Implementação em Hardware de um Controlador de Grupo de Elevadores Utilizando o Modelo PO-PC

Carlos Henrique Cortez Gomes
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Natal, Brasil
carlos.cortez.708@ufrn.edu.br

Haniel Lucas Machado Rocha
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Natal, Brasil
hani.lucas70@gmail.com

Jamilly Emilly da Silva Campelo
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Natal, Brasil
jamilly.campelo.700@ufrn.edu.br

Pedro Vinícius Barbosa Pereira
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Natal, Brasil
pedro.barbosa.703@ufrn.edu.br

***Abstract*—This paper presents the design and implementation of an elevator group controller (Level 2) using VHDL. The system adopts a Data Path (PO) and Control Path (PC) architecture to optimize vertical traffic efficiency. The dispatching algorithm employs a hybrid strategy combining Direction Priority and Nearest-Car logic, calculating a cost function based on the elevator's current state and call location. The architecture includes specific hardware modules, such as a Sweeper for call detection and a Finite State Machine (FSM) for orchestration. Behavioral simulations validated the system's ability to correctly assign elevators in various traffic scenarios, demonstrating the robustness of the modular hardware design.**

***Index Terms*—Elevator Control, VHDL, FPGA, PO-PC Model, Dispatching Algorithm.**

RESUMO

Este artigo apresenta o projeto e implementação de um controlador de grupo de elevadores (Nível 2) utilizando VHDL. O sistema adota uma arquitetura dividida em Parte Operativa (PO) e Parte de Controle (PC) para otimizar a eficiência do tráfego vertical. O algoritmo de despacho emprega uma estratégia híbrida combinando Prioridade por Direção e a lógica do Carro Mais Próximo (*Nearest-Car*), calculando uma função de custo baseada no estado atual do elevador e na localização da chamada. A arquitetura inclui módulos de hardware específicos, como um Varredor (*Sweeper*) para detecção de chamadas e uma Máquina de Estados Finitos (FSM) para orquestração. Simulações comportamentais validaram a capacidade do sistema de atribuir corretamente elevadores em variados cenários de tráfego, demonstrando a robustez do design modular de hardware.

I. INTRODUÇÃO

O transporte vertical eficiente é um requisito fundamental na arquitetura moderna e na urbanização densa. Sistemas de elevadores não devem apenas garantir a segurança mecânica e o deslocamento entre andares, mas também otimizar o fluxo de passageiros. À medida que o tráfego em edifícios aumenta, a lógica de despacho — a decisão de qual elevador deve atender a uma chamada — torna-se um problema complexo de otimização em tempo real.

O controle de elevadores é dividido em dois níveis hierárquicos: o controle local (Nível 1), responsável pela movimentação do motor, portas e segurança de um único elevador; e o controle de grupo (Nível 2), responsável pelo escalonamento e atribuição de chamadas externas. Enquanto o Nível 1 foca na execução mecânica, o Nível 2 foca na eficiência operacional, buscando minimizar métricas críticas como o tempo de espera do passageiro e o tempo total de viagem.

A implementação de tais sistemas em hardware digital requer metodologias robustas de design. A utilização de linguagens de descrição de hardware, como VHDL, aliada à metodologia de separação entre Parte Operativa (PO) e Parte de Controle (PC), permite a criação de sistemas modulares, escaláveis e verificáveis. O modelo PO-PC facilita o desenvolvimento ao segregar o fluxo de dados (registros, somadores, comparadores) da lógica de decisão sequencial (Máquinas de Estados Finitos).

Neste contexto, este trabalho apresenta o desenvolvimento de um algoritmo de escalonamento para um sistema de 3 elevadores. O projeto utiliza a lógica digital para implementar uma função de custo baseada na distância e direção dos carros, visando a otimização do atendimento às chamadas. A implementação visa integrar o controlador de alto nível (Nível 2) aos controladores de baixo nível (Nível 1) previamente desenvolvidos, consolidando um sistema completo de gerenciamento de tráfego vertical.

A. Objetivos

O objetivo geral deste trabalho é projetar, implementar e validar, através de simulação em VHDL, um algoritmo de controle e escalonamento para um sistema de elevadores. A proposta utiliza a modelagem PO-PC (Parte Operativa e Parte

de Controle) para integrar e gerenciar múltiplos controladores de nível 1.

