



**INSTITUTO
FEDERAL**

Santa Catarina

Câmpus
São José

Relatório

Cenário 2 - Travessia Controlada por Botoeira com Sinalização Noturna, Sinalização Piscante,
Avisos Sonoros

Rhenzo Hideki Silva Kajikawa
Matheus Pires Salazar

17 de Dezembro de 2023

Sumário

| | |
|---|-----------|
| 1. Introdução | 3 |
| 1.1. Objetivo | 3 |
| 1.2. Motivação | 3 |
| 2. Descrição do Projeto | 4 |
| 2.1. Cenário 2 - Descrição | 4 |
| 2.2. Procedimento | 5 |
| 2.3. Componentes utilizados | 6 |
| 2.4. Sistema Completo | 7 |
| 2.4.1. Elementos lógicos do sistema | 7 |
| 2.4.2. RTL Viewer | 10 |
| 3. Implementação na placa | 13 |
| 3.1. Simulações | 13 |
| 3.2. Pinagem | 15 |
| 4. Dificuldades | 15 |
| 5. Conclusão | 15 |

1. Introdução

1.1. Objetivo

Este projeto, feito na matéria de Dispositivos lógicos programáveis , tem como objetivo simular uma situação de transito , onde, como descrito, é feita a travessia de pedestres em diferentes cenários tanto de manhã quanto a noite . Além de tentar otimizar o transito , evitando fechamentos de semáforos da via principal de forma desnecessária.

1.2. Motivação

Em aula foram ensinados novos conceitos de VHDL, e agora como foi aplicado desde o mais básico até conceitos mais complexos. Dessa forma esse projeto visa certificar que foram aprendidos todos esses conhecimentos.

2. Descrição do Projeto

2.1. Cenário 2 - Descrição

Visa garantir uma travessia de pedestres diurna e noturna segura e consciente. Ao acionar a botoeira, será ativada uma iluminação branca sobre a faixa de passagem zebraada e nas áreas de espera dos pedestres, assegurando melhor visibilidade e segurança para o pedestre a noite. Simultaneamente, o semáforo emitirá sinais visuais e sonoros, indicando ao pedestre que o botão foi acionado com sucesso e alertando motoristas sobre a intenção de travessia. Durante a fase de liberação para veículos, o semáforo do pedestre permanecerá vermelho, economizando energia até que o botão seja acionado. Após a solicitação, os grupos focais do pedestre exibirão luz verde em ambos os lados da via, enquanto o semáforo dos carros exibirá sinal vermelho, garantindo a máxima segurança para os pedestres e reforçando a prioridade de travessia.

Para orientar pedestres de maneira eficaz, o semáforo do pedestre apresentará um contador regressivo, indicando o tempo restante para a travessia. O tempo total de travessia será ajustável, permitindo personalização conforme as necessidades locais. Nos últimos 30% do tempo, o sinal verde do semáforo do pedestre piscará, visualmente alertando que o tempo para a travessia está se encerrando. É importante ressaltar que a iluminação estará ativa apenas durante o tempo em que a botoeira foi acionada até 5 segundos após o término do tempo de travessia. Este ajuste visa otimizar o consumo de energia e garantir que a iluminação cumpra sua função apenas quando necessária.

2.2. Procedimento

Fora de aula foram decididas como seria a divisão do projeto. Foi decidido que era necessário ilustrar como seria feito a máquina de estados antes mesmo de começar o código em vhdl.

