SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE VIÇOSA · UFV
CAMPUS FLORESTAL

Trabalho-prático 1 - ISL

Sistema de pedágio inteligente

JOÃO GABRIEL BARBOSA SCHERER [EF06551]
SAMUEL NUNES FAUSTINO [EF06553]

Florestal - MG
2025

Sumário

1. Introdução

Este trabalho apresenta o desenvolvimento de um Sistema de Pedágio Inteligente capaz de classificar automaticamente veículos com base no número de eixos e no peso total.

O desenvolvimento seguiu as etapas tradicionais do design digital. Primeiro, analisamos o problema e definimos as condições para classificação dos veículos. Em seguida, construímos as tabelas verdade correspondentes e simplificamos as equações lógicas utilizando Mapas de Karnaugh e em seguida implementamos o circuito no Logisim para validar seu funcionamento. Após a validação da lógica, o sistema foi traduzido para a linguagem de descrição de hardware Verilog, simulado no Icarus Verilog e por fim visualizado no GTKWave, onde observamos as formas de onda e confirmamos o comportamento esperado.

A validação do circuito foi realizada no Logisim. Essa simulação visual permitiu verificar o funcionamento de cada componente antes da implementação em Verilog.

A saída do sistema utiliza um display de 7 segmentos que exibe o número da categoria do veículo identificado. O display pode mostrar os dígitos 1, 2, 3 ou a letra E, sendo esta última utilizada para indicar situações de erro ou casos não previstos no sistema

2. Organização

Para a organização do trabalho os módulos foram separados pelas funcionalidades em arquivos diferentes que descrevem a lógica de como cada modulo deverá funcionar.

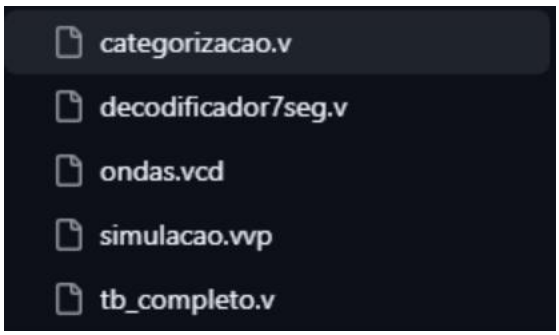


Foto referente a organização do trabalho

O arquivo categorizacao.v e descreve o circuito em verilog que considera as propriedades de peso (representadas pelos 4 bits **p3,p2,p1** e p0) e eixo (sendo representados por 2 bits **e1** e **e0**) para criar uma classificação dos veículos sendo essas, as saídas **c1** (carro de passeio), **c2** (caminhonete), **c3** (caminhão) e **E** (erro).

O arquivo decodificador7seg.v é a descrição em verilog do comportamento do circuito que gera uma representação visível em um display de sete segmentos utilizando como entradas **c1, c2, c3** e **E** (as saídas do arquivo categorizacao.v) gerando saídas de a até g que representam os segmentos do display (7 no total).

Os arquivos ondas.vcd e simulacao.vvp são arquivos gerados na compilação responsáveis pela simulação e representação em forma de ondas.

Por fim, o arquivo tb_completo.v é o testbench que oferece a simulação completa do sistema, de seu início na classificação de categorias de veículos até a sua representação no display de 7 segmentos, essa simulação é feita fazenda a verificação exaustiva de todas as 64 combinações das entradas de peso e eixo, isto é, os bits **p3,p2,p1,p0,e1** e **e0**.

3. Desenvolvimento

O sistema de classificação desenvolvido é determinado com base em duas variáveis primárias de entrada: o peso total em toneladas (P) e o número de eixos (E).

1. Definição das Variáveis de Entrada:

- **Número de Eixos (Variável E – 2 bits: E1 E0):**
  - 00: 2 eixos
  - 01: 3 eixos
  - 10: 4 eixos
  - 11: 5 ou mais eixos (não utilizamos, pois, seria redundante usar mais uma variável de saída para a mesma funcionalidade da saída anterior)
- **Peso Total (em Toneladas) (Variável P – 4 bits: P3 P2 P1 P0):**
  - A faixa de medição é de 0 a 15 toneladas.

2. Condições Lógicas de Classificação (Funcionalidade do Sistema):

A funcionalidade do sistema é regida pelas seguintes condições lógicas, onde E1 é o bit de maior peso da variável de eixos, e P é o valor do peso total:

- **Categoria 1 (C1):** O veículo é classificado como Categoria 1 se o peso total for menor ou igual a 7 toneladas e os dois bits para eixo forem iguais a 0.
- **Categoria 2 (C2):** O veículo é classificado como Categoria 2 se o peso total for menor ou igual a 12 toneladas e os bits para eixo forem iguais a 0 e 1 respectivamente.

