Projeto Final - Semaforo com FSM

Dispositivos Lógicos Progamáveis I

Arthur Cadore Matuella Barcella
Gabriel Luiz Espindola Pedro

18 de Dezembro de 2023

Sumário

1 Orientações **3**
1.1 Objetivos 3
1.2 Descrição do projeto - Sinalização Semafórica para Cruzamento com Passagem de Pedestres Controlada por Botoeira e Sincronizada com Vias Veiculares 3

2 Desenvolvimento **4**
2.1 Desenvolvimento da FSM: 4
2.2 Desenvolvimento dos componentes auxiliares: 5
2.3 Instanciando componentes: 8
2.4 TestBanch do Projeto: 8
2.5 Pinagem e Gravação na placa: 9
2.6 Não Atendimento das especificações 10

3 Referências bibliográficas: **11**

1 Orientações

O projeto será desenvolvido por equipes de até 2 estudantes, e cada equipe deverá escolher uma dos cenários propostos, ou até mesmo um cenário diferenciado desses. Cada projeto deverá envolver obrigatoriamente:

1.1 Objetivos

- Uso de um conjunto de mostradores de 7 segmentos de dois ou mais dígitos.
- Uso de leds para indicar mostrar os semáforos. O uso das GPIOs com circuitos de LEDs de cores verde, vermelho e amarelo é encorajado.
- A equipe poderá utilizar ambos kits disponibilizados no laboratório
- Usar as chaves para simular os sensores e botoeiras (sinais de entrada).
- Uma ou mais máquinas de estados finitos
- Deverá ser usado um projeto hierárquico, onde a entidade top level deverá apenas ter a instanciação de componentes, e eventuais adaptações ao hardware do kit.
- Todos os componentes e o sistema completo devem ser testados através de simulação no Modelsim.
- Os testes reais do sistema completo no kit devem ser filmados para demonstrar o funcionamento.
- Fazer uma análise das vantagens e deficiencias da solução proposta.

1.2 Descrição do projeto - Sinalização Semafórica para Cruzamento com Passagem de Pedestres Controlada por Botoeira e Sincronizada com Vias Veiculares

Este projeto visa apresentar uma solução para um cruzamento de vias, com a passagem de pedestres controlada por botoeira, integrada de maneira sincronizada com o fluxo de veículos. Abaixo, detalhamos as características tanto para pedestres quanto para carros.

- Fase Inicial: Os semáforos veiculares iniciam em sinal amarelo piscante em ambas as vias. O semáforo para pedestres permanece apagado até a ativação da botoeira, visando a economia de energia. apenas um led vermelho no centro da botoeira deve estar piscando para induzir o pedestre a acionar a boteira. Uma placa sobre a botoeira informa que é necessário acionar o botão para solicitar a passagem.
- Funcionamento no modo veicular sem solicitação de pedestre: O cruzamento deve alternar entre vermelho, amarelo e verde conforme programação para otimizar o fluxo de carros. O ciclo de passagem de pedestre deve ser iniciado se alguma das 4 boteiras existentes nos cantos do cruzamento for acionada.
- Solicitação de pedestre: Ao ser acionada uma das botoeiras dos Pedestres, se for a noite, será ativada a iluminação branca sobre a faixa de passagem zebrada e nas áreas de espera dos pedestres, assegurando melhor visibilidade e segurança para o pedestre a noite. Simultaneamente, o semáforo emitirá sinais sonoros, indicando ao pedestre que o botão foi acionado com sucesso e alertando motoristas sobre a intenção de travessia.

    1. Durante a fase de espera para dos pedestres, o semáforo do pedestre, que estava apagado economizando energia, acenderá em vermelho e mostrara um contador regresivo indicando o tempo faltante para a liberação da travessia.