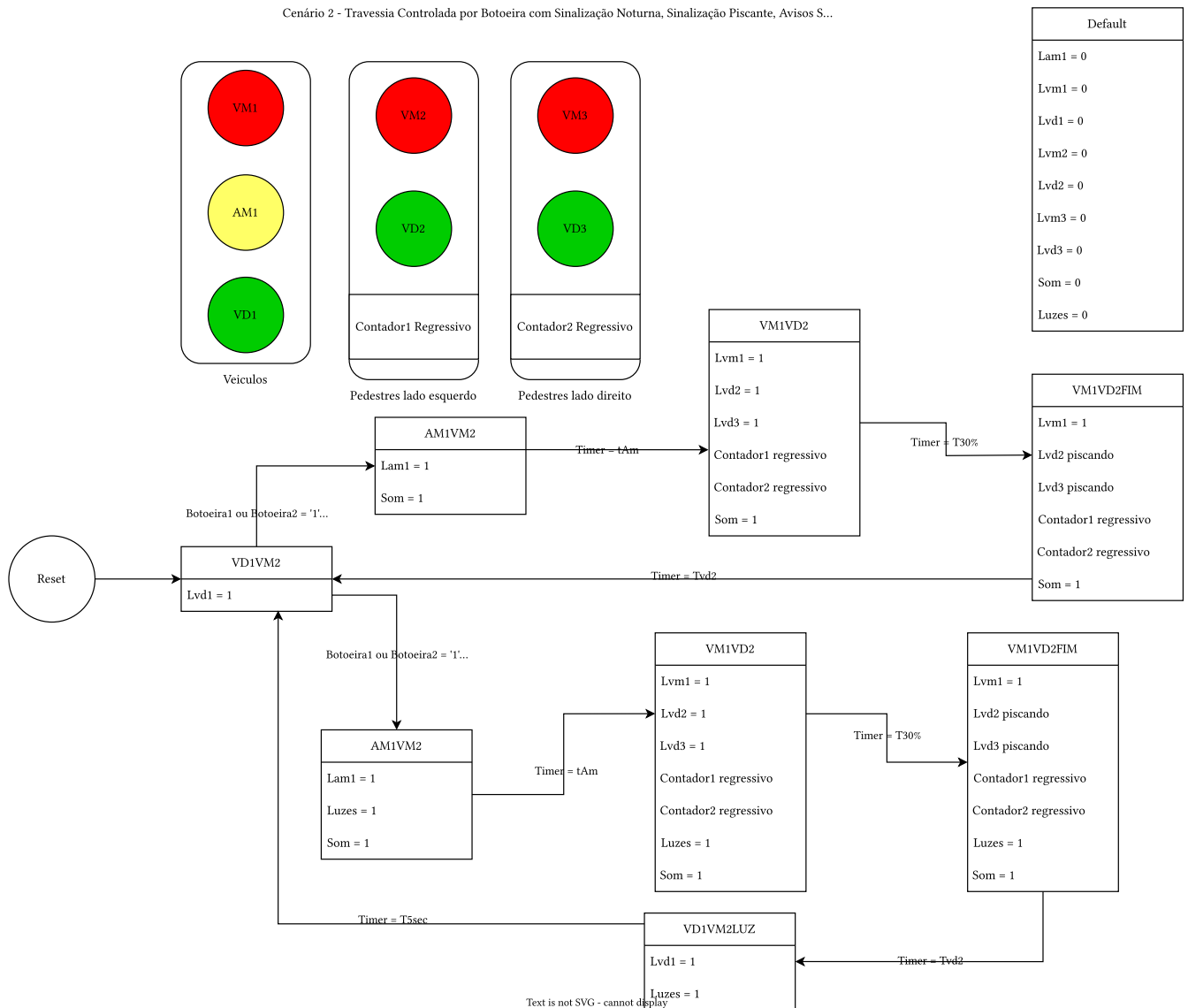


Figura 1: Elaboração das máquinas de estado
Fonte: Elaborada pelo autor

Nesse levantamento baseado nos semáforos visto em aula foi possível visualizar quais seriam as diferenças e semelhanças, além de ter ideia das entradas e saídas da máquina de estado do cenário 2.

Após pensar em como seria levantado a máquina de estado, fizemos uma varredura em quais componentes iriam ser necessários para a simular o cenário escolhido, e também atender as requisições feitas pelo professor.

Os componentes que foram escolhidos para compor o resto do projeto foram , um conversor bin2bcd , 2 conversores bcd2ssd e um divisor de clock.

A visualização do projeto ficou dessa forma:

