Universidade Federal de Santa Catarina
Centro Tecnológico - CTC
Departamento de Engenharia Elétrica - EEL

Gustavo Olegário (15100742)
Caio Pereira Oliveira (15100724)

TURMA 01208A

gustavo-olegario@hotmail.com
caiopoliveira@gmail.com

Relatório de Projeto Final EEL5105 2015.1

Florianópolis, 12 de Julho de 2015

Conteúdo

1. Introdução

A ideia do trabalho proposto foi de se implementar um código, em VHDL, que simulasse um jogo de batalha naval. O código foi feito para ser usado no kit ALTERA DE2 - Cyclone II EP2C35F672C6.

Para este projeto usamos: um contador, um conversor de clock de 50 MHz para 1 Hz, memórias ROM, um decodificador para display de sete segmentos, um comparador. O conversor de clock e a memória ROM foram disponibilizados previamente pelo professor.

A função do decodificador é transformar vetores de 4 bits em vetores de 6 bits que ativam os segmentos apropriados para mostrar o número correto.

O conversor de clock tem como tarefa transformar um clock de 50 MHz para 1 Hz, a fim de que a contagem de tempo de cada jogador possa ser feita em segundos.

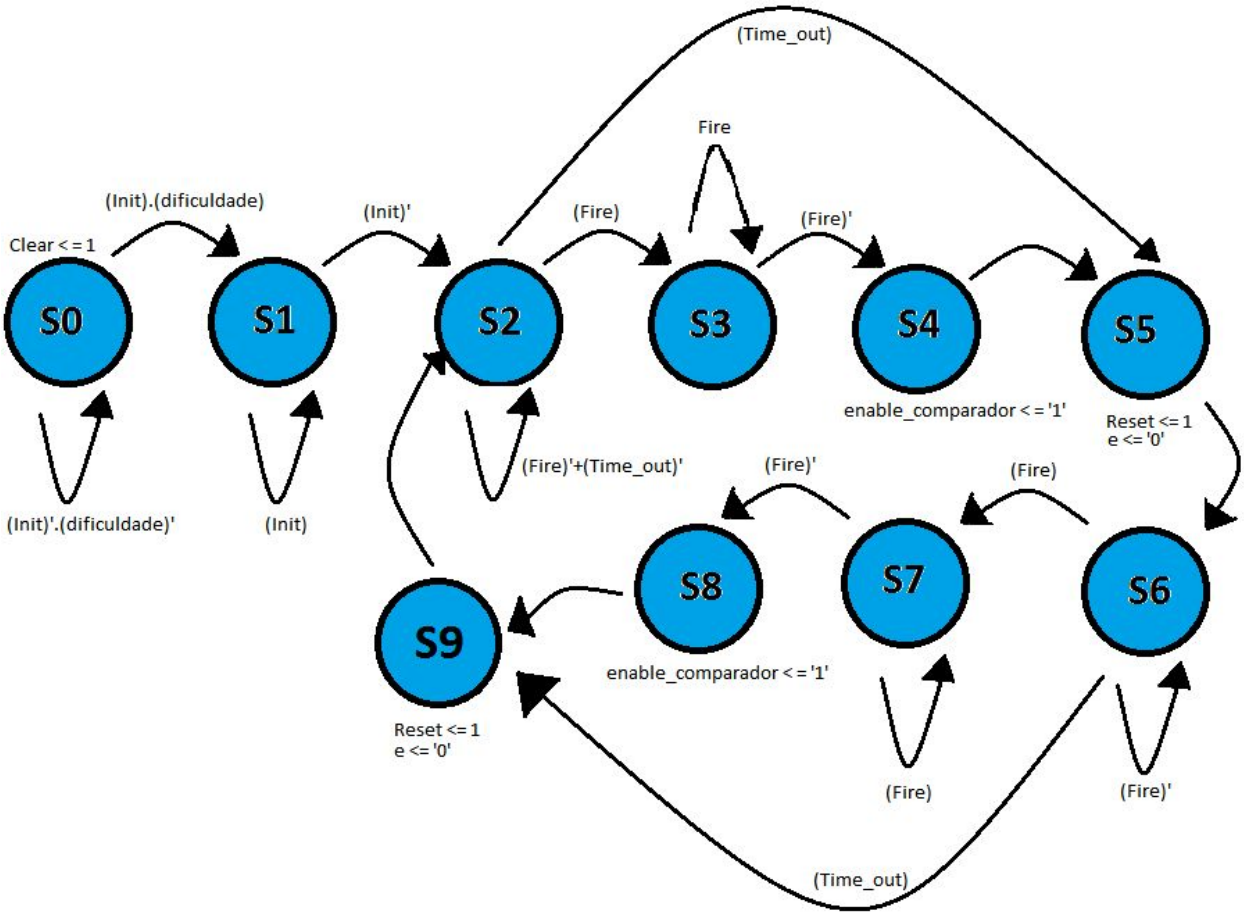
O contador calcula quanto tempo resta para o jogador fazer a sua jogada, baseado na dificuldade escolhida ao iniciar o jogo.

