

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Unidade 1

SISTEMAS DIGITAIS

Aula 1

Fundamentos de Sistemas Digitais

Fundamentos De Sistemas Digitais



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Na era da transformação digital, compreender os fundamentos dos sistemas digitais é mais do que uma habilidade técnica; é uma necessidade essencial para uma ampla gama de profissionais. Esta videoaula introduzirá os conceitos básicos e as diferenças entre sistemas digitais e analógicos, explorando a eletrônica digital e os sistemas de numeração. Você aprenderá a linguagem fundamental que permite a tecnologia digital operar e transformar nosso mundo. Desde a teoria até as aplicações práticas, esta aula é um convite para mergulhar no universo digital e equipar-se com conhecimentos que são críticos no cenário tecnológico atual. Junte-se a nós nesta jornada de aprendizado e descubra como os sistemas digitais formam a espinha dorsal da tecnologia moderna.

Ponto de Partida

No mundo contemporâneo, a distinção entre sistemas digitais e analógicos não é apenas técnica, mas fundamental para compreender as inovações tecnológicas que permeiam nossa vida diária. Enquanto os sistemas analógicos capturam e transmitem informações através de variações contínuas, representando a era inicial da eletrônica, os sistemas digitais utilizam dados em formatos binários para revolucionar o modo como processamos, armazenamos e

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



comunicamos informações. A transição da eletrônica analógica para a digital marcou o início de uma era de transformações sem precedentes, desde a simples execução de cálculos em calculadoras até a complexidade dos supercomputadores e a onipresença da internet.

A Figura 1 exemplifica a transformação tecnológica de métodos analógicos para digitais, uma mudança que revolucionou nossa interação diária com dados e informações. Representa a evolução dos registros manuais em papel para gráficos em dispositivos digitais, da computação simplificada em calculadoras ao processamento avançado simbolizado pelo ábaco, destacando o progresso da eletrônica.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

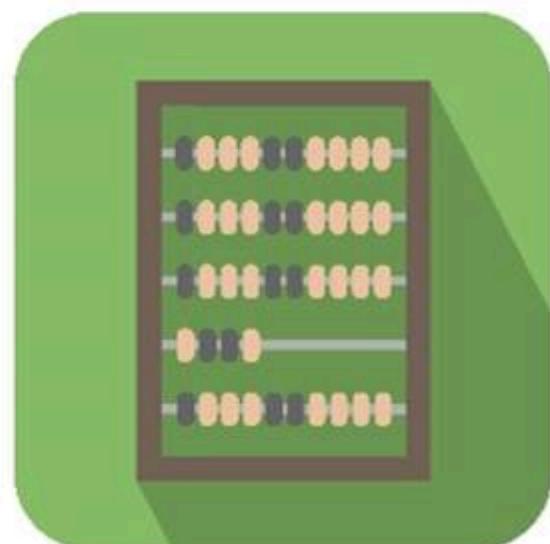


Figura 1 | Da analogia à digitalização: a evolução tecnológica na Era da Informação. Fonte: Freepik.

A eletrônica digital e os sistemas de numeração são os alicerces que sustentam esta revolução, permitindo avanços extraordinários em computação e telecomunicações. Mas como exatamente a superação das limitações dos sistemas analógicos pelos digitais catalisou não apenas avanços tecnológicos, mas também transformações sociais profundas?

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

No início do semestre, Ana Beatriz, uma estudante de engenharia de software com uma profunda fascinação pela história da tecnologia e seu impacto na sociedade, inicia um ambicioso projeto de conclusão de curso. Seu objetivo é explorar a transição dos sistemas analógicos para digitais, destacando como essa mudança influenciou o desenvolvimento de novas tecnologias de computação. Com uma curiosidade insaciável e uma abordagem meticulosa, Ana está especialmente interessada em entender as implicações sociais dessas transformações tecnológicas.

Para aprofundar sua pesquisa, Ana decide entrevistar Carlos Eduardo, um engenheiro eletrônico veterano cuja carreira começou na era dos sistemas analógicos. Carlos testemunhou a revolução digital em primeira mão, adaptando-se às novas tecnologias e contribuindo para o avanço dos sistemas digitais inovadores. Agora, como mentor de jovens engenheiros e estudantes, ele está disposto a compartilhar suas ricas experiências e insights sobre a evolução tecnológica. Suas conversas com Ana abrangem desde os limites dos sistemas analógicos, que restringiam o progresso tecnológico, até a inovação e as oportunidades trazidas pelos sistemas digitais e a crucial compreensão dos sistemas de numeração.

Essas discussões revelam os aspectos técnicos da transição e como ela afetou a vida das pessoas, transformando a maneira como interagimos, trabalhamos e nos divertimos. Carlos destaca a importância de se adaptar e continuar aprendendo, enfatizando que superar os desafios impostos pelos sistemas analógicos e abraçar as possibilidades dos digitais não apenas impulsionou a fronteira tecnológica, mas também deixou uma marca indelével na sociedade. Inspirada por Carlos, Ana percebe a importância de contribuirativamente para as inovações tecnológicas, uma lição que ela levará consigo ao avançar em sua própria carreira.

Vamos Começar!

Sistemas Digitais e Analógicos

Nossa jornada começa com a exploração dos fundamentos dos sistemas digitais, um tema que molda cada aspecto da tecnologia moderna. Ao longo desta seção, vamos mergulhar nos conceitos básicos que diferenciam os sistemas digitais dos analógicos, iluminando como esses princípios se aplicam no desenvolvimento e na operação de dispositivos e tecnologias que permeiam nosso cotidiano.

Sistemas Digitais *versus* Sistemas Analógicos

Primeiramente, é essencial compreender a distinção fundamental entre sistemas digitais e analógicos. Os sistemas analógicos operam com sinais contínuos que representam variáveis físicas, enquanto os sistemas digitais trabalham com dados expressos em binário, isto é, zeros e

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

uns (Lenz; Torres, 2019). Isso é exemplificado na Figura 2, com uma onda contínua senoidal e uma onda quadrada, representando respectivamente sinais analógicos e digitais. Essa diferença é crucial, pois determina como a informação é processada, armazenada e transmitida em dispositivos variados, desde relógios e rádios até computadores e smartphones.

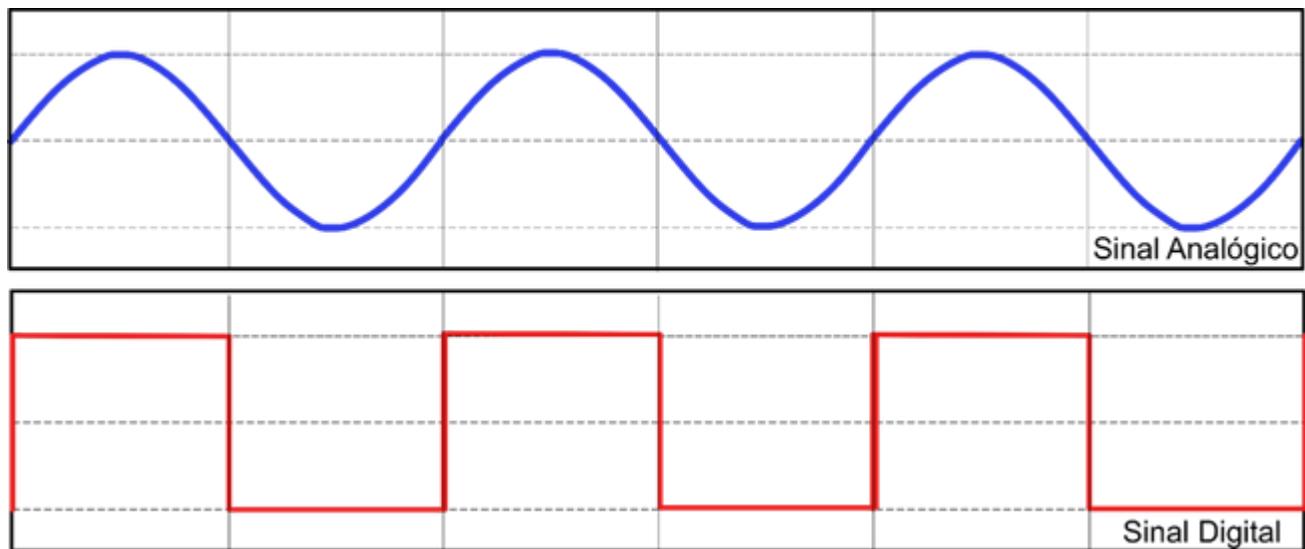


Figura 2 | Representação gráfica de sinais analógicos e digitais. Fonte: elaborada pelo autor.

O Quadro 1, por sua vez, detalha as diferenças-chave: sinais analógicos são contínuos e refletem medidas físicas, são sujeitos a mais interferências de ruído e têm menor consumo de largura de banda, enquanto os sinais digitais são discretos, robustos contra ruídos e com maior demanda de largura de banda, mas com consumo de energia geralmente mais baixo.

Comparação	Analógico	Digital
Sinal	Sinal analógico é contínuo e representa medidas físicas.	Sinais digitais são discretos e gerados por modulação digital.
Representação	Usa uma faixa contínua de valores.	Usa valores discretos ou descontínuos para representação.
Exemplo	Voz humana no ar, dispositivos eletrônicos analógicos.	Computadores, e outros dispositivos eletrônicos digitais.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Resposta ao ruído	Mais propenso a ser afetado, reduzindo a precisão.	Menos afetado.
Largura de banda	Geralmente, o processamento consome menos largura de banda.	Consume mais largura de banda para realizar a mesma informação.
Energia	Instrumento analógico consome grande energia.	Instrumento digital consome apenas energia negligível.

Quadro 1 | Comparaçāo entre sinais analógicos e digitais. Fonte: Idoeta e Capuano (2019).

A Revolução da Eletrônica Digital

A eletrônica digital transformou radicalmente a maneira como vivemos, trabalhamos e nos comunicamos. Exploraremos como os circuitos digitais, baseados em lógica binária, tornaram possível o desenvolvimento de computadores avançados, dispositivos móveis, e a própria internet. A capacidade de processar grandes volumes de dados com precisão e eficiência abriu caminho para inovações que seriam impossíveis com a tecnologia analógica.

Atualmente, a eletrônica digital é base de inovações como a internet das coisas (IoT) e a inteligência artificial (IA). Com dispositivos IoT, objetos comuns estão sendo transformados em agentes inteligentes capazes de comunicar-se entre si e com a nuvem, resultando em casas, cidades e indústrias mais inteligentes. Simultaneamente, a inteligência artificial está remodelando campos desde a medicina até a produção artística, oferecendo capacidades de aprendizado de máquina e processamento de linguagem natural que expandem as fronteiras do possível. Juntas, essas tecnologias estão não apenas moldando nosso presente, mas também definindo o futuro da interação humana com a máquina. A Figura 3 destaca uma residência transformada pela IoT e IA, na qual dispositivos interconectados melhoram a vida diária e prenunciam um futuro de interação avançada entre humanos e máquinas.

No coração desta revolução digital encontra-se a Lei de Moore, uma previsão feita por Gordon Moore em 1965, que observou que o número de transistores em um circuito integrado dobraria aproximadamente a cada dois anos. Esta lei tem se mantido verdadeira por décadas, impulsionando um crescimento exponencial na potência de processamento dos dispositivos eletrônicos. Graças a essa evolução, vivenciamos uma era em que os limites da computação são constantemente redefinidos, permitindo que máquinas aprendam, tomem decisões e até mesmo imitem formas de inteligência humana.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

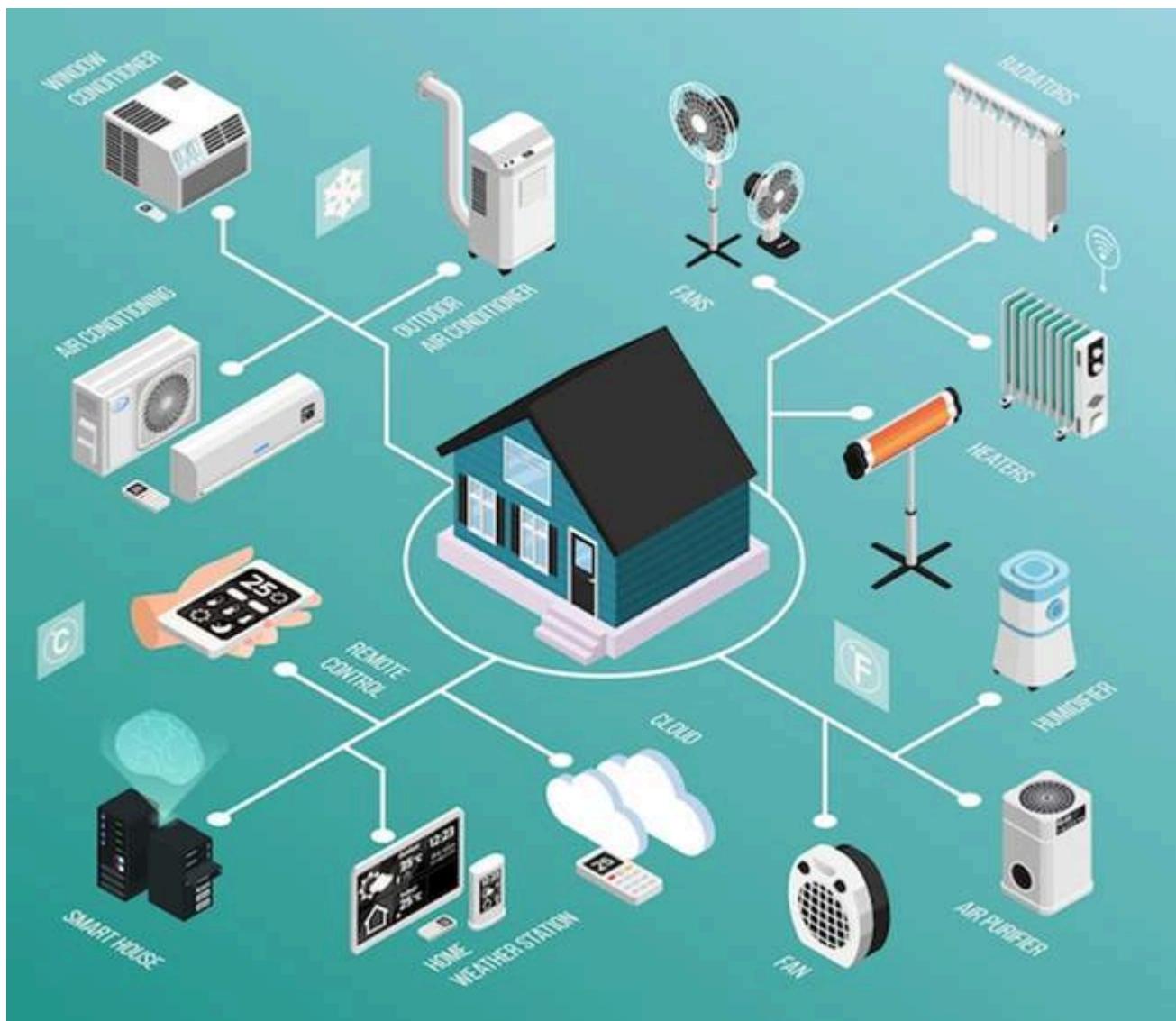


Figura 3 | A casa conectada: um panorama da internet das coisas. Fonte: Freepik.

Sistemas de Numeração: A Linguagem da Computação

Vamos agora abordar os sistemas de numeração, com especial atenção ao sistema binário – o idioma intrínseco da computação. Este sistema não só codifica números, mas é o alicerce sobre o qual textos, imagens e sons são convertidos em dados processáveis, formando assim a vasta gama de funcionalidades que nossos dispositivos eletrônicos modernos oferecem. Compreender esses sistemas é crucial para profissionais de informática e tecnologia, pois eles formam a base de operação e desenvolvimento dentro desses campos.

Os sistemas de numeração que predominam no cenário tecnológico atual são (Souza, 2018):

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- **Sistema decimal** (base 10): emprega dez símbolos, de 0 a 9, e é a espinha dorsal dos cálculos no dia a dia.
- **Sistema binário** (base 2): usa apenas 0 e 1 para representar dados, sendo a pedra angular da computação e da eletrônica digital.
- **Sistema octal** (base 8): com símbolos de 0 a 7, hoje é menos empregado, mas outrora foi uma ponte entre o entendimento humano e a complexidade binária.
- **Sistema hexadecimal** (base 16): com 16 símbolos, de 0 a 9 e de A a F, é preferido para programação e design de sistemas de computadores devido à sua eficiência em condensar extensas sequências binárias.

Sistema de Numeração	Base	Símbolos	Aplicações contemporâneas
Decimal	10	0-9	Matemática do dia a dia, medições financeiras
Binário	2	0, 1	Lógica de computadores, armazenamento de dados
Octal	8	0-7	Legado em programação, sistemas computacionais
Hexadecimal	16	0-9, A-F	Programação de sistemas, endereçamento de memória

Quadro 2 | Sistemas de numeração e suas aplicações contemporâneas. Fonte: Cruz (2022).

O quadro fornece um resumo claro dos diversos sistemas de numeração, suas bases numéricas, os símbolos utilizados e onde esses sistemas encontram utilidade no mundo da computação e tecnologia informacional atual.

Você está convidado a se engajar ativamente nesta exploração, questionando, ponderando e aplicando o conhecimento adquirido. Ao entender esses fundamentos, você estará mais preparado para navegar no mundo digital e, possivelmente, contribuir para as próximas ondas de

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

inovação tecnológica. Vamos juntos descobrir como os sistemas digitais e os sistemas de numeração formam a base sobre a qual o futuro da tecnologia está sendo construído.

Siga em Frente...

Fundamentos de Sistemas Digitais

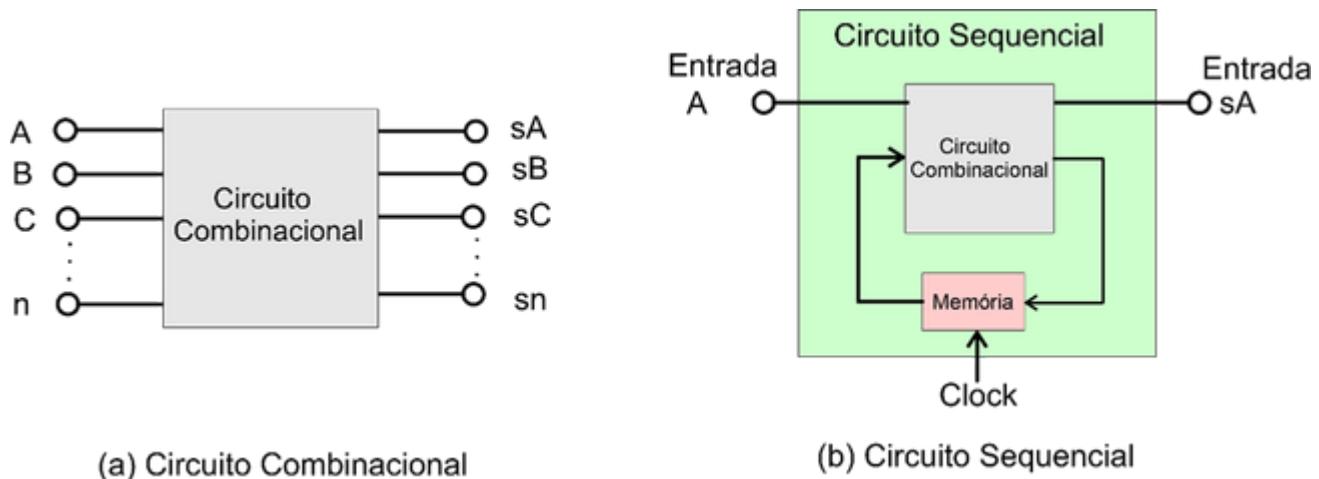
Os sistemas digitais são a espinha dorsal da nossa era computacional, suportando a infraestrutura de tecnologia que impulsiona inovações em todos os campos, desde comunicações móveis até avanços na medicina. A compreensão dos sistemas digitais é tanto uma habilidade técnica quanto uma linguagem universal na era da informação.

Tipos de Sistemas Digitais

No domínio dos sistemas digitais, existem duas categorias fundamentais: combinacionais e sequenciais. Os sistemas combinacionais, como o exemplificado na Figura 4(a), têm saídas que dependem unicamente das entradas atuais, sem memória de estados anteriores – um exemplo é a porta lógica AND, que emite alto (1) somente quando todas as suas entradas também estão em alto (1). Por outro lado, os sistemas sequenciais, mostrados na Figura 4(b), apresentam saídas que são determinadas pelas entradas atuais em conjunto com estados passados, devido à presença de memória no circuito. O flip-flop é um representante desta categoria, capaz de reter um estado até que seja modificado por um sinal de entrada (Idoeta; Capuano, 2019).

Sistemas digitais combinacionais são a base para operações lógicas e de decisão, enquanto sistemas digitais sequenciais são cruciais para tarefas que envolvem contagem, memória e processos temporais. O entendimento desses sistemas fornece a base para projetar tudo, desde calculadoras simples até supercomputadores complexos. Por exemplo, um microprocessador é uma integração complexa de sistemas combinacionais e sequenciais trabalhando juntos para realizar as operações de processamento de dados.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



(a) Círcuito Combinacional

(b) Círcuito Sequencial

Figura 4 | Fundamentos dos circuitos digitais: combinacionais e sequenciais. Fonte: elaborada pelo autor.

Relação entre Sistemas de Numeração e Sistemas Digitais

A relação entre sistemas de numeração e sistemas digitais é mais do que um mero exercício acadêmico; é um aspecto prático vital que permeia todos os aspectos da computação e da engenharia eletrônica. Cada sistema de numeração serve como uma ferramenta para diferentes propósitos em um sistema digital, e a habilidade de converter entre esses sistemas é uma competência essencial para os profissionais da área.