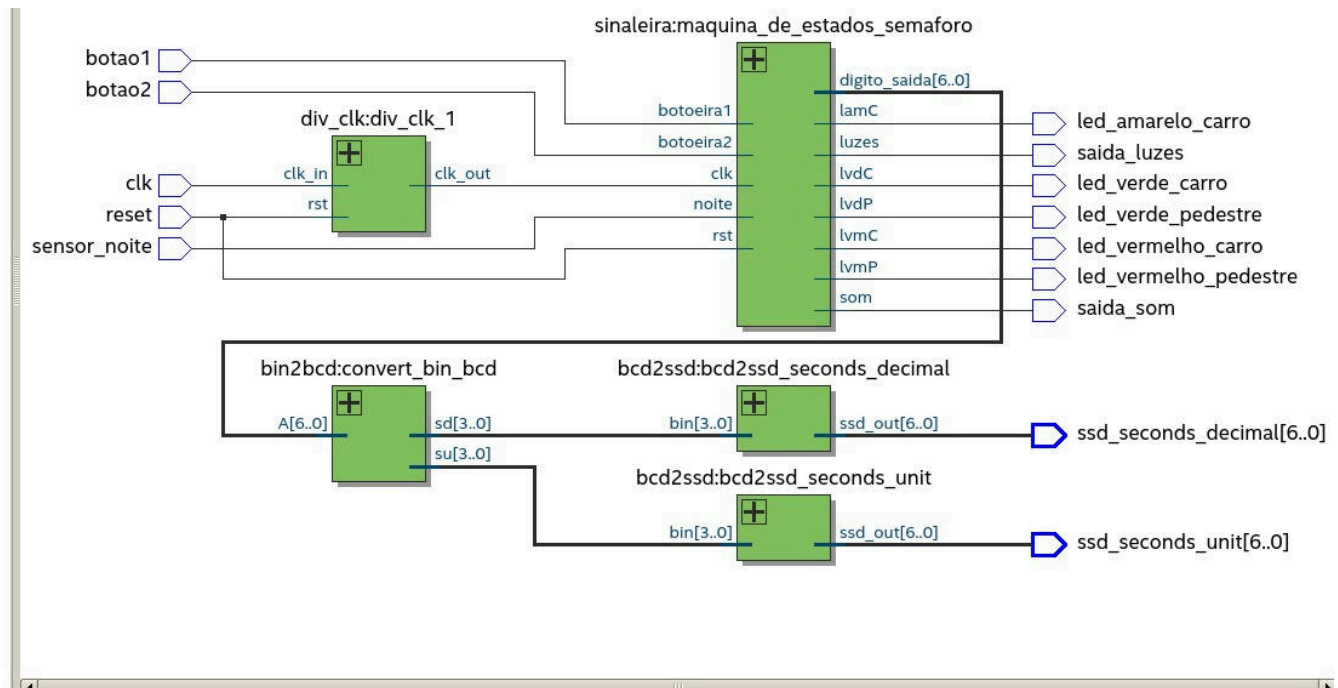


Figura 2: Elaboração das máquinas de estado
Fonte: Elaborada pelo autor

2.3. Componentes utilizados

O componentes utilizados para o projeto podem ser visualizados na Figura 2.

No total foram utilizados 2 displays de sete segmentos , 2 conversor de BCD para ssd , 1 conversor de Binário para BCD , 1 maquina de estados , 1 divisor de clock e 1 clock de 50 MHz , 2 botões para pedestres , 2 LEDs para simbolizar som e luz , 3 LEDs para semáforo de carros e mais 2 LEDs para semáforo de pedestres.

Como dito anteriormente , nesse projeto foi abordado maquinas de estados , a partir disso também foi trabalhado com o uso de 4 componentes menores.

Um divisor de clock com 2 entradas , reset e clock in , e uma saída clock out . Esse divisor tem como objetivo diminuir os pulsos de enable dos segundos , podendo ajustar toda contagem do sistema para diferentes clocks . Apenas trocando o valor genérico do componente “div”. Esse divisor de clock foi utilizado anteriormente no projeto do relógio que foi feito em sala , dessa maneira para otimizar o tempo foi decidido reutiliza-lo.

Para a maquina de estado foram criada com 5 entradas , a divisão delas sendo , 1 entrada padrão de reset, 1 entrada para o clock onde é conectada com o divisor de clock. 2 entradas que são das botoeiras para os pedestres que quando apertarem faz o estado mudar para abrir o semáforo dos pedestres e por ultimo uma entrada do sensor de noite.

Existem 8 saídas para a maquina de estado , 3 LEDs que servem para o semáforo dos carros , 2 LEDs para o semáforo dos pedestres junto a 1 saída de contagem e uma saída de som.

A saída de contagem tem como objetivo passar por um conversor binário para bcd , desse saem 2 pontos , 1 deles é para o decimal e outro a unidade que passa para um conversor bcd2ssd , esses são mostrados em 2 display de 7 segmentos que é habilitado quando o pedestre deve passar.

2.4. Sistema Completo

Após ter todos os componentes do projeto foi feita a junção deles.

2.4.1. Elementos lógicos do sistema

O numero de elementos lógicos por componente:

| | |
|------------------------------------|---|
| Flow Status | Successful - Fri Dec 15 14:31:58 2023 |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition |
| Revision Name | Cenario2 |
| Top-level Entity Name | div_clk |
| Family | Cyclone IV E |
| Device | EP4CE115F29C7 |
| Timing Models | Final |
| Total logic elements | 11 |
| Total registers | 7 |
| Total pins | 3 |
| Total virtual pins | 0 |
| Total memory bits | 0 |
| Embedded Multiplier 9-bit elements | 0 |
| Total PLLs | 0 |

Figura 3: RTL viewer do divisor de clock

Fonte: Elaborada pelo autor

Pode ser observado o uso de 11 elementos lógicos no divisor de clock

| | |
|------------------------------------|---|
| Flow Status | Successful - Fri Dec 15 14:39:38 2023 |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition |
| Revision Name | Cenario2 |
| Top-level Entity Name | bin2bcd |
| Family | Cyclone IV E |
| Device | EP4CE115F29C7 |
| Timing Models | Final |
| Total logic elements | 98 |
| Total registers | 0 |
| Total pins | 15 |
| Total virtual pins | 0 |
| Total memory bits | 0 |
| Embedded Multiplier 9-bit elements | 0 |
| Total PLLs | 0 |

Figura 4: RTL viewer do bin2bcd

Fonte: Elaborada pelo autor

No conversor bin2bcd foram utilizadas 98 unidades lógicas , esse valor alto pro conversor deve-se ao conversor binário para bcd que tem um custo alto para ser implementado de forma isolada.

| | |
|------------------------------------|---|
| Flow Status | Successful - Fri Dec 15 14:33:26 2023 |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition |
| Revision Name | Cenario2 |
| Top-level Entity Name | bcd2ssd |
| Family | Cyclone IV E |
| Device | EP4CE115F29C7 |
| Timing Models | Final |
| Total logic elements | 7 |
| Total registers | 0 |
| Total pins | 11 |
| Total virtual pins | 0 |
| Total memory bits | 0 |
| Embedded Multiplier 9-bit elements | 0 |
| Total PLLs | 0 |