- **Categoria 3 (C3):** O veículo é classificado como Categoria 3 se o peso total for maior que 12 toneladas e o bit E1 for igual a 1 (portanto, podendo ser tanto 10 quanto 11).
- **Categoria Erro (E):** O sistema deve indicar uma Categoria Erro para qualquer combinação de variáveis que não se encaixe nas Categorias C1, C2 ou C3.

3.1 Construção das tabelas verdade

A construção das tabelas verdade se basearam nas regras de classificação de veículos, formando as seguintes tabelas:
Tabela verdade para peso:

p3	p2	p1	p0	P<=7	P<=12	P>12
0	0	0	0	1	1	0
0	0	0	1	1	1	0
0	0	1	0	1	1	0
0	0	1	1	1	1	0
0	1	0	0	1	1	0
0	1	0	1	1	1	0
0	1	1	0	1	1	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	0	1

Tabela verdade para eixos:

E1	E0	2eixos	3eixos	4eixos_ou+
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	0	1

3.2 implementações dos mapas de karnaugh e funções booleanas simplificadas

Para cada saída da tabela verdade foi feito um mapa de karnaugh e sua simplificação booleana, como é possível observar abaixo:

3.2.1 K-maps para o peso

K-map para primeira saída:

p ≤ 7

p3p2 \ p1p0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

Sua função booleana simplificada ficou: ~p3

K-map da segunda saída:

p ≤ 12

p3p2 \ p1p0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	0	0	0
10	1	1	1	1

Sua função booleana simplificada ficou: ~p3+~p2+(~p1*~p0)

K-map da terceira saída:

p > 12

p3p2 \ p1p0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	1
10	0	0	0	0

Sua função booleana simplificada ficou: p0*p3*~p2 + p1*p3*~p2
Obs: seria possível simplificar mais a função para: (p0+p1)*(p3*~p2), porém optamos pela outra versão para facilitar o design do circuito lógico no Logisim.

3.2.2 K-maps para os eixos

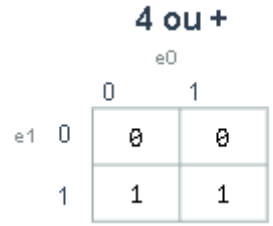
K-map para a primeira saída:

Sua função booleana simplificada ficou: ~e1*~e0

K-map para a segunda saída:

Sua função booleana simplificada ficou: ~e1*e0

K-map para a terceira saída:



Sua função booleana simplificada ficou: e1

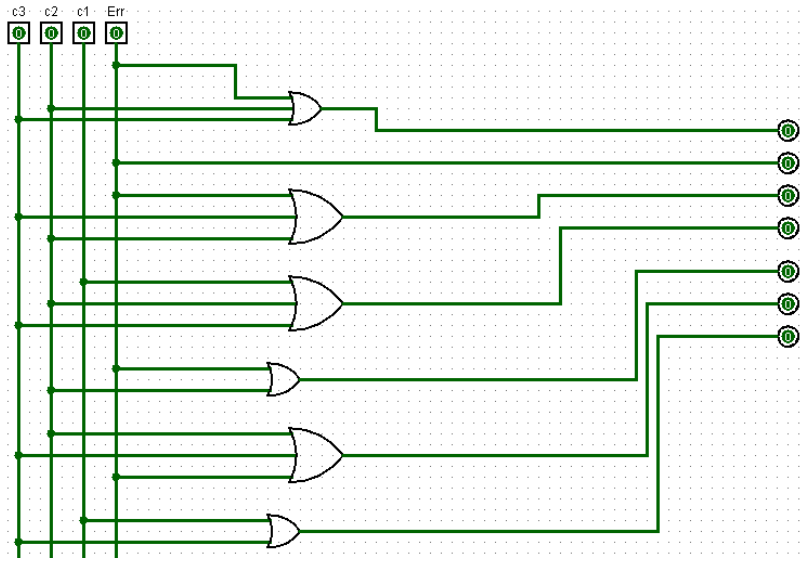
3.3 circuito logico no Logisim

As implementações anteriores serão uteis agora, pois para fazer o circuito logico, antes é necessário ter todas as funções preparadas. Para esse processo unimos cada saída ao seu respectivo parceiro de categoria, assim facilitando o trabalho de fazer uma só tabela verdade com 6 entradas.
Saídas concatenadas:

- P<=7 and 2eixos
- P<=12 and 3eixos
- P>12 and 4eixos_ou+

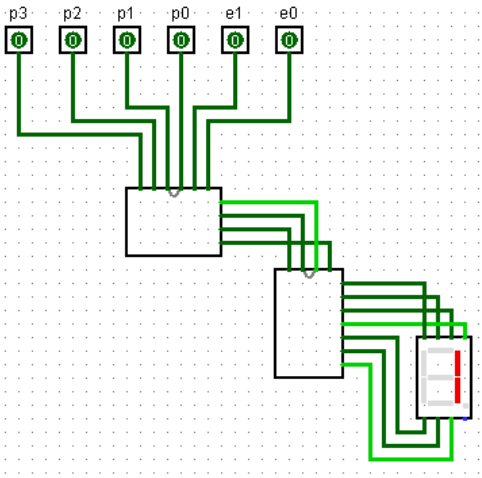
As funções lógicas ficaram dispostas da seguinte maneira:
Categoria 1: p3' · e1' · e0'
Categoria 2: (p3' + p2' + p1' · p0') · e1' · e0
Categoria 3: (p0 · p3 · p2' + p1 · p3 · p2') · e1
Categoria erro: (cat1+cat2+cat3)' (nor das 3 categorias anteriores)