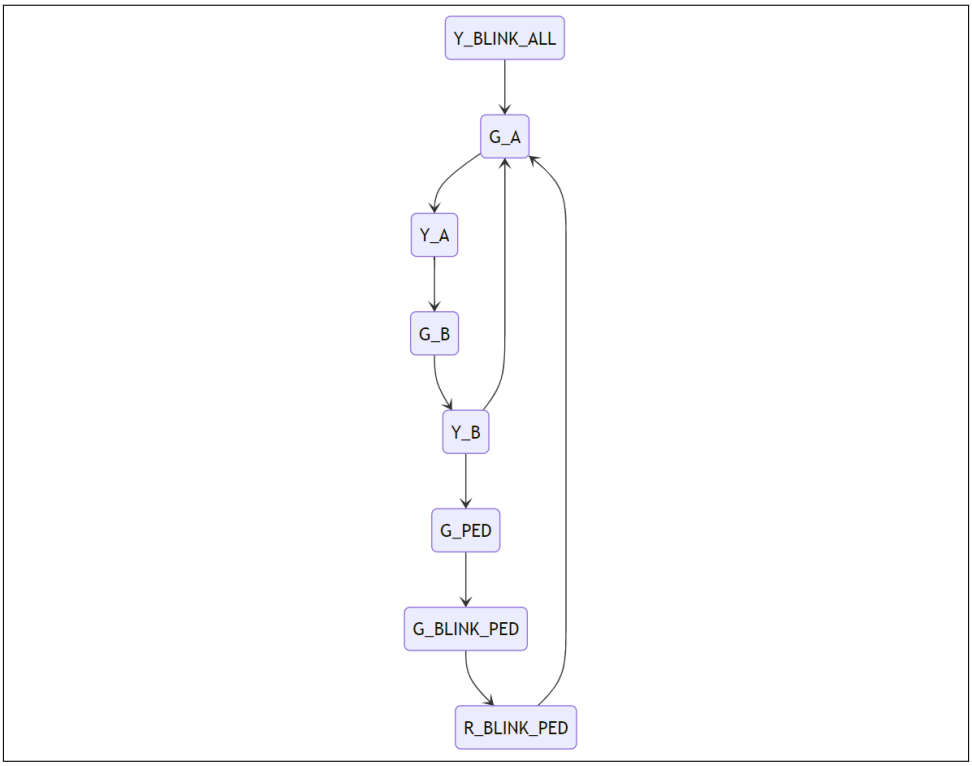
2. Na liberação, que é sincronizada com as vias, os grupos focais de pedestre exibirão luz verde em ambos os lados das faixas de pedestres, e indicará o tempo restante num contador regressivo.
3. Enquanto isso, os semáforos dos carros exibirão sinal vermelho em todas as direções, garantindo a máxima segurança para os pedestres.
4. Nos últimos 30% do tempo, sinal verde para pedestres piscará, alertando visualmente sobre término iminente da travessia.
5. Ao término do tempo de travessia configurado, semáforo para pedestres entra em vermelho piscante por 5 segundos, indicando retorno ao modo veicular.
6. Se botoeira for acionada novamente, inicia-se novo ciclo, sincronizando travessia de pedestres com o fluxo veicular.

2 Desenvolvimento

2.1 Desenvolvimento da FSM:

Inicialmente fizemos a estruturação da FSM resposável por realizar o controle dos componentes do semáforo. Estudamos os estados e as condições possiveis para então termos iniciarmos o código.

Figura 1: Finite State Machine principal



FONTE: Elaborado pelo autor

Os estados de cada máquina estão explicados abaixo:

- Y-BLINK-ALL: Estado inicial da máquina, onde os semaforos ficam piscando em amarelo. Este estado permanece em execução até que o enable seja habilitado, neste caso, o proximo estado é G-A.
- G-A: O estado G (Green) - A (Semaforo A) é habilita a cor verde no semafaro A e a cor vermelha no semaforo B, o proximo estado sempre é G-Y.

- Y-A: O estado Y (Yellow) - A (Semaforo A) é habilita a cor amarela no semafaro A e a cor vermelha no semaforo B, o proximo estado sempre é G-B.
- G-B: O estado G (Green) - B (Semaforo B) é habilita a cor verde no semafaro B e a cor vermelha no semaforo A, o proximo estado sempre é Y-B.
- Y-B: O estado Y (Yellow) - (Semaforo B) é habilita a cor amarela no semafaro B e a cor vermelha no semaforo A. Neste estado é necessário uma validação para que seja atribuido o proximo estado da FSM.
  - Caso pedestre tenha pressionado um dos 4 botões, a FSM irá receber um sinal por uma de suas entradas, este seinal será validado toda vez que a FSM atingir o estado Y-B.
  - Uma vez recebido sinal positivo, a FSM desvia seu fluxo de execução e passa para o estado G-PED.
  - Caso não receba nenhum sinal para abrir para os pedestres, retorna para o estado G-A, seguindo seu fluxo de execução normal.
- G-PED: O estado G (Green) - PED (Semaforo Pedestres) é habilita a cor vermelha em todos os semafaros de carros e a cor verde no semaforo para pedestres. Ao mesmo tempo, inicia um contador para verificar o tempo em que está no estado atual. O proximo estado sempre será G-BLINK-PED.
- G-BLINK-PED: O estado G (Green) - BLINK (Piscar) - PED (Semaforo Pedestres) é habilita a cor vermelha em todos os semafaros de carros e a cor verde (piscando) no semaforo para pedestres. Ao mesmo tempo, inicia um contador para verificar o tempo em que está no estado atual. O proximo estado sempre será R-BLINK-PED.
- R-BLINK-PED: O estado R (Red) - BLINK (Piscar) - PED (Semaforo Pedestres) é habilita a cor vermelha em todos os semafaros de carros e a cor vermelha (piscando) no semaforo para pedestres. Ao mesmo tempo, inicia um contador para verificar o tempo em que está no estado atual. Ao expirar o tempo do contador, o estado é passado para G-A novamente, fechando assim o loop.