Para alcançar essa meta, foram definidos os seguintes objetivos específicos:

- Desenvolver a Parte Operativa (PO), implementando componentes de hardware dedicados ao armazenamento e processamento de chamadas e calculadores de custo baseados na posição atual dos elevadores;
- Projetar a Parte de Controle (PC), elaborando uma Máquina de Estados Finitos (FSM) responsável por orquestrar a leitura das chamadas, o cálculo de custos e a atribuição do elevador responsável por cada requisição;
- Integrar os Níveis de Controle, realizando a interface entre o Escalonador (Nível 2) e os controladores individuais (Nível 1) para garantir a correta transmissão de comandos;
- Validar e analisar o desempenho através de simulações comportamentais, comprovando a robustez do algoritmo frente a cenários de múltiplas chamadas simultâneas.

II. Referencial Teórico

Uma decisão de projeto crucial no sistema de controle de elevadores é a escolha do algoritmo de escalonamento, que corresponde ao critério adotado pelo Escalonador (Nível 2) para designar qual elevador irá atender cada chamada externa. A eficiência deste algoritmo é fundamental para otimizar métricas como o tempo médio de espera e o tempo de percurso, o que se traduz em melhor qualidade de serviço para os usuários.

A literatura de controle de tráfego de elevadores apresenta diversas abordagens. Alguns dos algoritmos mais comuns incluem:

- *Nearest-Car:* É o algoritmo mais simples. O elevador escolhido para atender uma chamada é o que está fisicamente mais próximo do andar da chamada, independentemente da sua direção atual ou de outras chamadas em fila;
- *Fixed Zoning:* Divide o edifício em zonas verticais (baixas, médias e altas), e cada elevador é designado permanentemente para atender chamadas apenas na sua zona. É eficiente para edifícios muito altos com picos de tráfego específicos;
- *Prioridade por Direção (Collective Control):* Considera a direção atual do elevador e as chamadas internas já registradas. O elevador só para para atender uma nova chamada externa se ela estiver na sua direção atual e no seu caminho. Este é considerado um dos algoritmos básicos mais eficientes, pois minimiza mudanças de direção e percursos vazios;
- *Lógica Fuzzy ou Redes Neurais:* Abordagens mais avançadas que utilizam heurísticas ou aprendizado de máquina para calcular o custo de atendimento, levando em conta múltiplas variáveis como a carga do elevador e o tempo de espera projetado.

A. Algoritmo Adotado

No projeto em questão, foi adotada uma estratégia híbrida: o algoritmo de Prioridade por Direção combinado com o *Nearest-Car*. O funcionamento ocorre em duas etapas principais:

- *Cálculo da Prioridade (Cost Calculator):* Determina o custo com base no estado e direção. O custo ótimo ocorre quando o elevador está se movendo na mesma direção da chamada e ainda não a ultrapassou. O custo intermediário ocorre quando o elevador está parado. O pior custo é atribuído quando o elevador está na direção contrária ou já ultrapassou o andar;
- *Desempate por distância (Nearest-Car):* Se houver um empate no critério de custo entre dois ou mais elevadores, o desempate é feito escolhendo o elevador com a menor distância física até o andar da chamada.

A seleção desta abordagem híbrida é justificada por sua capacidade de equilibrar a eficiência de tráfego e a capacidade de resposta. Ao priorizar elevadores que já estão na direção correta, minimizam-se paradas desnecessárias e viagens vazias, reduzindo o consumo de energia e o tempo total de percurso. Além disso, o critério de desempate garante resposta rápida em situações onde múltiplos elevadores possuem o mesmo status.

III. Arquitetura da Parte Operativa

A Figura a seguir ilustra a arquitetura interna da Parte Operativa (PO) do Escalonador. Este módulo é responsável por todo o processamento de dados, cálculos aritméticos e roteamento de sinais do sistema, operando sob o comando da Parte de Controle (PC).

