Implementação em VHDL de um sistema de controle para 3 elevadores (controle em 2 níveis)

Objetivo

Projetar e implementar em **VHDL** um sistema digital que controla **3 elevadores** de um edifício em **dois níveis**:

1. **Nível 1 — Controlador individual** (por elevador): comando de abrir/fechar porta, subir/descer "X" andares, acionar/parar motor, indicação do andar atual em displays.

2. **Nível 2 — Supervisão / Escalonador**: gerencia chamadas (externas) e decide **qual elevador** atende cada chamada, priorização entre subir/descer, balanceamento de carga e otimização de tempo/viagens.

O sistema deve ser testado via **simulações digitais** (testbenches) e apresentado em relatório e vídeo.

Especificações funcionais

Parâmetros básicos

- **Número de elevadores**: 3
- **Número de andares**: 32 (andares 0 a 31) — codificação em binário (5 bits)
- **Tipos de chamadas externas**: botão **Subir** e botão **Descer** em cada andar (exceto andar topo/rodapé onde só há um tipo)
- **Chamadas internas**: painel de andares dentro de cada elevador (32 botões por cabine)
- **Indicadores**: display de 7 segmentos para mostrar andar atual e LED(s) para estado da porta (aberta/fechada), movimentação (subindo/descendo).
- **Temporalidades/temporizadores**: tempo de abertura/fechamento de porta, tempo de deslocamento entre andares — implementados por contadores.

Entradas sugeridas

- Sinais de chamada externa: call_up[0..31], call_down[0..31]
- Sinais de chamada interna por elevador: dest_request[i][0..31] (i = 0..2)
- Sensores por elevador: floor_sensor[i][k] (k = 0..31); door_open_closed[j] (j = 0..2)
- Sinais de controle: reset, clk

Saídas sugeridas

- Comandos para motor: move_up[i], move_down[i], motor_enable[i], brake[i] (i = 0..2)
- Comandos porta: door_open[i], door_close[i]
- Displays: seg7[j][0..6] (j = 0..2)

Arquitetura exigida (mínimo)

1. **Módulo top-level** que liga todos os blocos.
2. **Controller Local (por elevador)** — responsável por sequência de operações do elevador: receber ordens locais (do painel interno e do escalonador), controlar motor e portas, gerar status (andar e pronto/ocupado). Deve possuir máquina(s) de estado finito (FSM) bem documentadas.
3. **Escalonador / Supervisor (nível 2)** — recebe todas as chamadas externas/ internas, decide qual elevador atende (estratégia de alocação utilizada) e envia pedidos aos controllers locais.
4. **Interface de entrada/saída** para simulação (testbench).

Requisitos de implementação

- Projeto **síncrono** (clock global).
- Uso de **FSMs** claramente documentadas (diagrama de estados no relatório).
- **Parametrização**: número de andares e tempos devem ser parâmetros genéricos constantes.

- **Testbench completo** cobrindo: chamadas simples, concorrentes e comportamento de portas.
- Código limpo, comentários e organização modular (cada entidade em arquivo separado).

Entregáveis (obrigatórios)

1. **Código VHDL** completo (arquivos .vhd) organizados em pasta com script de simulação ou instruções claras para rodar as simulações.
    - Deve incluir: top-level, controladores locais, escalonador e testbenches.
2. **Relatório técnico (PDF)** explicando o projeto contendo:
    - Resumo e objetivos.
    - Arquitetura geral (diagrama de blocos).
    - Diagramas de estados das FSMs (por elevador e do escalonador).
    - Interface de sinais (descrição de entradas).
    - Estratégia de escalonamento (algoritmo escolhido — ex.: algoritmo nearest-car, prioridade por direção, round-robin, etc.) e justificativa.
    - Parâmetros adotados (nº de andares, tempos, etc.).
    - Exemplos de simulação: pelo menos **três cenários** com capturas de forma de onda e explicação step-by-step (ex.: chamada simultânea em andares diferentes).
    - Problemas encontrados e decisões de projeto.
    - Instruções para reproduzir simulações.
3. **Vídeo de apresentação (máx. 10 minutos)** demonstrando o funcionamento via simulação:
    - Explicação rápida da arquitetura;
    - Demonstração prática de pelo menos os cenários citados no relatório (mostrar formas de onda/logs e como as decisões do escalonador ocorrem);
    - Link para rodar as simulações localmente (ou gravação da execução).

Critérios de avaliação (peso total = 100%)

- **Funcionalidade** (30%): o sistema atende às especificações; controladores locais e escalonador funcionam corretamente nos cenários de testes.

- **Qualidade do projeto** (20%): modularidade, FSMs bem projetadas, utilização adequada de recursos síncronos e tratamento de condições limite.
- **Testes e execução** (15%): testbenches completos, cobertura dos cenários exigidos, clareza das simulações e das formas de onda apresentadas.
- **Relatório** (25%): clareza, diagramas, justificativas, documentação que permita reproduzir o trabalho.
- **Vídeo e apresentação** (10%): clareza na demonstração e capacidade de explicar o projeto e resultados.

Prazos e formato de entrega

- **Entrega final**: 29/10/25 — entregue um único arquivo .zip via tarefa no SIGAA.
- Nome do arquivo: TRAB_ELEVADORES_<alunos>.zip
- SOMENTE UM arquivo por grupo.
- Incluir um README.md na raiz com instruções de simulação (ferramenta/sintaxe: ModelSim/GHDL + gtkwave, etc.)
- Vídeo hospedado em link (YouTube, Google Drive) ou enviado no zip se o tamanho permitir.

Recomendações técnicas e dicas

- Defina desde o início a **interface entre escalonador e os controladores** (pedido simples: request_floor, direction, ack). Simplicidade ajuda a testar.
- Faça primeiro um **controlador local** que responde a comandos simples (subir, descer, abrir/fechar) e mostra andar — depois adicione o escalonador.
- Use **simulações em etapas**: unitárias (cada elevador), integração (escalonador + 3 elevadores), cenários complexos.

Observações finais

- Trabalho em grupo de no máximo 4 alunos.
- Plágio e cópia são proibidos — o professor poderá exigir demonstração ao vivo/sessão de perguntas.
- Qualquer variação relevante nas especificações adotadas deve ser comunicada e justificada no relatório.