Figura 5: RTL viewer do bcd2ssd

Fonte: Elaborada pelo autor

Para o conversor bcd para ssd foram utilizados 11 elementos , porém cada elemento trabalha de forma separada . Assim a estimativa para o total de elementos lógicos presente no projeto seria de 22 elementos lógicos para todo o grupo de conversores.

| Flow Summary | |
|------------------------------------|---|
| <<Filter>> | |
| Flow Status | Successful - Sat Dec 16 23:47:06 2023 |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition |
| Revision Name | Cenario2 |
| Top-level Entity Name | sinaleira |
| Family | Cyclone IV E |
| Device | EP4CE115F29C7 |
| Timing Models | Final |
| Total logic elements | 66 / 114,480 (< 1 %) |
| Total registers | 27 |
| Total pins | 19 / 529 (4 %) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 3,981,312 (0 %) |
| Embedded Multiplier 9-bit elements | 0 / 532 (0 %) |
| Total PLLs | 0 / 4 (0 %) |

Figura 6: RTL viewer da maquina de estado

Fonte: Elaborada pelo autor

A maquina de estado , descrito com o nome ‘sinaleira’ , sendo o componente mais complexo , ainda sim teve ao todo 66 elementos lógicos utilizados , um numero ainda sim menor que o próprio conversor bin2bcd utilizado.

| Flow Summary | |
|------------------------------------|---|
| <<Filter>> | |
| Flow Status | Successful - Sat Dec 16 23:45:25 2023 |
| Quartus Prime Version | 20.1.1 Build 720 11/11/2020 SJ Standard Edition |
| Revision Name | Cenario2 |
| Top-level Entity Name | cenario2 |
| Family | Cyclone IV E |
| Device | EP4CE115F29C7 |
| Timing Models | Final |
| Total logic elements | 159 / 114,480 (< 1 %) |
| Total registers | 27 |
| Total pins | 26 / 529 (5 %) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 3,981,312 (0 %) |
| Embedded Multiplier 9-bit elements | 0 / 532 (0 %) |
| Total PLLs | 0 / 4 (0 %) |

Figura 7: RTL viewer do cenário

Fonte: Elaborada pelo autor

Por fim temos o número de elementos lógicos do cenário completo. Ao todo foram utilizados menos elementos lógicos do que a soma deles de forma separada, isso porque o Quartus tem diferentes maneiras de otimizar os códigos que são dados a ele.

2.4.2. RTL Viewer

Aqui estão os RTLS viewers para cada componente:

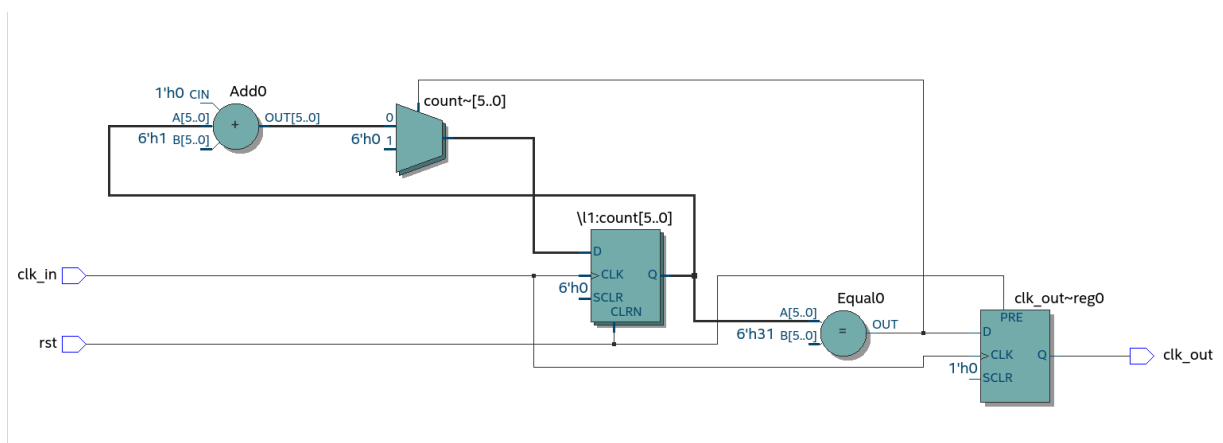


Figura 8: RTL viewer do divisor de clock

Fonte: Elaborada pelo autor

Um divisor de clock simples baseado em um contador.

