Universidade Federal do Rio Grande do Norte

Centro de Ciências Exatas e da Terra

Bacharelado em Ciência da Computação

DIM0128 - Circuitos Lógicos

Relatório do Sistema de Elevadores em VHDL

Autores: Luisa Ferreira de Souza Santos, Cícero Paulino de OLiveira Filho, Kauã do Vale Ferreira e Ryan David dos Santos Silvestre

Professor: Márcio Eduardo Kreutz

Natal – RN

Outubro de 2025

1 Resumo e Objetivos

O objetivo deste projeto é projetar e implementar em VHDL um sistema para controlar três elevadores em um edifício de 32 andares (0 a 31). O sistema é dividido em dois níveis hierárquicos, conforme especificado:

1. **Nível 1 (Controlador Local):** Um controlador individual para cada elevador, responsável por gerenciar as operações físicas como acionar o motor, controlar a abertura e fechamento das portas e registrar o andar atual.

2. **Nível 2 (Escalonador / Pai):** Um supervisor global que gerencia todas as chamadas externas (botões de subir/descer dos andares), decide qual dos três elevadores deve atender a cada requisição e envia os comandos para os controladores locais.

O projeto faz uso de Máquinas de Estados Finitos (FSMs) para o controle dos componentes, que serão validadas através de simulação com *testbenches* em VHDL.

2 Arquitetura Geral (Diagrama de Blocos)

A arquitetura do sistema segue o modelo Controlador de dois níveis exigido. O diagrama de blocos abaixo ilustra a conexão dos módulos principais:

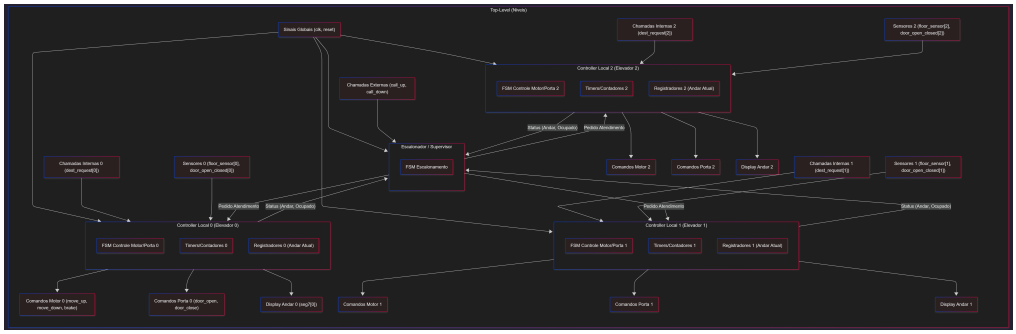

Figure 1: Diagrama de blocos do sistema de controle de três elevadores.

A arquitetura é composta por:

- **1x Módulo Escalonador (Supervisor):**
  - **Função:** Supervisiona todas as chamadas externas dos três elevadores, decidindo qual elevador atenderá cada requisição com base no algoritmo de escalonamento (descrito na Seção 4).

  - **Lógica:** Implementa a lógica de seleção de elevador e envia comandos para os controladores locais. Também monitora os estados atuais de cada elevador (andar, direção e estado da porta).

- **3x Módulos Controladores Locais (Elevador):**
  - **Função:** Coordena a operação de cada elevador, integrando os módulos de motor e porta. Recebe chamadas internas do elevador e requisições do escalonador, interpreta os sinais dos sensores e emite comandos apropriados para motor e a porta.
  - **Lógica:** Estruturado como uma FSM principal para gerenciar os estados do elevador: `IDLE` (ocioso), `MOVENDO`, `ABRINDO_PORTA`, `PORTA_ABERTA` e `FECHANDO_PORTA`.

- **3x Módulos de Componentes (Motor, Porta):**
  - **Funções:** São os "músculos" (componentes "burros") que obedecem aos comandos dos Controladores Locais. O motor contorla a movimentação do elevador, enquanto a porta gerencia a abertura e o fechamento.
  - **Lógica:**
    * **Motor:** Possui FSM interna para operação segura com estados como `PARADO`, `SUBINDO`, `DESCENDO` e `FREANDO`.
    * **Porta:** Controlada por lógica sequencial baseada em contadores e condicionais, garantindo abertura e fechamento seguros e temporizados.