A arquitetura foi projetada em estágios sequenciais de processamento. O fluxo de dados inicia-se com os vetores de chamadas externas, provenientes de um registrador externo. Os componentes principais são descritos a seguir:

- *Sweeper (Varredor):* Componente combinacional que varre os bits de entrada para identificar a próxima chamada pendente, alternando entre as chamadas de subir e descer. Ao encontrar uma chamada ativa, disponibiliza o andar e a direção, sinalizando à PC;
- *Target Register:* Registrador responsável por armazenar a chamada selecionada pelo Sweeper, garantindo a estabilidade dos dados (andar e direção alvo) durante o ciclo de cálculo;
- *Cost Calculator:* Existem três instâncias deste bloco, uma para cada elevador. Cada instância recebe o alvo e o estado atual do elevador correspondente para determinar a aptidão (custo) baseada na direção do movimento e posição relativa;
- *Distance Calculator:* Calcula, em paralelo, a distância física absoluta entre o andar atual do elevador e a chamada alvo, servindo como métrica de desempate;
- *Cost Comparator:* Centraliza os resultados, comparando os custos de aptidão (prioridade primária) e as distâncias (prioridade secundária) para determinar o índice do elevador vencedor;

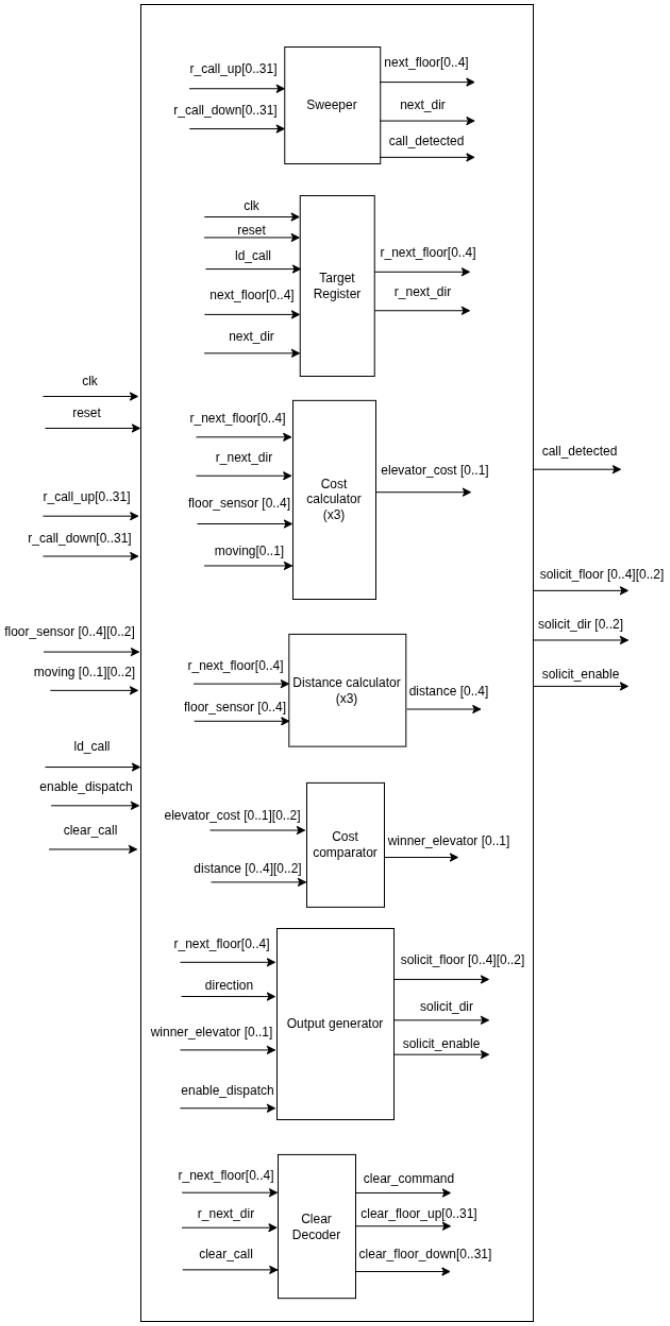

Fig. 1. Diagrama de Blocos da PO

- *Output Generator:* Atua como um roteador que, quando habilitado, direciona os comandos de solicitação especificamente para o controlador do elevador vencedor;
- *Clear Decoder:* Após o envio do comando, decodifica o andar e a direção atendidos para gerar um pulso específico que zera o bit correspondente no registrador de chamadas externas.