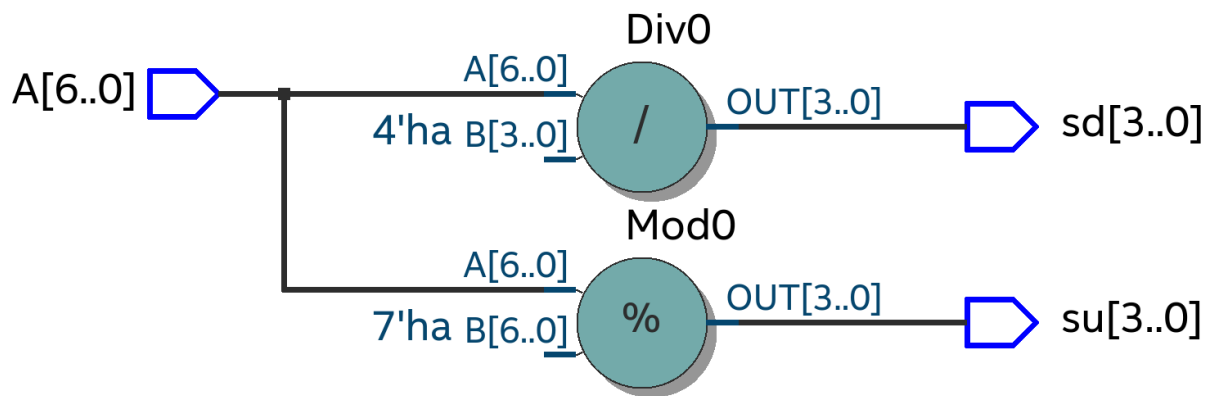


Figura 9: RTL viewer do bin2bcd

Fonte: Elaborada pelo autor

O uso de divisão e da operação Mod , duas operações custosas para o hardware.