3 Interface de Sinais Usadas

3.1 Top-Level

Interface de sinais planejada para o módulo `top-level` (que conectará todos os blocos) é baseada nos arquivos `.vhd` existentes:

Entradas Globais (do Testbench)

- `clk`, `rst`: Clock global e reset.
- `chamadas_externas_subir [0..30]`: Botões de subida dos andares.
- `chamadas_externas_descer [1..31]`: Botões de descida dos andares.
- `chamadas_internas_E1, E2, E3 [0..31]`: Painéis internos de cada elevador.
- `sensor_andar_E1, E2, E3 [0..31]`: Sensores de posição (simulados).

Saídas Globais (para Displays/Testbench)

- `display_andar_E1, E2, E3`: Indicador de andar atual.
- `display_porta_E1, E2, E3`: Indicador de porta (aberta ou fechada).
- `display_direcao_E1, E2, E3`: Indicador de movimento (subindo ou descendo).

Sinais Internos Chave (Entre Módulos)

- `req_do_escal_E*`: Saída do Escalonador para a entrada do Controlador Local.
- `cmd_motor_E*`: Saída do Controlador Local para a entrada `comando` do Motor.
- `cmd_porta_E*`: Saída do Controlador Local para a entrada `abre` da Porta.
- `sensor_mov_E*`: Saída `em_movimento` do Motor, ligada de volta ao Controlador Local.
- `sensor_porta_E*`: Saída `porta_aberta` da Porta, ligada de volta ao Controlador Local (e ao Motor para segurança).

3.2 Escalonador

Entradas Globais

- `clk`, `rst`: Clock global e reset.

- `pos_elevador_1`: Posição do elevador 1.
- `pos_elevador_2`: Posição do elevador 2.
- `pos_elevador_3`: Posição do elevador 3.

- `estado_elevador_1`: Estado do elevador 1.
- `estado_elevador_2`: Estado do elevador 2.
- `estado_elevador_3`: Estado do elevador 3.

- `requisicoes_externas_elevador_1`: Requisições externas do elevador 1.
- `requisicoes_externas_elevador_2`: Requisições externas do elevador 2.
- `requisicoes_externas_elevador_3`: Requisições externas do elevador 3.

Saídas Globais

- `requisicao_andar_elevador_1`: qual andar o elevador 1 deve ir.
- `requisicao_andar_elevador_2`: qual andar o elevador 2 deve ir.
- `requisicao_andar_elevador_3`: qual andar o elevador 3 deve ir.

3.3 Elevador

Entradas Globais

- `clk`, `rst`: Clock global e reset.

- `requisicoes_escalonador`: Chamadas do escalonador.
- `requisicoes_internas`: Chamas internas dos botões da cabine.

- `sensor_andar_atual`: Andar atual do elevador.
- `sensor_porta_aberta`: Estado da porta (aberta ou fechada).
- `sensor_movimento`: Estado do motor (parado, subindo ou descendo).

Saídas Globais

- `comando_motor`: Estado do elevador 3.
- `comando_porta`: Requisições externas do elevador 1.

Sinais Internos (Saída)

- `andar_atual`: Andar atual do elevador.
- `estado_motor`: Estado do motor (parado, subindo ou descendo).
- `estado_porta`: Estado da porta (aberta ou fechada).

3.4 Motor

Entradas Globais

- `clk`, `rst`: Clock global e reset.

- `comando`: Comando para a movimentação do elevador (parar, subir ou descer).
- `porta`: Estado atual da porta (aberta ou fechada)

Saídas Globais

- `em_movimento`: Indicação se o motor está em movimento.
- `direcao`: Direção para qual o motor está se movendo.
- `freio`: Indicação se o freio está em sendo utilizado.

