 = 8'b01100000;
181          4'h2: SevenSeg2 = 8'b11011010;
182          4'h3: SevenSeg2 = 8'b11110010;
183          4'h4: SevenSeg2 = 8'b01100110;
184          4'h5: SevenSeg2 = 8'b10110110;
185          4'h6: SevenSeg2 = 8'b10111110;
186          4'h7: SevenSeg2 = 8'b11100000;
187          4'h8: SevenSeg2 = 8'b11111110;
188          4'h9: SevenSeg2 = 8'b11110110;
189          4'hC: SevenSeg2 = 8'b01101110;
190          default: SevenSeg2 = 8'b00000000;
191      endcase
192      assign {HEX6[0], HEX6[1], HEX6[2], HEX6[3], HEX6[4], HEX6[5], HEX6[6], HEX6[7]} = ~SevenSeg2;
193
```

Código 8/11

```
194     reg [7:0] SevenSeg3;
195     always @(*) //display que mostra qual horário vai ser, "1" ou "2";
196     case(dHorario)
197         4'h0: SevenSeg3 = 8'b11111100;
198         4'h1: SevenSeg3 = 8'b01100000;
199         4'h2: SevenSeg3 = 8'b11011010;
200         4'h3: SevenSeg3 = 8'b11110010;
201         4'h4: SevenSeg3 = 8'b01100110;
202         4'h5: SevenSeg3 = 8'b10110110;
203         4'h6: SevenSeg3 = 8'b10111110;
204         4'h7: SevenSeg3 = 8'b11100000;
205         4'h8: SevenSeg3 = 8'b11111110;
206         4'h9: SevenSeg3 = 8'b11110110;
207         4'hC: SevenSeg3 = 8'b01101110;
208         default: SevenSeg3 = 8'b00000000;
209     endcase
210     assign {HEX4[0], HEX4[1], HEX4[2], HEX4[3], HEX4[4], HEX4[5], HEX4[6], HEX4[7]} = ~SevenSeg3;
211
212     reg [7:0] SevenSeg4;
213     always @(*) //primeiro display que vai indicar a hora;
214     case(display1)
215         4'h0: SevenSeg4 = 8'b11111100;
216         4'h1: SevenSeg4 = 8'b01100000;
217         4'h2: SevenSeg4 = 8'b11011010;
218         4'h3: SevenSeg4 = 8'b11110010;
219         4'h4: SevenSeg4 = 8'b01100110;
220         4'h5: SevenSeg4 = 8'b10110110;
221         4'h6: SevenSeg4 = 8'b10111110;
222         4'h7: SevenSeg4 = 8'b11100000;
223         4'h8: SevenSeg4 = 8'b11111110;
224         4'h9: SevenSeg4 = 8'b11110110;
225         4'hC: SevenSeg4 = 8'b01101110;
226         default: SevenSeg4 = 8'b00000000;
227     endcase
228     assign {HEX3[0], HEX3[1], HEX3[2], HEX3[3], HEX3[4], HEX3[5], HEX3[6], HEX3[7]} = ~SevenSeg4;
229
```

Código 9/11

```
230     reg [7:0] SevenSeg5;
231     always @(*) //segundo display que vai indicar a hora;
232     case(display2)
233         4'h0: SevenSeg5 = 8'b11111100;
234         4'h1: SevenSeg5 = 8'b01100000;
235         4'h2: SevenSeg5 = 8'b11011010;
236         4'h3: SevenSeg5 = 8'b11110010;
237         4'h4: SevenSeg5 = 8'b01100110;
238         4'h5: SevenSeg5 = 8'b10110110;
239         4'h6: SevenSeg5 = 8'b10111110;