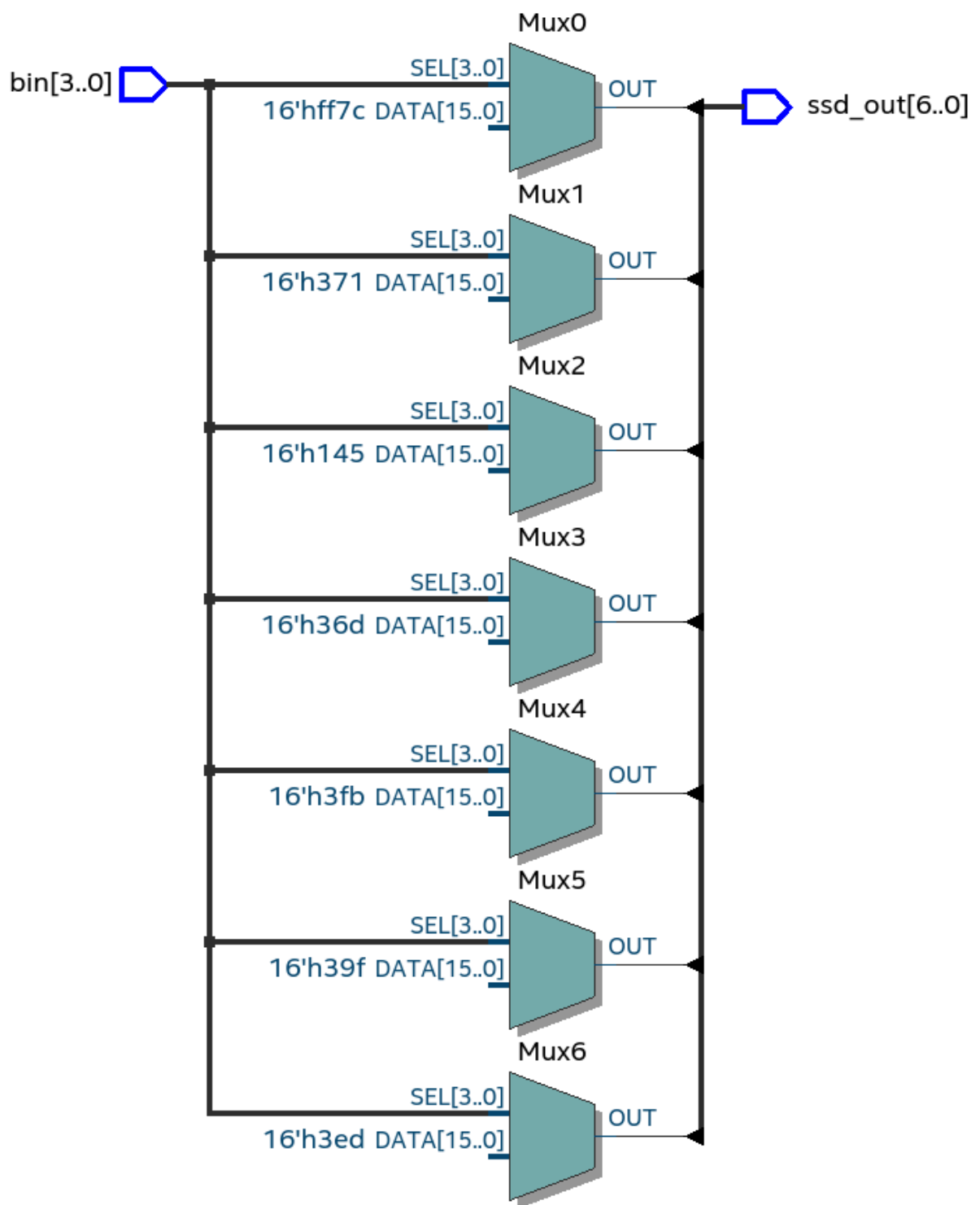


Figura 10: RTL viewer do bcd2ssd

Fonte: Elaborada pelo autor

A conversão de bcd para ssd , que são vários Mux

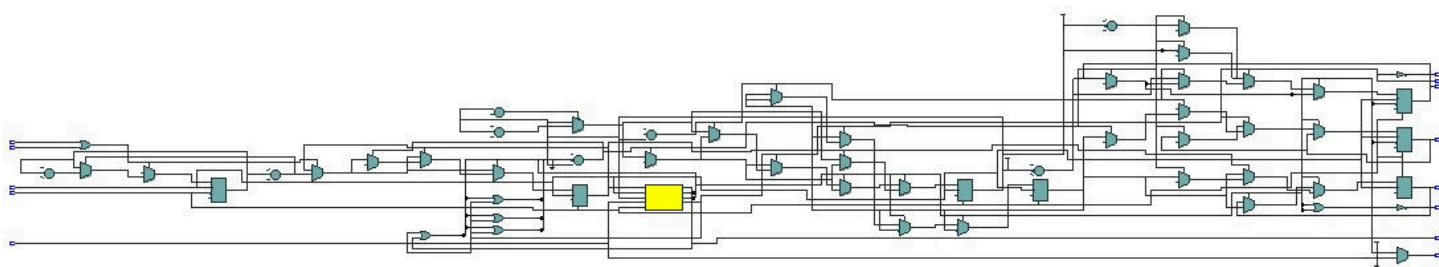


Figura 11: RTL viewer da sinaleira

Fonte: Elaborada pelo autor

A maquina de estado , que ficou com um nível de complexidade em questão de número de elementos aparecendo, e por fim não foi possível tirar print de como ficou os estados dentro da componente amarela isso pois o quartus quando aberta a componente da máquina de estados não mostrava os diferentes estados e suas conexões.

3. Implementação na placa

A implementação na placa foi feita em aula com o kit DE2-115 da TERASIC.