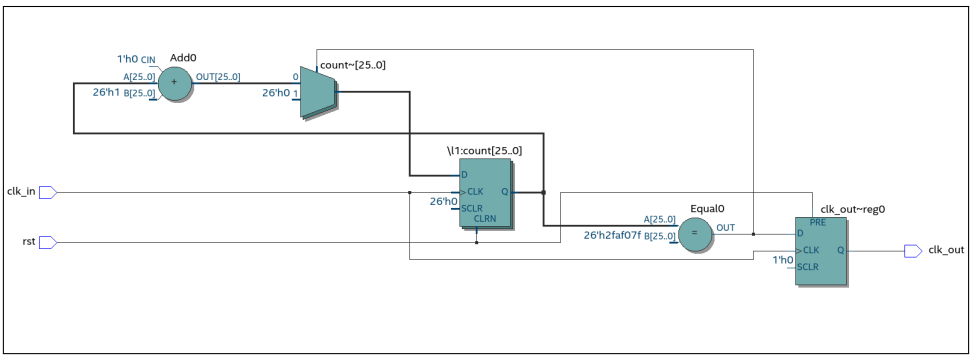
2.2 Desenvolvimento dos componentes auxiliares:

Assim que finalizamos o desenvolvimento da FSM, verificamos quais componentes auxiliares seriam necessários para dar andamento no projeto, para fazer o tratamento das entradas / saídas.

Os componentes auxiliares são os seguintes:

1. Div-Clk: O componente de DIV-CLK é utilizado para fazer a divisão do clock para todo o projeto. O componente é instanciado e recebe a entrada de clock diretamente do circuito gerador de clock. Em seguida, ele divide no tempo configurado (para este projeto, é utilizado 5000000, ou 50M, para obtenção de um clock de 1 segundo). Abaixo está o circuito contido dentro do componente de divisão do clock:

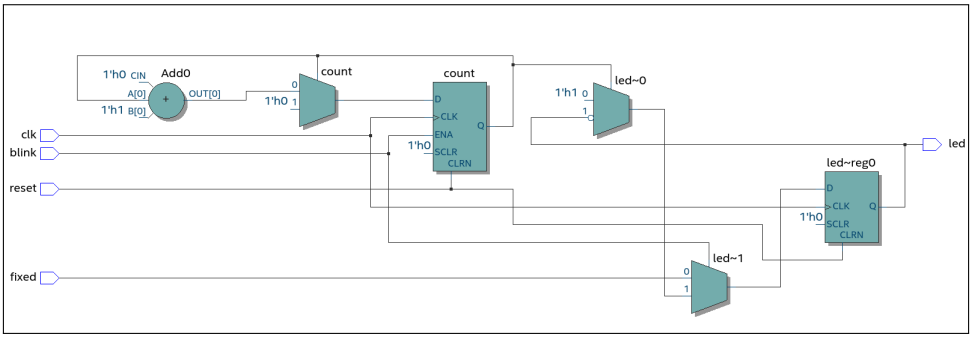
Figura 2: RTL - Divisor de Clock



FONTE: Elaborado pelo autor

2. blinking: O componente blinking é utilizado para realizar a piscagem de alguns leds no projeto. Seu objetivo é receber um sinal de piscagem ou de ascender (estáticamente) e propagar o estado solicitado para saída. Dessa forma, caso a FSM habilite o pino responsável por fazer um led piscar, este componente é responsável por criar um pulso de piscagem e propaga-lo para o led. O RTL do componente está descrito abaixo:

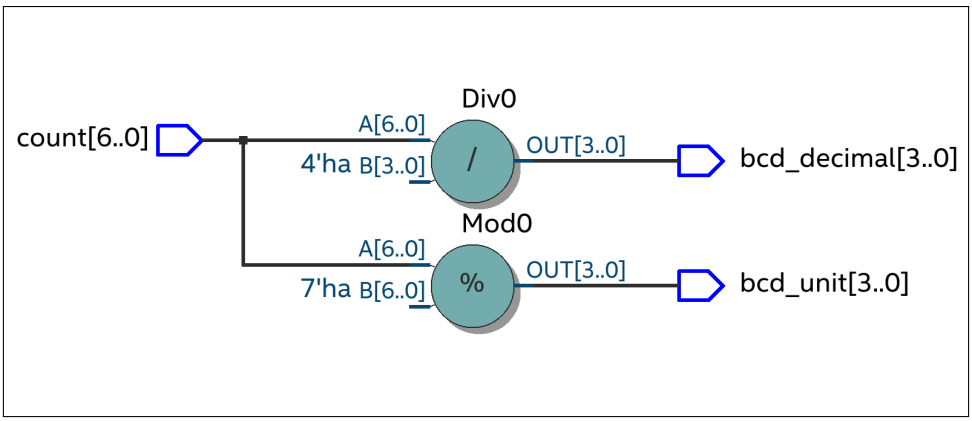
Figura 3: RTL - Blinking



FONTE: Elaborado pelo autor

3. bin2bcd: O componente bin2bcd é responsável por fazer a conversão entre binário (onde a contagem está sendo efetivamente realizada) para BCD (Binary-Coded Decimal), para obter cada casa decimal da contagem que esta sendo realizada com seu rescpetivo valor. Abaixo está o circuito correspondente ao componente de conversão de binário para BCD:

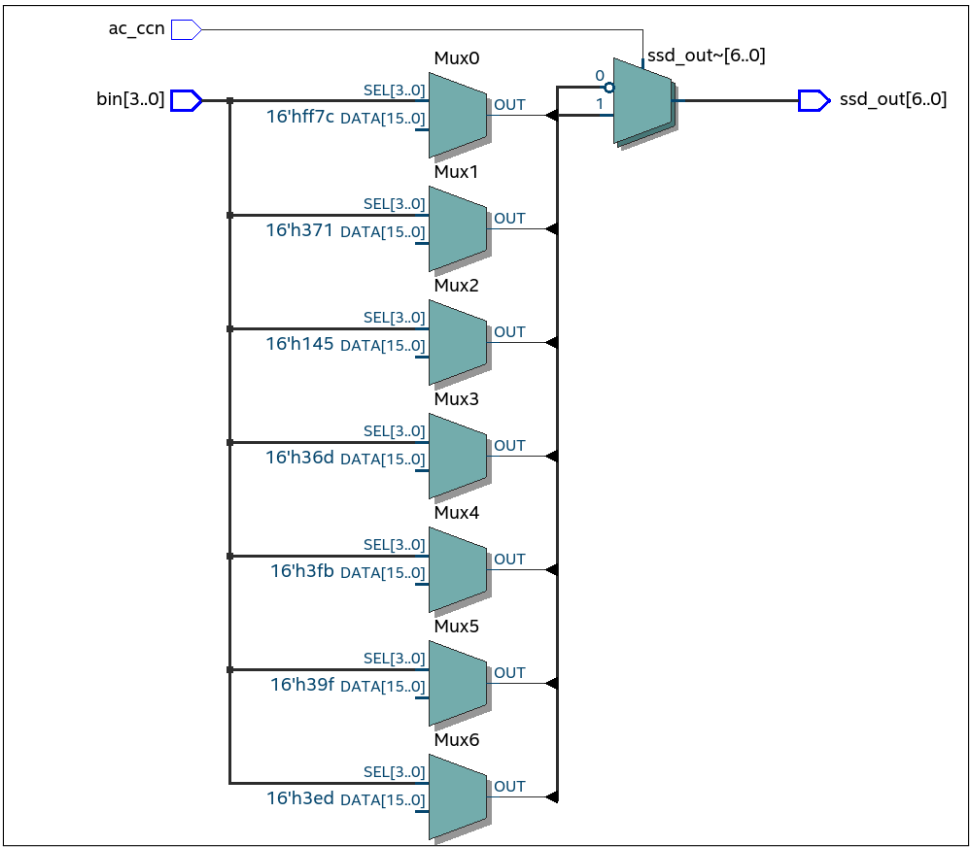
Figura 4: RTL - Bin to BCD



FONTE: Elaborado pelo autor

4. bcd2ssd: Uma vez com a contagem armazenada em BCD, outro componente é utilizado para conversão do código em BCD para SSD (Seven Segment Decoder), ou seja, uma representação em sete segmentos para posteriormente ser exibida através de um display. Para realizar essa conversão, utilizamos o circuito apresentado abaixo, o circuito faz apenas a verificação do valor (em BCD) recebido em sua entrada, para a respectiva saída representada em 7-Segmentos:

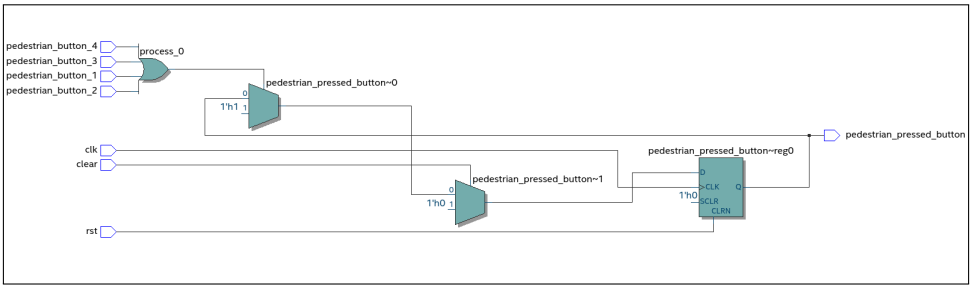
Figura 5: RTL - BCD to SSD



FONTE: Elaborado pelo autor

5. pedestrian-input: O componente pedestrian-input é responsável por receber as entradas das botoeiras (pedestres) e realizar seu tratamento para enviar apenas um sinal a FSM apontando que um cliente deseja atravessar a via. Uma vez que a FSM tenha entrado no estado para a passagem de pedestres, um pulso é enviado a entrada "clear"deste componente para limpar o flip-flop. Abaixo está o RLT deste componente:

Figura 6: RTL - Pedestrian Input



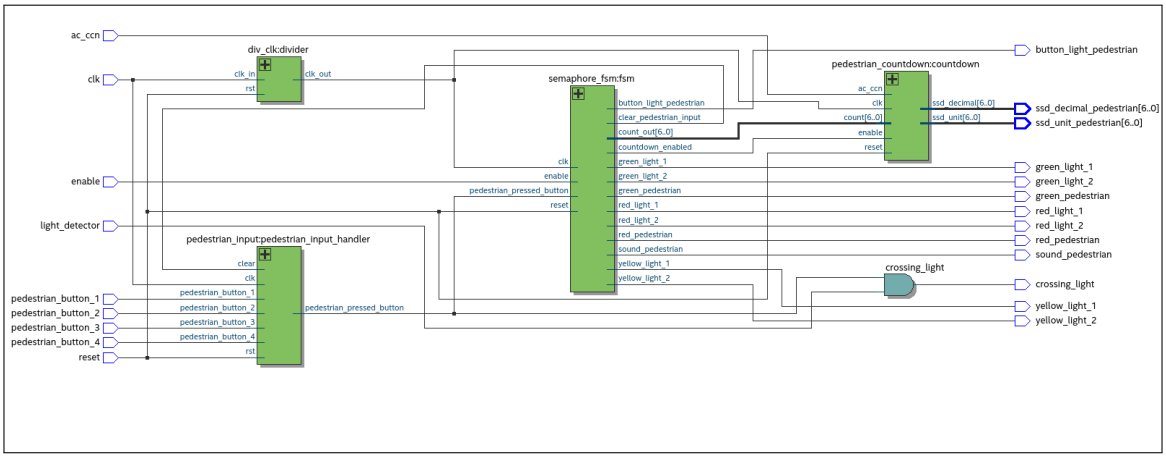
FONTE: Elaborado pelo autor

2.3 Instanciando componentes:

Com todos os componentes devidamente estruturados e testados, iniciamos o desenvolvimento do código VHDL principal (main.vhd), cujo objetivo é instanciar todos os demais componentes e fazer sua interconexão.

Abaixo está o RTL do projeto (main.vhd) mostranto todas os componentes utilizados e suas conexões:

Figura 7: RTL - Arquivo principal do projeto (main.vhd)