O comparador analisa a jogada do usuário e compara ela com o local na memória onde estão os navios e avisa ao jogador se a jogada teve algum efeito ou não.

A memória ROM tem como tarefa guardar as posições de navio escolhida por cada jogador.

2. Temporizador

Para implementar nosso temporizador, usando VHDL comportamental, criamos uma máquina de estados finitos que controla o funcionamento do resto do circuito através de enables e resets.



Descrição dos estados:

S0 - Estado inicial da FSM, todas as variáveis são zeradas e então espera pela dificuldade ser escolhida e o botão INIT apertado.

S1 - Grava a dificuldade escolhida, impedindo-a de ser alterada no decorrer do jogo, inicia o processo de reset do cronômetro e espera o usuário soltar o botão INIT.

S2 - Termina de resetar o cronômetro, determina o jogador atual como Jogador 1 e passa as informações necessárias para os displays de 7 segmentos (Endereço de memória atual (Switches 15 e 14), linha de tiro (Switches 13 a 0) e tempo restante).

Espera o fim do turno do jogador 1, que acaba caso ele aperte FIRE (indo para o estado S3, que contabiliza o tiro) ou caso o tempo restante chegue a 0 (indo para o estado S5 fazendo o jogador 1 “perder a vez”).

S3 - Espera o jogador soltar o botão FIRE

S4 - Ativa o enable do comparador e do contador de pontos.

S5 - Reseta as variáveis referentes ao turno do jogador 1, inicia o reset do cronômetro e desativa o enable do comparador e do contador de pontos.

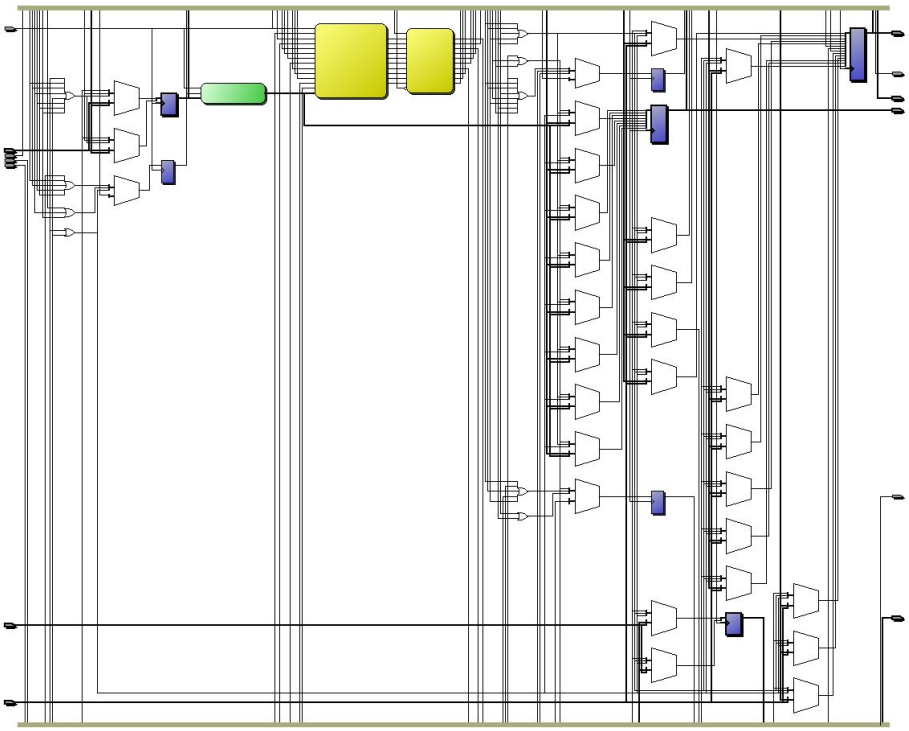
S6 - Termina de resetar o cronômetro, determina o jogador atual como Jogador 2 e passa as informações necessárias para os displays de 7 segmentos (Endereço de memória atual (Switches 15 e 14), linha de tiro (Switches 13 a 0) e tempo restante).