3.1. Simulações

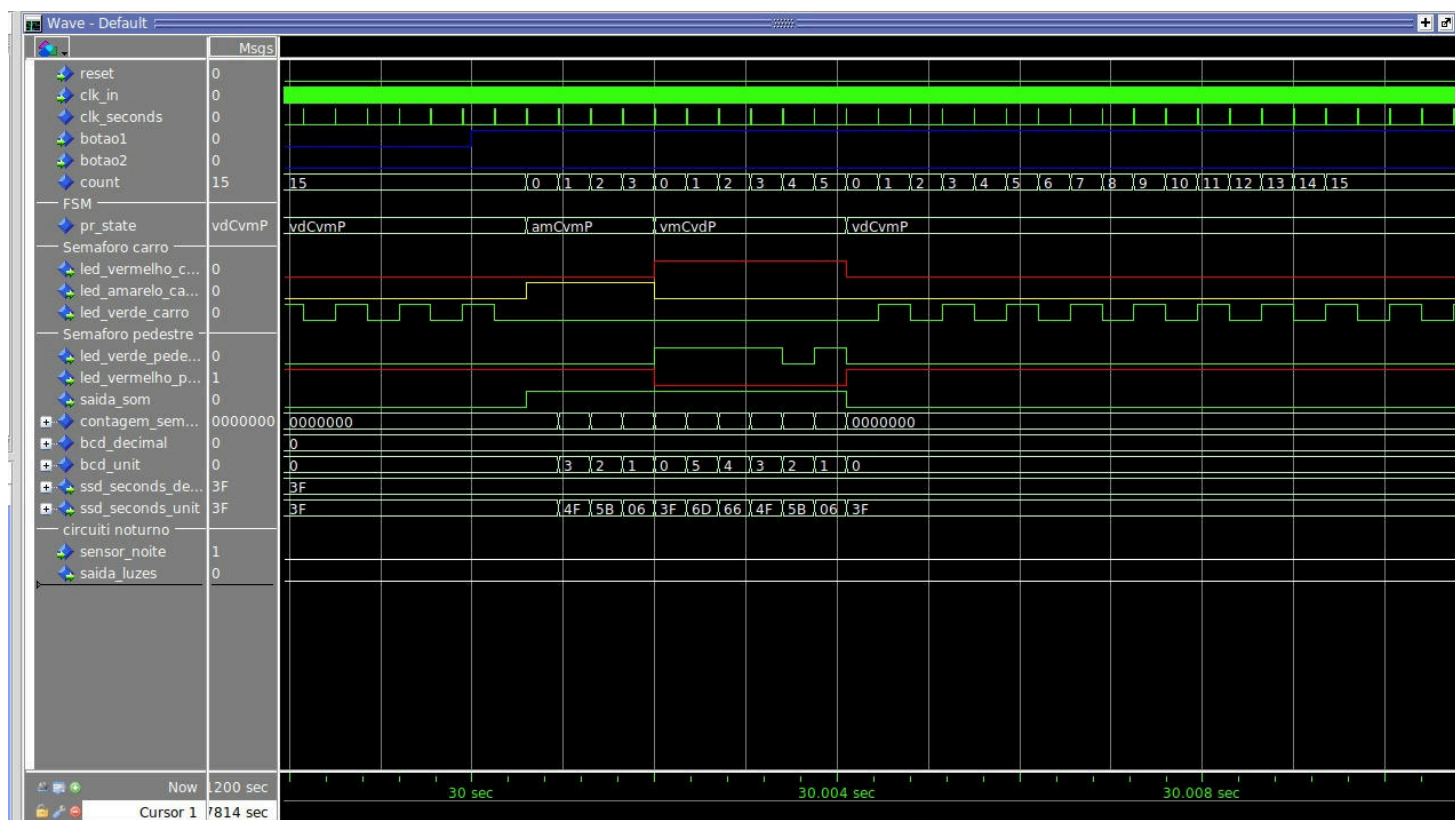


Figura 12: Simulação Dia

Fonte: Elaborada pelo autor

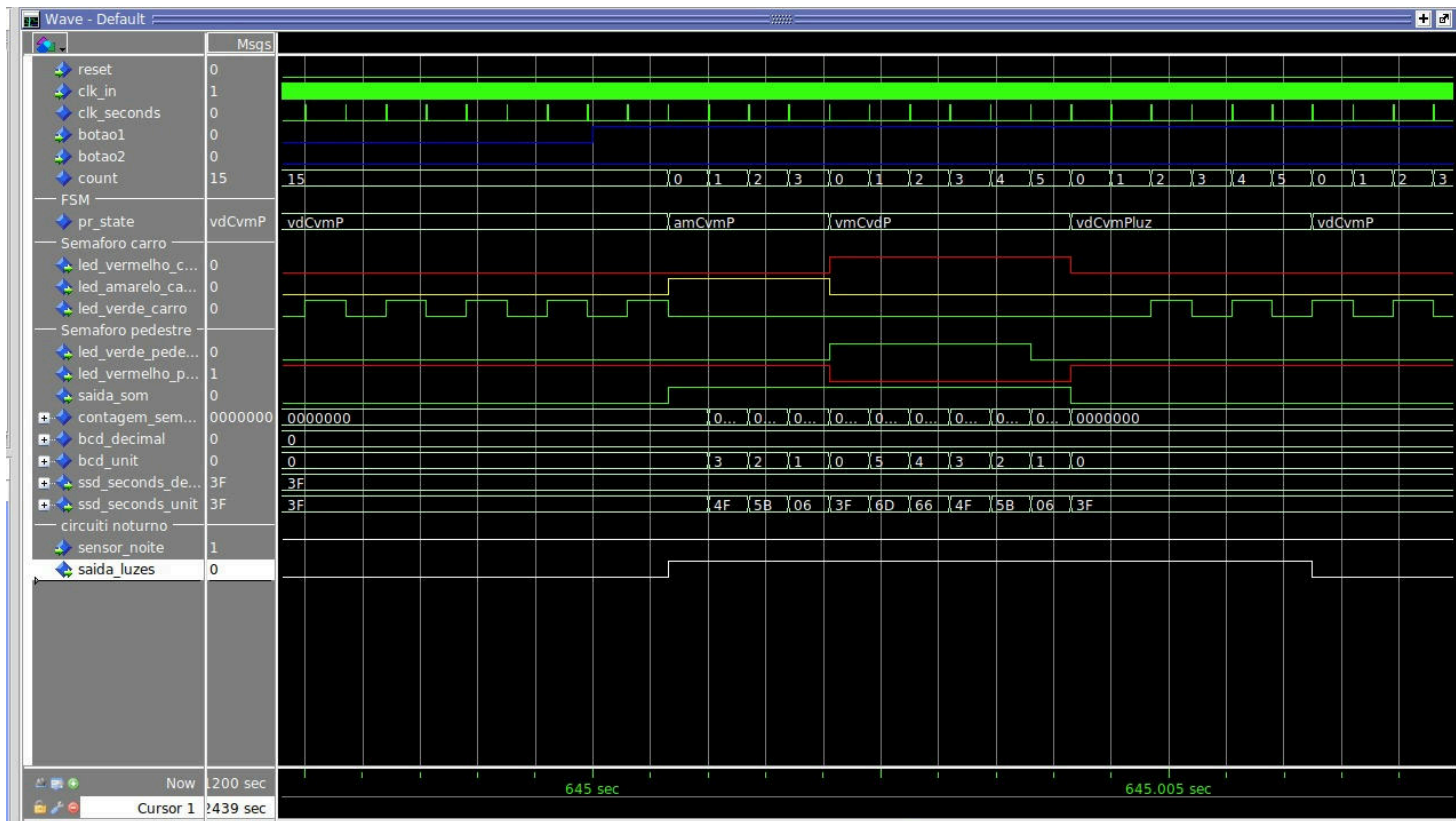


Figura 13: Simulação Noite

Fonte: Elaborada pelo autor

Estas são as 2 simulações feitas , visando testar as diferentes possibilidades e erros que podem acontecer em um caso mais proximo da realidade.