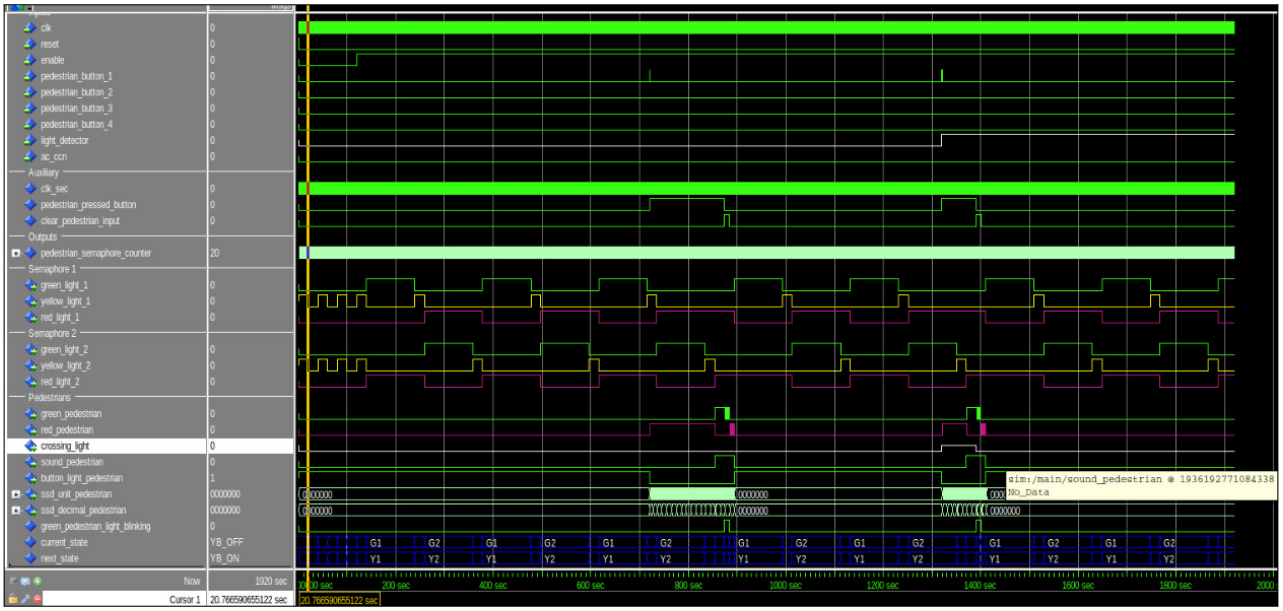
FONTE: Elaborado pelo autor

2.4 TestBanch do Projeto:

Uma vez com a montagem do projeto finalizada, realizamos testes do modelsim a fim de verificar se as entradas / saídas estavam se comportando conforme o esperado.

Utilizamos um tb.do e wave.do para automatizar o processo de inicialização do testbanch do projeto, abaixo está a simulação em operação:

Figura 8: Simulação do projeto - Modelsim



FONTE: Elaborado pelo autor

2.5 Pinagem e Gravação na placa:

Para realizar a gravação do projeto em uma FPGA, antes de tudo, é necesário realizar o mapeamento dos pinos para as entradas fisicas disponiveis. Como está sendo utilizando o kit de desenvolvimento, verificamos a documentação (ver referências) na wiki para entender quais portas deveriam ser mapeadas para quais pinos, conforme exibido abaixo:

Figura 9: Mapeamento dos pinos de entrada da FPGA

Pino	Name FPGA	Signal
Clock	PIN_Y2	CLOCK_50
Reset	PIN_M23	Push-button[0]
Enable	PIN_AB28	SW[0]
Pedestrian_button_1	PIN_AC24	Slide Switch[10]
Pedestrian_button_2	PIN_AB24	Slide Switch[11]
Pedestrian_button_3	PIN_AB23	Slide Switch[12]
Pedestrian_button_4	PIN_AA24	Slide Switch[13]
light_detector	PIN_AA23	Slide Switch[14]
ac_ccn	PIN_Y23	Slide Switch[17]

FONTE: Elaborado pelo autor

Figura 10: Mapeamento dos pinos de saida (padrão) da FPGA

Descrição	Pino	Name FPGA	Signal
Luz Verde - Semafaro 1	green_light_1	PIN_G19	LEDR[0]
Luz Verde - Semafaro 2	yellow_light_1	PIN_F19	LEDR[1]
Luz Amarela - Semafaro 1	red_light_1	PIN_E19	LEDR[2]
Luz Amarela - Semafaro 2	green_light_2	PIN_F18	LEDR[4]
Luz Vermelha - Semafaro 1	yellow_light_2	PIN_E18	LEDR[5]
Luz Vermelha - Semafaro 2	red_light_2	PIN_J19	LEDR[6]
Luz Verde - Pedestre	green_pedestrian	PIN_J17	LEDR[8]
Luz Vermelha - Pedestre	red_pedestrian	PIN_G17	LEDR[9]
Iluminação da travessia	crossing_light	PIN_H16	LEDR[11]
Sinal sonoro - Pedestre	sound_pedestrian	PIN_J16	LEDR[12]
Luz Botão - Pedestre	button_light_pedestrian	PIN_H17	LEDR[13]