Conversões entre sistemas de numeração são essenciais para os engenheiros e programadores que trabalham com hardware e software. Por exemplo, os dados representados no sistema decimal pelos usuários frequentemente precisam ser convertidos para o sistema binário para processamento interno do computador. Além disso, o octal e o hexadecimal são usados para simplificar a visualização e manipulação de *strings* binárias longas, especialmente em contextos de programação e design de sistemas.

- Convertendo decimal para binário

A conversão de decimal para binário é realizada por meio de divisões sucessivas por 2, anotando o resto e lendo os restos de trás para frente ao final do processo. Por exemplo, para converter o número decimal 13 para binário, dividimos 13 por 2, o que nos dá 6 com um resto de 1. Continuamos dividindo o quociente por 2 até que o quociente seja 0, anotando os restos. Assim, obtemos a sequência binária 1101, exemplificado em 5(a).

- Convertendo binário para hexadecimal

Para converter binário para hexadecimal, agrupamos os bits binários em conjuntos de quatro, começando da direita para a esquerda, e então atribuímos a cada grupo de quatro um dígito hexadecimal correspondente, de 0 a F. Por exemplo, o número binário 11011011010 seria agrupado (0110 1101 1010), e então convertido para hexadecimal como 6DA, mostrado em 5(b).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- Convertendo hexadecimal para decimal

A conversão de hexadecimal para decimal envolve multiplicar cada dígito hexadecimal pelo valor de 16 elevado à potência da posição do dígito (começando do zero à direita), conforme o exemplo em 5(c).

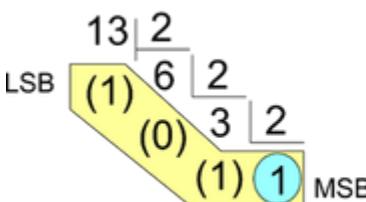
(a) Decimal para Binário	(b) Binário para Hexadecimal	(c) Hexadecimal para Decimal
	11011011010 0110 1101 1010 6 D A	$B3 = 3 \times 16^0 + B^1 \times 16^1$ $B3 = 3 + 11 \times 16$ $B3 = 179$ $B3_H = 179_D$
$13_D = 1101_B$	$11011011010_B = 6DA_H$	

Figura 5 | Conversões numéricas. Fonte: elaborada pelo autor.

A conversão entre sistemas de numeração é uma operação matemática e uma funcionalidade essencial incorporada em software, e empregada em tempo de execução por sistemas digitais para uma ampla gama de finalidades, desde o armazenamento simples de números até operações avançadas de criptografia e compressão de dados. Portanto, compreender como essas conversões ocorrem e sua aplicação prática é crucial para estudantes e profissionais de tecnologia. Dominar essa habilidade facilita a compreensão da manipulação e representação de dados e capacita a criação de algoritmos e hardware eficientes para processamento de informações. Com esse domínio, você estará preparado para enfrentar os desafios do mundo digital e contribuir para a inovação tecnológica.

Vamos Exercitar?

Transformações Tecnológicas

No início da aula lançamos um desafio ao estudante, inspirado nas conversas entre Ana Beatriz, uma estudante de engenharia de software, e Carlos Eduardo, um engenheiro eletrônico experiente: compreender os limites dos sistemas analógicos e como a transição para sistemas digitais, junto com uma profunda compreensão dos sistemas de numeração, permitiu superar essas barreiras. Este desafio evidencia a evolução tecnológica e ressalta as transformações sociais significativas impulsionadas por essa mudança de paradigma. Durante suas discussões, Ana e Carlos focaram os limites dos sistemas analógicos, que são mais suscetíveis ao ruído e têm uma capacidade limitada de processamento de dados, e como a adoção dos sistemas digitais ofereceu uma maneira de processar e armazenar dados de forma mais eficiente e precisa, abrindo novos caminhos para inovações e impactos sociais.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Ao longo da aula exploramos os conteúdos fundamentais discutidos por Ana e Carlos, incluindo a distinção entre sistemas digitais e analógicos, a importância da eletrônica digital e a compreensão dos sistemas de numeração como base para superar as limitações analógicas. A digitalização permite que a informação seja processada, armazenada e transmitida sem degradação significativa, promovendo uma expansão exponencial das capacidades computacionais e de comunicação. A conversão entre diferentes sistemas de numeração exemplifica a flexibilidade e eficiência dos sistemas digitais, aplicáveis desde operações aritméticas simples até complexas aplicações em áreas como criptografia e inteligência artificial. Ana e Carlos ilustram como essa transição não apenas solucionou problemas técnicos, mas também abriu horizontes para inovações em diversos campos.

Encorajamos o estudante a refletir, junto com Ana e Carlos, a respeito de outras áreas nas quais a transição do analógico para o digital pode gerar transformações profundas. Por exemplo, na saúde, como a digitalização de registros médicos e o uso de diagnósticos assistidos por IA, discutidos por Ana em seu projeto com a ajuda de Carlos, podem melhorar significativamente o atendimento ao paciente. Ou na educação, área em que as conversas entre os dois destacaram como as plataformas digitais podem personalizar a aprendizagem para atender às necessidades individuais dos estudantes. Essa reflexão não só reforça a compreensão dos conceitos discutidos, mas também estimula a aplicação desses princípios para solucionar problemas e inovar em diversos campos, seguindo o exemplo de curiosidade e inovação de Ana e a sabedoria e experiência de Carlos.

Saiba mais

Para aprimorar seu entendimento acerca de sistemas numéricos e códigos digitais, sugerimos dedicar atenção às páginas 10 a 26 da Seção 1, Unidade 1 do livro [Eletônica: circuitos analógicos e digitais](#), de Charles William Polizelli Pereira. Este livro destaca-se por sua abordagem didática e detalhada dos temas, essenciais para quem deseja se aprofundar na eletrônica e computação. Disponível em português, facilita a compreensão e a aplicação prática dos conceitos abordados, representando um recurso imprescindível para estudantes e profissionais da área. Para explorar os conceitos em profundidade, concentre-se nas páginas indicadas.

Referências

- CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. edição. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

SOUZA, D. B. da C. *et al.* **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

Aula 2

Portas Lógicas e Álgebra Booleana

Portas Lógicas E Álgebra Booleana



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Descubra os alicerces da tecnologia moderna nesta videoaula essencial que trata de portas lógicas, álgebra booleana e o mapa de Karnaugh. Estes conceitos não apenas formam a base do processamento digital, mas também são cruciais para a inovação e eficiência em diversas áreas profissionais. Aprenda como essas ferramentas podem otimizar projetos e soluções tecnológicas, preparando você para desafios avançados no cenário digital. Não perca a chance de elevar suas habilidades e compreensão; junte-se a nós nesta jornada de aprendizado!

Ponto de Partida

Em uma manhã típica no campus da universidade, Ana Beatriz pensa em seu próximo projeto: a otimização de um sistema de segurança digital para um centro comunitário local, visando melhorar sua eficiência e confiabilidade. O desafio é considerável: o sistema atual é desatualizado, consome muita energia e ocupa muito espaço físico. Ela sabe que para atingir seu objetivo precisará aplicar conceitos avançados de portas lógicas e álgebra booleana, mas sente-se insegura em relação ao ponto pelo qual deve começar.

Ao compartilhar suas preocupações com Carlos Eduardo, Ana é apresentada ao cerne da problematização: "Como podemos utilizar o conhecimento de portas lógicas, álgebra booleana e o mapa de Karnaugh para simplificar o circuito de segurança e torná-lo mais eficiente em termos de consumo de energia e espaço, enfrentando os desafios de eficiência, velocidade e miniaturização de dispositivos digitais?"

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Carlos, percebendo a oportunidade de ensino, propõe um desafio prático para Ana. Ele sugere que eles abordem o projeto do sistema de segurança como um caso de estudo para aplicar e demonstrar a eficácia desses conceitos teóricos. "Vamos transformar este desafio em uma oportunidade de aprendizado" diz Carlos: "Aplicando a teoria à prática, você não apenas resolverá um problema real, mas também ganhará uma compreensão profunda dos princípios que formam a base da tecnologia digital moderna."

Ana e Carlos mergulham no projeto, começando com uma revisão das portas lógicas existentes no circuito e explorando como a álgebra booleana pode ser usada para simplificar a lógica do circuito. Carlos introduz o mapa de Karnaugh como uma ferramenta poderosa para visualizar e minimizar expressões booleanas complexas, facilitando a identificação de simplificações que poderiam tornar o circuito mais compacto e eficiente.

À medida que mergulhamos na era digital, a compreensão das estruturas que formam a base da tecnologia digital se torna imperativa. Portas lógicas, álgebra booleana e o mapa de Karnaugh não são apenas componentes teóricos: são as ferramentas que moldam, simplificam e potencializam os sistemas digitais que definem nossa realidade tecnológica. Esses conceitos possibilitam a execução de operações básicas em circuitos e a otimização de processos complexos que são fundamentais para a inovação e eficiência tecnológica.

Vamos Começar!

Portas Lógicas e a Álgebra Booleana

As portas lógicas e a álgebra booleana constituem a base da computação e do processamento de dados, e oferecem as ferramentas necessárias para desvendar e otimizar a complexidade dos sistemas digitais que definem nosso mundo tecnológico.

As portas lógicas são os componentes fundamentais dos circuitos digitais, atuando como os neurônios do cérebro digital. Elas realizam operações básicas de lógica booleana – AND, OR, NOT, NAND, NOR, XOR, XNOR –, processando sinais binários (0s e 1s) para produzir um resultado específico. A Figura 1 ilustra a representação gráfica dessas portas lógicas e suas respectivas tabelas-verdade. Cada tipo de porta lógica tem uma função única, determinada pela tabela-verdade que descreve suas operações (Idoeta, 2019).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

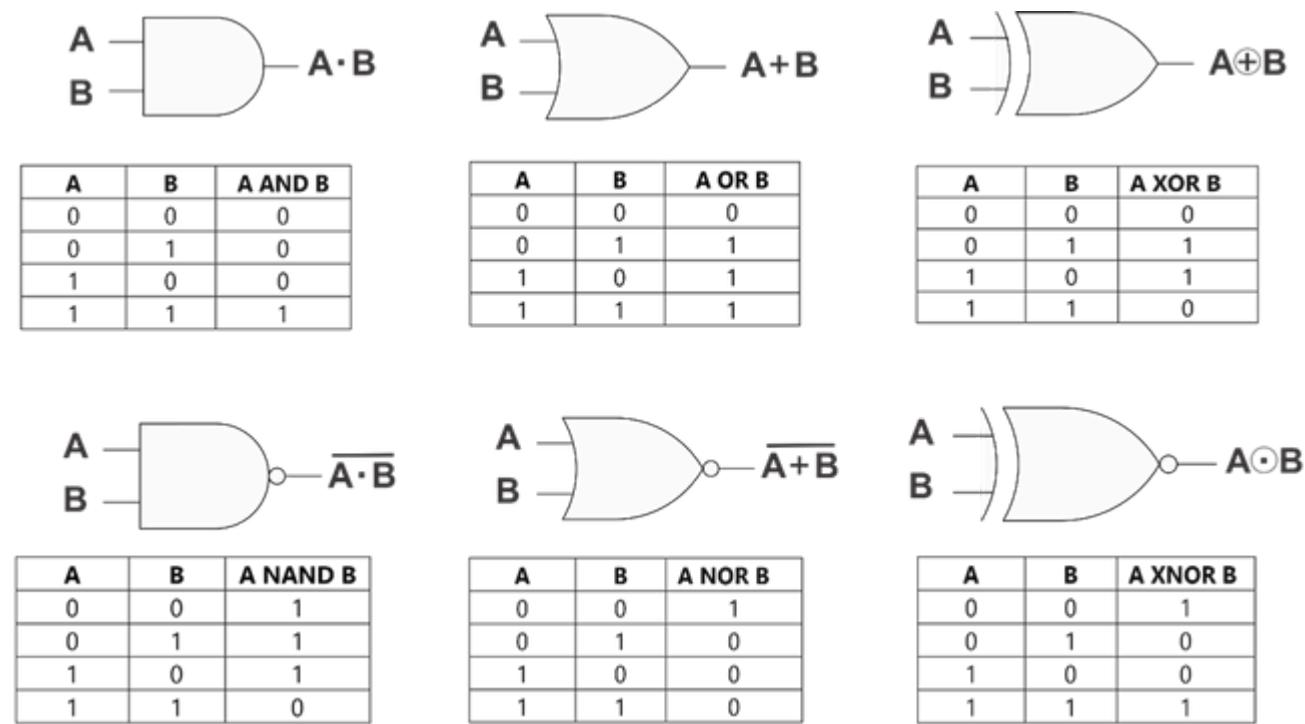


Figura 1 | Símbolos e tabelas-verdade de portas lógicas básicas. Fonte: elaborada pelo autor.

A importância das portas lógicas reside em sua capacidade de executar operações complexas de computação por meio da combinação de funções simples. Elas são os tijolos que constroem os circuitos digitais, desde calculadoras simples até supercomputadores avançados. Entender como essas portas funcionam e como são combinadas para formar circuitos mais complexos é crucial para qualquer profissional da área de tecnologia.

Exemplos práticos do uso de portas lógicas incluem a construção de um circuito de soma, que combina portas AND, OR e XOR para realizar adições binárias, ou sistemas de segurança que exigem uma sequência específica de entradas para ser desbloqueado.

Álgebra Booleana

A álgebra booleana permite a representação e manipulação de expressões lógicas, facilitando a simplificação e análise de circuitos digitais. Por meio de suas leis e propriedades, como a lei da dualidade e os teoremas de De Morgan, é possível otimizar circuitos para que sejam mais eficientes em termos de velocidade, espaço e consumo de energia (Idoeta, 2019).

1. Axiomas:

- **Identidade da adição** (axioma 1): em uma configuração representada na Figura 2, temos uma lâmpada (L) conectada em série a um interruptor (A) e um segundo interruptor (B)

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

conectado em paralelo a A. Quando o interruptor B está desligado (posição 0), a lâmpada segue o estado do interruptor A ($L = A$). No entanto, quando o interruptor B é acionado, a lâmpada fica acesa, independentemente da posição do interruptor A, demonstrando a operação da lógica OR, também conhecida como "OU".

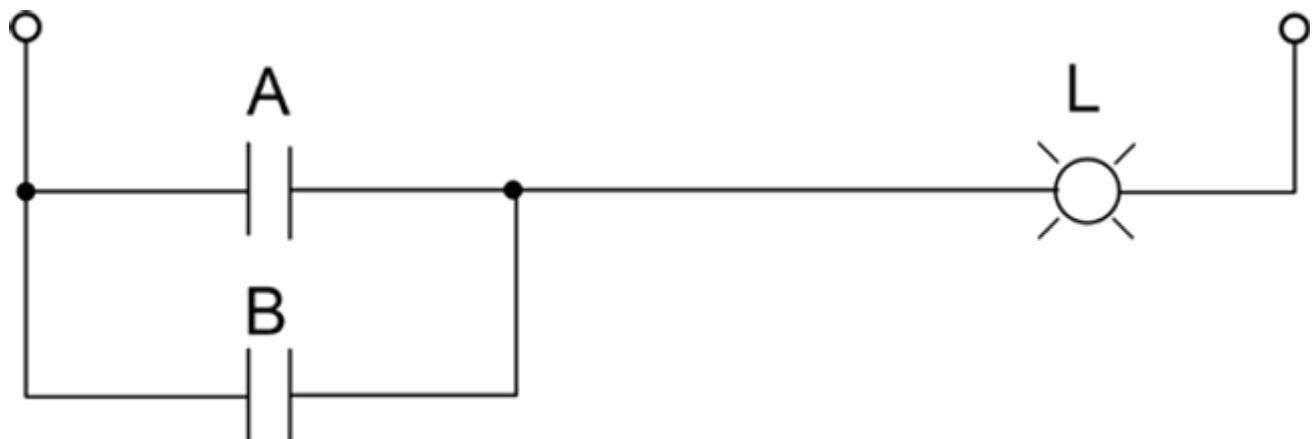


Figura 2 | Exemplo da identidade da adição. Fonte: elaborada pelo autor.

- **Identidade da multiplicação** (axioma 2): em uma configuração ilustrada na Figura 3, na qual uma lâmpada está em série com as chaves A e B, a lâmpada L permanece acesa somente quando ambas as chaves (A e B) estão acionadas, o que representa a operação AND, ou "E". Como exemplo, se a chave B estiver constantemente acionada ($B = 1$), podemos afirmar que $L = A \cdot B = A \cdot 1$. Nesse cenário, a lâmpada depende exclusivamente do estado de A para permanecer acesa.

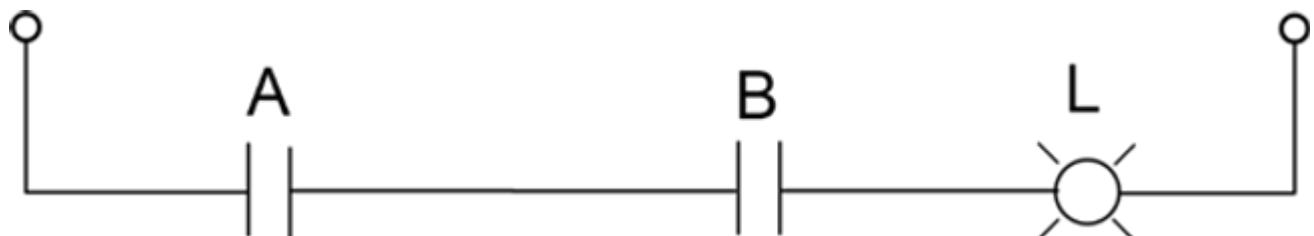


Figura 3 | Exemplo da identidade da multiplicação. Fonte: elaborada pelo autor.

- **Lei do Complemento** (axioma 3): na configuração apresentada na Figura 4, em que temos a chave A em série com a lâmpada L e uma segunda chave ($A\bar{}$), que é o complemento de A, em paralelo com a primeira, aplicam-se as seguintes condições: quando A está desligada (0), $A\bar{}$ está ligada (1), resultando na lâmpada acesa ($A + A\bar{ } = 1$). Por outro lado, quando A está ligada (1), $A\bar{}$ está desligada (0), mantendo a lâmpada acesa.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

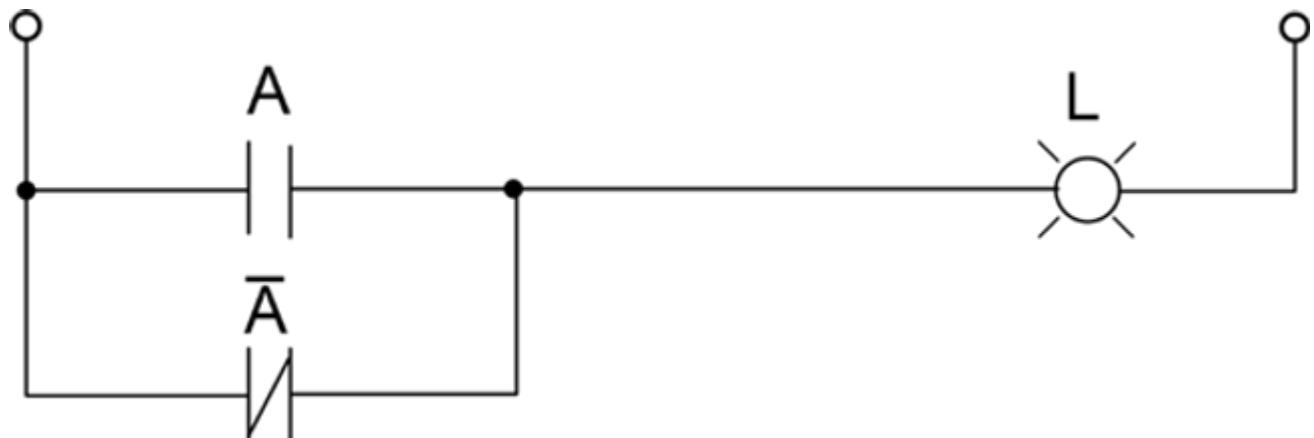


Figura 4 | Exemplo da Lei do Complemento – chaves em paralelo. Fonte: elaborada pelo autor.

Além disso, é fundamental destacar que a disposição de duas chaves complementares em série com a lâmpada equivale a realizar a operação de multiplicação entre uma variável A e seu complemento. Isso resulta em zero ($A \cdot A\bar{=} = 0$), o que implica que a lâmpada não acenderá, como exemplificado na Figura 5.



Figura 5 | Exemplo da Lei do Complemento – chaves em série. Fonte: elaborada pelo autor.

2. Propriedades:

Agora, exploraremos as propriedades da álgebra booleana, abordando comutatividade, associatividade e distributividade.

- A **comutatividade**, representada pela troca da ordem das chaves em série ou em paralelo, é uma propriedade fundamental. Independentemente de ligarmos A e B ou B e A em série ou paralelo, o resultado na lâmpada é o mesmo. Essa propriedade pode ser expressa por uma equação: $A + B = B + A$. Em outras palavras, a ordem das chaves não afeta o resultado da iluminação da lâmpada.
- A **associatividade** diz respeito à maneira como agrupamos as chaves, seja em série, como $(A + B) + C$, ou em paralelo, como $(A \cdot B) \cdot C$. Importante notar que a ordem de agrupamento não afeta o resultado na lâmpada. Essa propriedade pode ser expressa por equações: $(A + B) + C = A + (B + C)$ e $(A \cdot B) \cdot C = A \cdot (B \cdot C)$.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- A **distributividade** é análoga à forma como conectamos chaves em relação a uma lâmpada. Podemos conectar as chaves A, B e C de duas maneiras diferentes: $A \cdot (B + C)$ ou $(A \cdot B) + (A \cdot C)$. É importante notar que a lâmpada sempre apresentará o mesmo resultado, independentemente da maneira como as chaves são distribuídas.

A Tabela 1 formalmente resume os principais axiomas e propriedades dessa álgebra.