Temos 2 imagens , uma representando uma simulação voltada para o dia , onde não se deve ter a saída de luzes , já a outra imagem é feita a simulação representando a noite , dessa forma é possível ver alterações em algumas saídas , como o ativamento de uma luz noturna para o pedestre.

Nessas imagens também é possível observar um caso tratado em aula , caso o botão da botoeira emperre , o fluxo dos carros deve ser mantido até a botoeira ser concertado , por isso foi deixada a botoeira de forma continua para averiguar se foi possível alcançar o que foi desejado.

3.2. Pinagem

| | | | | | | | | | | |
|-----|---------------------|--------|----------|---|-------|-----------------|---------------|-------------|--|--|
| in | botao1 | Input | PIN_N21 | 6 | B6_N2 | 2.5 V (default) | 8mA (default) | | | |
| in | botao2 | Input | PIN_R24 | 5 | B5_N0 | 2.5 V (default) | 8mA (default) | | | |
| in | clk | Input | PIN_Y2 | 2 | B2_N0 | 2.5 V (default) | 8mA (default) | | | |
| out | led_amarelo_carro | Output | PIN_G16 | 7 | B7_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | led_verde_carro | Output | PIN_G15 | 7 | B7_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | led_verde_pedestre | Output | PIN_J16 | 7 | B7_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | led_vermelho_carro | Output | PIN_H15 | 7 | B7_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | led_verm...pedestre | Output | PIN_H17 | 7 | B7_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| in | reset | Input | PIN_M23 | 6 | B6_N2 | 2.5 V (default) | 8mA (default) | | | |
| out | saida_luzes | Output | PIN_G17 | 7 | B7_N1 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | saida_som | Output | PIN_J15 | 7 | B7_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| in | sensor_noite | Input | PIN_AB28 | 5 | B5_N1 | 2.5 V (default) | 8mA (default) | | | |
| out | ssd_seco...cimal[6] | Output | PIN_AA14 | 3 | B3_N0 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco...cimal[5] | Output | PIN_AG18 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco...cimal[4] | Output | PIN_AF17 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco...cimal[3] | Output | PIN_AH17 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco...cimal[2] | Output | PIN_AG17 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco...cimal[1] | Output | PIN_AE17 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco...cimal[0] | Output | PIN_AD17 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco..._unit[6] | Output | PIN_AC17 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco..._unit[5] | Output | PIN_AA15 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco..._unit[4] | Output | PIN_AB15 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco..._unit[3] | Output | PIN_AB17 | 4 | B4_N1 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco..._unit[2] | Output | PIN_AA16 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco..._unit[1] | Output | PIN_AB16 | 4 | B4_N2 | 2.5 V (default) | 8mA (default) | 2 (default) | | |
| out | ssd_seco..._unit[0] | Output | PIN_AA17 | 4 | B4_N1 | 2.5 V (default) | 8mA (default) | 2 (default) | | |

Figura 14: pinagem

Fonte: Elaborada pelo autor

4. Dificuldades

Nesse projeto houveram alguns contra-tempos . Alguns desses como a implementação de uma maquina de estado , pensar na botoeira travada e a criação diferentes situações na simulação. Estes foram alguns dos maiores desafios encontrados durante este projeto.

As maquinas de estados se mostraram desafiadoras , pois além de um conteúdo novo , foi-se colocado de forma aos alunos desenvolverem suas próprias máquinas sem o auxílio do professor. Assim necessitando de um maior planejamento e visualização de como deveriam ser feitas os diferentes estados.

A implementação na botoeira foi feita de maneira simples , porém não se tinha pensado em diferentes situações cotidianas como o travamento de uma botoeira, para implementá-lo foi decidido utilizar um contador que contaria o intervalo de tempo que a botoeira ficaria ativa , após um determinado tempo , as botoeiras seriam desabilitadas até que houve-se alguma intervenção que tirasse os inputs das botoeiras.

A criação de diferentes cenários para testagem do cenário se mostrou extremamente complicado uma vez que envolvia diferentes métodos para diferentes falhas. Além disso ao primeiro contato sempre é pensado em cenários mais ideais , deixando complexo a visão de como implementar cenário que podem comprometer o fluxo desejado.

5. Conclusão

Com esse projeto foi possível colocar em prática todo o conhecimento aprendido nesse semestre. Desde da composição de entidades menores , até maquinas de estados que exigem um maior planejamento .

Com este projeto também é possível visualizar o potencial de utilizar o vhd1 para compor um cenário em uma linguagem lógica que pode representar entradas , saídas e diferentes estados.

Por tanto , com a finalização deste projeto foi possível visualizar e aplicar as maquinas de estado em um código , além de uni-lo com o uso de componentes . Tudo isso em um cenário plausível que faz parte do cotidiano.