240         4'h7: SevenSeg5 = 8'b11100000;
241         4'h8: SevenSeg5 = 8'b11111110;
242         4'h9: SevenSeg5 = 8'b11110110;
243         4'hC: SevenSeg5 = 8'b01101110;
244         default: SevenSeg5 = 8'b00000000;
245     endcase
246     assign {HEX2[0], HEX2[1], HEX2[2], HEX2[3], HEX2[4], HEX2[5], HEX2[6], HEX2[7]} = ~SevenSeg5;
247
248     reg [7:0] SevenSeg6;
249     always @(*) //terceiro display que vai indicar a hora;
250     case(display3)
251         4'h0: SevenSeg6 = 8'b11111100;
252         4'h1: SevenSeg6 = 8'b01100000;
253         4'h2: SevenSeg6 = 8'b11011010;
254         4'h3: SevenSeg6 = 8'b11110010;
255         4'h4: SevenSeg6 = 8'b01100110;
256         4'h5: SevenSeg6 = 8'b10110110;
257         4'h6: SevenSeg6 = 8'b10111110;
258         4'h7: SevenSeg6 = 8'b11100000;
259         4'h8: SevenSeg6 = 8'b11111110;
260         4'h9: SevenSeg6 = 8'b11110110;
261         4'hC: SevenSeg6 = 8'b01101110;
262         default: SevenSeg6 = 8'b00000000;
263     endcase
264     assign {HEX1[0], HEX1[1], HEX1[2], HEX1[3], HEX1[4], HEX1[5], HEX1[6], HEX1[7]} = ~SevenSeg6;
265
```

Código 10/11

```
266         reg [7:0] SevenSeg7;
267         always @(*) //quarto display que vai indicar a hora;
268         case(display4)
269             4'h0: SevenSeg7 = 8'b11111100;
270             4'h1: SevenSeg7 = 8'b01100000;
271             4'h2: SevenSeg7 = 8'b11011010;
272             4'h3: SevenSeg7 = 8'b11110010;
273             4'h4: SevenSeg7 = 8'b01100110;
274             4'h5: SevenSeg7 = 8'b10110110;
275             4'h6: SevenSeg7 = 8'b10111110;
276             4'h7: SevenSeg7 = 8'b11100000;
277             4'h8: SevenSeg7 = 8'b11111110;
278             4'h9: SevenSeg7 = 8'b11110110;
279             4'hC: SevenSeg7 = 8'b01101110;
280             default: SevenSeg7 = 8'b00000000;
281         endcase
282         assign {HEX0[0], HEX0[1], HEX0[2], HEX0[3], HEX0[4], HEX0[5], HEX0[6], HEX0[7]} = ~SevenSeg7;
283
284
285     endmodule
286
287     /*  SW -> switches de 0 à 17;
288         LEDR -> leds de 0 à 17;
289         KEY -> botões de 0 à 3;
290         HEX0 -> display 0;
291         HEX1 -> display 1;
292         HEX2 -> display 2;
293         HEX3 -> display 3;
294         HEX4 -> display 4;
295         HEX5 -> display 5;
296         HEX6 -> display 6;
297         HEX7 -> display 7;
```

Código 11/11

3.2 Diagrama de fluxo

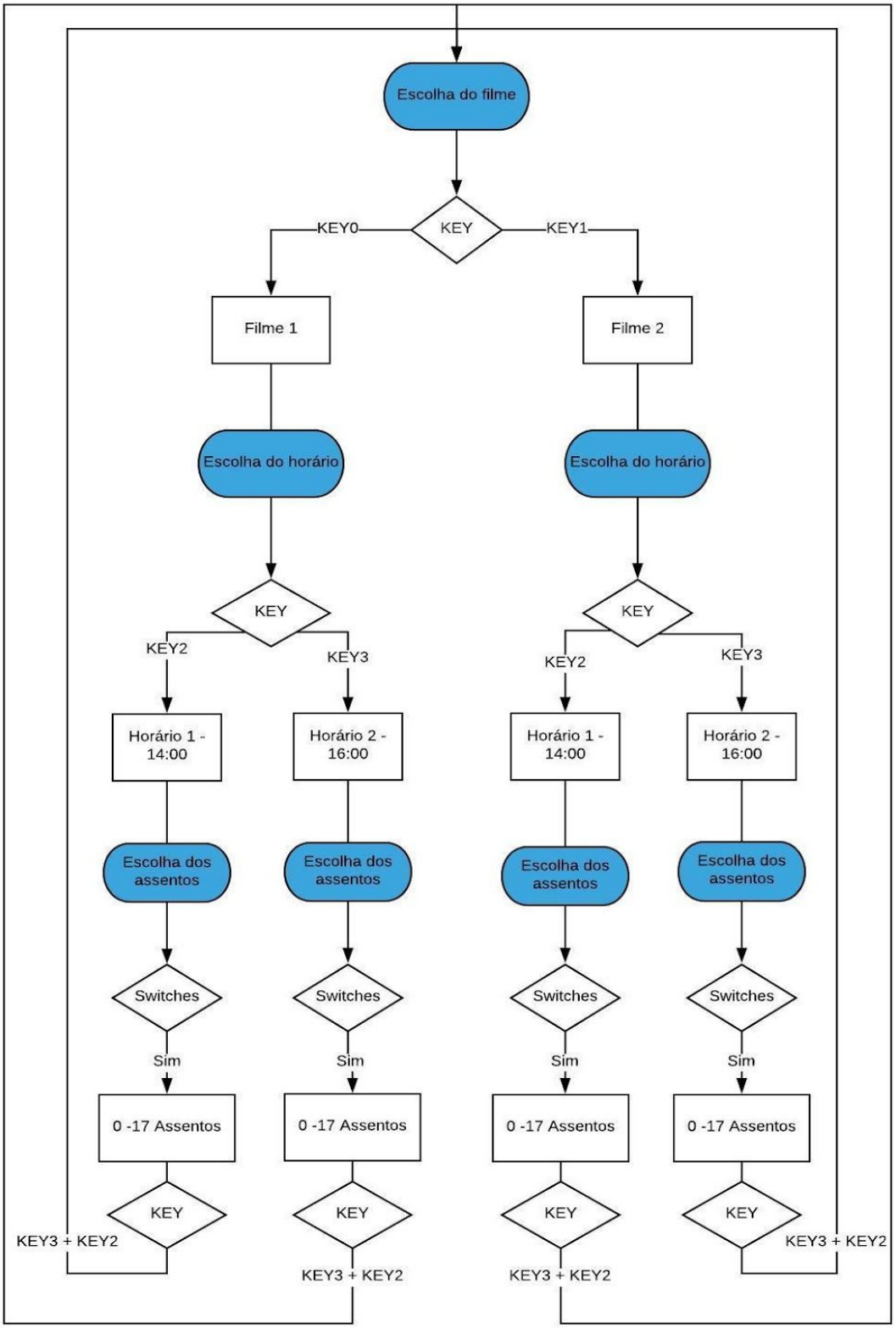

Figura 2 – Diagrama de fluxo

4. Execução do Projeto, Testes e Resultados