Axioma/Propriedade	Descrição
Identidade da Adição	$A + 0 = A$. $A + 1 = 1$. $A + A = A$.
Identidade da Multiplicação	$A \cdot 1 = A$. $A \cdot 0 = 0$. $A \cdot A = A$.
Lei do Complemento	$A + A\bar{=} = 1$. $A \cdot A\bar{=} = 0$.
Comutatividade	$A + B = B + A$. $A \cdot B = B \cdot A$.
Associatividade	$(A + B) + C = A + (B + C)$. $(A \cdot B) \cdot C = A \cdot (B \cdot C)$.
Distributividade	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$. $A + (B \cdot C) = (A + B) \cdot (A + C)$.

Tabela 1 | Axiomas e propriedades de Boole. Fonte: elaborada pelo autor.

Na Álgebra de Boole, a simplificação de funções lógicas complexas é uma tarefa fundamental, e para isso recorremos a diversos teoremas, leis e propriedades que permitem reduzir a complexidade das expressões.

Teorema da Absorção:

O Teorema da Absorção é uma ferramenta essencial para remover termos redundantes em expressões lógicas. Ele se expressa em duas regras fundamentais:

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

$$A + (A \cdot B) = A.$$

$$A \cdot (A + B) = A.$$

Dupla Negação:

O princípio da dupla negação estabelece que a negação dupla de uma variável lógica é igual à própria variável:

$$A \overline{\overline{A}} = A$$

Propriedades Idempotentes:

As propriedades idempotentes afirmam que operações repetidas com uma variável não alteram seu valor:

$$A + A = A$$

$$A \cdot A = A$$

Elementos Absorventes:

Dois elementos absorventes desempenham um papel importante na simplificação de expressões:

- 0 é um elemento absorvente para a multiplicação: $A \cdot 0 = 0$.
- 1 é um elemento absorvente para a adição: $A + 1 = 1$.

Negações do Zero e do Um:

As negações do zero e do um são propriedades que mostram que a negação desses valores é o oposto:

$$\overline{0} = 1 \quad \overline{1} = 0$$

Leis de De Morgan:

As Leis de De Morgan são menos intuitivas que as regras anteriores e estabelecem as seguintes regras:

$$A \cdot B = A + B$$

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

$$A + B = A \bullet B$$

Siga em Frente...

O mapa de Karnaugh, desenvolvido inicialmente por Edward Veitch e aperfeiçoado posteriormente por Maurice Karnaugh, oferece uma abordagem valiosa para simplificar circuitos lógicos ao realizar reduções nas expressões lógicas. Essa ferramenta gráfica é particularmente eficaz na simplificação de expressões lógicas em circuitos digitais, destacando-se em equações booleanas que envolvem duas, três ou quatro variáveis (Idoeta, 2019).

É comum agrupar as células do mapa de Karnaugh em pares ou em uma quadra, o que simplifica as expressões booleanas. Isso é alcançado agrupando células adjacentes que contenham o valor "1".

A SOP (soma de produtos) e POS (produto de somas) são duas abordagens diferentes para simplificar expressões booleanas usando o mapa de Karnaugh. Neste momento focaremos a soma de produtos ao explorar os mapas de Karnaugh.

Na forma SOP (soma de produtos), as células agrupadas no mapa de Karnaugh representam termos "AND" que, quando combinados, compõem a expressão booleana, envolvendo produtos de variáveis.

1. Tipos de mapa de Karnaugh:

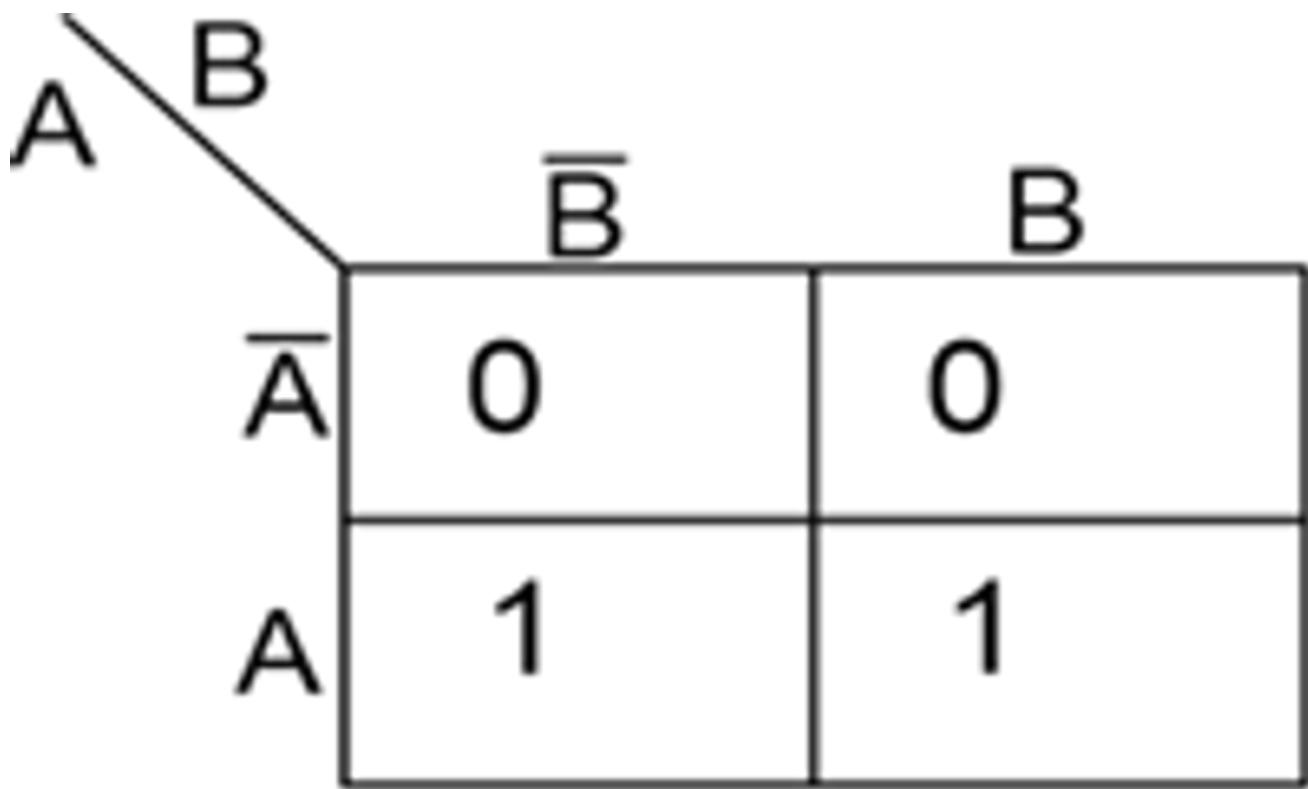
A seguir, descreveremos os mapas de Karnaugh para duas, três e quatro variáveis.

• Duas variáveis

Para duas variáveis, o mapa de Karnaugh é representado por uma tabela contendo quatro células que abrangem todas as possíveis combinações binárias dessas variáveis. As combinações com A e B são organizadas em uma matriz de 2x2, conforme Figura 5.

Exemplo: deixar no word: $F(A,B) = A \cdot B + A \cdot \bar{B}$, pode ser representado na Figura 5, preenchendo os termos das células com 1.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Forma SOP

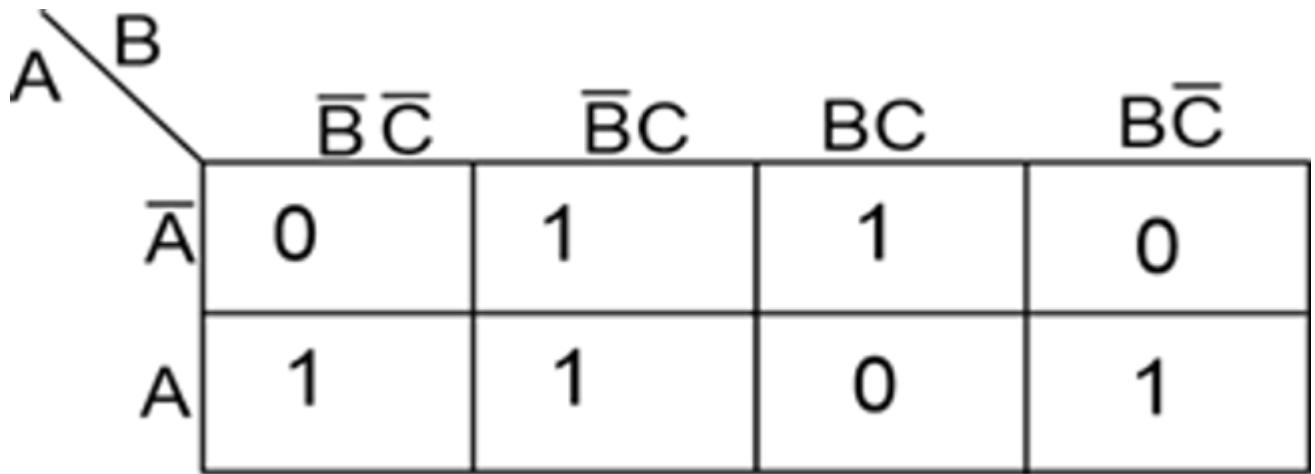
Figura 5 | Mapa de Karnaugh para $F(A,B)$. Fonte: elaborada pelo autor.

- **Três variáveis:**

Com três variáveis: A, B e C, o mapa tem oito células. As células podem ser agrupadas em pares, quadras ou oitavas (o que é uma simplificação direta).

Exemplo: $F(A, B, C) = A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}\bar{C}$

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Forma SOP

Figura 6 | Mapa de Karnaugh para $F(A, B, C)$. Fonte: elaborada pelo autor.

- **Quatro variáveis:**

Para quatro variáveis, o mapa de Karnaugh tem 16 células, representando todas as combinações possíveis de valores binários de quatro bits. As células podem ser agrupadas de várias maneiras, como pares, quadras, oitavas ou grupo de 16, dependendo da lógica da função booleana.

$$\text{Exemplo: } F(A, B, C, D) = A\bar{D} \cdot B\bar{D} \cdot C\bar{D} \cdot D\bar{D} + A\bar{D} \cdot B \cdot C \cdot D\bar{D} + A \cdot B\bar{D} \cdot C \cdot D\bar{D} + A \cdot B \cdot C \cdot D$$

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

\overline{AB}	\overline{CD}	$\overline{C}\overline{D}$	$C\overline{D}$	$C\overline{D}$
$\overline{A}\overline{B}$	1	0	0	0
$\overline{A}B$	0	0	0	1
$A\overline{B}$	0	0	1	0
AB	0	0	0	1

Forma SOP

Figura 7 | Forma SOP para $F(A, B, C, D)$ descrita. Fonte: elaborada pelo autor.

A simplificação de uma expressão booleana usando o mapa de Karnaugh (mapa de K) envolve os seguintes passos, conforme descrito no quadro a seguir:

Etapa	Descrição do passo
1 – Criar mapa de Karnaugh	a. Liste as variáveis booleanas envolvidas.
	b. Organize as variáveis em uma tabela.
	c. Preencha a tabela com os valores da função lógica.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

2 – Agrupar 1's no mapa	a. No mapa de Karnaugh, agrupe os 1s adjacentes.
3 – Escreva a expressão simplificada	<p>a. Escreva a expressão simplificada usando os grupos formados.</p> <p>b. Combine os termos dos grupos com operadores OR.</p>
4 – Verifique a expressão simplificada	<p>a. Substitua a expressão original pela expressão simplificada.</p> <p>b. Verifique o funcionamento da expressão simplificada com todas as combinações de entrada.</p>

Quadro 1 | Procedimento de resolução com mapa de Karnaugh. Fonte: Idoeta (2019).

Termos de “não importa” (*don't care*) são termos na expressão que não afetam a saída e podem ser ignorados. Verifique a expressão minimizada em busca de termos de não importa e removê-los, se estiverem presentes.

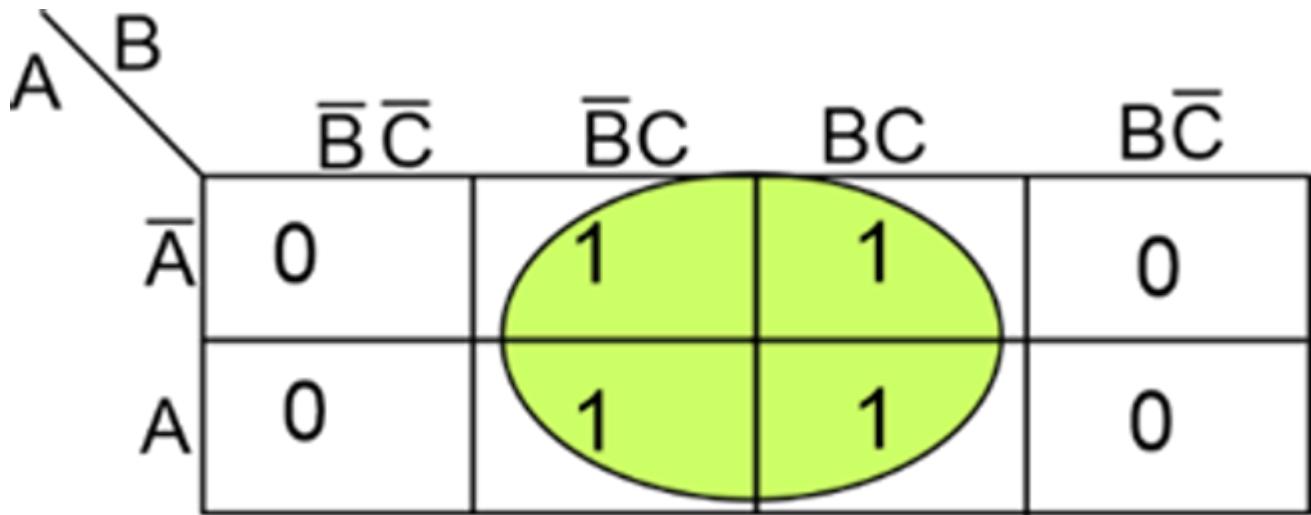
A seguir, exploraremos uma variedade de exemplos que ilustram a metodologia de simplificação de expressões lógicas usando a Álgebra Booleana e o mapa de Karnaugh, oferecendo a oportunidade de aplicar e reforçar o conhecimento adquirido até o momento.

Exemplo 1 – Simplifique a expressão lógica original

$$S = A\bar{B}\bar{C} + A\bar{B}C + A\bar{B}C + AB\bar{C}$$

Com o mapa de Karnaugh, marcamos “1” em cada termo da expressão S e identificamos uma quadra.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Forma SOP

Figura 8 | Forma SOP para $F(A, B, C)$ descrita. Fonte: elaborada pelo autor.

Na quadra formada, observamos qual o termo que não muda. Na quadra há os pares A e $A\bar{A}$, B e $B\bar{B}$. Significa que esses termos podem ser eliminados, restando o termo C , que é o resultado da simplificação.

Exemplo 2 – Use a álgebra booleana para simplificar a expressão lógica: $A \cdot (A + B + C)$

Usando a distributiva, podemos expandir a expressão da seguinte forma:

$$A \cdot A + A \cdot B + A \cdot C.$$

Agora, observe que $A \cdot A$ é equivalente a A (pelo Teorema da Absorção, que diz que $A \cdot A = A$). Então, podemos simplificar a expressão da seguinte maneira:

$$A + A \cdot B + A \cdot C.$$

Agora, aplicando a distributiva novamente, podemos agrupar o termo A em comum:

$$A \cdot (1 + B + C).$$

Como $(1 + B + C)$ é sempre igual a 1 ou verdadeiro, podemos simplificar ainda mais:

$$A \cdot 1 = A.$$

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Portanto, a expressão original " $A \cdot (A + B + C)$ " é equivalente a " A " após a simplificação, demonstrando o uso do Teorema da Absorção no processo.

Exemplo 3 – Use a álgebra booleana para simplificar a expressão lógica: $X + (X \cdot Y) + (X \cdot Z)$

Usando o Teorema da Absorção: $X + (X \cdot Y) + (X \cdot Z) = X$ (aplicando a regra 1 do Teorema da Absorção às duas últimas partes).

Exemplo 4 – Use a álgebra booleana, para simplificar a expressão lógica $((A + B))$.

Ao aplicar a dupla negação a essa expressão, temos que:

$$(A + B) = A + B$$

Exemplo 5 – Use a álgebra booleana, para simplificar a expressão lógica $Y + Y$

Se tivermos a variável Y somada a ela mesma, a expressão permanece inalterada: $Y + Y = Y$.

Vamos Exercitar?

Ampliando os Horizontes com a Tecnologia

Ao aplicar os conceitos aprendidos, Ana consegue não apenas simplificar o circuito, mas também projetar uma solução que utiliza menos energia e ocupa menos espaço, superando as expectativas do centro comunitário. O sucesso do projeto serve como um momento de aprendizado crucial para Ana, que agora vê como os conceitos teóricos de sistemas digitais têm aplicações práticas significativas, permitindo inovações tecnológicas que impactam a sociedade de maneira positiva.

O projeto se torna um ponto de referência para Ana, simbolizando a ponte entre o conhecimento acadêmico e a aplicação prática. Carlos, satisfeito, reforça a ideia de que o verdadeiro entendimento vem da aplicação do conhecimento para resolver problemas do mundo real. A problematização, integrada à narrativa desde o início, guia o desenvolvimento da aula e destaca a importância de adaptar e aplicar conceitos teóricos para enfrentar e superar desafios práticos no campo da tecnologia digital.

Refletindo acerca da problematização apresentada, entendemos que eficiência, velocidade e miniaturização de dispositivos digitais são desafios prementes na era digital. As portas lógicas, a

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



álgebra booleana e o mapa de Karnaugh são fundamentais nesse contexto, pois permitem o projeto de circuitos otimizados que atendem a essas necessidades.

Tomemos, por exemplo, o processo de simplificação de um circuito responsável por um sistema de segurança eletrônico. Originalmente, o circuito poderia exigir uma série complexa de portas lógicas. No entanto, aplicando a álgebra booleana e o mapa de Karnaugh, podemos simplificar a expressão lógica do circuito, reduzindo o número de portas necessárias. Isso economiza espaço e materiais, além de aumentar a velocidade do circuito devido à diminuição da latência resultante da redução dos componentes pelo qual um sinal deve passar.

Com esta abordagem, resolvemos o problema imediato de otimizar um circuito existente, e pavimentamos o caminho para dispositivos mais eficientes. Você está convidado a aplicar esses princípios na resolução de problemas similares; explore como a simplificação e a eficiência podem ser alcançadas em outros sistemas digitais e pense em como essas técnicas podem ser utilizadas para inovações futuras.

Saiba mais

Para você que deseja mergulhar ainda mais fundo nos fundamentos dos circuitos digitais e expandir seu conhecimento sobre a lógica que impulsiona a tecnologia que usamos todos os dias, temos uma recomendação especial. O livro [*Circuitos Digitais*](#), de Rodrigo Vinícius Mendonça Pereira e Hugo Tanzarella Teixeira, é uma leitura essencial para todos que buscam compreender a teoria e a prática por trás dos dispositivos que moldam nosso mundo moderno.

Na Seção 1.1, "Conceitos de circuitos digitais", disponível da página 9 à 27, você encontrará uma explanação detalhada dos princípios de operação dos circuitos digitais, incluindo uma discussão que aborda as portas lógicas e a álgebra booleana tratadas nesta aula.

Referências

- CRUZ, E. *et al.* **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. edição. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. *et al.* **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Aula 3

Circuitos Combinacionais

Circuitos Combinacionais



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Explore o universo dos circuitos digitais com nossa videoaula detalhada, focada nos circuitos combinacionais e sequenciais. Esses componentes formam o coração dos sistemas digitais modernos, e nosso módulo educativo visa demonstrar a importância desses circuitos na eletrônica e no processamento de dados. Este é o momento para fortalecer sua capacidade em eletrônica digital e se posicionar na vanguarda da inovação tecnológica. Aproveite esta oportunidade para ampliar suas habilidades e desvendar os segredos do mundo digital. Convidamos você a avançar conosco nesta jornada educativa e transformadora.

Ponto de Partida

Na era atual, marcada por constantes avanços tecnológicos, compreender as bases da tecnologia digital é fundamental. Isso inclui os circuitos combinacionais e sequenciais, além da lógica subjacente que os orienta. Esses elementos não são apenas ideias teóricas distantes; eles são, de fato, componentes vitais que alimentam, aperfeiçoam e aumentam o poder dos sistemas digitais que encontramos em quase todos os aspectos do nosso dia a dia.

Ana Beatriz e Carlos Eduardo têm se destacado na resolução de problemas complexos e fascinantes mundos dos circuitos combinacionais e flip-flops. Nesta fase de sua jornada educativa, Ana se depara com um novo desafio que coloca suas habilidades à prova, enquanto Carlos oferece sua experiência inestimável para guiá-la através dos intrincados conceitos e aplicações desses componentes fundamentais dos sistemas digitais.

Ana está entusiasmada com o projeto de um novo sistema de controle de acesso baseado em identificação digital, que requer a integração de circuitos combinacionais para o processamento

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

de sinais e flip-flops para a memória temporária. No entanto, ela rapidamente percebe que, apesar de sua familiaridade com os conceitos básicos, aplicar esses componentes em um projeto funcional é um desafio completamente diferente.

Carlos propõe que eles abordem o projeto como uma série de etapas de aprendizado, começando com a revisão dos princípios dos circuitos combinacionais e terminando com a implementação de flip-flops para criar uma memória temporária eficaz.

Carlos questiona: "Como podemos projetar um sistema de controle de acesso que não apenas seja eficiente e rápido, mas também compacto e confiável, usando circuitos combinacionais para processamento de sinal e flip-flops para armazenamento temporário?". Este desafio ressalta a necessidade de uma compreensão profunda e aplicação prática de conceitos teóricos para superar limitações técnicas e atender às demandas de sistemas digitais modernos.

