MÁQUINAS DE ESTADOS FINITOS COMO ESTRATÉGIA DIDÁTICA NO ENSINO DE FPGA APLICADA A UM ELEVADOR DIDÁTICO[1]

FINITE STATE MACHINES AS AN FPGA TEACHING STRATEGY APPLIED TO AN EDUCATIONAL ELEVATOR

Lucas Duarte[2]

Resumo: O ensino de linguagens de descrição de *hardware* e programação concorrente apresenta desafios significativos em cursos de Engenharia Elétrica, frequentemente resultando em elevada carga cognitiva e desmotivação dos estudantes. Para reduzir esse problema, este artigo propõe uma metodologia de aprendizagem incremental baseada no desenvolvimento de um controlador de elevador utilizando Máquinas de Estados Finitos. O projeto foi implementado em VHDL e validado no kit de desenvolvimento Terasic DE10-Lite. O trabalho contou ainda com o suporte de um protótipo físico para testes de acionamento de potência e condicionamento de sinais. A metodologia, inspirada em princípios de engenharia reversa, dividiu o sistema em seis etapas progressivas, partindo de um movimento autônomo básico até alcançar um sistema modular com lógica de despacho e intertravamentos de segurança inspirados em aplicações industriais. O particionamento lógico do código facilitou o isolamento de falhas e a simplificação do processo de depuração. Conclui-se que a decomposição de projetos complexos em módulos testáveis apresenta potencial didático para a consolidação do raciocínio paralelo e aproxima a teoria digital da prática laboratorial.

Palavras-chave: FPGA; VHDL; Máquina de Estados Finitos; Aprendizagem Incremental; Ensino de Engenharia.

Abstract: The teaching of hardware description languages and concurrent programming presents significant challenges in Electrical Engineering courses, often resulting in high cognitive load and student demotivation. To mitigate this problem, this paper proposes an incremental learning methodology based on the development of an elevator controller using Finite State Machines. The project was implemented in VHDL and validated on the Terasic DE10-Lite development kit. The work was also supported by a physical prototype for tests involving power actuation and signal conditioning. The methodology, inspired by reverse engineering principles, divided the system into six progressive stages, starting with basic autonomous movement and progressing toward a modular system with industrial-inspired safety interlocks and dispatch logic. The logical partitioning of the code facilitated fault isolation and simplified the debugging process. It is concluded that the decomposition of complex projects into testable modules shows didactic potential for consolidating parallel reasoning and bringing digital theory closer to laboratory practice.

Keywords: FPGA; VHDL; Finite State Machine; Incremental Learning; Engineering Education.

---

[1] Artigo apresentado ao curso de Bacharelado em Engenharia Elétrica do Instituto Federal de Santa Catarina, para obtenção do título de bacharel em Engenharia Elétrica 2026. Orientador: Prof. Michael Klug, Doutor.

[2] Discente do curso de Bacharelado em Engenharia Elétrica do Instituto Federal de Santa Catarina. lucas.d07@aluno.ifsc.edu.br.

1 INTRODUÇÃO

O ensino de sistemas digitais modernos em cursos de Engenharia Elétrica baseia-se fortemente no uso de dispositivos de lógica programável, conhecidos como FPGAs (*Field Programmable Gate Arrays*), e linguagens de descrição de *hardware* (*Hardware Description Languages* - HDLs), com destaque para o VHDL (VHSIC *Hardware Description Language*), em que VHSIC corresponde a *Very High Speed Integrated Circuit*. Contudo, a transição do paradigma de programação sequencial, típico de microcontroladores tradicionais, para o modelo de pensamento concorrente e paralelo inerente ao *hardware* digital exige um esforço cognitivo elevado por parte dos estudantes nos semestres iniciais (PEDRONI, 2004; WAKERLY, 2005). Esta mudança abrupta de percepção lógica frequentemente resulta em desmotivação e dificuldades na retenção de conceitos práticos de circuitos síncronos durante as atividades laboratoriais.

Como alternativa para reduzir essa barreira pedagógica e aproximar a teoria da prática, o desenvolvimento de projetos práticos contextualizados baseados na metodologia de Aprendizagem Baseada em Projetos (*Project-Based Learning* - PBL) atua como um importante elemento facilitador (PRINCE, 2004). Entre os estudos de caso aplicáveis ao ensino técnico, o controle de um sistema de elevador destaca-se por apresentar uma sequência intuitiva de estados, eventos discretos bem definidos e proximidade com a vivência cotidiana do estudante. A complexidade do sistema completo, no entanto, pode sobrecarregar o aluno se introduzida de forma integral, demandando estratégias de particionamento lógico.

Para contornar esse obstáculo, este artigo propõe uma metodologia de aprendizagem incremental inspirada em princípios de engenharia reversa. O sistema de controle de um elevador de quatro andares é decomposto em uma sequência progressiva de seis etapas de complexidade. A abordagem parte de uma lógica de movimentação autônoma básica (Etapa 1) e evolui gradualmente até a consolidação de um sistema modular integrado (Etapa 6), que incorpora máquinas de estados finitos (*Finite State Machines* - FSMs) para o controle da cabine e da porta, memória de registro de chamadas e um algoritmo síncrono de decisão para o despacho de rotas, permitindo uma escrita de código mais estruturada e eficiente (JASINSKI, 2016).

A validação da estrutura proposta foi realizada por meio de síntese lógica, análise RTL e validação funcional utilizando o *software* Intel Quartus Prime Lite Edition, com posterior programação do FPGA no kit de desenvolvimento Terasic DE10-Lite, baseado no FPGA Intel MAX 10 (TERASIC, 2016; MAXFIELD, 2004). A fim de estreitar a relação entre a abstração do código e o comportamento eletromecânico real, utilizou-se também um protótipo físico em maquete funcional, acionado por circuitos de potência e sensores de presença. O foco deste trabalho reside na análise qualitativa do potencial didático dessa divisão modular, avaliando como o isolamento de falhas em blocos independentes pode simplificar o processo de depuração (*troubleshooting*) para estudantes em fase inicial de contato com VHDL e FPGAs.

A estrutura deste artigo reflete diretamente a progressão da proposta metodológica. Na seção seguinte, apresenta-se a fundamentação teórica sobre o comportamento síncrono e a modelagem de máquinas de estados em *hardware* digital. Os critérios de particionamento lógico das seis etapas incrementais e os detalhes de integração com a maquete física são descritos na terceira seção. Posteriormente, a quarta seção dedica-se à análise dos resultados funcionais e aos

cenários práticos de depuração no *hardware*, enquanto as considerações finais sintetizam as contribuições da proposta e indicam possíveis aprimoramentos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, são apresentados os conceitos que estruturam o desenvolvimento deste projeto. O texto aborda inicialmente a tecnologia da FPGA e a natureza concorrente da linguagem VHDL, avançando para a teoria das Máquinas de Estados Finitos. Em seguida, discute-se a aplicação da aprendizagem incremental e das metodologias ativas para reduzir a carga cognitiva no ensino de engenharia. Por fim, a seção detalha o modelo lógico do elevador, utilizado como estudo de caso, ilustrando como as regras de controle e os requisitos de segurança operam em um sistema orientado a eventos.

2.1 FPGA

Os dispositivos lógicos programáveis transformaram a abordagem de projetos eletrônicos. Diferentemente de microcontroladores tradicionais, que em geral executam instruções de forma sequencial, a descrição em HDL destinada a um FPGA é sintetizada em estruturas digitais que operam de maneira paralela no *hardware* (MAXFIELD, 2004). Na prática do laboratório, o foco do estudante muda: ele deixa de se preocupar apenas com o fluxo do algoritmo e passa a considerar a alocação de recursos lógicos, registradores, interconexões e pinos físicos do dispositivo. Essa dinâmica exige a compreensão de como os blocos lógicos programáveis, compostos por tabelas de pesquisa (LUTs, do inglês *Look-Up Tables*), registradores e interconexões configuráveis, são utilizados para mapear funções booleanas e implementar circuitos digitais (TOCCI; WIDMER; MOSS, 2011). Trata-se de uma transição mental abrupta, que exige ferramentas de ensino específicas para a compreensão do paralelismo do *hardware* (JASINSKI, 2016; SASS; SCHMIDT, 2010).