Esta estrutura modular permite que a Parte de Controle gerencie o tempo e a sequência das operações, enquanto a Parte Operativa resolve a complexidade da lógica de seleção e roteamento.

IV. Máquina de Estados da Parte de Controle

A Parte de Controle (PC) foi implementada como uma Máquina de Estados Finitos (FSM) composta por quatro estados principais, responsáveis por orquestrar o fluxo de dados da Parte Operativa. O funcionamento segue a lógica descrita abaixo:

- *Idle:* É o estado inicial de espera. Ocorre enquanto o sinal `call_detected` (originado do componente `sweeper`) permanece em nível baixo, indicando que não há chamadas ativas nos vetores `r_call_up` e `r_call_down`;
- *Loading Call:* Quando o sinal `call_detected` é ativado, a máquina transita para este estado. Neste momento, o andar requisitado é carregado na estrutura comportamental da Parte Operativa para processamento;
- *Enable Dispatch:* Na próxima borda de subida do *clock*, o sistema avança para este estado, liberando a transmissão da chamada de andar para o elevador escolhido (vencedor do cálculo de custo);
- *Clear Call:* O estado final libera a conexão do componente `clear_decoder` para o `external_calls_register`. Esta ação reseta a chamada atendida no registrador e retorna a FSM para o estado *Idle*, reiniciando o ciclo.

V. Simulação e Resultados

Para demonstrar o funcionamento e validar a lógica do controlador de Nível 2, foram analisados quatro casos principais de tráfego através de simulação.

A. Cenário 1: Chamada Única de Subida

O primeiro caso avalia uma requisição única de subida. Conforme demonstrado na figura, o andar requisitado é o 10. O algoritmo selecionou o terceiro elevador, pois este se encontrava mais próximo (parado no andar 7).

Observa-se na forma de onda que os sinais representativos dos estados da PC ligam e desligam sequencialmente, em sincronia com o processamento da Parte Operativa. Adicionalmente, verifica-se que o sinal `solicit_enable` opera atrelado ao `enable_dispatch`, garantindo o envio correto do comando.

B. Cenário 2: Chamada Única de Descida

O segundo caso é semelhante ao primeiro, porém o vetor `r_next_floor` processa uma chamada de descida. A estabilidade da transição de estados se mantém, validando a lógica para ambas as direções.

C. Cenário 3: Chamadas Simultâneas

O terceiro cenário apresenta uma condição mista, com requisitos de chamada subindo e descendo ocorrendo simultaneamente. O sistema prioriza e processa as chamadas sequencialmente, mantendo a integridade lógica do primeiro caso até que a PC retorne ao estado *Idle*.