Com a compilação do código e envio para o FPGA, perceberam-se diversos problemas, como o *bouncing*, que está localizado na escolha do horário 2, pois utilizamos o mesmo botão, KEY3, para selecionar o horário 2 e para confirmar as cadeiras, mesmo tendo tentado alterar para KEY0 e KEY1 ao mesmo tempo para tentar solucionar e não ter resolvido. O modo pensado após apresentação do projeto para tratar o *bouncing* foi utilizar os dois botões KEY3 e KEY2 para confirmar os assentos, assim funcionando corretamente.

Além disso, o grupo não obteve disponibilidade de FPGA, sendo necessário compartilhá-lo com outro grupo, tendo assim pouco tempo para realizar os testes e resolvê-los antes da entrega do projeto. Devido a isso, também não foi possível a implementação do LCD, porém, foi transferido o que seria mostrado nele para os Displays de 7 segmentos, com os dois primeiros displays mostrando o filme e o número do filme escolhido com “F1” ou “F2” e os próximos seis mostrando os horários selecionados “H1 1400” ou “H2 1600”.

Outro problema encontrado foi o caso da implementação do pagamento dos ingressos, que segundo a lógica do grupo, aparentava estar correto, porém ao transferir para o FPGA, descobriu-se que ele não entrava no *“if”* que inciava o bloco do pagamento, assim comprometendo este trecho do projeto, já que não tinha disponibilidade suficiente do FPGA para testes.

Sendo assim, considerando que o nosso e outro grupo ficaram sem o equipamento necessário para a realização do projeto, precisando dividir com outros dois grupos que acabaram tendo que sacrificar tempo deles com o mesmo para nos emprestar, tivemos dificuldade para conseguir terminá-lo. Contudo, os projetos foram interessantes e com temáticas atraentes, porém seria melhor se nos seguintes o número de grupos fosse menor, tendo equipamentos suficientes para todos os grupos ou que a universidade conseguisse disponibilizar mais deles para a quantidade estipulada para os grupos.

5. Conclusões

Devido ao fato do compartilhamento do FPGA com outro grupo, não houve tempo suficiente para realizar os testes necessários com o código e o equipamento, como a implementação do pagamento corretamente e o funcionamento do LCD. Além disso, um integrante do grupo precisou se ausentar do curso temporariamente, porém ainda foi possível se obter um resultado satisfatório em relação ao projeto.

Sendo assim, o projeto conseguiu atingir grande parte do que deveria, apesar dos problemas que surgiram ao longo do seu desenvolvimento.

6. Referências

[1] - https://www.fpga4fun.com/Opto3.html <Acesso em: 07/06/2018>