Além da flexibilidade de prototipagem, a adoção tecnológica de FPGAs justifica-se pela sua capacidade de processamento paralelo, essencial para sistemas que demandam baixa latência. Essa característica torna o domínio desta tecnologia indispensável para o desenvolvimento de soluções complexas de controle e processamento de sinais digitais (MONMASSON et al., 2011).

2.2 LINGUAGEM VHDL

A barreira inicial no aprendizado de FPGAs costuma estar associada à própria forma de descrição do *hardware*. O VHDL permite descrições estruturais, comportamentais e de fluxo de dados, mas sua lógica de funcionamento difere das linguagens de programação sequenciais. O grande desafio didático é fazer o aluno compreender que uma linha de código em VHDL não representa, necessariamente, uma instrução aguardando sua vez de execução, mas pode representar uma conexão, um registrador, uma função lógica ou um processo concorrente implementado no *hardware* (MAXFIELD, 2004; JASINSKI, 2016).

Pedroni (2004) evidencia a diferença entre *software* e *hardware* ao discutir o comportamento de variáveis e sinais em VHDL. Enquanto variáveis declaradas em processos são atualizadas de forma imediata dentro do fluxo sequencial do processo, os sinais (*signals*) representam conexões lógicas ou estados internos e,

no modelo de simulação da linguagem, têm suas atualizações efetivadas após ciclos de avaliação, conhecidos como *delta cycles*. Essa característica ajuda a explicar por que diferentes blocos de um projeto em VHDL podem operar de forma concorrente no *hardware*. É essa concorrência estrutural que permite a um sistema embarcado monitorar botões externos, acionar circuitos de potência e atualizar interfaces visuais simultaneamente, sem depender de um único fluxo sequencial de execução.

2.3 MÁQUINAS DE ESTADOS FINITOS (FSM)

Para controlar de forma segura e previsível um sistema em que várias ações acontecem ao mesmo tempo, as Máquinas de Estados Finitos assumem o papel de núcleo do projeto. Wakerly (2005) define a FSM como um modelo matemático fundamental para o design digital, pois ela avalia os sinais de entrada e o histórico do sistema para decidir qual será a próxima ação. No contexto da linguagem VHDL, o uso de máquinas de estados permite que a lógica de decisão seja separada do restante do código (PEDRONI, 2004). Essa organização é essencial para evitar que estímulos simultâneos nos botões levem o sistema a operar de forma incorreta ou a entrar em condições inválidas.

A literatura clássica divide as FSMs nos modelos de Mealy e Moore. Enquanto na máquina de Mealy as saídas podem reagir de forma combinacional aos estímulos externos, o modelo de Moore estabelece que as saídas dependam apenas do estado atual. Pedroni (2010) destaca que, no projeto de sistemas digitais, a adoção de FSM do tipo Moore, quando associada ao registro das saídas e à sincronização das entradas através do sinal de *clock*, contribui para maior previsibilidade das saídas e reduz a ocorrência de transições indesejadas associadas a *glitches* combinacionais. Na prática de um protótipo físico de elevador, essa estabilidade é necessária, pois reduz o risco de transições indevidas nos sinais de controle. Assim, quando combinada com sincronização de entradas, registro de saídas, intertravamentos e temporizações de segurança, ou *dead-time*, a FSM contribui para evitar comandos conflitantes de "subir" e "descer" na Ponte H.

2.4 APRENDIZAGEM INCREMENTAL E METODOLOGIAS ATIVAS

A quantidade de novos conceitos exigida pela combinação de *hardware* reconfigurável, concorrência e temporização costuma sobrecarregar estudantes de engenharia, especialmente quando os tópicos são apresentados de forma puramente matemática (SWELLER; AYRES; KALYUGA, 2011). Ao investigar o uso de metodologias ativas no ensino de engenharia, Prince (2004) conclui que a adoção da aprendizagem baseada em projetos ajuda a conectar a teoria abstrata ao raciocínio aplicado, melhorando a compreensão dos alunos.

A proposta deste trabalho aplica esse conceito por meio do particionamento lógico do sistema. Em vez de confrontar o aluno com a escolha de rotas, prioridades e sensores simultaneamente, o desenvolvimento é dividido em camadas incrementais (SASS; SCHMIDT, 2010). A cada nova estrutura inserida no VHDL, torna-se possível obter uma validação física observável na placa DE10-Lite ou no protótipo mecânico. Essa estratégia isola as potenciais zonas de erro, transformando o processo de depuração em uma investigação focada e podendo reduzir a frustração comum no aprendizado inicial da linguagem (RAJ, 2018).

2.5 O MODELO DO ELEVADOR COMO SISTEMA ORIENTADO A EVENTOS

Para projetar um controlador digital, é necessário compreender o funcionamento do objeto de estudo e identificar quais condições físicas devem ser representadas pela lógica de controle. O sistema de um elevador opera por meio da coordenação de componentes integrados, como a cabine de transporte, o motor de tração, as portas automatizadas, os sensores de posição e os botões de chamada (NISE, 2012). Embora o movimento mecânico da cabine seja contínuo durante a subida ou a descida, o controlador digital interpreta o funcionamento do sistema por meio de eventos discretos, como o acionamento de uma chamada, a detecção de um pavimento por sensor ou a confirmação de fechamento da porta.

Dessa forma, o elevador pode ser modelado como um sistema discreto orientado a eventos, no qual condições relevantes de operação são associadas a estados lógicos e as mudanças de comportamento ocorrem a partir de critérios de transição bem definidos (HARRIS; HARRIS, 2012). Essa característica torna o sistema adequado à utilização de Máquinas de Estados Finitos, pois permite representar separadamente a condição atual do controlador, as regras responsáveis pela definição do próximo estado e os comandos associados à operação dos atuadores.

Conforme discutido, a estrutura de Moore é adequada para essa aplicação porque as saídas de movimentação são associadas ao estado atual da máquina (WAKERLY, 2005). Em implementações que envolvem dispositivos físicos, essa separação, em conjunto com a sincronização das entradas e mecanismos de filtragem de sinais, favorece maior previsibilidade no acionamento dos atuadores (BROWN; VRANESIC, 2009). No controlador desenvolvido, essa organização permite que comandos de subida, descida e parada sejam relacionados às condições operacionais representadas pela FSM, reduzindo a dependência direta das saídas em relação a variações momentâneas das entradas.

Além do transporte básico, a lógica de um elevador exige regras destinadas a impedir combinações inadequadas de comandos. Essas restrições podem ser implementadas por meio de intertravamentos lógicos, responsáveis por condicionar a execução de uma ação à confirmação de determinadas condições do sistema (TOCCI; WIDMER; MOSS, 2011). Conceitualmente, essa camada impede, por exemplo, que o movimento vertical seja autorizado sem a confirmação de fechamento da porta e permite que condições prioritárias, como emergência ou limite de carga, se sobreponham às ordens normais de atendimento. O sistema também incorpora uma temporização de segurança antes da reversão do sentido do motor, estabelecendo um intervalo de parada entre comandos opostos de movimentação (ROTH JR.; JOHN, 2008).

Embora o protótipo desenvolvido não reproduza a escala de potência, a redundância dos circuitos de proteção ou os requisitos de certificação presentes em elevadores comerciais, o modelo de controle preserva relações funcionais encontradas em sistemas de transporte vertical. Entre essas relações estão o registro de chamadas, a identificação da posição da cabine por pavimento, o atendimento condicionado à posição e ao sentido de deslocamento, a coordenação entre o controle da cabine e o controle da porta e a utilização de intertravamentos para restringir o movimento diante de condições inadequadas.

No controlador implementado, a confirmação de porta fechada atua como requisito para o movimento vertical, a detecção de obstrução interfere na sequência de fechamento da porta e as condições de emergência e limite de peso possuem prioridade sobre as ordens normais de despacho. Além disso, a lógica de

estacionamento prevê o retorno da cabine ao térreo após um período de ociosidade. Dessa forma, a proximidade entre o modelo didático e um elevador real está concentrada nas relações funcionais de controle, no encadeamento dos comandos e na interação entre sensores, decisões lógicas e atuadores, e não na equivalência normativa ou construtiva com um equipamento comercial.