3.5 Porta

Entradas Globais

- `clk`, `rst`: Clock global e reset.
- `comando`: Comando para a movimentação da porta (abrir ou fechar).
- `motor_mov`: Estado atual do motor (em movimento ou parado)

Saídas Globais

- `porta_aberta`: Estado atual da porta (aberta ou fechada).

4 Estratégia de Escalonamento

O método de *Scan* consiste em percorrer todos os andares a partir de um sentido inicial, seja para cima ou baixo, mudando o sentido apenas ao chegar ao térreo, cobertura ou ao andar mais alto. Esse método *"escaneia"* todos os andares de modo que ele para apenas no andar em que foi solicitado a parada para subir ou descer do elevador.

5 Parâmetros Adotados

- Número de andares: 32 (0 a 31)
- Timer da porta: 250.000.000 ciclos de clock (se o clock for de 50MHz = 5 segundos)
- Outros parâmetros: tempos de abertura/fechamento de porta, tempo de deslocamento entre andares, etc.

6 Exemplos de Simulação

Deverão ser incluídos pelo menos três cenários com capturas de forma de onda e explicação passo a passo, por exemplo:

- Chamadas simultâneas em andares diferentes.

- Conflito de chamadas no mesmo andar.
- Teste de segurança da porta durante movimento.

7 Problemas e Decisões de Projeto

De início tivemos três opções de método de escalonamento, ambos visando características que beneficiariam o projeto, tais características seriam: simplicidade, otimização para aspectos específicos e eficiência.

7.1 Scan

O método de *Scan* consiste em percorrer todos os andares a partir de um sentido inicial, seja para cima ou baixo, mudando o sentido apenas ao chegar ao térreo ou cobertura. Esse método *"escaneia"* todos os andares de modo que ele para apenas no andar em que foi solicitado a parada para subir ou descer do elevador.

Observamos também outras duas possibilidades de implementar o *Scan*. A primeira consistia em limitar os pontos de troca de sentido a chamada mais alta ou mais baixa ao invés das extremidades do prédio. A segunda trava do posicionamento inicial dos elevadores: o *elevador 0* teria posição inicial no térreo, o *elevador 1* ficaria no andar 15 (centro do prédio de 32 andares) e o *elevador 2* ficaria na cobertura. Nesse sentido os elevadores seguiriam a mesma lógica do *Scan* básico, no entanto o elevador *1* pegaria o *overflow* dos elavadores adjacentes, de modo que seria um elevador para tratamento de sobrecarga.

7.2 Precedência e proximidade

Essa abordagem analisa o estado atual de cada um dos três elevadores (andar, direção, ocupado/pronto) e decide qual elevador é o mais adequado analisando o custo, proximidade e direção para assim atender a uma nova solicitação. Após a decisão, ele envia as ordens específicas para o *Controller Local* do elevador selecionado, que executa a sequência de operações para completar a viagem

7.3 Vantagens e desvantagens

A abordagem do *Scan* facilita a implementação do escalonador por possuir uma estratégia mais *lazy* para lidar com o problema. No entanto essa estratégia também possui um custo que se dá na eficiência (velocidade e tempo de espera) de atendimento de requisições. Para diminuir esses custos, a adição de estratégias de posicionamento e cálculos de distância, mostraram-se relativamente benéficos.

No entanto, a maior eficiência se encontra na abordagem onde a eurística é mais fortemente empregada. No caso da abordagem de proximidade e precedência o gerenciamento é feito de uma forma mais rebuscada, priorizando tempo de espera e velocidade. Porém a complexidade do circuito, principalmente na implementação estrutural, dificulta a implantação do mesmo.

8 Instruções para Reproduzir as Simulações

O projeto é rodado utilizando o guia fornecido em aula, com GHDL e GTK Wave.

- Analisar arquivos: `ghdl -a <seu_projeto>.vhdl <seu_testbench>.vhdl`
- Elaborar testbench: `ghdl -e <seu_testbench>`
- Rodar simulação e gerar arquivo de onda: `ghdl -r <seu_testbench> --vcd=wave.vcd`
- Visualização no GTK Wave: `gtkwave wave.vcd`