Ana começa delineando o projeto, identificando as entradas (sinais de identificação digital) e as saídas (comandos de acesso permitido ou negado). Ela usa circuitos combinacionais para analisar os sinais de entrada e determinar a resposta apropriada. A tarefa se complica quando Ana precisa projetar um mecanismo de memória temporária que registre tentativas de acesso para análise futura, uma aplicação perfeita para os flip-flops.

Carlos introduz Ana ao processo de design de circuitos, desde a concepção inicial até a simulação. Eles revisam juntos a tabela-verdade para o sistema de controle de acesso, simplificam as expressões lógicas usando mapas de Karnaugh e, finalmente, selecionam os componentes adequados para o circuito.

Ao longo desse processo, Ana e Carlos não apenas expandem seus conhecimentos, mas também fortalecem sua colaboração e habilidades de resolução de problemas, preparando-se para enfrentar os desafios futuros da tecnologia digital com confiança e determinação.

Vamos Começar!

Circuitos Combinacionais

Sistemas digitais empregam circuitos lógicos, classificáveis em dois tipos principais: circuitos combinacionais e circuitos sequenciais. Os circuitos combinacionais usam portas lógicas para realizar cálculos com base nas entradas que são fornecidas. Eles são como calculadoras eletrônicas que produzem resultados com base nas informações inseridas. Imagine que você tem variáveis matemáticas e deseja realizar operações como soma, subtração, multiplicação, etc. O circuito combinacional calcula e apresenta o resultado correspondente com base nas entradas que você fornece (Idoeta, 2019).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Uma diferença importante entre circuitos combinacionais e circuitos sequenciais é que os combinacionais não têm memória. Eles não retêm informações sobre entradas anteriores ou eventos passados. Em vez disso, calculam as saídas apenas com base nas entradas atuais. Isso os torna ideais para tarefas que não dependem do histórico, o que explica sua importância na eletrônica digital.

O projeto de um circuito combinacional segue as etapas listadas:

1. Atribuição de símbolos às variáveis de entrada e saída.
2. Determinação da tabela-verdade a partir da especificação do problema.
3. Obtenção de equações simplificadas com base na tabela-verdade e mapa de Karnaugh.
4. Se necessário, mapeamento do circuito para a biblioteca de portas disponíveis.
5. Criação do desenho do circuito com base nas equações e no mapeamento realizado.

A seguir, abordaremos o projeto de circuitos combinacionais com um exemplo prático: criaremos um circuito que avalia um número binário de 3 bits e determina se ele é menor ou igual ao número decimal 3.

Antes de começarmos o projeto de um circuito combinacional, é fundamental estabelecer com clareza as suas funções. Isso inclui a identificação das entradas e saídas necessárias.

Passos para projetar um circuito combinacional:

1. Definir as variáveis de entrada e saída

No exemplo, projetamos um circuito para verificar se um número binário de 3 bits é menor ou igual a 3. Conforme ilustrado na Figura 1, as entradas são os três bits do número (a, b e c), e a saída é um único bit (y), que indica se o número é menor ou igual a 3 (1 para verdadeiro, 0 para falso).

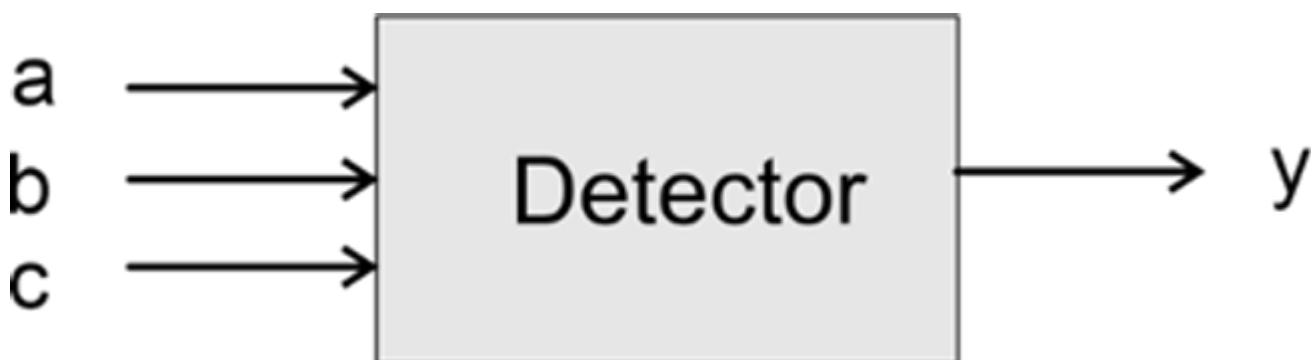


Figura 1 | Exemplo de circuito combinacional. Fonte: elaborada pelo autor.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

2. Identificação do problema (tabela-verdade)

O próximo passo é criar uma tabela-verdade que liste todas as combinações possíveis das entradas e seus resultados correspondentes. Neste exemplo, a saída y será 1 para números de 0 a 3 e 0 para números de 4 a 7.

Bloco 1

Decimal	a	b	c
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Bloco 2

y	
1	
1	
1	
1	
0	
0	

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

	0
	0

Tabela 1 | Tabela-verdade para o problema apresentado. Fonte: elaborada pelo autor.

3. Determinação das equações lógicas simplificadas

A tabela-verdade é uma ferramenta poderosa na criação de equações lógicas que descrevem o comportamento de um circuito. No exemplo dado, a saída y pode ser representada pela seguinte expressão lógica: $a\bar{a} \cdot b\bar{a} \cdot c\bar{a} + a\bar{a} \cdot b\bar{a} \cdot c + a\bar{a} \cdot b \cdot c\bar{a} + a\bar{a} \cdot b \cdot c$. Esses dados são posteriormente utilizados no mapa de Karnaugh para obter uma expressão simplificada.

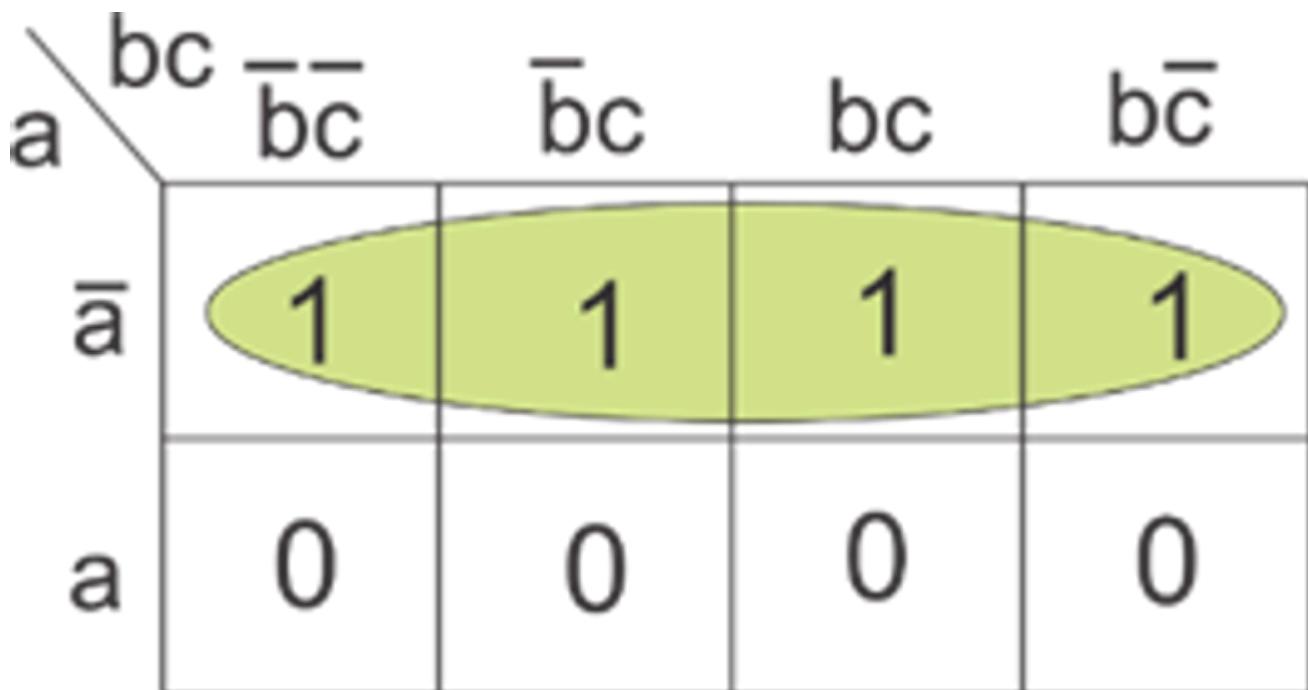


Figura 2 | Mapa de Karnaugh da Tabela 1. Fonte: elaborada pelo autor.

A simplificação revela a capacidade de agrupar os “1s” em uma quadra. Dentro dessa quadra, as variáveis b e c aparecem nas duas formas: b , $b\bar{a}$, c e $c\bar{a}$. No entanto, a variável $a\bar{a}$ permanece constante. Portanto, a quadra é equivalente à variável c , que representa a expressão simplificada de y . Em suma, podemos afirmar que $y = a\bar{a}$.

4. Mapeamento de componentes

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Após a obtenção das equações lógicas, é de extrema importância a seleção dos componentes eletrônicos apropriados, como portas lógicas. Normalmente, usamos portas AND, OR e NOT para construir o circuito. No entanto, em nosso exemplo específico, há necessidade de uma porta inversora, uma vez que a saída y está diretamente conectada à entrada $a\bar{x}$.

5. Desenho do circuito

Por fim, o circuito é desenhado de acordo com as especificações, as entradas a , b e c e a saída y , conforme mostrado na Figura 3. Observe que nesse exemplo é necessário o uso de apenas uma porta inversora.

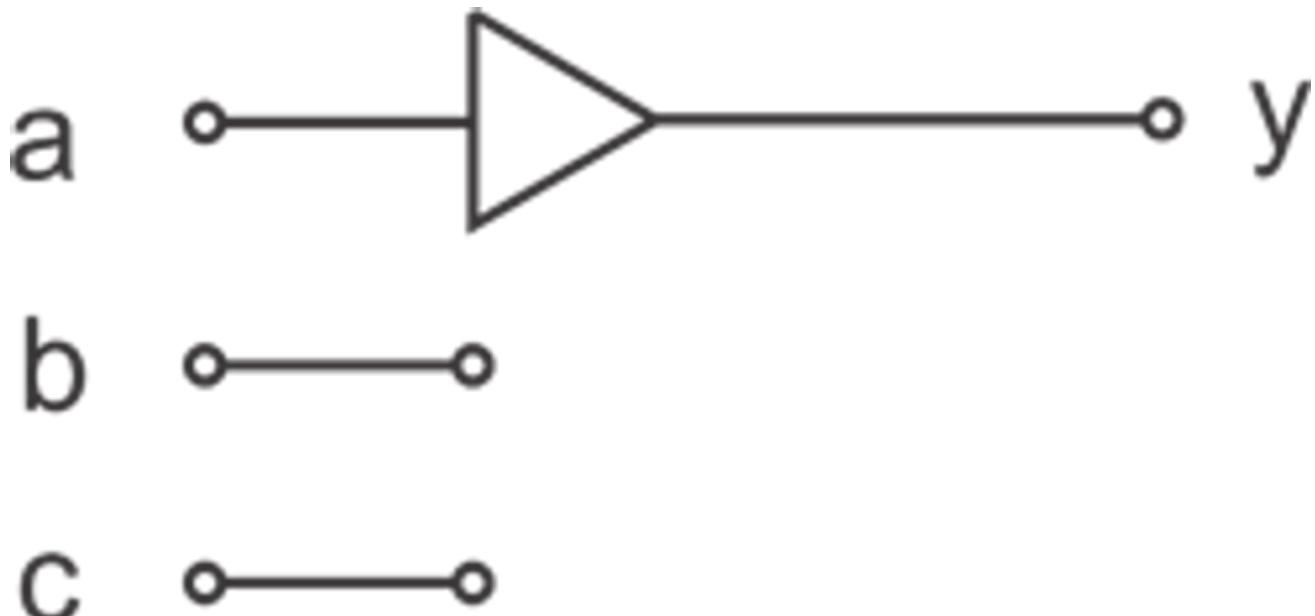


Figura 3 | Circuito simplificado para a expressão $y = c$. Fonte: elaborada pelo autor.

Isso ilustra o projeto de circuitos combinacionais, que inclui a clara definição das entradas e saídas, a formulação de equações lógicas, a seleção de componentes adequados e a montagem do circuito final. Os circuitos combinacionais desempenham um papel vital em sistemas digitais, tomando decisões lógicas instantâneas com base em suas entradas atuais.

Circuitos comparadores são componentes usados para a tomada de decisões com base em comparações entre entradas digitais. Funcionam tipicamente comparando os valores de duas ou mais entradas binárias e gerando uma saída que indica qual delas é maior, menor ou se são iguais. Essa saída é muitas vezes expressa em termos de níveis lógicos, em que um resultado "alto" pode representar verdadeiro ou falso, enquanto um resultado "baixo" representa o oposto. Essa capacidade de comparar números binários é fundamental em circuitos combinacionais, nos quais as decisões lógicas são cruciais (Idoeta, 2019).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Exemplo:

Projetar um circuito combinacional para comparar a magnitude relativa de dois números binários de 2 bits. Neste contexto, consideremos o projeto de um comparador de palavras com dois dígitos, representados como A e B, onde A é formado por A1A0 e B é formado por B1B0. Nesse cenário, a saída do comparador pode assumir três condições distintas: A < B, A = B e A > B, como ilustrado na Tabela 2.

Solução:

Para realizar a comparação entre dois números de 2 bits, representados como A (A1 A0) e B (B1 B0), podemos empregar uma tabela-verdade abrangente que detalha todas as combinações potenciais dos 4 bits sob análise. Isso nos permite visualizar de forma abrangente e sistemática o comportamento do circuito comparador, considerando cada possível estado das entradas A e B e suas respectivas relações de magnitude.

Bloco 1

Decimal	A1	A0	B1
0	0	0	0
1	0	0	0
2	0	0	1
3	0	0	1
4	0	1	0
5	0	1	0
6	0	1	1
7	0	1	1
8	1	0	0
9	1	0	0
10	1	0	1

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

11	1	0	1
12	1	1	0
13	1	1	0
14	1	1	1
15	1	1	1

Bloco 2

B0	A < B	A = B	A > B
0	0	1	0
1	1	0	0
0	1	0	0
1	1	0	0
0	0	0	1
1	0	1	0
0	1	0	0
1	1	0	0
0	0	0	1
1	0	0	1
0	0	1	0
1	1	0	0
0	0	0	1

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

1	0	0	1
0	0	0	1
1	0	1	0

Tabela 2 | Tabela-verdade do comparador. Fonte: elaborada pelo autor.

Temos um circuito lógico com três condições, e representado por três expressões lógicas:

- Quando $A < B$ (mintermos 1, 2, 3, 6, 7 e 11): $\Sigma(A < B) = \Sigma(1, 2, 3, 6, 7, 11)$
- Quando $A = B$ (mintermos 0, 5, 10, 15): $\Sigma(A = B) = \Sigma(0, 5, 10, 15)$
- Quando $A > B$ (mintermos restantes): $\Sigma(A > B) = \Sigma(4, 8, 9, 12, 13, 14)$

A partir da tabela-verdade apresentada, podemos proceder à construção dos mapas de Karnaugh, levando em consideração os mintermos. Neste caso, uma vez que temos três colunas de saída, originam-se três expressões que necessitam de simplificação. Para visualizar esse processo, a figura a seguir ilustra os mapas de Karnaugh e suas respectivas simplificações.



Figura 4 | Mapas de Karnaugh do circuito comparador. Fonte: elaborada pelo autor.

Obtêm-se as seguintes equações simplificadas:

- Quando $A < B$: $A_1 \cdot A_0 \cdot B_0 + A_0 \cdot B_1 \cdot B_0 + A_1 \cdot B_1$
- Quando $A = B$: $(A_0 \oplus B_0) \cdot (A_1 \oplus B_1)$
- Quando $A > B$: $A_0 \cdot B_1 \cdot B_0 + A_1 \cdot A_0 \cdot B_0 + A_1 \cdot B_1$

Note que a expressão usando XOR na condição $A = B$ é obtida aplicando o Teorema de De Morgan.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

O desenho do circuito é representado na Figura 5, fornecendo uma representação visual que facilita a compreensão e a análise do circuito em questão. A figura serve como um guia visual essencial para identificar componentes, conexões e a disposição geral do circuito, tornando mais acessível a interpretação e discussão do seu funcionamento e design.

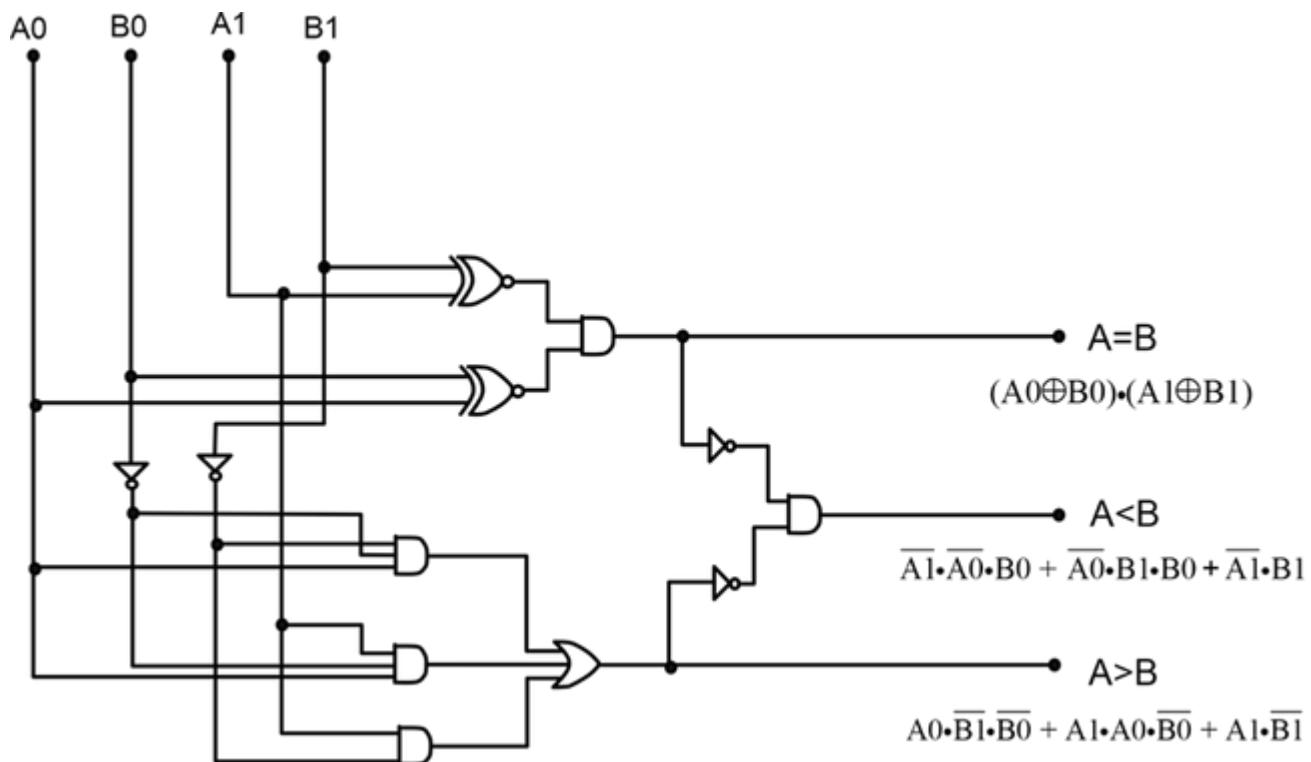


Figura 5 | Representação lógica do circuito comparador. Fonte: elaborada pelo autor.

Circuitos aritméticos são componentes essenciais em sistemas digitais, processadores, calculadoras e outras aplicações que envolvem cálculos em números binários. Projetados para eficiência, garantem precisão e velocidade nas operações matemáticas, desempenhando um papel fundamental no funcionamento destes sistemas.

Siga em Frente...

Circuitos de memória e flip-flop

Os elementos de memória são fundamentais na arquitetura dos sistemas digitais, especialmente nos computadores, onde armazenam e processam grandes quantidades de informações. Esses elementos são categorizados de acordo com sua função, capacidade de armazenamento, e

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

capacidade de leitura e escrita. A informação fornecida abrange uma variedade desses elementos, destacando sua importância, funcionamento, e aplicações específicas.

Os **registradores** são elementos de memória de alta velocidade, mas com capacidade de armazenamento muito limitada. Eles são capazes de armazenar apenas uma palavra (um conjunto de bits) por vez. Apesar de sua velocidade, a necessidade de armazenar grandes quantidades de dados em sistemas digitais exige soluções mais amplas, como as memórias (Idoeta, 2019).

As **memórias** são conjuntos específicos de circuitos capazes de armazenar um grande número de palavras simultaneamente. Existem dois tipos principais de memórias: ROM (*Read-Only Memory*) e RAM (*Random-Access Memory*). A ROM permite apenas a leitura de seu conteúdo, que é gravado durante a fabricação ou por um usuário posteriormente, mas não pode ser modificado. A RAM, por outro lado, permite tanto a leitura quanto a escrita, podendo ter seu conteúdo alterado indeterminadas vezes (Idoeta, 2019).

No nível mais fundamental, os **latches** são elementos primitivos de memória utilizados em microeletrônica para armazenar bits de informação. Eles funcionam com base no princípio de realimentação e podem apresentar problemas como a condição de corrida quando mal projetados. Para evitar esses problemas, existem variações como o latch-SR sensível a nível e o latch-SR do tipo D, que evita a indefinição de estado por meio de uma porta inversora, garantindo que as entradas não sejam 1 ao mesmo tempo quando a porta de enable está desabilitada.