Para organizar essas relações de forma estruturada, adota-se uma arquitetura modular. O sistema é dividido em blocos com funções específicas: uma FSM principal para o controle do movimento da cabine, uma FSM secundária dedicada à porta, registradores de memória para armazenar as chamadas e um módulo de decisão, denominado despachante, responsável por avaliar as requisições, a posição da cabine e o sentido de deslocamento. Essa separação conceitual mantém o código organizado, favorece o isolamento de falhas e constitui o alicerce técnico para a abordagem didática incremental apresentada na metodologia.

3 METODOLOGIA E IMPLEMENTAÇÃO

Esta seção detalha as escolhas tecnológicas e as estratégias pedagógicas adotadas para o desenvolvimento do controlador de elevador baseado em Máquinas de Estados Finitos. Para apoiar o ensino prático de programação concorrente em VHDL, a metodologia estruturou-se em três pilares fundamentais. Primeiramente, define-se a infraestrutura de *hardware* e *software* utilizada como base do processamento lógico. Em seguida, detalha-se a integração do algoritmo com o protótipo, etapa que introduz desafios reais de engenharia, como o condicionamento de sensores e o acionamento de potência. Por fim, apresenta-se a abordagem didática inspirada na engenharia reversa, em que o sistema completo é particionado em uma sequência incremental de etapas definida conforme sua complexidade. Para o estudo de caso do elevador, essa aplicação resultou em seis etapas progressivas.

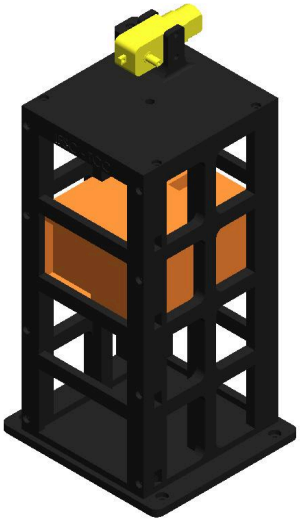
3.1 INFRAESTRUTURA DE DESENVOLVIMENTO

A execução deste projeto utilizou como plataforma central o kit de desenvolvimento Terasic DE10-Lite, equipado com a FPGA Intel MAX 10 (10M50DAF484C7G). A escolha desta placa justifica-se pela vasta disponibilidade de periféricos integrados, como chaves deslizantes (*switches*), botões de pressão e visores (*displays*) de sete segmentos, que permitem a validação visual da lógica sem a necessidade inicial de circuitos externos. Para a descrição do hardware e a síntese lógica, empregou-se o software Intel Quartus Prime Lite Edition. Além da compilação modular do código VHDL, esse ambiente foi utilizado para o mapeamento físico dos sinais de entrada e saída por meio da ferramenta *Pin Planner*, permitindo a correta interface entre a descrição lógica do projeto e os pinos físicos da FPGA.

3.2 INTEGRAÇÃO COM O MODELO FÍSICO (PROTÓTIPO)

Antes da integração elétrica e lógica, o protótipo físico foi planejado em ambiente CAD 3D, permitindo visualizar a estrutura mecânica, a disposição da cabine e os principais elementos de movimentação. A modelagem utilizada como referência para essa etapa é apresentada na Figura 1.

Figura 1 – Modelagem do protótipo



Fonte: elaborado pelo autor.

Esta etapa de projeto mecânico permitiu o dimensionamento das guias lineares, o posicionamento das roldanas e do motor DC de tração. A prototipagem virtual contribuiu para reduzir erros de montagem e para adequar o posicionamento mecânico da cabine às necessidades de leitura dos sensores e acionamento do controle em VHDL. Para compor essa integração de hardware, foram empregados os componentes detalhados na Figura 2.

Figura 2 – Componentes utilizados na integração eletrônica do protótipo

(a) Kit Terasic DE10-Lite (b) Sensor TCRT5000 (c) Módulo Ponte H L298N

Fonte: adaptado de Terasic ([s. d.]), Viana (2022) e Viana (2023).

Além da validação em placa, o projeto contempla a integração com um protótipo físico de elevador, que atua como uma ferramenta didática complementar. O controle do motor principal de elevação da cabine é realizado por meio de um módulo de potência (Ponte H), que permite controlar a direção e a parada do motor por sinais digitais da FPGA. Já o mecanismo da porta é acionado de forma independente por um servomotor.

A detecção de posição da cabine e o monitoramento dos andares são executados por sensores ópticos reflexivos infravermelhos (modelo TCRT5000) instalados em cada pavimento. Diferente da simulação puramente lógica, o uso do protótipo físico exige que o estudante lide com o condicionamento de sinais e com instabilidades de leitura associadas ao ambiente físico e à comutação dos atuadores, favorecendo o desenvolvimento de competências práticas de integração de hardware.

Para promover a compatibilidade elétrica entre a FPGA, a Ponte H e os sensores, a interface de pinos de entrada e saída de propósito geral, ou GPIO, da placa foi configurada no padrão elétrico 3,3 V LVTTL. Na prática laboratorial, essa integração exige a unificação da referência de terra, ou GND, entre a placa de desenvolvimento, os sensores e a fonte de alimentação externa dos motores. Esse aterramento comum é fundamental para estabilizar os níveis lógicos e reduzir a ocorrência de leituras instáveis ou falsos disparos nos sensores ópticos.

3.3 ESTRATÉGIA INSPIRADA NA ENGENHARIA REVERSA E SEQUÊNCIA INCREMENTAL

A metodologia proposta baseia-se na desconstrução didática de um sistema completo. Partindo da complexidade global para a simplificação modular, em uma lógica inspirada na engenharia reversa, o conteúdo é organizado em uma sequência incremental na qual novas funcionalidades são introduzidas somente após a validação dos conceitos trabalhados anteriormente.

O número de etapas não constitui um parâmetro fixo da metodologia. A quantidade e o escopo de cada etapa devem ser definidos de acordo com a complexidade do sistema, os requisitos do projeto e os conceitos de hardware que se pretende isolar durante o processo de aprendizagem. Dessa forma, um sistema de menor complexidade pode demandar uma sequência mais curta, enquanto projetos com maior número de subsistemas ou interfaces podem exigir um particionamento mais detalhado.

Neste estudo de caso, a análise dos conceitos envolvidos no controle do elevador resultou na organização do desenvolvimento em seis etapas. Essa divisão permitiu introduzir progressivamente a estrutura básica da FSM, a interação com entradas físicas, a retenção síncrona de chamadas, a concorrência entre máquinas de estados, a separação da lógica de despacho e, por fim, os intertravamentos e a integração com o protótipo. A Tabela 1 apresenta as etapas definidas para esta aplicação da metodologia.

Tabela 1 – Síntese das etapas da metodologia incremental

Etapa	Função Adicionada	Conceitos de VHDL e hardware trabalhados	Validação
1	Base Visual e Movimento Autônomo	FSM contínua, divisores de *clock* e decodificador para *displays* de sete segmentos.	DE10-Lite
2	Interação Reativa	Leitura de entradas físicas (chaves) e interrupção do ciclo autônomo.	DE10-Lite
3	Registro de Chamadas	Elementos de memória síncrona (flip-flops) e retenção de dados temporários.	DE10-Lite
4	Controle da Porta	Instanciação paralela, múltiplas FSMs concorrentes e temporização de estados.	DE10-Lite
5	Lógica de Despacho e Decisão de Rotas	Algoritmo de decisão de rotas e separação entre módulo de controle e de execução.	DE10-Lite
6	Intertravamentos e Validação no Protótipo	Intertravamentos lógicos, tratamento de ruído (filtragem temporal) e lógica de proteção.	DE10-Lite + protótipo

Fonte: elaborado pelo autor.

A seguir, detalha-se o escopo estrutural de cada etapa, destacando os conceitos de VHDL inseridos na construção da máquina e os diagramas de transição de estados correspondentes.