FONTE: Elaborado pelo autor

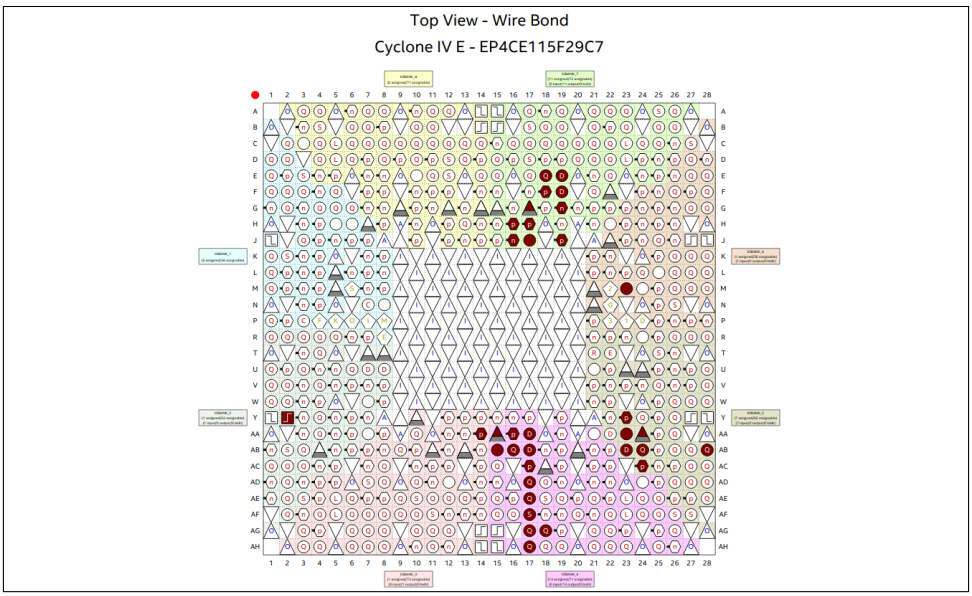
Figura 11: Mapeamento dos pinos de saida (para os displays) da FPGA

Descrição	Pino	Name FPGA	Signal
Segmento A - Unidade	ssd_unit_pedestrian_a	PIN_AA17	HEX6[0]
Segmento B - Unidade	ssd_unit_pedestrian_b	PIN_AB16	HEX6[1]
Segmento C - Unidade	ssd_unit_pedestrian_c	PIN_AA16	HEX6[2]
Segmento D - Unidade	ssd_unit_pedestrian_d	PIN_AB17	HEX6[3]
Segmento E - Unidade	ssd_unit_pedestrian_e	PIN_AB15	HEX6[4]
Segmento F - Unidade	ssd_unit_pedestrian_f	PIN_AA15	HEX6[5]
Segmento G - Decimal	ssd_unit_pedestrian_g	PIN_AC17	HEX6[6]
Segmento A - Decimal	ssd_decimal_pedestrian_a	PIN_AD17	HEX7[0]
Segmento B - Decimal	ssd_decimal_pedestrian_b	PIN_AE17	HEX7[1]
Segmento C - Decimal	ssd_decimal_pedestrian_c	PIN_AG17	HEX7[2]
Segmento D - Decimal	ssd_decimal_pedestrian_d	PIN_AH17	HEX7[3]
Segmento E - Decimal	ssd_decimal_pedestrian_e	PIN_AF17	HEX7[4]
Segmento F - Decimal	ssd_decimal_pedestrian_f	PIN_AG18	HEX7[5]
Segmento G - Decimal	ssd_decimal_pedestrian_g	PIN_AA14	HEX7[6]

FONTE: Elaborado pelo autor

Após realizar esse mapeamento, a configuração do pin-planner ficou da forma apresentada abaixo:

Figura 12: Mapeamento dos pinos - PinPlanner



FONTE: Elaborado pelo autor

Lembrando que para realizar o mapeamento, como estamos utilizando o kit de desenvolvimento **DE2-115**, a familia FPGA a ser utiliza é **Cyclone IV E - ALTERA**, sendo que o device dentro desta família é o **EP4CE115F29C7**

Ao final do mapeamento, a tabela do Pin-Planner ficou conforme apresentado abaixo:

Figura 13: Mapeamento dos pinos - PinPlanner

reset	Input	PIN_M23	6	B6_N2	PIN_F17	2.5 V (default)		8mA (default)			
pedestrian_button_4	Input	PIN_AA24	5	B5_N2	PIN_AE15	2.5 V (default)		8mA (default)			
pedestrian_button_3	Input	PIN_AB23	5	B5_N2	PIN_AF15	2.5 V (default)		8mA (default)			
pedestrian_button_2	Input	PIN_AB24	5	B5_N2	PIN_AD15	2.5 V (default)		8mA (default)			
pedestrian_button_1	Input	PIN_AC24	5	B5_N2	PIN_AC15	2.5 V (default)		8mA (default)			
light_detector	Input	PIN_AA23	5	B5_N2	PIN_H16	2.5 V (default)		8mA (default)			
enable	Input	PIN_AB28	5	B5_N1	PIN_J16	2.5 V (default)		8mA (default)			
clk	Input	PIN_Y2	2	B2_N0	PIN_J1	2.5 V (default)		8mA (default)			
ac_ccn	Input	PIN_Y23	5	B5_N2	PIN_D16	2.5 V (default)		8mA (default)			
yellow_light_2	Output	PIN_F18	7	B7_N1	PIN_G19	2.5 V (default)		8mA (default)	2 (default)		
yellow_light_1	Output	PIN_E19	7	B7_N0	PIN_H21	2.5 V (default)		8mA (default)	2 (default)		
ssd_unit_pedestrian[0]	Output	PIN_AA17	4	B4_N1	PIN_G15	2.5 V (default)		8mA (default)	2 (default)		
ssd_unit_pedestrian[1]	Output	PIN_AB16	4	B4_N2	PIN_G20	2.5 V (default)		8mA (default)	2 (default)		
ssd_unit_pedestrian[2]	Output	PIN_AA16	4	B4_N2	PIN_H19	2.5 V (default)		8mA (default)	2 (default)		
ssd_unit_pedestrian[3]	Output	PIN_AB17	4	B4_N1	PIN_B18	2.5 V (default)		8mA (default)	2 (default)		
ssd_unit_pedestrian[4]	Output	PIN_AB15	4	B4_N2	PIN_H15	2.5 V (default)		8mA (default)	2 (default)		
ssd_unit_pedestrian[5]	Output	PIN_AA15	4	B4_N2	PIN_J15	2.5 V (default)		8mA (default)	2 (default)		
ssd_unit_pedestrian[6]	Output	PIN_AC17	4	B4_N2	PIN_D17	2.5 V (default)		8mA (default)	2 (default)		
ssd_decimal...destrian[0]	Output	PIN_AD17	4	B4_N2	PIN_C16	2.5 V (default)		8mA (default)	2 (default)		
ssd_decimal...destrian[1]	Output	PIN_AE17	4	B4_N2	PIN_G21	2.5 V (default)		8mA (default)	2 (default)		
ssd_decimal...destrian[2]	Output	PIN_AG17	4	B4_N2	PIN_A18	2.5 V (default)		8mA (default)	2 (default)		
ssd_decimal...destrian[3]	Output	PIN_AH17	4	B4_N2	PIN_F15	2.5 V (default)		8mA (default)	2 (default)		
ssd_decimal...destrian[4]	Output	PIN_AF17	4	B4_N2	PIN_J19	2.5 V (default)		8mA (default)	2 (default)		
ssd_decimal...destrian[5]	Output	PIN_AG18	4	B4_N2	PIN_J17	2.5 V (default)		8mA (default)	2 (default)		
ssd_decimal...destrian[6]	Output	PIN_AA14	3	B3_N0	PIN_C15	2.5 V (default)		8mA (default)	2 (default)		
sound_pedestrian	Output	PIN_J16	7	B7_N2	PIN_D15	2.5 V (default)		8mA (default)	2 (default)		
red_pedestrian	Output	PIN_G17	7	B7_N1	PIN_E17	2.5 V (default)		8mA (default)	2 (default)		
red_light_2	Output	PIN_J19	7	B7_N2	PIN_G18	2.5 V (default)		8mA (default)	2 (default)		
red_light_1	Output	PIN_E18	7	B7_N1	PIN_G22	2.5 V (default)		8mA (default)	2 (default)		
green_pedestrian	Output	PIN_J17	7	B7_N2	PIN_E15	2.5 V (default)		8mA (default)	2 (default)		
green_light_2	Output	PIN_F19	7	B7_N0	PIN_G16	2.5 V (default)		8mA (default)	2 (default)		
green_light_1	Output	PIN_G19	7	B7_N2	PIN_H17	2.5 V (default)		8mA (default)	2 (default)		
crossing_light	Output	PIN_H16	7	B7_N2	PIN_A17	2.5 V (default)		8mA (default)	2 (default)		
button_light_pedestrian	Output	PIN_H17	7	B7_N2	PIN_B17	2.5 V (default)		8mA (default)	2 (default)		

FONTE: Elaborado pelo autor

2.6 Não Atendimento das especificações

- Semafaro de pedestres conta o tempo para cada sub-estado de travessia do pedestre, ou seja, para o estado de travessia G-PED, o contador mostra o tempo deste estado, em seguida, no estado G-BLINKING-PED, o contador mostra o tempo deste estado. A solução seria separar todos os estados do pedestre em uma segunda máquina de estado, atendendo assim a especificação requerida.

3 Referências bibliográficas:

- Interfaces de entrada e saída da DE2-115
- Especificações sobre displays de 7 Segmentos
- Simulação Funcional usando o ModelSim