Após esse primeiro passo entramos na parte do display de 7 segmentos, na qual as saídas anteriores se tornam as entradas deste novo. Utilizamos 'ous' para realizar a saída correta de cada led do display.



Circuito relacionado ao display (cada saída representa uma parte do display).

Ao final tem-se o circuito completo, utilizando o circuito de categorização e o circuito do display de 7 segmentos:



3.4 módulos Verilog e visualização em forma de ondas

Em Verilog, um módulo representa um bloco de hardware que pode ter entradas e saídas e contém a lógica necessária para realizar uma função específica. Cada módulo pode ser independente ou instanciado dentro de outro módulo maior, permitindo organizar o projeto de forma hierárquica. No caso do sistema de pedágio inteligente, os módulos `s1`, `s2` e `s3` realizam verificações lógicas sobre o peso e o número de eixos do veículo, enquanto o módulo `error` indica quando uma combinação inválida ocorre. O módulo principal `circuito` integra esses blocos e gera sinais de controle que serão usados em outros módulos, como o `sete_segmentos`, que converte esses sinais em padrões para um display de 7 segmentos.

O *testbench* é um módulo separado que não faz parte do hardware real, mas serve para simular o funcionamento do projeto. Nele usamos registradores para colocar valores nas entradas do circuito e fios para ler as respostas. Nesse testbench do pedágio, há um loop que passa por todas as combinações possíveis de peso e número de eixos, testando cada caso, atribuindo esses valores às entradas do circuito e registrando as saídas. Durante a simulação, comandos como `$monitor` exibem as alterações no terminal, enquanto `$dumpfile` e `$dumpvars` criam um arquivo de formas de onda que pode ser visualizado no GTKWave, permitindo acompanhar a evolução dos sinais ao longo do tempo. Essa visualização ajuda a compreender como o circuito reage a diferentes entradas de maneira sequencial.

4. Compilação e Execução

A compilação e execução do arquivo deverá ser feita através do terminal dentro do diretório da pasta ‘’trabalho-ISL/’’, primeiramente a compilação do código é feita por meio da linha de comando:

- iverilog -o simulacao.vvp categorizacao.v decodificador7seg.v tb_completo.v

Já para execução dos módulos (assim também como para a criação de um arquivo **.vcd** no qual será usado para criar uma simulação de onda, que pode ser vista por meio do programa gtkwave) deve ser usado o comando:

- vvp simulacao.vvp

E por fim, para abrir a simulação de onda do comportamento lógico do sistema será utilizado:

- gtkwave ondas.vcd

5. Resultados

Como resultado, foi concluída a implementação de um módulo digital que permite a classificação de veículos em categorias que podem ser usadas para sistemas de identificação de veículos e cobrança automática de tarifas e pedágios, tais categorias sendo influenciada pelos valores de peso e eixo coletados por sensores na pista.

O sistema contou com uma implementação em verilog da lógica combinacional responsável por categorizar os carros e mostrar a dada categoria em um display de sete segmentos, assim como a elaboração de um circuito com o mesmo objetivo e lógica.

6. Conclusão

No desenvolvimento do projeto, começamos criando tabelas verdade separadas para cada saída (s1, s2 e s3) com base no peso e no número de eixos dos veículos. Em seguida, simplificamos essas equações usando Mapas de Karnaugh e unimos as funções reduzidas em um único circuito, o que tornou a implementação mais organizada e eficiente. Depois, implementamos o circuito no Logisim para validar sua lógica, garantindo que cada categoria de veículo fosse corretamente identificada. Para a visualização dos resultados, utilizamos um display de 7 segmentos, onde as saídas podiam mostrar os números 1, 2 ou 3, ou E em caso de erro, tornando o acompanhamento do sistema mais direto. Após essa validação, descrevemos todo o projeto em Verilog, simulamos no Icarus Verilog e visualizamos as formas de onda no GTKWave, confirmando que o comportamento do circuito correspondia ao esperado para todas as combinações de entrada.

7. Referências

CILETTI, Michael D. *Advanced Digital Design with the Verilog HDL*. 2. ed. Upper Saddle River: Pearson Prentice Hall, 2010.

ANTHROPIC. *Claude AI* [inteligência artificial generativa]. Disponível em: https://claude.ai/. Acesso em: 15 out. 2025. (utilizado para gerar os mapas de karnaugh para a documentação)