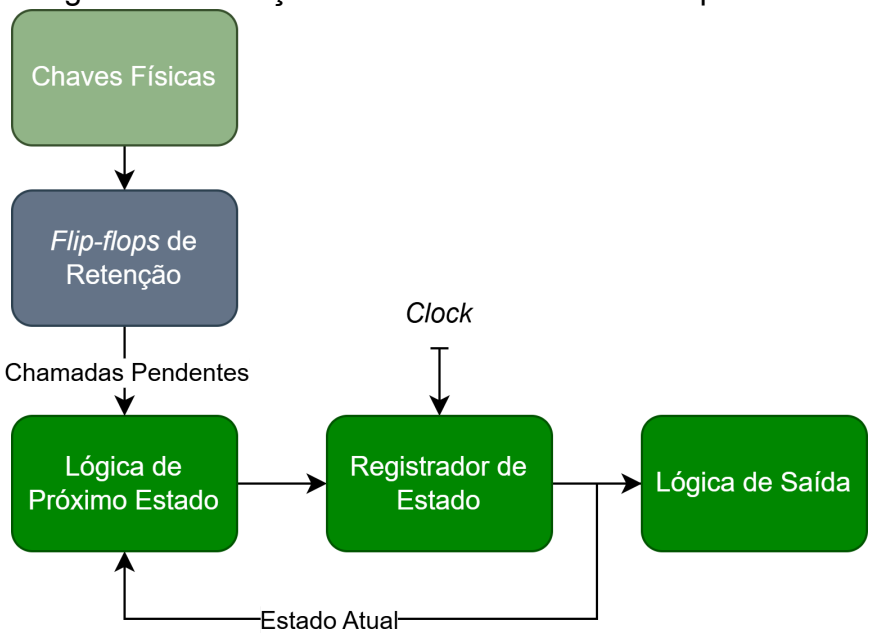
3.3.1 Etapa 1: Base Visual e Movimento Autônomo

Nesta etapa, o objetivo didático é introduzir os primeiros elementos estruturais de uma FSM em *hardware*: o registrador de estado, a lógica de próximo estado e a lógica de saída. O estudante compreende que a máquina atua sobre um bloco de memória, baseado em *flip-flops*, responsável por armazenar a condição atual do sistema. A partir desse estado armazenado, a lógica de próximo estado define qual será a condição seguinte da máquina, sendo as transições atualizadas de forma estritamente síncrona a cada borda ativa do sinal de *clock*.

Como a primeira etapa ainda não depende de eventos externos, a FSM é construída com uma sequência autônoma de estados, representando a passagem ordenada da cabine pelos pavimentos simulados. Dessa forma, o estudante observa que o comportamento do circuito não resulta de uma sequência de comandos executados por *software*, mas da atualização sucessiva de estados em *hardware*.

Em paralelo, implementa-se a lógica de saída baseada no modelo de Moore. Nesse modelo, a ativação dos periféricos depende do estado atual armazenado no registrador, o que isola as saídas de variações instantâneas nas entradas e favorece a estabilidade necessária para o acionamento de componentes físicos. Como saídas da FSM, são acionados decodificadores lógicos para os *displays* de sete segmentos da placa DE10-Lite, permitindo a visualização da transição contínua entre os pavimentos simulados. O estudante valida o comportamento do circuito observando o sequenciamento autônomo da lógica implementada. Essa estrutura inicial, formada pelo registrador de estado, pela lógica de próximo estado e pela lógica de saída, constitui o núcleo sobre o qual são inseridas as entradas físicas e a memória de chamadas nas etapas seguintes. A evolução estrutural correspondente às Etapas 1 a 3 é sintetizada na Figura 3.

Figura 3 – Evolução estrutural da FSM nas etapas 1 a 3



Fonte: elaborado pelo autor.

3.3.2 Etapa 2: Interação Reativa e a Ausência de Retenção

Preservando a estrutura síncrona do registrador de estado, esta fase rompe o ciclo autônomo para introduzir a leitura de estímulos externos. O bloco combinacional da lógica de próximo estado, que antes operava de forma isolada, passa a avaliar não apenas o estado atual retroalimentado, mas também os níveis lógicos das chaves físicas da placa. No código VHDL, isso se traduz na inserção de estruturas de decisão vinculadas às portas de entrada mapeadas.

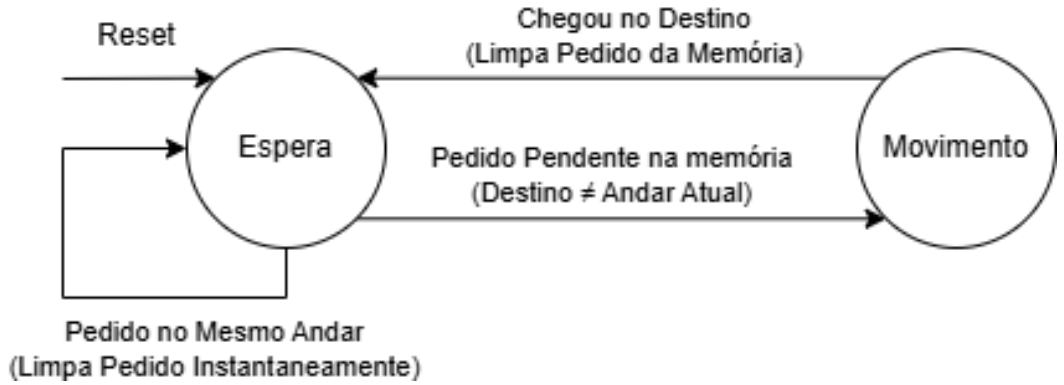
O sistema torna-se reativo à condição presente da entrada: a lógica de próximo estado define a condição de movimento apenas enquanto a chave externa estiver ativada. Caso a chave seja liberada, a condição deixa de ser verdadeira e o bloco combinacional define o retorno ao estado de parada, atualizado no ciclo de *clock* seguinte. Conforme representado na Figura 3, as entradas físicas passam a fornecer novas condições à lógica de próximo estado, mas ainda não existe um bloco responsável por armazenar as requisições. Essa implementação permite ao estudante identificar a diferença entre um controle reativo sem retenção de chamada e um sistema que necessita de memória adicional.

3.3.3 Etapa 3: Registro de Chamadas e Atendimento

A terceira etapa soluciona a limitação reativa ao introduzir blocos de memória dedicados externos ao núcleo da FSM principal. No ambiente de desenvolvimento, o estudante compreende que o sistema necessita de registradores adicionais, baseados em flip-flops, para operar em paralelo com a FSM. Esses blocos auxiliares memorizam o pulso momentâneo dos botões e mantêm as solicitações de chamada ativas. A incorporação dessa memória à estrutura desenvolvida nas etapas anteriores é representada na Figura 3.

Com essa adição, a lógica de próximo estado da máquina principal deixa de avaliar diretamente as chaves e passa a ler as saídas dos registradores de chamadas. Quando a cabine atende ao respectivo pavimento, a lógica de controle emite um sinal síncrono para limpar exclusivamente a solicitação correspondente. O funcionamento específico do registro, retenção e atendimento das chamadas é detalhado na Figura 4. Do ponto de vista pedagógico, separar o armazenamento de dados do núcleo de controle de estados introduz princípios de hierarquia e organização em hardware digital.

Figura 4 – Diagrama Lógico da Etapa 3



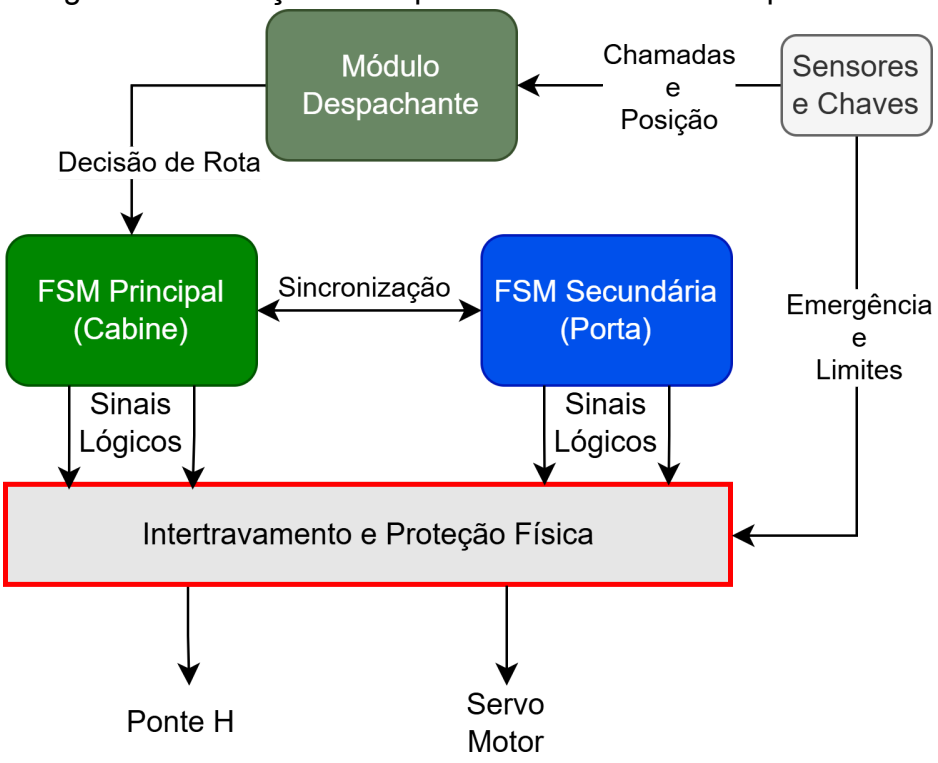
Fonte: elaborado pelo autor.