Espera o fim do turno do jogador 2, que acaba caso ele aperte FIRE (indo para o estado S3, que contabiliza o tiro) ou caso o tempo restante chegue a 0 (indo para o estado S9 fazendo o jogador 2 “perder a vez”).

S7 - Espera o jogador soltar o botão FIRE.

S8 - Ativa o enable do comparador e do contador de pontos.

S9 - Reseta as variáveis referentes ao turno do jogador 2, inicia o reset do cronômetro e desativa o enable do comparador e do contador de pontos.



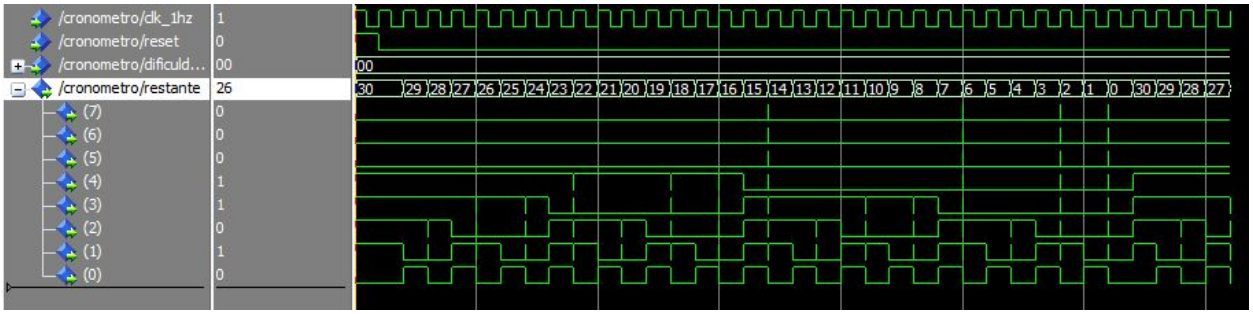
Descrição RTL do temporizador

2.1. Cronômetro

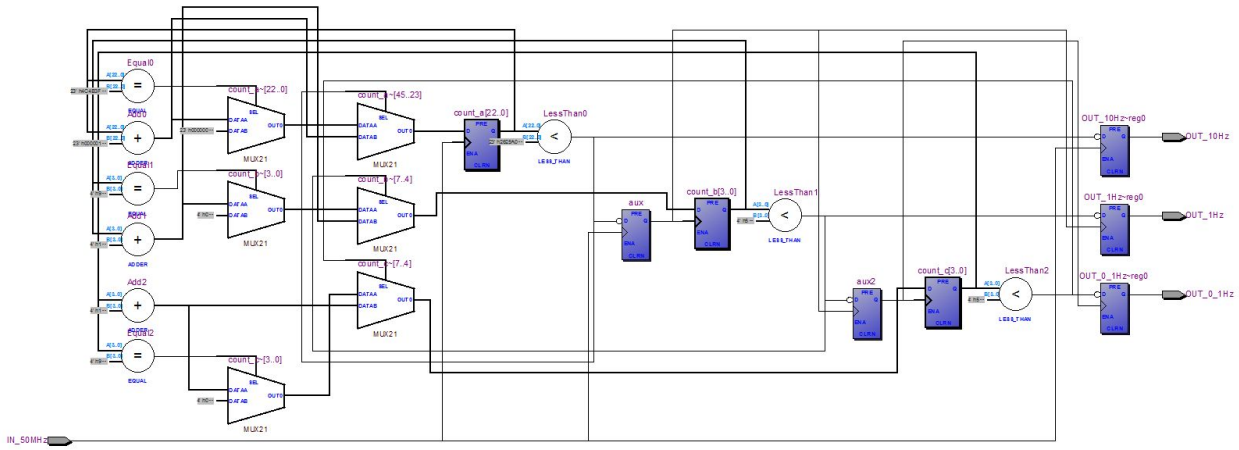
A ideia para esse cronômetro, como foi dito anteriormente, é poder informar quantos segundos o usuário possui para fazer a sua jogada.