Os **flip-flops**, por sua vez, são uma evolução dos latches, oferecendo uma unidade básica de armazenamento de bit mais estável e confiável. Um flip-flop do tipo D, por exemplo, é sensível à borda do sinal de clock e transmite o sinal de entrada para a saída quando o clock varia. Essa estrutura é fundamental para armazenar dados de forma segura em sistemas digitais e é utilizada em uma variedade de aplicações, desde o armazenamento de memória até a supressão de ruídos em sinais digitais. A seguir, exploraremos os principais tipos de flip-flops: S-R, J-K e D, destacando suas funcionalidades, gestão de estado e aplicações práticas.

O flip-flop S-R opera na borda de subida do clock, alterando seu estado com base nas entradas S (set) e R (reset) após essa transição. A tabela-verdade e as formas de onda ilustram como, dependendo das entradas S e R durante a borda de subida do clock, a saída (Q) pode variar. É crucial evitar que S e R estejam altos simultaneamente para prevenir estados indefinidos (Figura 6).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

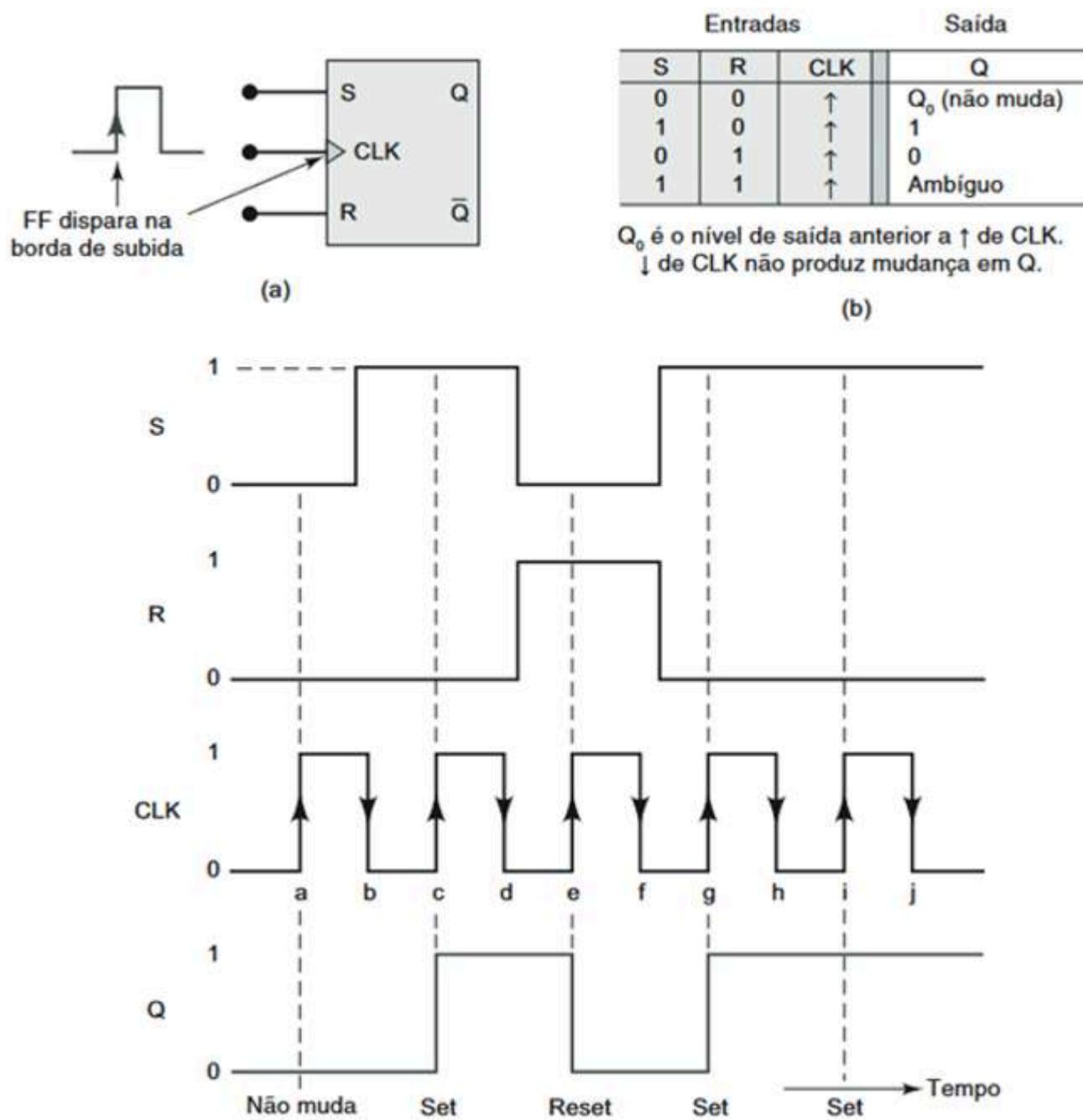


Figura 6 | Flip-flop S-R com clock que responde apenas à borda de subida do pulso de clock; (a) tabela-verdade; (b) formas de onda típicas. Fonte: Tocci e Widmer (2011, p. 188).

O flip-flop J-K, uma evolução do S-R, tem entradas J e K que eliminam os estados ambíguos, permitindo que o estado do flip-flop mude a cada borda de subida do clock quando ambas estão altas. Sua versatilidade o torna ideal para contadores binários e outras aplicações que necessitam de mudanças de estado (Figura 7).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

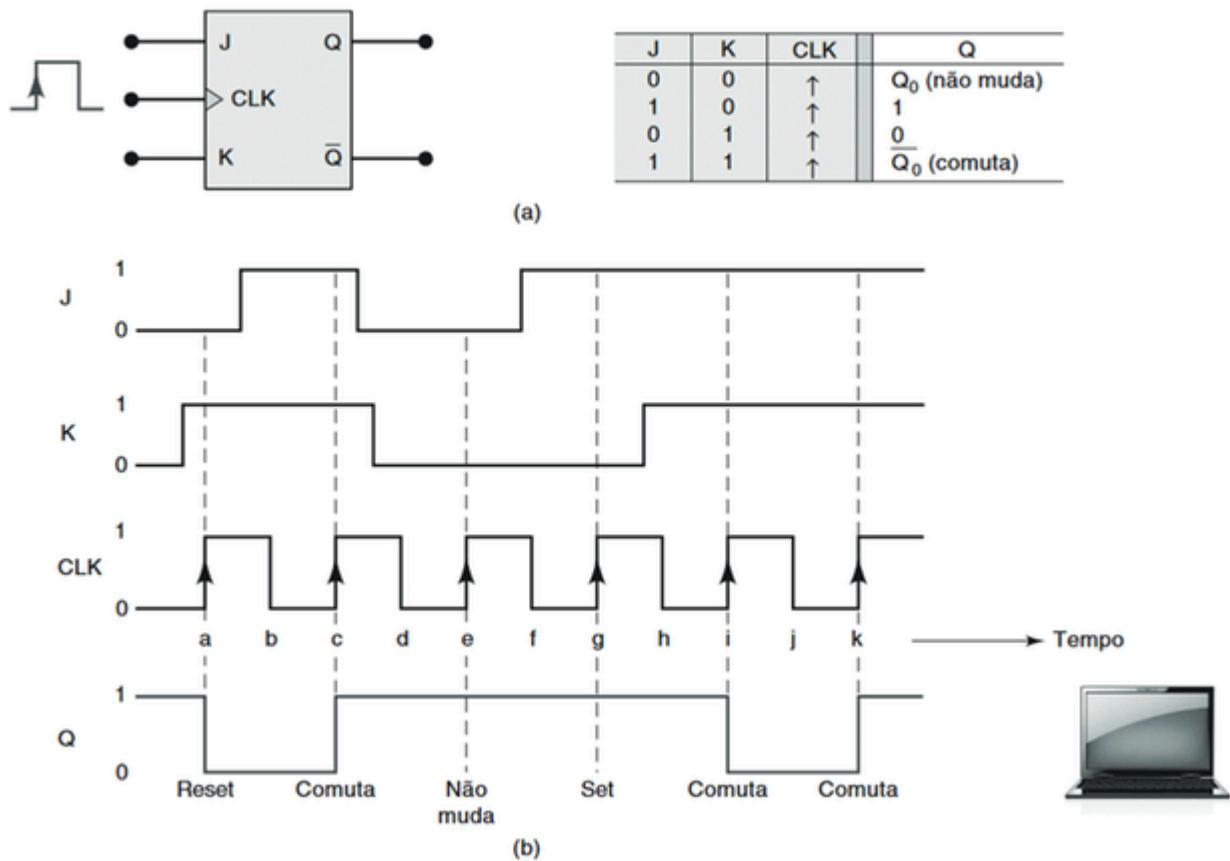


Figura 7 | (a) Flip-flop J-K com clock que responde apenas às bordas positivas do clock; (b) formas de ondas. Fonte: Tocci e Widmer (2011, p.191).

O flip-flop D é notável pela sua simplicidade, com uma única entrada de controle, D (Data), fazendo com que a saída Q assuma o valor de D na borda de subida do clock. Essa previsibilidade o torna perfeito para situações que exigem que a saída siga a entrada, mas somente atualizada nas bordas de subida do clock (Figura 8).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

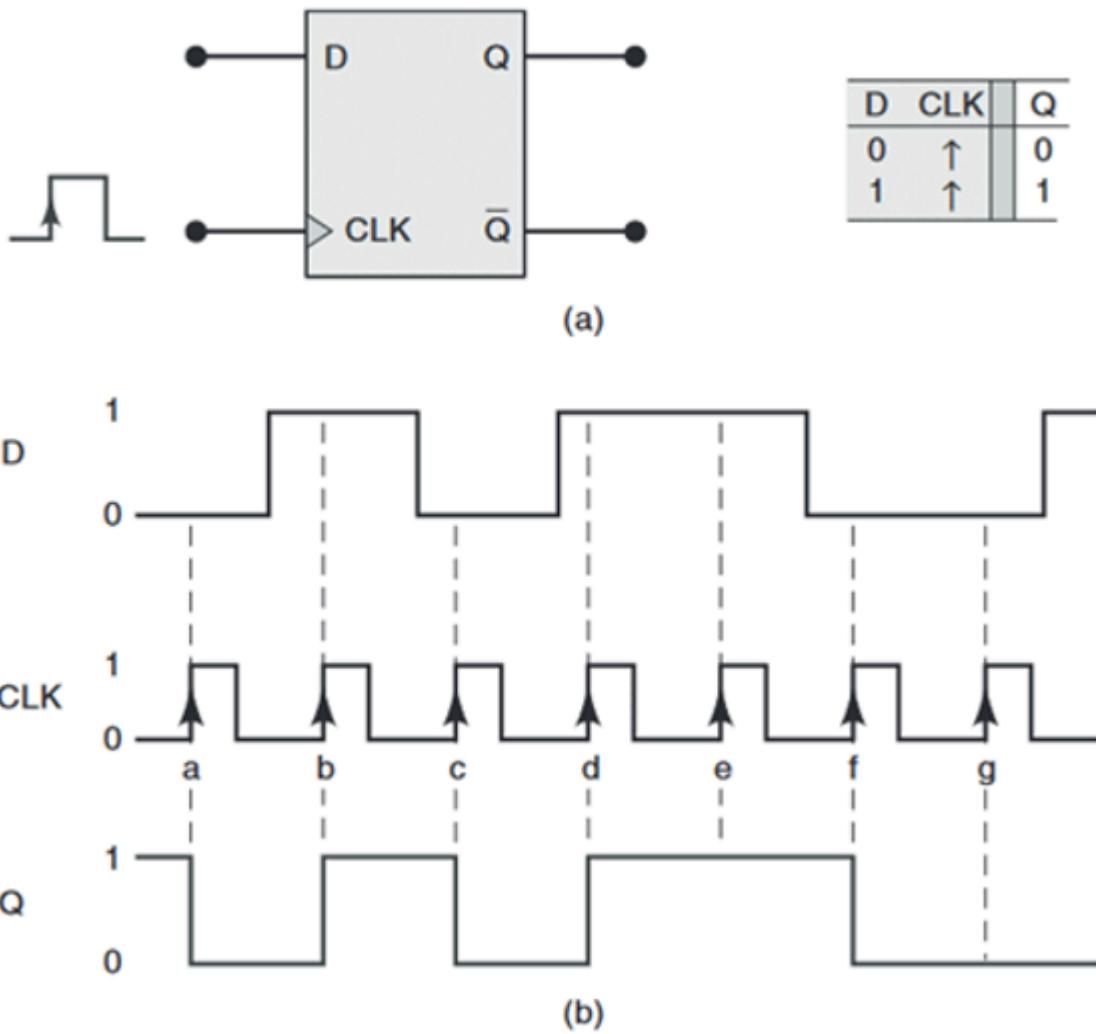


Figura 8 | Flip-flop D com clock que responde apenas à borda de subida do pulso de clock; (a) tabela-verdade; (b) formas de onda típicas. Fonte: Tocci e Widmer (2011, p. 193).

Pontos importantes a serem observados incluem a sensibilidade dos flip-flops às bordas de clock, seja na subida ou, em casos específicos, na descida, e o controle de estado, em que as entradas determinam se a saída muda para alto, baixo ou alterna, dependendo do tipo de flip-flop e das condições das entradas. Eles são aplicados em diversos circuitos digitais, como sistemas de temporização, contadores e registradores de memória, fornecendo uma base para compreender a lógica sequencial e o design de circuitos digitais.

Vamos Exercitar?

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Circuitos: Eficiência, Velocidade, Miniaturização

Após várias iterações e testes, Ana e Carlos conseguem desenvolver um sistema de controle de acesso robusto e confiável. Ana percebe a importância da precisão no design de circuitos combinacionais e a utilidade dos flip-flops na criação de memória temporária para sistemas digitais. Carlos enfatiza a relevância dos conceitos aprendidos, não apenas para o projeto em questão, mas como fundamentos essenciais para o avanço da tecnologia digital. Ele encoraja Ana a continuar explorando e aplicando esses conceitos em projetos futuros, reforçando a ideia de que a inovação tecnológica é impulsionada pela compreensão e aplicação criativa do conhecimento. Ana conclui o projeto com uma nova apreciação pela complexidade e pelo poder dos circuitos digitais. O sucesso do sistema de controle de acesso serve como um testemunho de sua jornada de aprendizado e da importância de uma sólida base teórica aliada à experimentação prática.

Este capítulo da história de Ana Beatriz e Carlos Eduardo destaca como a problematização e a aplicação prática de conceitos teóricos são cruciais no campo da eletrônica digital. Eles demonstram que, com a orientação correta e um compromisso com o aprendizado contínuo, é possível superar desafios técnicos complexos e contribuir para o avanço da tecnologia digital.

Ao longo de nossa jornada, destacamos a importância dos circuitos combinacionais e sequenciais, fundamentais para enfrentar os desafios atuais em eficiência energética, agilidade de processamento e miniaturização de dispositivos. Esses elementos impulsionam avanços em áreas como inteligência artificial, telecomunicações e sistemas computacionais, contribuindo para a sustentabilidade e o progresso tecnológico. Três aspectos essenciais merecem destaque: a eficiência energética dos circuitos sequenciais, a velocidade de processamento dos circuitos combinacionais e a miniaturização de dispositivos eletrônicos. Estamos apenas começando a explorar as vastas possibilidades da eletrônica digital, e você está convidado a desafiar o convencional e imaginar novas inovações que possam surgir desses conceitos. Juntos, podemos moldar o futuro, utilizando de forma criativa e inteligente a eletrônica digital.

Saiba mais

Se você está interessado em aprofundar seus conhecimentos nos fundamentos dos circuitos digitais e explorar a lógica por trás da tecnologia que permeia nosso cotidiano, temos uma sugestão imperdível. O livro [*Circuitos Digitais*](#), de Rodrigo Vinícius Mendonça Pereira e Hugo Tanzarella Teixeira, representa um recurso valioso para aqueles que desejam entender tanto a teoria quanto a aplicação prática dos componentes que constituem a espinha dorsal do mundo tecnológico em que vivemos.

Especialmente na Seção 3.1, intitulada "Flip-flops", que se estende da página 123 à 143, é oferecida uma visão aprofundada de como os circuitos digitais funcionam, complementando os

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

tópicos que tratam de circuitos de memória discutidos em nossas aulas. Este segmento do livro traz clarificações detalhadas e insights da operação desses componentes cruciais. Para começar sua jornada de descoberta, acesse o livro e mergulhe na leitura.

Referências

- CRUZ, E. *et al.* **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. edição. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. *et al.* **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.
- TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. edição. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 4

Circuitos Complexos

Circuitos Complexos



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Explore o mundo dos circuitos digitais nesta videoaula abrangente sobre contadores, registradores e conversores. Esses componentes desempenham um papel importante em uma variedade de aplicações práticas, desde deslocamento de bits até processamento de dados e controle de sistemas. Venha conosco descobrir como esses conceitos fundamentais são aplicados no dia a dia dos profissionais de eletrônica, preparando-o para enfrentar desafios e

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

inovar em sua carreira. Junte-se a nós nesta jornada de aprendizado e descoberta, e leve suas habilidades para o próximo nível.

Ponto de Partida

Circuitos Sequenciais

No quarto capítulo da jornada educacional de Ana Beatriz e Carlos Eduardo, a atenção se volta para os circuitos complexos, mais especificamente, contadores, registradores e conversores. Esses componentes, essenciais para a operação e eficácia de sistemas digitais modernos, apresentam um novo conjunto de desafios e oportunidades de aprendizado.

Ana, agora mais confiante em suas habilidades, se depara com o desafio de projetar um sistema de gerenciamento de energia para um pequeno satélite. Este projeto requer não apenas uma contagem precisa dos eventos, mas também um armazenamento e processamento eficiente dos dados coletados no espaço, onde as condições são imprevisíveis e a precisão é crítica.

Carlos sugere que esse projeto pode se beneficiar significativamente da aplicação prática dos conceitos de contadores, registradores e conversores. Ele ressalta que para garantir a confiabilidade do sistema em condições variáveis, é crucial entender como esses componentes podem ser otimizados para eficiência energética, precisão na contagem e na conversão de dados.

Carlos propõe a seguinte questão: "Como podemos desenvolver um sistema de gerenciamento de energia que seja eficiente, preciso e confiável em condições espaciais, utilizando contadores para monitoramento de eventos, registradores para armazenamento de dados e conversores para a manipulação de dados?" Este desafio destaca a necessidade de uma aplicação inovadora dos conceitos aprendidos, adaptando-os às exigências únicas do ambiente espacial.

Ana começa pelo desenho do sistema, identificando como os contadores podem ser usados para monitorar o consumo de energia e os ciclos operacionais dos instrumentos a bordo. Ela utiliza registradores para armazenar dados críticos que precisam ser analisados para a manutenção da eficiência energética do satélite. Por fim, ela integra conversores para facilitar a comunicação entre os diferentes sistemas a bordo, que operam com diferentes formatos de dados.

Carlos auxilia Ana na seleção dos tipos apropriados de contadores, destacando a importância dos contadores síncronos pela sua precisão e confiabilidade. Ele também aconselha o uso de registradores de deslocamento que podem efetivamente gerenciar o fluxo de dados entre os sistemas do satélite.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

No desenrolar deste capítulo, Ana e Carlos mergulham no intrincado mundo dos contadores, registradores e conversores, essenciais para a operação de sistemas digitais avançados. Diante do desafio de projetar um sistema de gerenciamento de energia para um satélite, eles exploram como esses componentes podem ser aplicados de forma eficiente e precisa. Esta jornada destaca a importância da adaptação dos conceitos teóricos à realidade prática, preparando-os para enfrentar os desafios únicos do ambiente espacial.

Vamos Começar!

Contadores

Os contadores são circuitos digitais que encontram ampla aplicação em projetos de sistemas eletrônicos, especialmente para operações de temporização e sequenciamento. Eles podem ser classificados em dois tipos principais: síncronos e assíncronos (ou *ripple counters*), além de variações como contadores em anel. O diagrama da Figura 1 mostra a estrutura básica de um contador síncrono, no qual o flip-flop tipo D armazena o estado atual do contador e é atualizado a cada pulso do clock. A lógica combinacional define o próximo estado do contador com base no estado atual. Este arranjo garante que as transições de estado sejam feitas de forma ordenada e previsível, característica essencial para a operação confiável do contador em aplicações de temporização e sequenciamento (Idoeta, 2019).

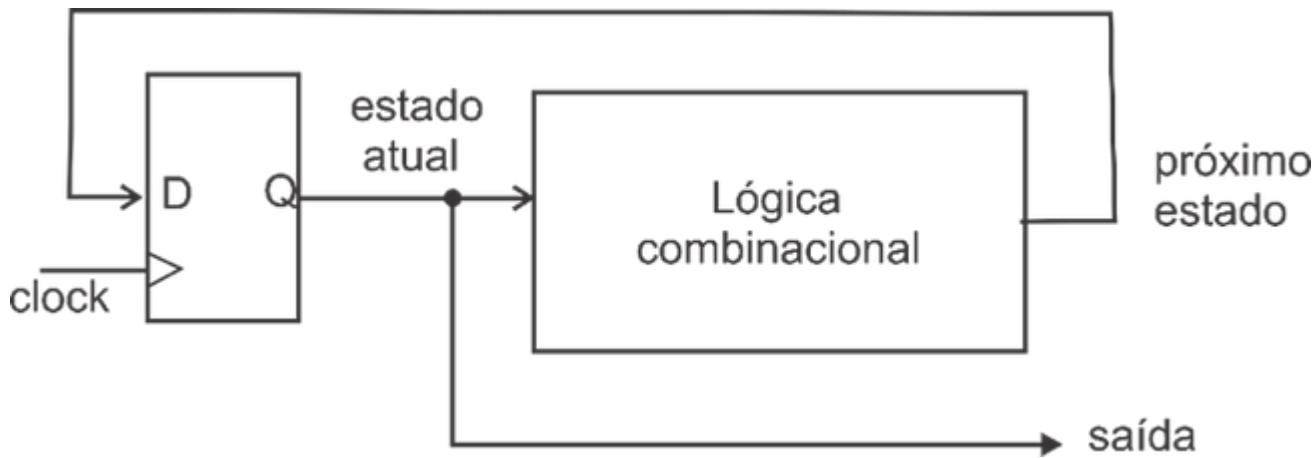


Figura 1 | Modelo de um contador síncrono. Fonte: elaborada pelo autor.