3.3.4 Etapa 4: Concorrência e Varredura por Prioridade

Com o registro de chamadas consolidado, a metodologia avança para demonstrar a concorrência característica das FPGAs. Diferentemente de uma implementação sequencial em software, na qual tarefas podem ficar condicionadas à ordem de execução do programa ou a atrasos bloqueantes, a arquitetura de hardware permite a operação simultânea de múltiplos circuitos síncronos.

Nessa etapa, adiciona-se uma segunda FSM dedicada ao controle da porta, com estados e temporizações próprios. Enquanto a FSM principal controla o deslocamento da cabine, a máquina secundária administra a sequência de abertura e fechamento da porta. Essa inserção marca o início da arquitetura modular desenvolvida nas Etapas 4 a 6, cuja evolução é sintetizada na Figura 5.

Figura 5 – Evolução da arquitetura modular nas etapas 4 a 6



Fonte: elaborado pelo autor.

A decisão de rotas, entretanto, ainda utiliza nessa fase uma cadeia condicional baseada em prioridade fixa. Essa limitação é mantida de forma intencional na sequência didática, permitindo ao estudante observar que a concorrência estrutural do hardware não elimina a necessidade de organizar adequadamente a lógica de decisão.

3.3.5 Etapa 5: Lógica de Despacho e Decisão de Rotas

A quinta etapa busca superar a limitação da varredura linear por meio da integração de componentes estruturais e da separação de responsabilidades. Nessa fase, o sistema passa a ser organizado em blocos com funções distintas, separando o circuito responsável pela execução do movimento daquele dedicado à decisão de rotas. Para isso, insere-se um módulo lógico combinacional denominado despachante.

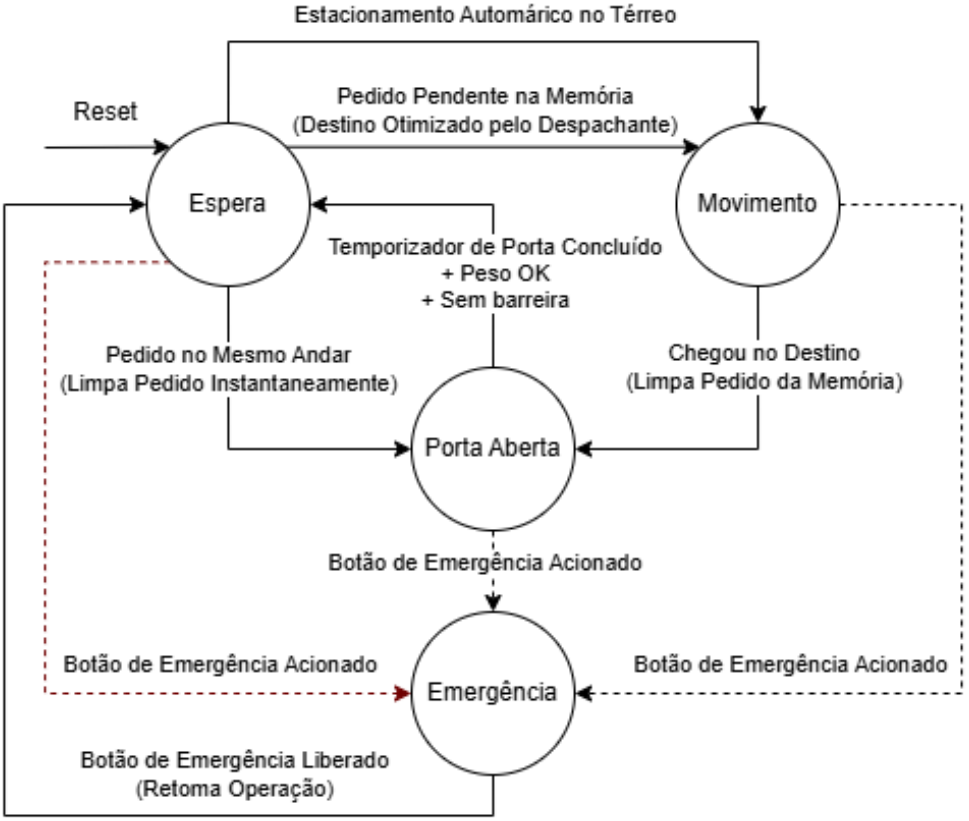
Esse bloco recebe as chamadas armazenadas nos registradores, a posição atual da cabine e o sentido do movimento, processando essas informações para gerar sinais que indicam à FSM principal a existência de uma requisição válida e a direção a ser seguida. Na arquitetura apresentada na Figura 5, o módulo despachante passa a ocupar a interface de decisão entre as chamadas e a FSM principal. Ao delegar o cálculo de rotas a um componente externo, simplifica-se a lógica de próximo estado da máquina responsável pelo movimento e reforça-se a organização hierárquica do projeto.

3.3.6 Etapa 6: Intertravamentos e Validação do Protótipo

A sexta etapa integra a lógica estruturada ao modelo físico, adicionando uma camada de intertravamentos inspirada em aplicações industriais. No contexto do projeto de hardware, as regras de proteção funcional são incorporadas à lógica de controle como critérios prioritários de bloqueio e autorização de movimento. Conforme representado na arquitetura da Figura 5, essa camada passa a atuar entre os sinais de controle produzidos pelas FSMs e o acionamento dos atuadores físicos.

Os intertravamentos associados ao botão de emergência, à confirmação de porta fechada e às condições de limite passam a ter prioridade sobre as ordens normais provenientes da lógica de despacho. Também são consideradas temporizações de segurança antes da inversão do motor, utilizando estados intermediários para evitar comandos abruptos aos atuadores. O comportamento específico dessas condições de proteção e sua prioridade em relação ao fluxo normal do sistema são detalhados na Figura 6.

Figura 6 – Lógica de intertravamentos e proteção da Etapa 6



Fonte: elaborado pelo autor.

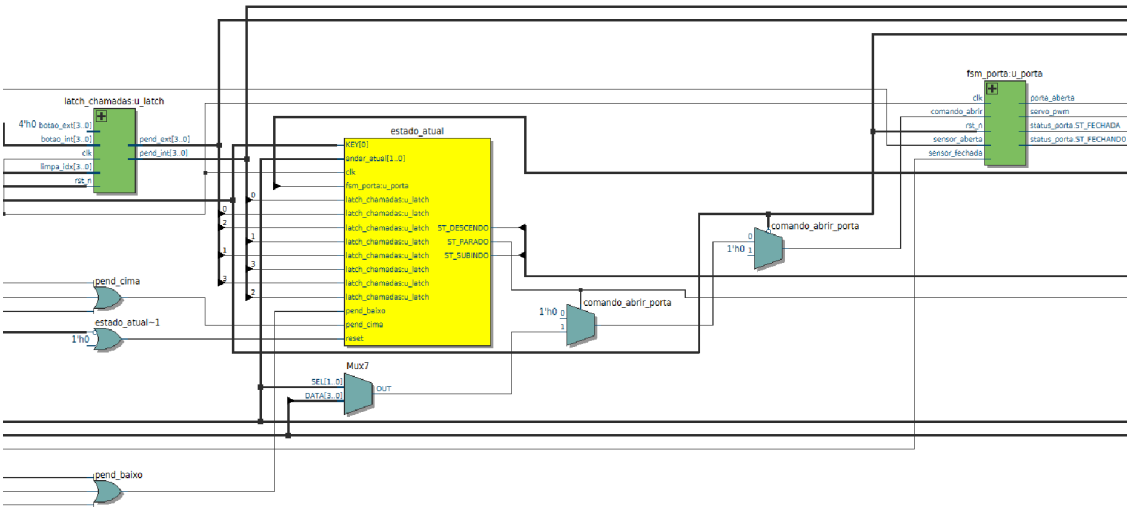
4 RESULTADOS E DISCUSSÕES

A proposta deste artigo é estabelecer um referencial didático e técnico para a aplicação do projeto em disciplinas de sistemas digitais. Portanto, os resultados aqui discutidos concentram-se na validação da implementação estrutural em *hardware*, na resolução de problemas físicos de integração e na observação qualitativa do potencial didático proporcionado pelo particionamento lógico.