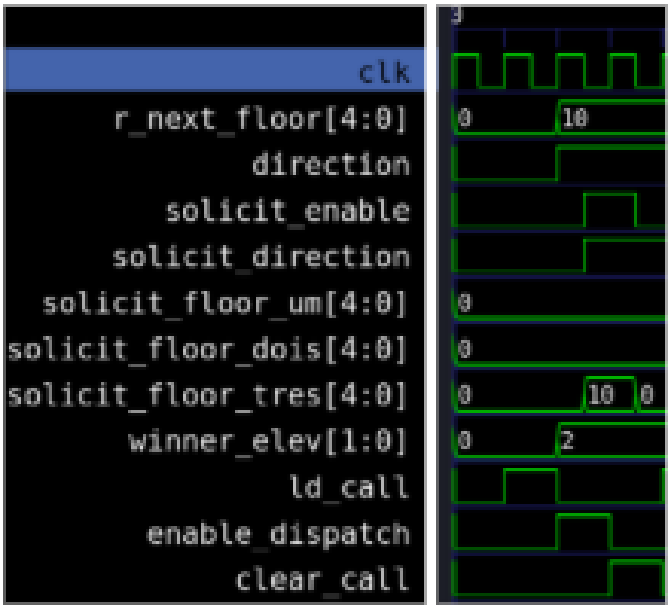

Fig. 2. Resultados do primeiro caso da simulação

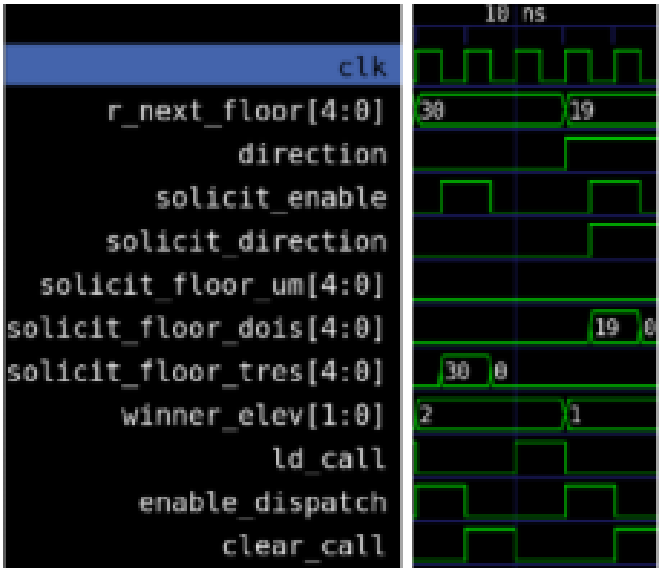

Fig. 3. Resultados do terceiro caso da simulação

D. Cenário 4: Múltiplas Chamadas

O quarto caso submete o sistema a múltiplas chamadas de subida e descida. O funcionamento segue o padrão dos casos anteriores, alternando entre as direções e repetindo o ciclo de limpeza (*Clear Call*) várias vezes, até que todas as requisições sejam atendidas e o sistema se estabilize em *Idle*.

VI. Conclusão

Como foi preciso apenas adaptar o nível 2 do último projeto, a implementação da parte operativa e a parte de controle não trouxe muitas dificuldades, pois a maioria dos componentes já estavam completados e os que precisavam ser

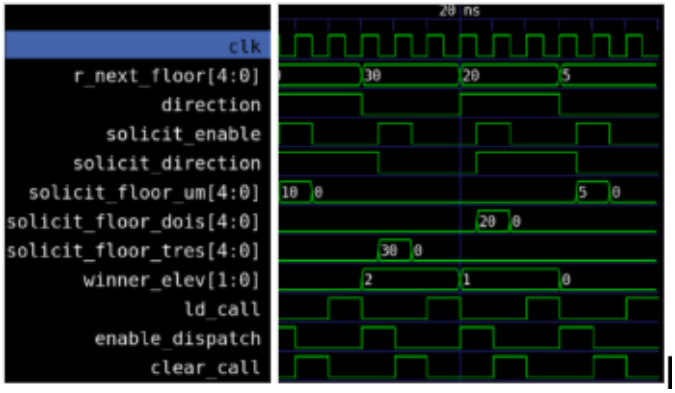

Fig. 4. Resultados do quarto caso da simulação

adicionados eram simples, com exceção do *sweeper* e *external calls register*, que necessitavam de lógicas específicas e únicas.

Ao geral, dividir o funcionamento em parte operativa e parte de controle facilitou o processo de criação, depuração e modificação do código, pois garantindo que as partes funcionavam, o nível 2 todo funcionava.

Referências

[1] R. D. Peters, "Elevator dispatching," in *Elevator Technology 9, Proc. of ELEVCON '98*, 1998, pp. 126–135.

[2] L. Al-Sharif, "Introduction to elevator group control," *Lift and Escalator Library*, 2016. [Online]. Disponível: liftescalatorlibrary.org.

[3] S. T. Imrak, A. G. Gungor, and R. M. Leblebicioglu, "Determination of the next stopping floor in elevator traffic control by means of neural networks," *Elevator World*, vol. 53, no. 4, pp. 140–146, 2005.

[4] P. Cortes, J. Larrañeta, and L. Onieva, "Genetic algorithm for controller in elevator groups analysis," in *Proc. of the 10th IFAC Symposium on Transportation Systems*, vol. 36, no. 14, 2003, pp. 191–196.