O clock de 50 MHz , o qual foi usado como base do projeto, é muito rápido e, por si só, não seria possível indicar corretamente o tempo restante. Sendo assim, o clock de 50 MHz é ligado num conversor de clock realiza uma contagem até 25 mil, incrementando o valor atual em 1 a cada ciclo de clock, e então inverte o clock de 1 Hz, recomeçando a contagem.

Esse novo clock de 1 Hz será utlilizado como clock para o cronômetro que mais tarde será codificado para o display de sete segmentos. Como podemos perceber, esse método nos permite criar um clock de 1 Hz mesmo dispondo de apenas um clock de 50 MHz e outras unidades lógicas. O sinal de reset assíncrono do cronômetro nos permite recomeçar a contagem, independemente do estado que estivermos.



Simulação do cronômetro

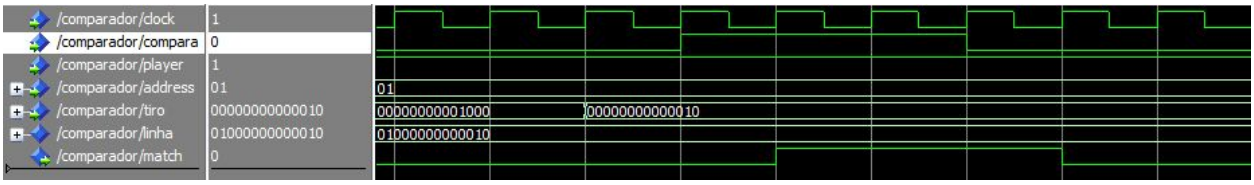


Circuito do conversor de clock

3. Comparador

O objetivo do comparador é gerar um bit de saída para o contador de pontos chamado MATCH, que será posto em nível lógico alto caso a o tiro do jogador acerte uma posição de navio determinado na memória. Para isso, usamos um misto de VHDL comportamental e estrutural, comportamental para manter a sincronia do circuito e estrutural para ter um circuito mais simples.

Durante a partida, os usuários escolherão uma coordenada fazendo o uso das chaves para isso. Após a escolha ter sido feita, o comparador verificará se o "input" está correto, fazendo uma comparação com sua memória ROM e alterando o bit MATCH de acordo.



Simulação do comparador, mostrando que a saída MATCH somente será '1' caso o enable (variável "compara") esteja em '1' e o tiro acerte alguma posição de memória (variável "linha") que guarde um navio.