4.1 VALIDAÇÃO ESTRUTURAL

A compilação e a síntese lógica do projeto completo, correspondente à Etapa 6, foram realizadas no *software* Intel Quartus Prime Lite Edition. A ferramenta RTL Viewer permitiu visualizar a organização estrutural do projeto em blocos distintos, associados à máquina de estados principal, à máquina da porta e ao módulo despachante, interligados por sinais de controle e decisão, conforme apresentado na Figura 7.

Figura 7 – Recorte do RTL *Viewer* da Etapa 6



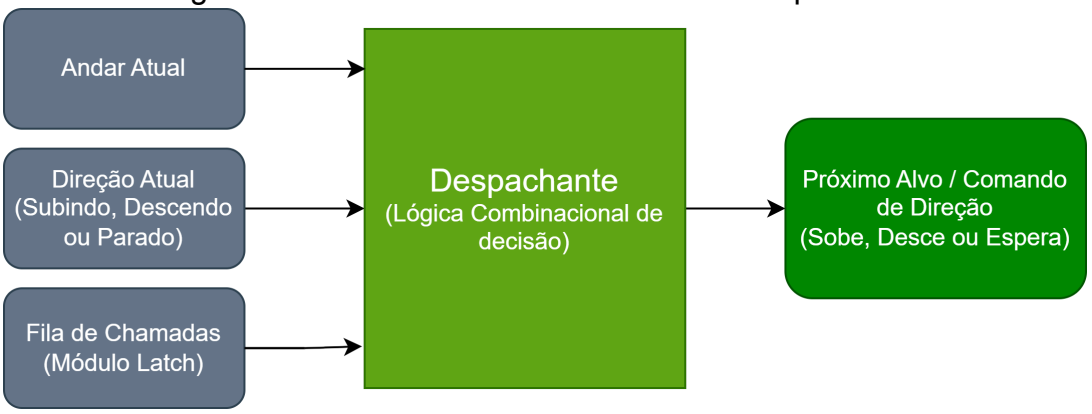
Fonte: elaborado pelo autor.

Essa visualização indica que a divisão do código VHDL não se limita à organização textual do arquivo, refletindo-se também na estrutura lógica sintetizada para a FPGA. A separação entre a máquina de estados principal, a máquina da porta e o módulo despachante favorece o estudo compartimentado do sistema, permitindo que cada bloco seja analisado, modificado e depurado de forma isolada. Dessa forma, a validação estrutural reforça a viabilidade técnica da arquitetura proposta para uso em kits didáticos de FPGA.

4.2 PARALELISMO E LÓGICA DE DESPACHO

A organização modular adotada permitiu separar a execução dos estados de movimento da lógica responsável pela decisão de rotas. Enquanto a FSM principal controla as condições operacionais da cabine, o módulo despachante avalia continuamente as chamadas pendentes, a posição atual e o sentido de deslocamento. A relação entre essas entradas e a decisão fornecida à máquina principal é apresentada na Figura 8.

Figura 8 – Entradas e saídas do módulo despachante



Fonte: elaborado pelo autor.

O despachante foi implementado como um bloco de lógica combinacional sem clock próprio. Dessa forma, sua decisão é recalculada a partir das condições presentes em suas entradas, enquanto a atualização dos estados da FSM principal permanece condicionada ao clock do sistema. Essa separação permite distinguir a lógica de decisão da lógica de execução, evitando a concentração das regras de varredura de chamadas diretamente na máquina responsável pelo movimento.

Durante os testes práticos, observou-se que a FSM da porta realizava de forma independente a temporização configurada para abertura. Simultaneamente, a FSM principal permanecia responsável pelo controle da cabine e o módulo despachante continuava avaliando as chamadas pendentes e as condições utilizadas na decisão de rota. Esse comportamento evidencia a operação concorrente dos diferentes blocos implementados no FPGA.

Para organizar o atendimento das requisições sem concentrar todas as condições na FSM principal, as regras de decisão do módulo despachante foram definidas conforme apresentado na Tabela 2.

Tabela 2 – Regras de decisão do módulo despachante

Condição de Entrada	Decisão de Rota do Despachante
Cabine parada e há chamada acima	Altera direção para SUBIR.
Cabine parada e há chamada abaixo	Altera direção para DESCER.
Cabine subindo e há chamada pendente acima	Mantém direção de SUBIDA.
Cabine subindo e há chamadas apenas abaixo	Conclui ciclo atual e inverte a direção para DESCER.
Cabine descendo e há chamada pendente abaixo	Mantém direção de DESCIDA.
Cabine descendo e há chamadas apenas acima	Conclui o atendimento no sentido atual e inverte a direção para SUBIR.
Nenhuma chamada pendente registrada	Permanece no estado de ESPERA.

Fonte: elaborado pelo autor.

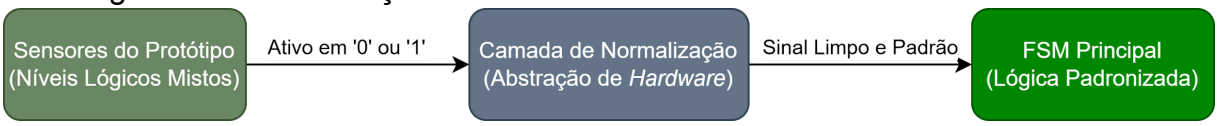
As regras adotadas preservam o sentido de deslocamento enquanto existirem solicitações pendentes na direção atual e permitem a inversão do movimento quando o atendimento nesse sentido é concluído. A utilização de um módulo específico para essa função reduz a quantidade de verificações inseridas diretamente na lógica de transição da FSM principal e reforça a separação de responsabilidades entre os blocos.

4.3 IMPACTO DO PROTÓTIPO NO CONDICIONAMENTO DE SINAIS

A transição do acionamento exclusivo dos displays da placa para o controle eletromecânico do protótipo exigiu adaptações, evidenciando as condições não ideais do ambiente físico. A integração demandou o mapeamento dos pinos e a adequação dos sinais provenientes dos sensores externos antes de sua utilização pelos módulos de controle.

Uma das estratégias adotadas consistiu na criação de sinais internos normalizados. Como os dispositivos físicos podem representar uma condição ativa em nível lógico alto ou baixo, a interpretação elétrica das entradas foi concentrada em uma camada anterior ao núcleo de controle. O fluxo dessa organização é apresentado na Figura 9.

Figura 9 – Normalização das entradas físicas antes do núcleo de controle



Fonte: elaborado pelo autor.

A camada de normalização converte os sinais provenientes do protótipo em representações lógicas padronizadas. Dessa forma, uma condição válida, como a detecção de um pavimento ou a confirmação da posição da porta, passa a ser representada internamente por nível lógico ativo independentemente da polaridade original do dispositivo empregado. Essa separação reduz o acoplamento entre as características elétricas dos sensores e as regras descritas nas FSMs.

No código implementado, constantes de configuração definem os níveis lógicos considerados ativos para os sensores de pavimento e de posição da porta. Assim, alterações no tipo ou na polaridade de uma entrada podem ser concentradas na interface de normalização, sem exigir a modificação das condições distribuídas pelo núcleo de controle.