Contadores Síncronos

Contadores síncronos avançam seus estados em sincronismo com o sinal de relógio. Cada flip-flop (elemento de armazenamento) recebe o pulso de relógio simultaneamente, garantindo uma transição de estado coerente e previsível. Essa característica elimina os atrasos de propagação

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

que podem levar à instabilidade em contadores assíncronos. Exemplos incluem contadores módulo-2, módulo-4, módulo-8 e módulo-16, cada um referindo-se ao número de estados únicos que o contador pode representar antes de retornar ao seu estado inicial. A sequência de contagem e a lógica de transição de estado são definidas por tabelas de verdade e implementadas usando flip-flops e lógica combinacional.

Com base na Figura 2, pode-se observar que o contador síncrono módulo-2 é construído utilizando um flip-flop tipo D com a saída Q retroalimentada através de uma porta NOT para a entrada D. A cada borda de subida do sinal de clock, a entrada D recebe o inverso do estado atual de Q, causando a mudança do estado de Q na próxima borda de subida. Este comportamento resulta na alternância da saída Q entre 0 e 1 a cada ciclo do relógio, efetivamente dividindo a frequência do sinal de relógio por 2, pois são necessários dois ciclos do relógio para que a saída complete uma sequência de 0 a 1 e retorne a 0. A tabela à direita ilustra a sequência de estados do flip-flop (estado atual Q e estado seguinte D, confirmando a operação de contagem módulo-2 (Idoeta, 2019).

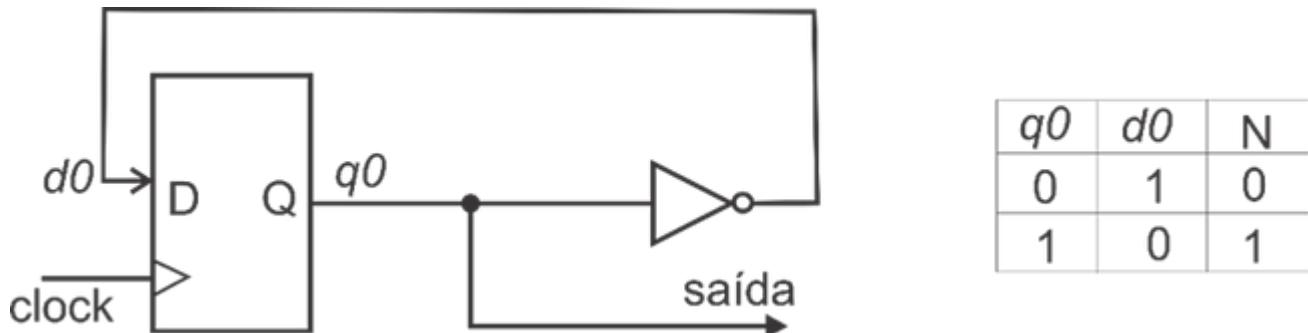
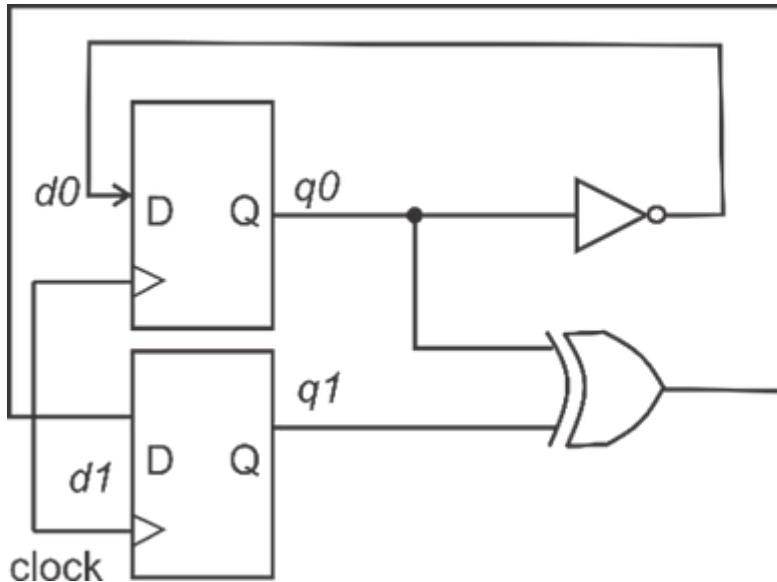


Figura 2 | Diagrama de um contador síncrono módulo-2 com tabela de transição de estados. Fonte: elaborada pelo autor.

Para o contador síncrono módulo-4, a configuração utiliza dois flip-flops tipo D em cascata, com a saída do primeiro conectada à entrada do segundo. O contador avança na sequência binária 00, 01, 10, 11 e retorna a 00, resultando na contagem de 0 a 3 (quatro estados distintos), o que divide a frequência do relógio de entrada por 4.

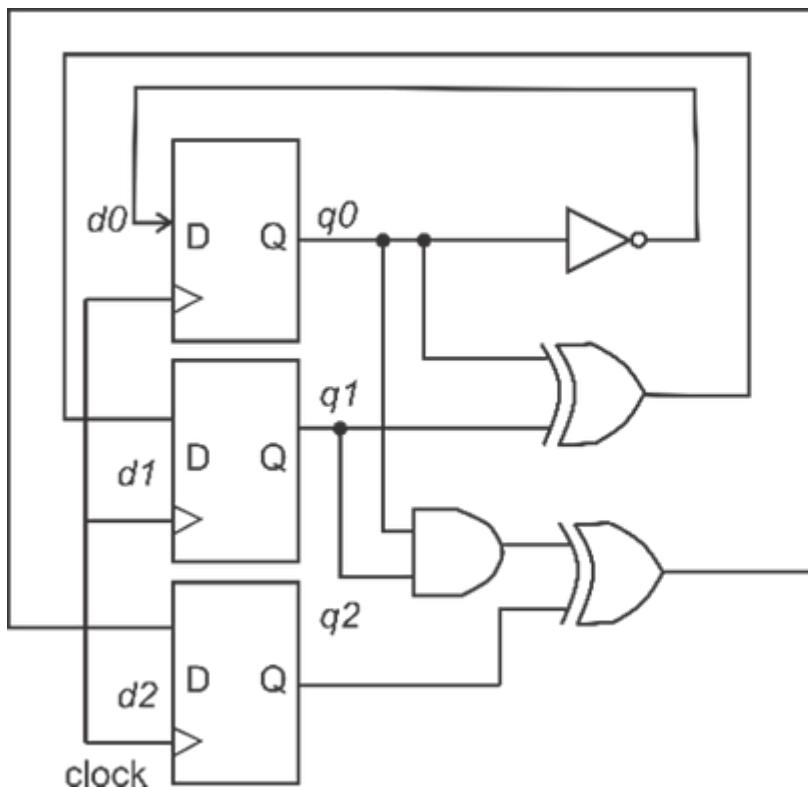
SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



q_1	q_0	d_1	d_0	N
0	0	0	1	1
0	1	1	0	2
1	0	1	1	3
1	1	0	0	0

Figura 3 | Diagrama de um contador síncrono módulo-4 com tabela de transição de estados. Fonte: elaborada pelo autor.

O contador síncrono módulo-8 da Figura 4 utiliza uma série de três flip-flops para realizar a contagem de 0 a 7, correspondendo a uma sequência binária de 000 a 111.



q_2	q_1	q_0	d_2	d_1	d_0	N
0	0	0	0	0	1	1
0	0	1	0	1	0	2
0	1	0	0	1	1	3
0	1	1	1	0	0	4
1	0	0	1	0	1	5
1	0	1	1	1	0	6
1	1	0	1	1	1	7
1	1	1	0	0	0	0

Figura 4 | Diagrama de um contador síncrono módulo-8 com tabela de transição de estados. Fonte: elaborada pelo autor.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

As formas de onda, na Figura 5, ilustram como a saída de cada flip-flop q_0 , q_1 , e q_2 muda em resposta ao sinal de relógio (clock), mostrando a progressão da contagem em estados binários. Após atingir o estado 111, o contador reinicia a sequência para 000, completando o ciclo de contagem e dividindo a frequência do relógio por 8. A linha "Saída" representa a saída combinada dos três flip-flops, fornecendo uma representação visual clara de como a contagem progride ao longo do tempo com cada pulso de relógio (Idoeta, 2019).

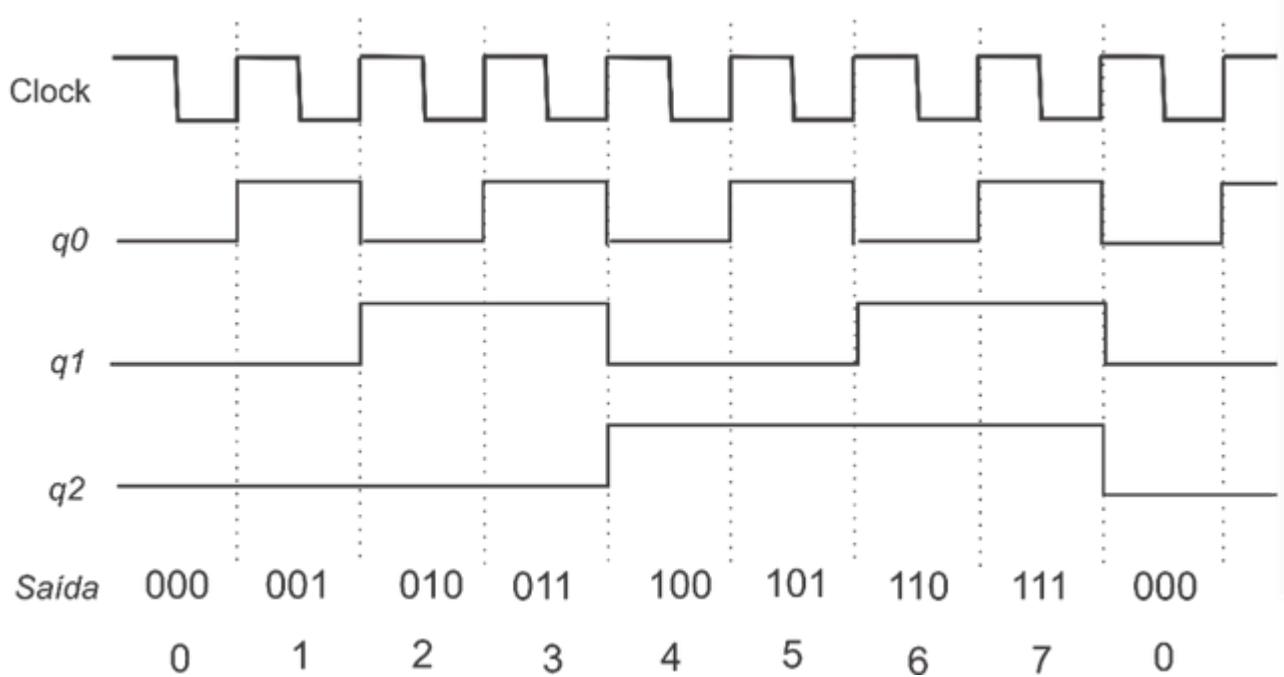


Figura 5 | Formas de onda de um contador síncrono módulo-8 e progressão de estados binários. Fonte: elaborada pelo autor.

Contadores Assíncronos (*Ripple Counters*)

Em contraste, os contadores assíncronos não sincronizam a transição de estado de todos os flip-flops com o sinal de relógio. Em vez disso, o estado de cada flip-flop é alterado pela transição do estado do flip-flop anterior, criando um efeito de "ondulação" através do contador. Embora mais simples e usando menos hardware que os contadores síncronos, eles são suscetíveis a problemas de temporização devido a atrasos de propagação, o que pode levar a estados indesejados ou a uma contagem imprecisa (Idoeta, 2019).

A Figura 6 representa um contador assíncrono, também conhecido como *ripple counter*. Nesse tipo de contador, cada flip-flop é acionado pela borda de descida da saída do flip-flop imediatamente anterior, o que causa o efeito de "ripple" ou propagação do sinal através da cadeia de flip-flops, daí o nome. Isso é evidenciado no diagrama de tempo pela transição retardada das saídas q_1 e q_2 em relação à saída q_0 e ao sinal de clock.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

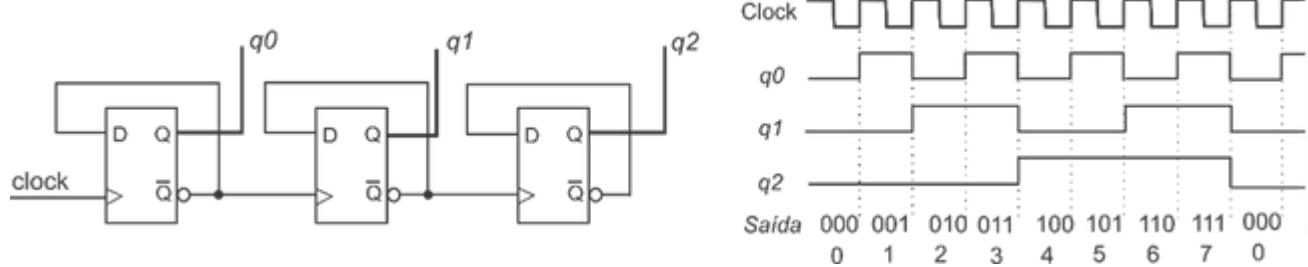


Figura 6 | Circuito e diagrama de tempo de um contador assíncrono (ripple counter) módulo. Fonte: elaborada pelo autor.

Contador em Anel

O contador em anel é um tipo de circuito digital que utiliza uma configuração inovadora de flip-flops para realizar contagem sequencial, conforme mostrado na Figura 7. Esta disposição garante que, em qualquer momento, apenas um flip-flop esteja no estado lógico alto (1). O estado alto se propaga sequencialmente de um flip-flop para o próximo a cada ciclo de clock, efetuando uma “rotação” do sinal por todo o anel. Para iniciar a contagem em anel, os flip-flops são carregados com uma condição inicial específica, usualmente com o primeiro flip-flop em estado alto e os demais em estado baixo (0). A saída de cada flip-flop serve como entrada para o seguinte, criando a dinâmica de “caminhada” do estado alto. Contadores em anel são particularmente valiosos para a geração de sequências de controle e para a administração precisa de temporização em aplicações digitais, devido à sua natureza previsível e cíclica.

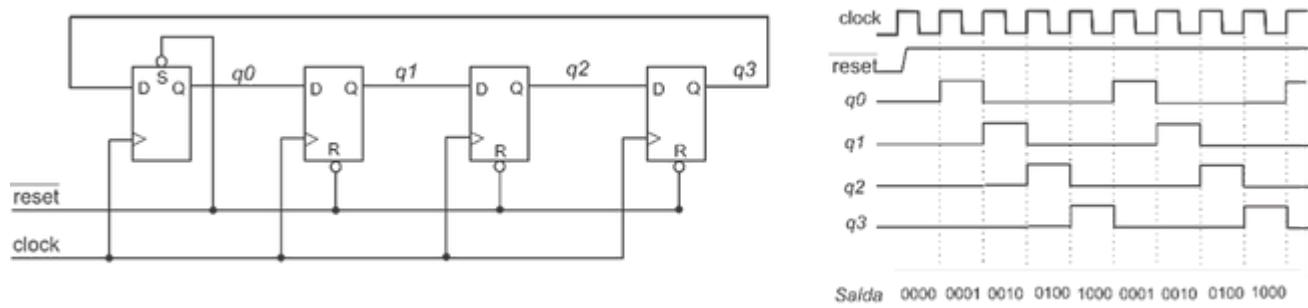


Figura 7 | Diagrama de circuito e formas de onda de um contador em anel de 4 bits com reset. Fonte: elaborada pelo autor.

O projeto de um contador depende de requisitos específicos, incluindo velocidade, precisão da contagem e complexidade do hardware. Contadores síncronos são preferidos em aplicações de alta velocidade e precisão devido à sua estabilidade e previsibilidade, enquanto contadores assíncronos e em anel encontram uso em aplicações que requerem simplicidade de design e menor uso de recursos.

Siga em Frente...

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Registradores e conversores

Os registradores de deslocamento desempenham funções importantes no processamento de dados, conversão de formatos de dados, e temporização em circuitos. Eles manipulam dados permitindo o deslocamento de bits para a direita ou esquerda, além de possibilitarem a carga e leitura de dados em modos série ou paralelo. Essa flexibilidade torna os registradores de deslocamento ferramentas poderosas em design de circuitos digitais.

Registradores de Deslocamento Série-Paralelo

A Figura 8 adiante mostra um circuito de registrador de deslocamento série-paralelo de 4 bits. Esses registradores são utilizados para armazenar e deslocar dados; eles recebem dados serialmente, ou seja, bit a bit em sequência, através da entrada "d". Com cada pulso de clock, os dados são deslocados de um flip-flop para o seguinte. Após quatro pulsos de clock, todos os bits são transferidos para as saídas q0, q1, q2 e q3, podendo então ser lidos paralelamente.

Os registradores de deslocamento série-paralelo são fundamentais em aplicações que requerem a conversão de dados de série para paralelo, como mencionado no texto fornecido. Isso permite que um sistema que recebe dados de forma serial, como uma interface de comunicação, possa depois manipular esses dados simultaneamente, facilitando o processamento por parte de outros componentes digitais que operam paralelamente. Eles desempenham um papel crucial na conversão de dados, atuando efetivamente como conversores série-paralelo (Idoeta, 2019).

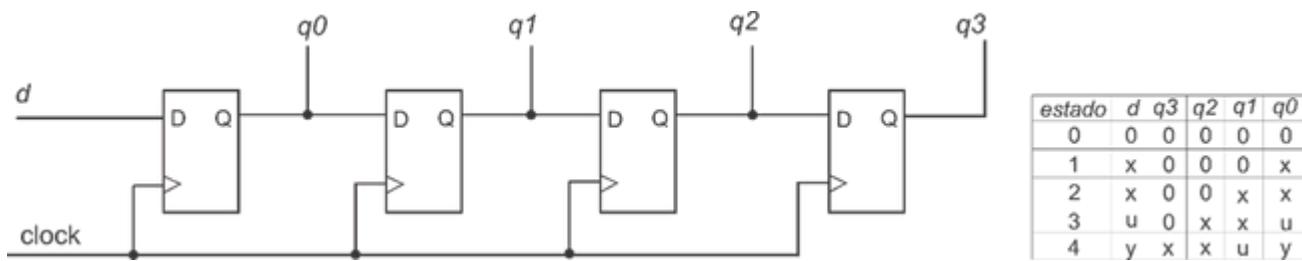


Figura 8 | Circuito de registrador de deslocamento série-paralelo de 4 bits e tabela de estados. Fonte: elaborada pelo autor.

Registradores de Deslocamento Paralelo-Série

Por outro lado, os registradores de deslocamento paralelo-série funcionam ao receber um grupo de bits de uma vez (em paralelo), armazenando-os internamente, e então liberando esses bits sequencialmente (um bit por ciclo de relógio) em modo série. Essa funcionalidade é fundamental para a conversão de dados de paralelo para série, permitindo a transmissão de dados armazenados internamente em um dispositivo para outro dispositivo através de uma única linha de comunicação.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

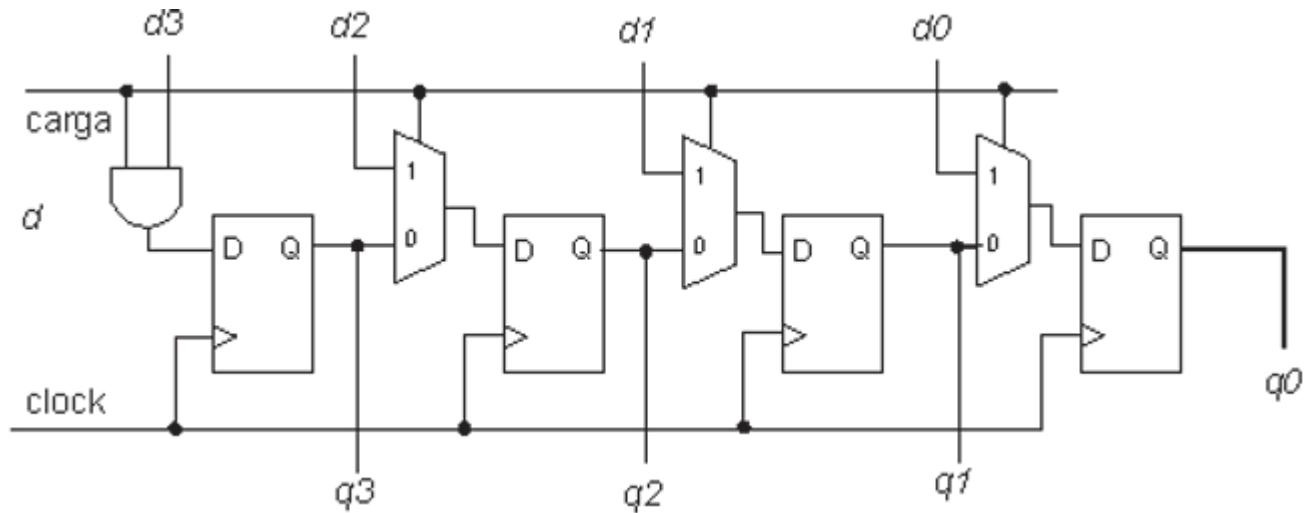


Figura 9 | Registrador de deslocamento paralelo-série. Fonte: elaborada pelo autor.