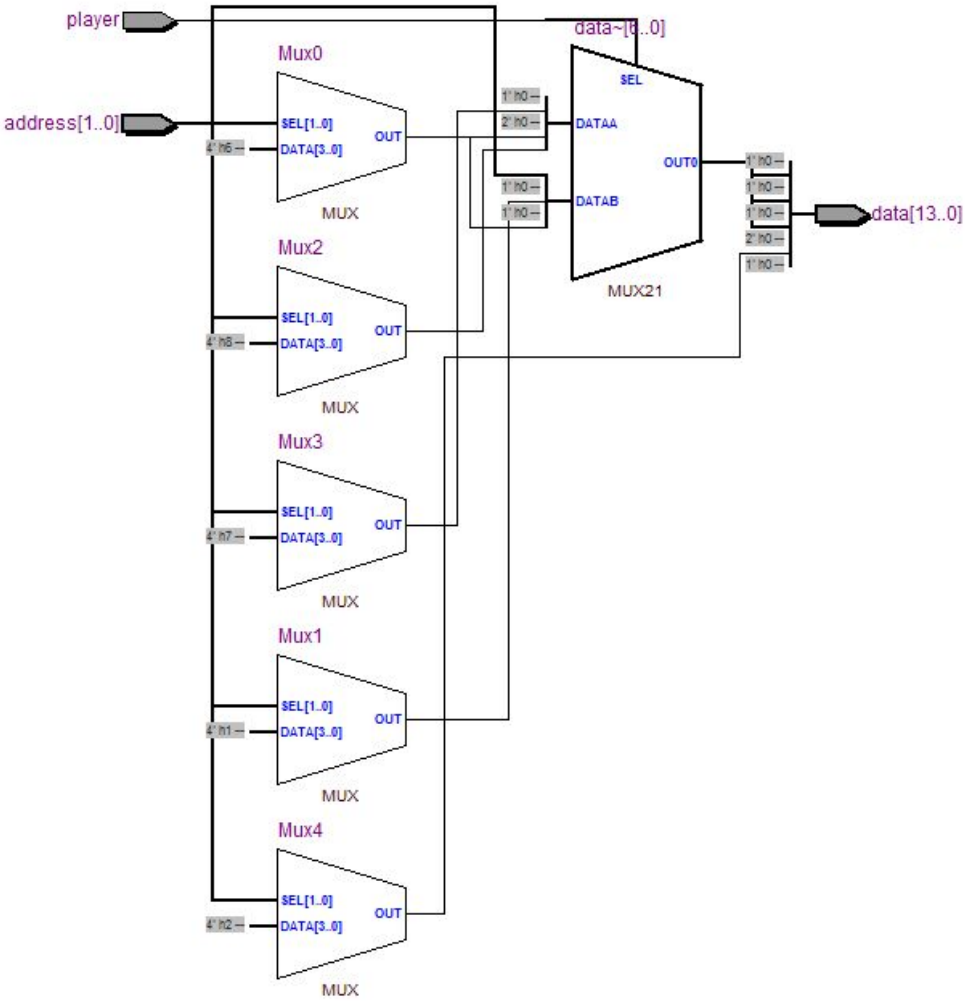
4. Memória

Para este trabalho, utilizamos uma memória ROM (Read-only memory) disponibilizada pelo professor com algumas alterações nossas.

Esta memória tem como entradas: um std_logic chamado player, que assume '0' para o jogador 1 e '1' para o jogador 2 e um std_logic_vector de 2 bits chamado address que define a linha desejada da matriz. Como uníca saida, contém um std_logic_vector de 14 bits que possui o conteúdo da linha determinada pelas entradas.

Entre as linhas 57 e 76 do arquivo ROM.vhd, está definido o processo que de acordo com as entradas, determina a saída.

As coordenas dos navios de cada jogador são escolhidas pelos usuários antes do início da partida. Os jogadores devem alterar o código nas matrizes, colocando 1’s para os locais onde se encontram uma das coordenadas do navio e 0’s nas posições com água.



Circuito da memória ROM

5. Coordenadas

O display de sete segmentos da placa possui grande importância neste projeto. Para que pudéssemos usá-lo corretamente, fizemos um decodificador que recebe um vetor de 4 bits do sistema hexadecimal e o converte para um vetor de 7 bits que indica quais segmentos devem ser ligados ou desligados para que o número desejado apareça no display.

O conversor para sete segmentos foi feito utilizando VHDL comportamental.

```
entity decod is
port (C: in std_logic_vector(3 downto 0);
 S: out std_logic_vector(6 downto 0)
 );
end decod;

architecture circuito of decod is
begin
-- decodifica binario para 7 segmentos
S <=  "1000000" when C = "0000" else
"1111001" when C = "0001" else
"0100100" when C = "0010" else
"0110000" when C = "0011" else
"0011001" when C = "0100" else
"0010010" when C = "0101" else
"0000010" when C = "0110" else
"1111000" when C = "0111" else
"0000000" when C = "1000" else
"0011000" when C = "1001" else
"0001000" when C = "1010" else
"0000011" when C = "1011" else
"1000110" when C = "1100" else
"0100001" when C = "1101" else
"0000110" when C = "1110" else
"0001110" when C = "1111" else
"1111111";
end circuito;
```