Além da padronização lógica, a integração física evidenciou instabilidades de leitura associadas aos sensores e à comutação dos atuadores. Quando necessário, foram empregados mecanismos de sincronização e filtragem temporal para evitar que variações transitórias fossem interpretadas como eventos válidos pela lógica de controle. Também foram considerados intervalos de parada antes de comandos sucessivos de inversão do motor DC, adequando a resposta lógica às características do sistema eletromecânico.

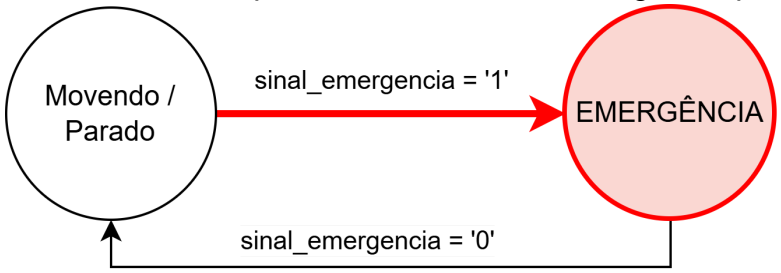
4.4 INTERTRAVAMENTOS E SEGURANÇA

A etapa final da integração incorporou condições prioritárias de proteção à lógica de controle. Diferentemente das chamadas de pavimento, que participam da

decisão normal de despacho, sinais associados à emergência, ao fechamento da porta e ao limite de peso atuam como restrições sobre a movimentação da cabine.

Entre essas condições, o sinal de emergência foi tratado por meio de um estado específico da FSM. Quando a condição é detectada, a máquina é direcionada ao estado `EMERGENCIA` e o comando de direção assume `DIR_PARADA`. Enquanto o sinal permanece ativo, o comando de movimentação é mantido bloqueado. Após sua liberação, a FSM retorna obrigatoriamente ao estado `PARADO` antes de voltar ao fluxo normal de atendimento, conforme apresentado na Figura 10.

Figura 10 – Tratamento prioritário do sinal de emergência pela FSM



Fonte: elaborado pelo autor.

O comportamento apresentado na Figura 10 representa uma das formas pelas quais as condições de proteção receberam prioridade sobre as ordens normais de despacho. A confirmação física de fechamento da porta também foi utilizada como requisito para autorizar transições aos estados de movimentação vertical. De forma semelhante, a ativação do limite de peso, simulada por uma entrada digital, bloqueia novas partidas enquanto a condição permanecer ativa.

A barreira óptica atua especificamente sobre a sequência de fechamento da porta. Quando uma obstrução é detectada durante esse processo, a FSM secundária altera sua sequência de estados e comanda a reabertura, evitando a continuidade do fechamento sob a condição detectada.

Para avaliar o comportamento integrado diante de chamadas simultâneas e condições de bloqueio, foram executados os cenários funcionais apresentados na Tabela 3.

Tabela 3 – Cenários de testes funcionais executados na maquete

Cenário de Teste	Comportamento Esperado	Resultado Observado
Chamada para andar superior durante descida	O sistema retém a chamada, mas conclui a rota prioritária em andamento.	Verificado. FSM priorizou o sentido atual.
Acionamento concomitante de múltiplos andares	O módulo despachante organiza o atendimento linear conforme o sentido da viagem.	Verificado. Lógica de despacho coletivo executada.
Obstrução da porta fechando	A FSM secundária comanda a reabertura da porta no ciclo de controle seguinte.	Verificado. Reversão efetuada sem comandos lógicos conflitantes.
Ativação do sensor de excesso de peso (simulado)	Sistema bloqueia partidas e força a manutenção do estado de porta aberta.	Verificado. Lógica de proteção inibiu o motor.

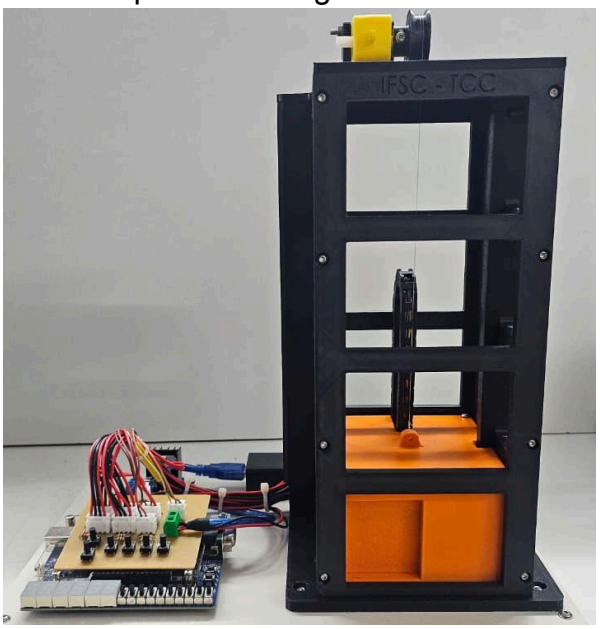
Fonte: elaborado pelo autor.

Os resultados observados indicaram que as regras prioritárias foram aplicadas conforme o comportamento definido para o protótipo. A separação entre lógica de despacho, FSM principal e FSM da porta permitiu que cada condição interferisse no bloco correspondente sem exigir a concentração de todos os tratamentos em uma única estrutura de controle.

4.5 CONTRIBUIÇÃO DIDÁTICA DO PROTÓTIPO FÍSICO

O uso do protótipo como extensão física do sistema de controle permitiu observar de forma concreta os efeitos das decisões lógicas implementadas em VHDL. Ao contrário da validação limitada aos LEDs e displays integrados à DE10-Lite, a maquete tornou visíveis o movimento da cabine, a atuação da porta e a resposta dos sensores, aproximando a abstração do código do comportamento eletromecânico observado na Figura 11.

Figura 11 – Protótipo físico integrado ao sistema de controle



Fonte: elaborado pelo autor.

A utilização do protótipo também favoreceu a identificação de limitações práticas que não se evidenciam com a mesma clareza em validações exclusivamente lógicas, como instabilidades de leitura, necessidade de filtragem digital e ajustes de temporização entre comandos sucessivos. Dessa forma, o protótipo desempenhou papel relevante não apenas como recurso ilustrativo, mas como instrumento de validação funcional e apoio didático à compreensão do comportamento concorrente do *hardware*.

4.6 O POTENCIAL DA METODOLOGIA NO ISOLAMENTO DE FALHAS

Uma evidência relevante em favor do particionamento lógico ocorreu durante a fase de validação técnica do sistema. Na Etapa 6, o projeto previa a inclusão de uma rotina automatizada em que a cabine deveria retornar ao andar térreo após um período determinado de ociosidade. Durante os ensaios em bancada, observou-se uma anomalia: o temporizador atingia o limite, mas a cabine permanecia estática.

Caso a arquitetura estivesse menos particionada, o diagnóstico demandaria o rastreamento integral de dezenas de sinais. Contudo, devido ao isolamento modular promovido pelas etapas anteriores, a causa foi identificada rapidamente: a FSM principal respondia adequadamente às chamadas externas, mas estava isolada do sinal interno gerado pelo temporizador. A correção consistiu apenas na inclusão do respectivo sinal na hierarquia de prioridades do módulo despachante, sem a necessidade de refatorar as lógicas consolidadas de controle do motor ou da porta.

Esse relato reforça a proposta central do artigo, pois evidencia que a construção de sistemas digitais em módulos favorece a limitação do escopo de análise durante a depuração. Ao restringir a investigação às interfaces recém-adicionadas, a metodologia pode contribuir para reduzir o esforço de identificação de falhas e favorecer a transição de um processo baseado em tentativa e erro para uma investigação lógica orientada por evidências.

5 CONSIDERAÇÕES FINAIS

O ensino de linguagens de descrição de *hardware* e o desenvolvimento em FPGAs impõem desafios cognitivos significativos devido à mudança do paradigma de programação sequencial para a execução concorrente. Diante desse cenário, este artigo apresentou uma abordagem didática baseada no particionamento lógico, utilizando o controle de um protótipo de elevador por Máquinas de Estados Finitos (FSMs) como estudo de caso.