A Figura 10 a seguir ilustra de maneira simplificada os dois tipos de registradores de deslocamento: série-paralelo e paralelo-série. No registrador série-paralelo, a entrada serial (D0) é convertida em saídas paralelas (Q0 a Q3) após uma série de ciclos de clock, permitindo a manipulação de bits individuais simultaneamente. Inversamente, o registrador paralelo-série recebe entradas paralelas (D0 a D3), armazena esses dados e, em seguida, os libera bit a bit de forma serial pela saída (Q3), ideal para a transmissão sequencial de dados através de um único canal de comunicação. Esta funcionalidade bidirecional destes registradores é crucial para a conversão de dados dentro de sistemas digitais, servindo tanto como conversores série-paralelo quanto paralelo-série, dependendo da direção do fluxo de dados (Idoeta, 2019).

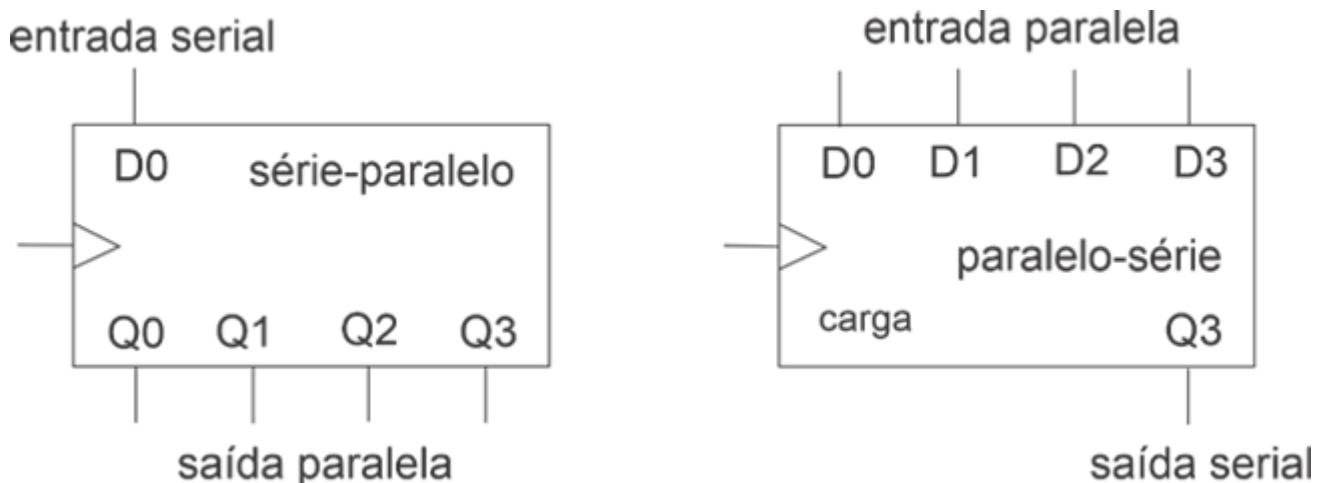


Figura 10 | Diagrama de registradores de deslocamento série-paralelo e paralelo-série. Fonte: elaborada pelo autor.

A escolha entre diferentes tipos de registradores de deslocamento depende dos requisitos específicos do projeto, incluindo a necessidade de conversão de dados, a velocidade de

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

operação, e a complexidade do processamento de dados desejado. A habilidade de manipular dados de maneira flexível torna os registradores de deslocamento componentes indispensáveis no projeto de sistemas digitais modernos.

Vamos Exercitar?

Contadores e registradores na prática

Com a orientação de Carlos, Ana desenvolve um protótipo do sistema de gerenciamento de energia. Os testes revelam a eficácia dos contadores na monitoração precisa dos eventos, dos registradores no armazenamento eficiente de dados e dos conversores na otimização da comunicação de dados. Ana percebe a importância de compreender não apenas a teoria por trás desses componentes, mas também suas aplicações práticas em projetos reais.

Carlos destaca a jornada de aprendizado de Ana, do básico ao complexo, enfatizando como a compreensão profunda dos sistemas digitais e a capacidade de aplicar esses conceitos em situações reais são fundamentais para qualquer engenheiro. Ele encoraja Ana a continuar explorando e aplicando seus conhecimentos em novos desafios.

A experiência no projeto do sistema de gerenciamento de energia para o satélite marca um ponto de virada na educação de Ana. Ela agora tem o conhecimento teórico necessário e a confiança para aplicar esses conceitos em projetos complexos e em condições desafiadoras.

Ao aplicar estes conceitos fundamentais, garantimos uma operação precisa do sistema de controle e abrimos portas para a inovação. Refletindo acerca de outras possíveis soluções, podemos considerar a implementação de algoritmos de correção de erro para lidar com as variações do clock ou a utilização de técnicas de design redundante para aumentar a confiabilidade do sistema.

Encorajamos você a pensar nesses conteúdos e a considerar como eles podem ser aplicados em outras situações práticas. Quais outras aplicações em sistemas digitais podem se beneficiar desses princípios? Como você pode usar o que aprendeu nesta aula para inovar seus próprios projetos?

Agora é sua vez de explorar e aplicar esses conceitos. Vamos avançar no seu entendimento e habilidades práticas. Continue a jornada de aprendizado e veja até onde ela pode levar você e suas inovações no campo da eletrônica digital.

Saiba mais

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Para aprofundar seu entendimento dos conceitos abordados nesta aula e expandir seu conhecimento no fascinante mundo dos circuitos digitais, recomendamos o capítulo "Contadores e Registradores" do livro [*Circuitos Digitais*](#). Este capítulo oferece uma visão detalhada dos mecanismos internos e das aplicações práticas de contadores e registradores em circuitos digitais, complementando o que foi discutido durante nossa videoaula.

Você pode acessar o capítulo recomendado e outros conteúdos relacionados na Biblioteca Virtual, começando na página 123 até a página 152.

Dê o próximo passo no seu aprendizado, explorando as nuances e aplicações práticas que os circuitos digitais oferecem. Boa leitura e descobertas!

Referências

- CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. edição. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. et al. **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Nesta videoaula de encerramento, revisitaremos os pilares dos sistemas digitais, abordando a importância de cada conceito na construção de uma base sólida para futuros projetos tecnológicos. Destacaremos como o conhecimento de portas lógicas, álgebra booleana, circuitos combinacionais, contadores, registradores e conversores podem ser aplicados no mundo profissional, abrindo portas para inovações e soluções eficazes. Junte-se a nós para consolidar sua aprendizagem e preparar-se para aplicar esses conhecimentos em sua carreira. Não perca!

Ponto de Chegada

Fundamentação da Eletrônica Digital

Olá, estudante! Durante as aulas, você mergulhou profundamente nos conceitos de sistemas digitais, abrangendo desde as fundamentações teóricas dos sistemas digitais e analógicos, passando por portas lógicas e álgebra booleana, até a aplicação prática em circuitos combinacionais, contadores, registradores e conversores. Essa jornada de conhecimento é vital para desenvolver a competência necessária nesta unidade: a capacidade de compreender e projetar sistemas digitais complexos, integrando teoria e prática para criar soluções inovadoras em tecnologia digital.

Ao desenvolver essa competência, você é encorajado a aplicar os conhecimentos adquiridos de forma criativa e crítica, visando não apenas compreender os conceitos estudados, mas também ser capaz de projetar e otimizar circuitos digitais. Esta habilidade é crucial para enfrentar e resolver desafios reais de engenharia, permitindo-lhe contribuir significativamente para avanços tecnológicos no campo da eletrônica digital.

No domínio dos sistemas digitais, distinguem-se os circuitos combinacionais e sequenciais, ambos essenciais na construção de estruturas lógicas e no armazenamento de estados, respectivamente. Os desafios de eficiência energética, velocidade de processamento e miniaturização de dispositivos são diretamente influenciados pela capacidade de inovar no design e aplicação destes circuitos.

A aplicação prática destes conceitos é ilustrada no processo de design de circuitos combinacionais, envolvendo a criação de tabelas-verdade e mapas de Karnaugh para simplificar a lógica dos circuitos, otimizando os recursos eletrônicos disponíveis. Circuitos como os flip-flops D, sensíveis às bordas de clock, são vitais em sistemas sequenciais, facilitando a construção de sistemas de temporização e registradores de memória, ampliando as possibilidades de aplicação da lógica sequencial.

Este caminho percorrido tanto reforça sua compreensão teórica quanto aprimora suas habilidades práticas. Está claro que os conhecimentos adquiridos acerca de sistemas digitais, desde os fundamentos até os aspectos mais complexos de sua aplicação, são essenciais para

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

sua formação e atuação profissional. Continue explorando, questionando e aplicando esses conceitos, pois cada desafio superado é um passo adiante em sua jornada no vasto campo da eletrônica e tecnologia digital.

É Hora de Praticar!

Imagine que você é um engenheiro de software trabalhando em uma startup de tecnologia chamada **HealthTech Innovations**. A empresa está desenvolvendo um dispositivo vestível para monitoramento contínuo da saúde, voltado para pacientes com condições crônicas. O dispositivo deve ser capaz de medir sinais vitais como frequência cardíaca, níveis de oxigênio no sangue e temperatura corporal, processar os dados em tempo real, e enviar alertas em caso de anomalias.

Personagens e Empresas Envoltas

- **Engenheiro de Software (Você):** Responsável pelo desenvolvimento do sistema de monitoramento e processamento de dados.
- **Engenheiro de Hardware (Carlos):** Cuida da integração dos sensores e da arquitetura de hardware do dispositivo.
- **Médico Consultor (Dr. Silva):** Fornece orientação sobre os parâmetros de saúde que devem ser monitorados.
- **HealthTech Innovations:** Empresa desenvolvedora do dispositivo.

Problema Proposto

O dispositivo vestível deve ser capaz de:

- Coletar dados de múltiplos sensores em tempo real.
- Processar os sinais vitais e identificar anomalias que possam indicar problemas de saúde.
- Enviar alertas para o paciente e o médico automaticamente em caso de detecção de valores fora dos padrões normais.

Como garantir que o dispositivo processa e analisa os dados dos sensores em tempo real de forma eficiente?

Quais circuitos e componentes digitais são mais adequados para o processamento e armazenamento dos dados vitais?

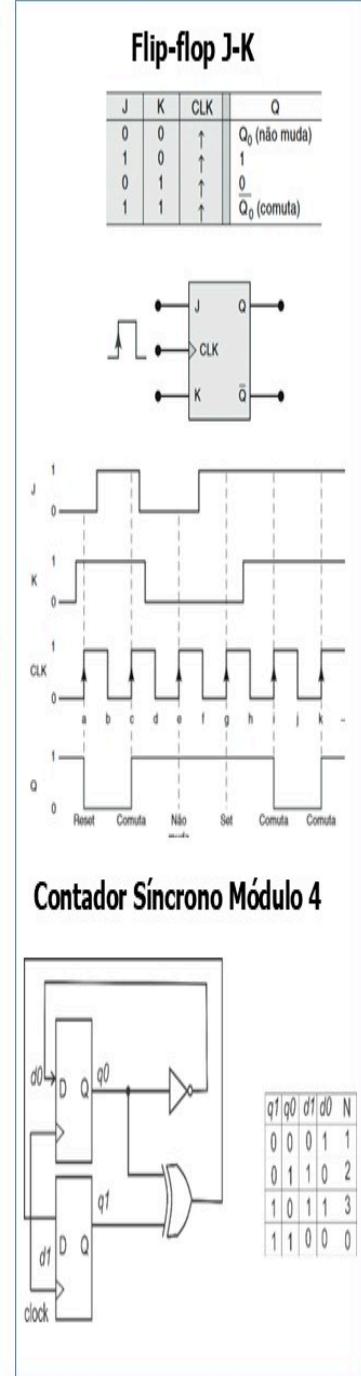
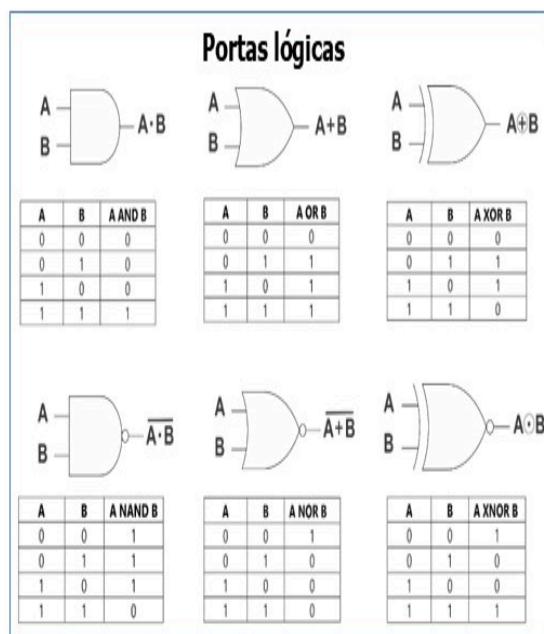
Como garantir a confiabilidade e precisão dos dados coletados e processados pelo dispositivo?

Olá estudante, chegamos ao encerramento da unidade!

Vamos realizar a experiência presencial que irá consolidar os conhecimentos adquiridos? É a oportunidade perfeita para aplicar, na prática, o que foi aprendido em sua disciplina. Vamos

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

transformar teoria em vivência e tornar esta etapa ainda mais significativa. Não perca essa chance única de colocar em prática o conhecimento adquirido.



SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital.** 42. edição. São Paulo: Érica, 2019.
LENZ, M. L.; TORRES, F. E. **Microprocessadores.** Porto Alegre: SAGAH, 2019.
SOUZA, D. B. da C. et al. **Sistemas digitais.** Porto Alegre: SER – SAGAH, 2018.

Unidade 2

MICROPROCESSADOR

Aula 1

Introdução aos Microprocessadores

Introdução Aos Microprocessadores



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Descubra o coração pulsante da tecnologia moderna em nossa próxima videoaula sobre microprocessadores! Exploraremos o funcionamento interno desses componentes cruciais, desde a unidade lógica e aritmética até a memória cache, ilustrando como eles processam e executam instruções que impulsionam o mundo digital. Esta aula não apenas solidifica conceitos técnicos, mas também ponteia a teoria com aplicações práticas, preparando você para inovações no campo da engenharia e computação. Junte-se a nós para desvendar os segredos dos sistemas digitais e elevar suas habilidades ao próximo nível.

Ponto de Partida

No coração da inovação tecnológica, os microprocessadores emergem como fundamentais para a nossa interação diária com a tecnologia. Eles são a força motriz por trás de dispositivos que

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



vão de smartphones a sistemas de automação doméstica, e compreendê-los é vital não só para engenheiros e desenvolvedores, mas para todos nós.

Ana Beatriz, guiada por Carlos Eduardo, ilustrados na Figura 1, mergulha na prática, utilizando os conceitos dos microprocessadores para automatizar uma tarefa doméstica, como o controle de iluminação via sensores de movimento. Este projeto prático aprofunda seu entendimento da teoria e destaca a necessidade crítica de integrar eficazmente os componentes do sistema para solucionar problemas reais.

Este entendimento aprofundado é crucial para qualquer um que aspire ser engenheiro ou entusiasta da tecnologia. Ao nos juntarmos a Ana e Carlos nesta jornada educativa, somos convidados a explorar os fundamentos dos sistemas digitais e a aplicar esse conhecimento de forma prática, impulsionando nossas carreiras e contribuindo para o avanço tecnológico.

Prepare-se para mergulhar no mundo dos sistemas digitais e descubra, assim como Ana, as inúmeras oportunidades de aplicar esses ensinamentos de maneira prática. Esta é a sua chance de avançar na jornada de aprendizado e inovação. Vamos juntos desvendar as máquinas de estados e os circuitos complexos, empunhando a tecnologia para criar um futuro melhor.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Figura 1 | Ana e Carlos: parceria em automação residencial. Fonte: Cogna IA.

Vamos Começar!

Entendendo os Microprocessadores

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Ao mergulharmos no universo dos sistemas digitais, nosso ponto de partida são os microprocessadores. Esses componentes, também conhecidos como CPUs, são verdadeiramente o cérebro dos computadores, vide Figura 2, desempenhando um papel crucial no processamento e execução de programas. Mas o que realmente faz um microprocessador funcionar? Como ele se integra aos vastos sistemas digitais que definem a nossa era tecnológica?

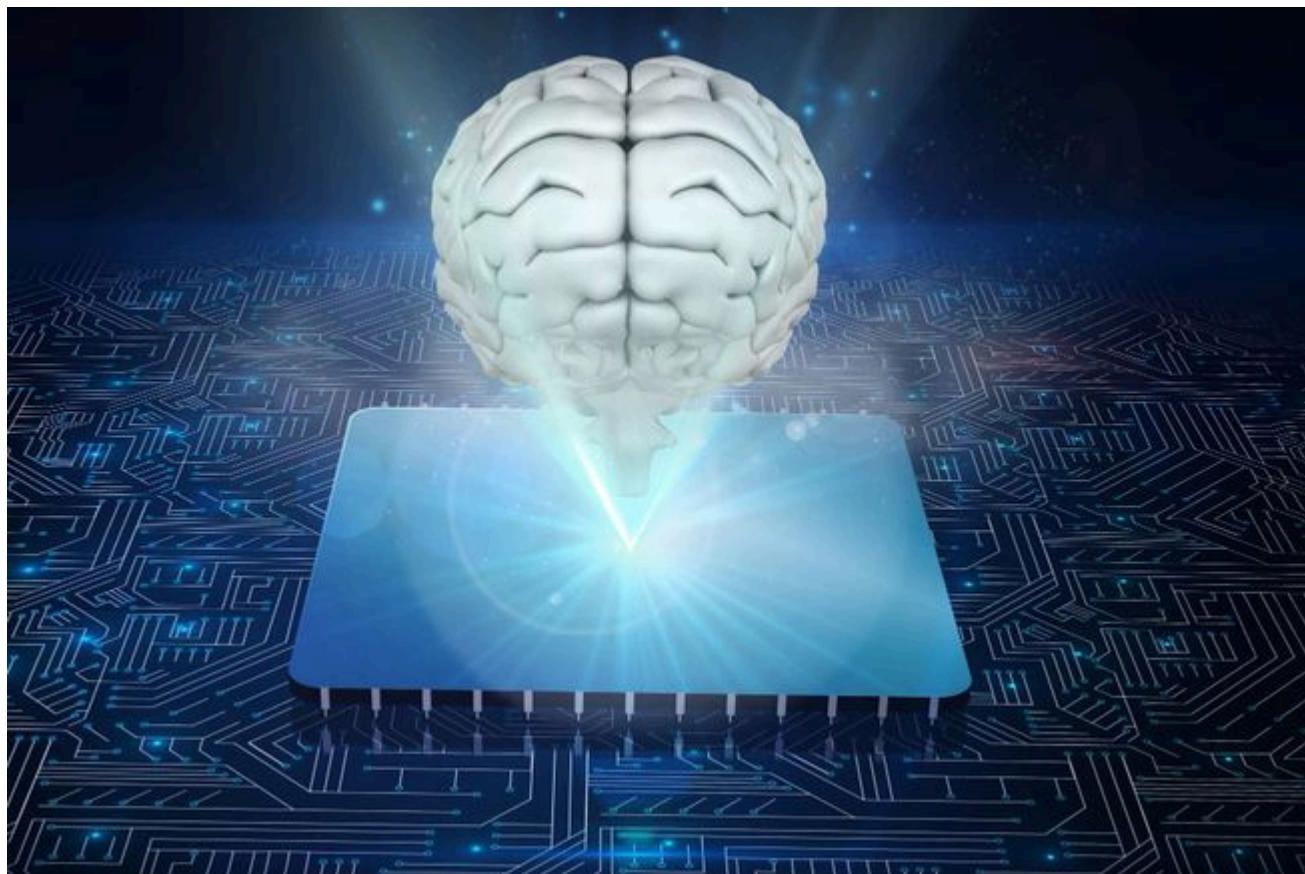


Figura 2 | Microprocessador: cérebro do computador. Fonte: Freepik.

A Figura 3 ilustra a arquitetura fundamental de uma unidade central de processamento (CPU, do termo em inglês *Central Processing Unit*), o componente essencial que orquestra as funções de um computador. Central para este esquema são a unidade lógica/aritmética (ULA) e o bloco de controle, ambos responsáveis pelas operações críticas de processamento e execução de decisões. A ULA executa cálculos matemáticos e operações lógicas, enquanto os registradores facilitam o armazenamento temporário de dados.

O fluxo de dados é simples: informações são recebidas pela CPU, processadas pela ULA sob a coordenação do bloco de controle, e, finalmente, os resultados são enviados para fora do

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

sistema como saída. A seguir, encontramos a memória, essencial para armazenar tanto as instruções a serem executadas quanto os dados a serem processados.

A atividade dentro da unidade funcional de controle é meticulosamente sincronizada, com o contador de instrução (CI) rastreando a sequência de execução e o registrador de instruções (RI) mantendo a instrução atual. O decodificador de instruções traduz as instruções em ações compreensíveis para a CPU. Este arranjo demonstra a complexidade e precisão do processamento digital e reflete a elegância do design que torna possível a computação moderna.