Trecho do VHDL do decodificador para 7 segmentos

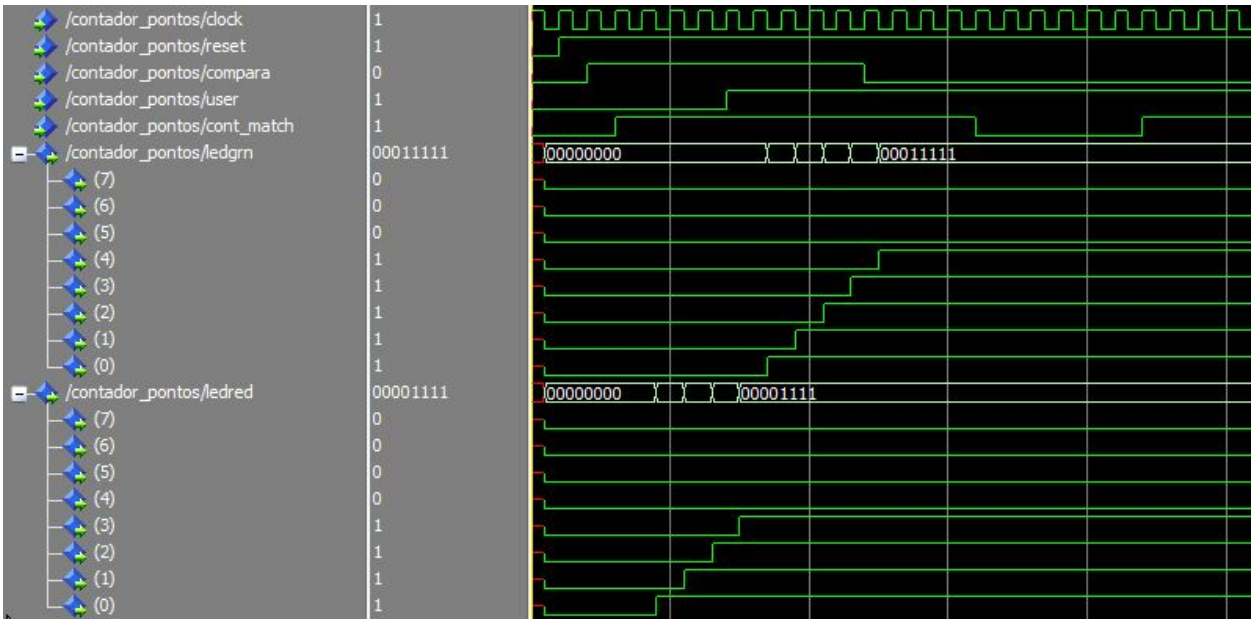
6. Contador de pontos

Para a indicar a contagem de pontos de cada jogador, foram usados 8 LEDs verdes e mais 8 LEDs vermelhos. Sempre que um jogador consegue acertar uma coordenada do navio adversário um de seus LEDs são acesos.

Para armazenar a pontuação de cada jogador, utilizando VHDL comportamental, criamos um contador de pontos que possui dois vetores de 8 bits (um para LEDR e outro para LEDG), quando o enable do comparador estiver ativo e o MATCH estiver em ‘1’, baseado no bit que indica o jogador, um dos vetores sofrerá um Left Logical Shift, e sua posição 0 será completada com um ‘1’.

Outro fato que deve ser considerado é que se um jogador atirar em uma posição em que havia atirado previamente, onde havia um navio, o jogo pontuará para o usuário novamente. Isso

acontece porque nossa memória suporta apenas leitura, e não escrita. Para deixar o jogo mais real, necessitaríamos de uma memória RAM, que é mais complexa.



Simulação do contador de pontos, mostrando que um ponto só será contabilizado se o enable (variável “compara”) e o match (variável “cont_match”) estiverem ativos ao mesmo tempo.

7. Resultados e conclusões

Analisando o projeto como um todo, não houve nenhuma unidade que apresentasse problema. Fizemos todas as simulações solicitadas, bem como os testes na placa. Em momento algum tivemos problema nos testes finais e os resultados foram todos dentro do esperado. A maioria das unidades lógicas construídas foram associadas com a FSM, visto que ela é a base para todo o projeto. O comparador foi usado junto com a memória ROM para que ele possa validar, ou não, a jogada do usuário. O decodificador recebe informações do usuário e da FSM e as repassa para o display de sete segmentos. O clock de 50 MHz é usado para sincronizar todo o circuito e é também usado para gerar o clock de 1 Hz. Já o clock de 1 Hz é usado no cronômetro para contar o tempo restante do usuário. Quanto ao contador de pontos, ele é sempre atualizado depois que o comparador é utilizado para verificar uma jogada.

Anexo A – Observações

Durante o desenvolvimento do projeto, nós encontramos alguns problemas, mas todos foram resolvidos. A maior dificuldade, sem dúvida, foi ter feito a FSM visto que ela era a parte principal do projeto. As demais unidades lógicas não foram tão trabalhosas, visto que já haviam sido trabalhadas em aulas passadas. A ideia do projeto em si é excelente, pois consegue abordar todos os assuntos trabalhados durante o semestre.

Programas usados no desenvolvimento do trabalho:

-Quartus II

-Model Sim - Altera

-Microsoft Paint

-Google Drive/Docs

Material de consulta:

-PDFs das aulas de laboratório

Equipamento utilizado:

-Kit ALTERA DE2 - Cyclone II EP2C35F672C6.