A aplicação da metodologia ao estudo de caso foi organizada em seis etapas progressivas e evidenciou que a desconstrução da complexidade de um sistema digital favorece a clareza arquitetural. Ao isolar módulos com funções específicas, como o controle do movimento da cabine, a FSM da porta, o armazenamento síncrono de chamadas e a lógica de despacho, reduziu-se a interdependência direta entre as responsabilidades implementadas no código. O número de etapas, entretanto, não constitui uma característica fixa da metodologia, devendo ser adaptado à complexidade e aos conceitos envolvidos em cada projeto.

No tratamento de falhas, o particionamento lógico mostrou-se particularmente útil para delimitar o escopo da depuração. O caso observado no sistema de estacionamento automático demonstrou esse aspecto: embora o temporizador de ociosidade concluísse sua contagem, a cabine permanecia parada devido à ausência do respectivo sinal na lógica de prioridade do módulo despachante. Como as FSMs de movimento e da porta já estavam isoladas em módulos previamente validados, a investigação pôde ser concentrada na interface recém-integrada. Nesse contexto, a metodologia favoreceu um processo de *troubleshooting* orientado pela análise dos blocos e de suas interconexões, reduzindo a necessidade de uma busca ampla e não direcionada por falhas.

Além da abstração lógica, a integração do código VHDL com a maquete eletromecânica permitiu expor condições não ideais do ambiente físico. A necessidade de implementar adequações ao domínio de *clock*, como registradores de sincronização para tratar entradas assíncronas provenientes dos sensores ópticos, e temporizações de segurança, ou *dead-time*, para reduzir o risco de transitórios de corrente na comutação da Ponte H, aproximou a prática laboratorial das restrições encontradas em projetos de *hardware* digital. A síntese no *software* Quartus Prime Lite Edition permitiu verificar a viabilidade da arquitetura implementada, indicando que a abordagem modular é aplicável a kits didáticos de FPGA, como a placa DE10-Lite.

Como o presente trabalho limitou-se a uma análise qualitativa da viabilidade técnica da arquitetura modular proposta e do processo de integração, o estudo não mensurou numericamente o impacto pedagógico em sala de aula. Como propostas para trabalhos futuros, sugere-se a aplicação formal desta sequência incremental em turmas regulares de engenharia, visando à coleta de dados quantitativos sobre retenção de conhecimento, compreensão de concorrência em *hardware* e redução do tempo médio de depuração entre os estudantes. Adicionalmente, recomenda-se a expansão do projeto para incorporar lógicas de comunicação em rede e interfaces gráficas de supervisão.

Conclui-se, portanto, que o uso de FSMs, combinado a um fluxo de aprendizagem incremental e à validação em protótipo físico, apresenta potencial didático relevante. A metodologia organiza projetos digitais complexos em estruturas progressivas e auditáveis, auxiliando o estudante na transição da teoria abstrata para o domínio prático do desenvolvimento de *hardware* reconfigurável.

REFERÊNCIAS

BROWN, S.; VRANESIC, Z. **Fundamentals of Digital Logic with VHDL Design**. 3. ed. New York: McGraw-Hill, 2009.

HARRIS, D.; HARRIS, S. **Digital Design and Computer Architecture**. 2. ed. Burlington: Morgan Kaufmann, 2012.

JASINSKI, R. **Effective Coding with VHDL: Principles and best practice**. 1. ed. Cambridge: The MIT Press, 2016.

MAXFIELD, C. **The Design Warrior's Guide to FPGAs: Devices, Tools and Flows**. Burlington: Elsevier, 2004.

MONMASSON, E. et al. FPGAs in Industrial Control Applications. **IEEE Transactions on Industrial Informatics**, Nova York, v. 7, n. 2, p. 224-243, 2011.

NISE, N. S. **Engenharia de Sistemas de Controle**. 6. ed. Rio de Janeiro: LTC, 2012.

PEDRONI, V. A. **Circuit Design with VHDL**. Cambridge: The MIT Press, 2004.

PEDRONI, V. A. **Eletrônica digital moderna e VHDL**. Rio de Janeiro: Elsevier, 2010.

PRINCE, M. Does Active Learning Work? A Review of the Research. **Journal of Engineering Education**, Washington, v. 93, n. 3, p. 223-231, 2004.

RAJ, A. A. B. **FPGA-Based Embedded System Developer's Guide**. 1. ed. New York, USA: Taylor & Francis, CRC Press, 2018.

ROTH JR., C. H.; JOHN, L. K. **Digital Systems Design Using VHDL**. 2. ed. Boston: Thomson/Cengage Learning, 2008.

SASS, R.; SCHMIDT, A. **Embedded Systems Design with Platform FPGAs: Principles and Practices**. 1. ed. Massachusetts, USA: Morgan Kaufmann Publishers, 2010.

SWELLER, J.; AYRES, P.; KALYUGA, S. **Cognitive Load Theory**. New York: Springer, 2011.

TERASIC TECHNOLOGIES. **DE10-Lite Development and Education Board**. Terasic, [s. d.]. Disponível em: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1021. Acesso em: 11 fev. 2026.

TERASIC TECHNOLOGIES. **DE10-Lite User Manual**. Hsinchu, Taiwan. Disponível em: https://www.terasic.com. Acesso em: 16 maio 2025.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas Digitais: Princípios e Aplicações**. 11. ed. São Paulo: Pearson Prentice Hall, 2011.

VIANA, C. C. Como controlar motor DC utilizando o Driver Ponte H L298N. **Blog da Robótica**, 15 maio 2023. Disponível em: https://www.blogdarobotica.com/2023/05/15/como-controlar-motor-dc-utilizando-o-driver-ponte-h-l298n/. Acesso em: 22 abr. 2026.

VIANA, C. C. Como usar o sensor infravermelho TCRT5000 com Arduino. **Blog da Robótica**, 30 jun. 2022. Disponível em: https://www.blogdarobotica.com/2022/06/30/como-usar-o-sensor-infravermelho-tcrt5000-com-arduino/. Acesso em: 22 abr. 2026.

WAKERLY, J. F. **Digital Design: Principles and Practices**. 4. ed. Upper Saddle River: Prentice Hall, 2005

APÊNDICE A - REPOSITÓRIO DO CÓDIGO-FONTE (GITHUB)

Devido à extensão dos arquivos e à divisão estrutural em seis etapas incrementais somadas à versão adaptada para o protótipo, o código-fonte completo desenvolvido neste trabalho foi disponibilizado em um repositório digital de versionamento.

A adoção desta prática visa facilitar a reprodutibilidade do projeto por outros estudantes e professores, permitindo o *download* direto dos arquivos em formato VHDL e do mapeamento de pinos (Pin Planner) compatível com o *software* Intel Quartus Prime Lite Edition.

O código completo, organizado em pastas conforme as etapas descritas na metodologia, pode ser acessado publicamente por meio do seguinte endereço eletrônico.

Link para o repositório: https://github.com/lucas07duarte/fsm-elevador-vhdl

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me conceder a força, a resiliência e a sabedoria necessárias para superar os desafios ao longo de toda a graduação.

Ao meu orientador, Prof. Michael Klug, pela paciência, pela confiança depositada neste trabalho e por todo o direcionamento técnico. Sua orientação e apoio técnico contribuíram de forma significativa para a realização deste trabalho.

Aos professores do curso de Engenharia Elétrica do IFSC Joinville, pelos ensinamentos teóricos e práticos que fundamentaram a minha jornada acadêmica, e aos colegas de turma, pelas valiosas trocas de conhecimento.

Um agradecimento especial e profundo aos meus pais, Sidnei e Rosana, pelo apoio incondicional, pelos sacrifícios que fizeram para que eu pudesse chegar até aqui e por sempre acreditarem no meu potencial. Esta conquista é nossa.

Aos amigos e a todos que, de forma direta ou indireta, contribuíram para este projeto, o meu muito obrigado.

LUCAS DUARTE

MÁQUINAS DE ESTADOS FINITOS COMO ESTRATÉGIA NO ENSINO DE FPGA APLICADA A UM ELEVADOR DIDÁTICO

Este trabalho foi julgado adequado para obtenção do título em Bacharel em Engenharia Elétrica, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

Joinville, 01/07/2026.

Prof. Michael Klug, Dr
Orientador
Instituto Federal de Santa Catarina

Prof. Jeferson Luiz Curzel, Dr
Instituto Federal de Santa Catarina

Prof. Nivaldo Theodoro Schiefler Junior, Dr
Instituto Federal de Santa Catarina