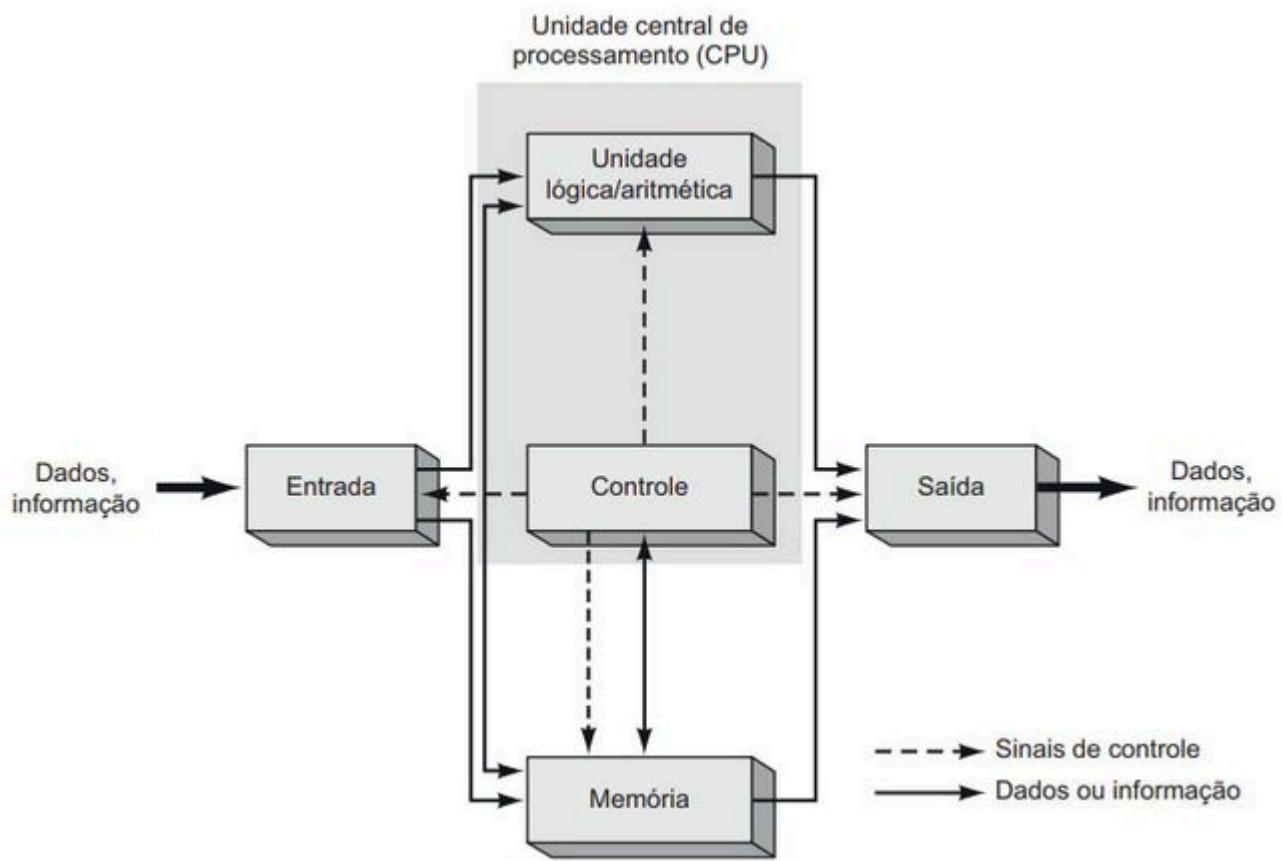


Figura 3 | Diagrama funcional de uma unidade central de processamento (CPU). Fonte: Tocci e Widmer (2011, p. 18).

A Figura 4 apresenta um sistema digital interconectado por barramentos, componentes essenciais para a comunicação eficiente em um computador. O barramento de dados é uma via expressa por onde as informações são transferidas entre o microprocessador e outros dispositivos, como contadores e decodificadores. Cada dispositivo, seja um contador de 8 bits ou um conjunto de buffers, conecta-se a esse barramento, permitindo a troca de dados.

Por outro lado, o barramento de endereços é responsável por direcionar esses dados para os locais corretos, identificando as posições de memória a serem lidas ou escritas. O barramento de

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

controle gerencia o fluxo dessas operações, enviando sinais específicos que coordenam quando e como os dados são transferidos. Esses barramentos e a memória cache formam a espinha dorsal de um sistema computacional eficaz, garantindo que os dados sejam processados de forma rápida e precisa.

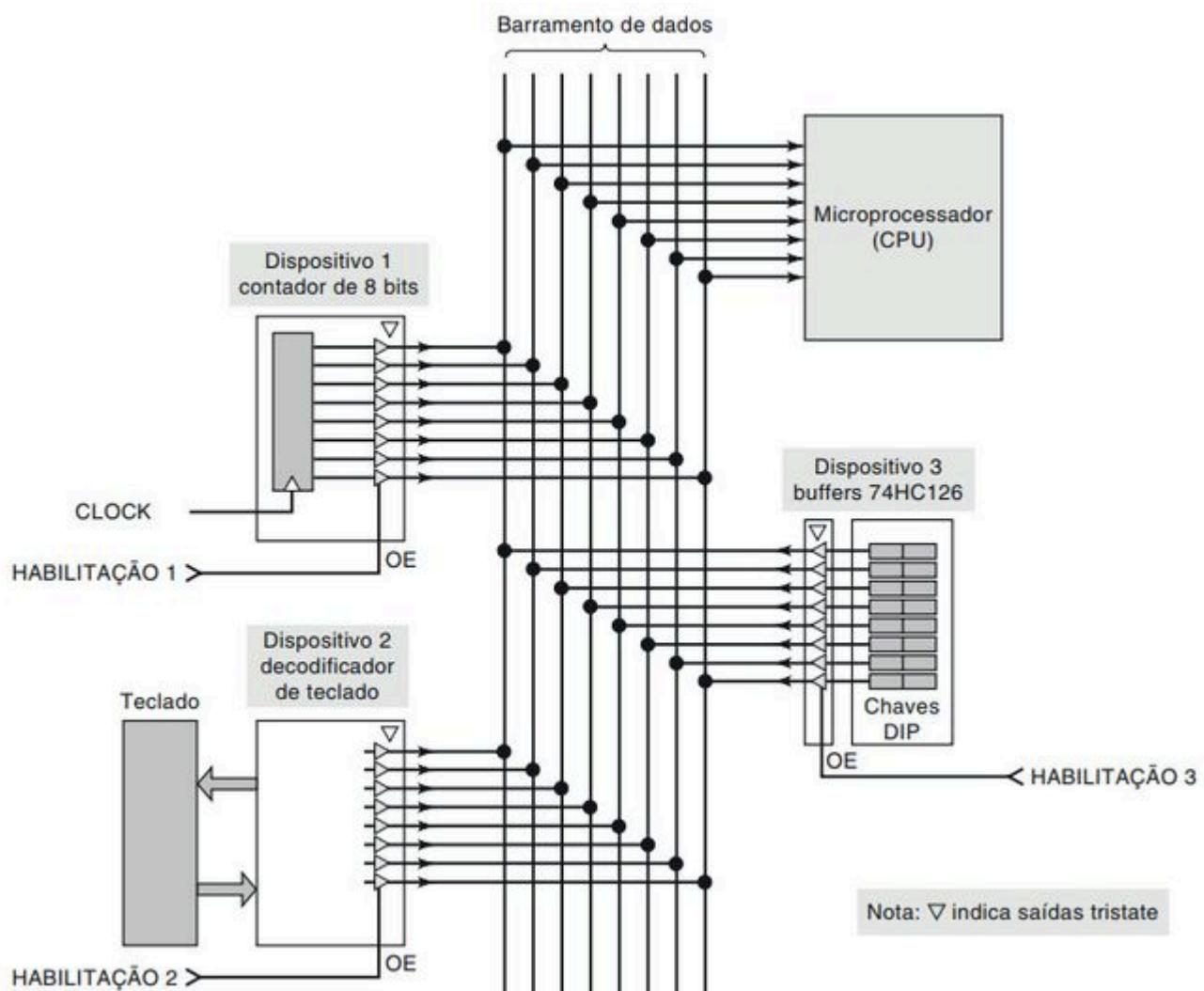


Figura 4 | Tipos de barramentos de dados. Fonte: Tocci e Widmer (2011, p. 545).

A compreensão profunda desses componentes e de como eles interagem é fundamental para qualquer aspirante a engenheiro ou entusiasta de tecnologia. Ao explorar esses fundamentos, Ana não só adquire um conhecimento valioso que serve de base para inovações tecnológicas futuras, mas também desenvolve habilidades práticas aplicáveis em uma variedade de contextos profissionais.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Siga em Frente...

Máquinas de Estados

À medida que prosseguimos com nosso aprendizado em engenharia de computação e projeto de sistemas, mergulhamos no estudo das máquinas de estados finitos (FSM – *Finite State Machines*), estruturas abstratas essenciais para modelar dispositivos eletrônicos e sistemas computacionais que operam com um conjunto limitado de estados. Essas FSM são vitais no desenvolvimento de circuitos sequenciais, em que o comportamento do sistema é ditado tanto pelas entradas atuais quanto pelo registro histórico dessas entradas em seus estados. A Figura 5 ilustra dois exemplos de máquinas de estados finitos: uma máquina de Mealy e uma máquina de Moore. Vamos detalhar cada um dos elementos presentes nesses diagramas.

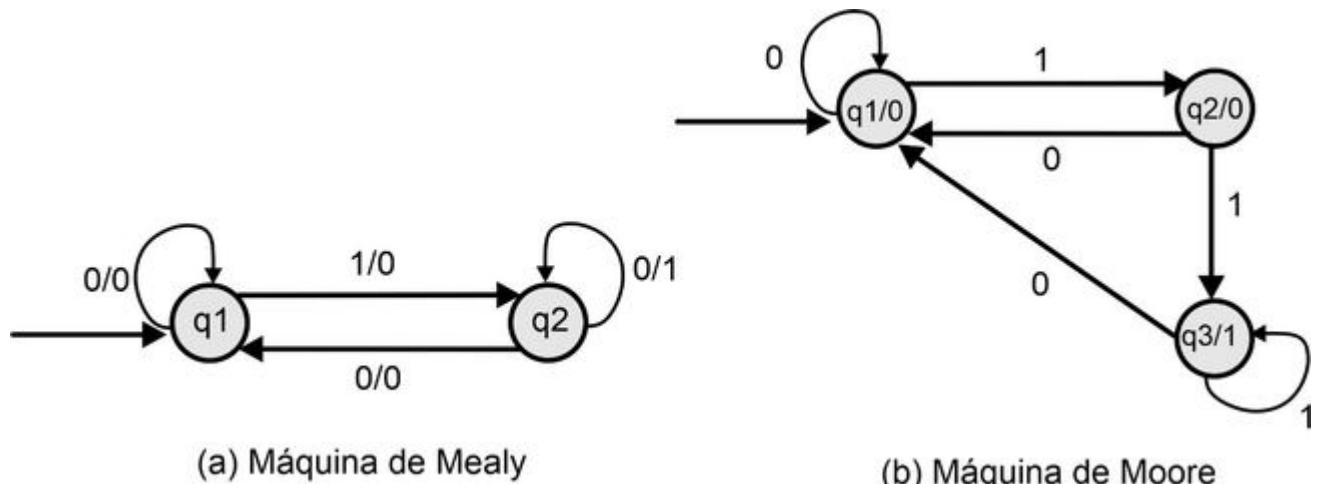


Figura 5 | Comparação entre máquinas de estados finitos: (a) Mealy; (b) Moore. Fonte: elaborada pelo autor.

Máquina de Mealy:

Em uma máquina de Mealy, a lógica de operação pode ser imaginada como uma sequência de comandos interativos em um videogame, em que cada botão pressionado (entrada) resulta em uma ação imediata da personagem (saída), e essa ação pode variar dependendo do contexto ou "estado" do jogo. Veja uma explicação detalhada integrando o exemplo:

1. Estados: imagine que cada círculo no diagrama é um local diferente no mapa do jogo (por exemplo, "q1", "q2", "q3"). A personagem do jogo está em um desses locais em um determinado momento.
2. Transições: as setas entre os círculos representam os movimentos possíveis que a personagem pode fazer de um local para outro. Na máquina de Mealy, cada movimento (transição) é acompanhado por uma ação específica (saída), que é indicada nas setas.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

3. Saídas: a ação (saída) é determinada pelo local atual (estado) e pelo comando específico (entrada) que você dá. Por exemplo, se sua personagem está no local "q1" e você pressiona o botão "A" (entrada "0"), ela pula (saída "0") imediatamente, representado na seta como "0/0". Isso significa que a ação de pular está diretamente ligada ao comando dado, independentemente de a personagem mudar de local (estado) ou não.

Isso permite que, enquanto no jogo, a personagem responda instantaneamente ao comando, sem atraso para a transição de estado. Em termos de diagrama, você veria a notação "entrada/saída" sobre a seta que liga "q1" a "q2", que representa a entrada recebida e a saída imediata, ilustrando a capacidade da máquina de Mealy de produzir saídas dinâmicas que são sensíveis às entradas em tempo real.

Máquina de Moore:

Imaginemos que você está novamente no controle de uma personagem em um videogame. Cada local do mapa onde a personagem pode estar é representado por um estado no diagrama (q1, q2, q3 etc.).

1. Estados: cada círculo é um "checkpoint" ou local no jogo onde a personagem pode realizar uma ação específica.
2. Transições: as setas são os caminhos que a personagem pode seguir de um checkpoint para outro. Cada caminho é acionado por um comando que você dá através do controle do jogo.
3. Saídas: na máquina de Moore, a ação que a personagem executa em cada checkpoint é predeterminada e não muda, independentemente do comando que você usou para chegar lá. Por exemplo, no estado "q1", não importa se você chegou lá pulando, correndo ou andando, a personagem sempre fará a ação associada a "q1" (por exemplo, recarregar a arma, indicado por "q1/0" onde "0" é a ação de recarregar).

Agora, se você move a personagem para o próximo checkpoint (q2), ela realizará a ação associada a esse novo estado, como acionar uma armadilha (indicado por "q2/0"). Somente quando a personagem chega ao estado "q3", ela realizará a ação de celebrar (indicado por "q3/1"), e essa ação só ocorre porque a personagem alcançou o estado "q3", não porque você pressionou um botão específico.

Neste estilo de videogame, a máquina de Moore garante que a personagem só executará ações específicas em locais específicos do mapa, e essas ações não mudarão com base nas entradas dos jogadores durante a transição de estados. Na máquina de Mealy, a saída é uma propriedade da transição (isto é, da combinação do estado atual e da entrada recebida). Em resumo, a diferença-chave entre as duas máquinas pode ser observada no Quadro 1.

Critério	Máquina de Moore	Máquina de Mealy
----------	------------------	------------------

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Dependência da saída	Apenas do estado atual.	Do estado atual e da entrada atual.
Localização da saída	Saída situada nos estados.	Saída situada nas transições.
Número de estados	Geralmente requer mais estados.	Geralmente requer menos estados.
Requisitos de hardware	Menor para implementação de circuito.	Maior para implementação de circuito.
Velocidade de reação	Reage mais lentamente às entradas (um ciclo de clock depois).	Reage mais rapidamente às entradas.
Geração de saída/estado	Geração de saída e estado síncronos.	Geração de saída assíncrona.
Design	Mais fácil de projetar.	Mais difícil de projetar.
Mudança de entrada	Se a entrada muda, a saída não muda.	Se a entrada muda, a saída também muda.
Comparação de estados	Tem mais ou o mesmo número de estados que a máquina de Mealy.	Tem menos ou o mesmo número de estados que a máquina de Moore.

Quadro 1 | Características da máquina de Moore e da máquina de Mealy. Fonte: elaborado pelo autor.

Exemplo: consideremos uma máquina de Estados (ME) projetada para identificar um padrão específico: uma sequência de dois “1” seguidos por um “0”. O funcionamento dessa máquina é ilustrado em um diagrama na Figura 6, que detalha como ela processa a entrada e gera a saída. A saída “s” se torna “1” apenas quando o padrão desejado é detectado na entrada “e”.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

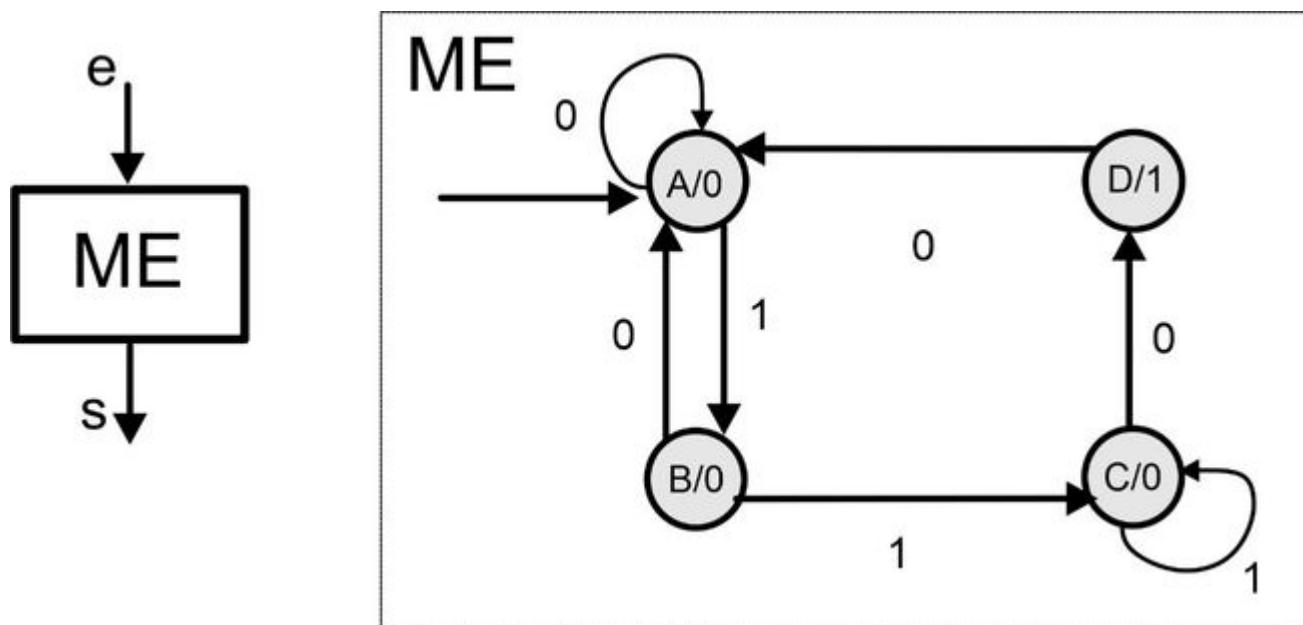


Figura 6 | FEM Moore para detectar a sequência 110. Fonte: elaborada pelo autor.

Estado A (A/0): este é o estado inicial em que a máquina começa a operação. A saída é 0, indicando que a sequência desejada ainda não foi detectada.

Estado B (B/0): se a máquina receber o dígito 1 enquanto estiver no estado A, ela se moverá para o estado B, mantendo a saída 0, já que apenas o primeiro 1 da sequência foi recebido.

Estado C (C/0): se outro 1 for recebido enquanto estiver no estado B, a máquina transita para o estado C. A saída permanece 0 porque, embora tenhamos recebido dois 1s, ainda não vimos um 0 subsequente para completar a sequência.

Estado D (D/1): se a máquina receber 0 enquanto estiver no estado C, ela se move para o estado D, e a saída se torna 1, indicando que a sequência "110" foi detectada com sucesso.

Em qualquer ponto, se uma entrada não for a esperada para continuar a sequência, a FSM "reinicia" para o estado A ou para o estado apropriado que considera as últimas entradas válidas.

A escolha entre uma máquina de Moore e uma máquina de Mealy muitas vezes depende do comportamento desejado para o sistema e das especificações do projeto. Por exemplo, se as saídas precisam ser imediatamente responsivas às entradas, uma máquina de Mealy pode ser mais apropriada. Se a estabilidade da saída é mais importante e não deve mudar com entradas transitórias, então uma máquina de Moore pode ser preferível.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Vamos Exercitar?

Transformações Tecnológicas

Neste projeto prático, Ana Beatriz e Carlos Eduardo enfrentaram o desafio de automatizar o controle de iluminação doméstica usando sensores de movimento, aplicando conceitos de microprocessadores. Este projeto ilustra a importância de integrar componentes de sistema de maneira eficaz para resolver problemas reais, destacando a aplicação prática dos conhecimentos teóricos em microprocessadores.

Inicialmente, Ana e Carlos selecionaram os componentes essenciais: sensores de movimento para detecção de presença, um microprocessador como o cérebro do sistema para processar os sinais dos sensores, e um sistema de iluminação composto por lâmpadas ou LEDs controlados pelo microprocessador. A etapa de desenvolvimento envolveu a programação do microprocessador para interpretar os sinais dos sensores e a criação de uma lógica de controle que determinasse a ativação ou desativação da iluminação com base na detecção de movimento.

Durante os testes e a otimização do sistema, Ana e Carlos ajustaram a sensibilidade dos sensores e os parâmetros do microprocessador para assegurar uma resposta adequada e eficiente do sistema de iluminação. Com o sistema funcionando como esperado, eles refletiram acerca da importância da seleção e programação corretas dos componentes para o sucesso do projeto, e de como a aplicação prática dos conceitos de microprocessadores possibilitou a resolução de um problema cotidiano.

Este projeto não apenas resolveu uma necessidade diária, mas também serviu como um exemplo concreto do valor da aplicação prática dos conhecimentos teóricos em tecnologia. Ana e Carlos demonstraram a relevância dos sistemas digitais e microprocessadores além do ambiente acadêmico, incentivando os estudantes a considerar outras aplicações práticas dos conceitos aprendidos, como no desenvolvimento de sistemas de automação avançados ou sistemas de segurança. Assim, este projeto prático reforça a importância da integração eficaz de componentes digitais e abre caminho para futuras inovações, encorajando os estudantes a explorar novas possibilidades tecnológicas.

Saiba mais

Os sistemas digitais e máquinas de estado são conceitos fundamentais na área da computação e engenharia eletrônica, desempenhando um papel crucial no desenvolvimento e na implementação de circuitos lógicos complexos. Esses sistemas permitem a representação, análise e síntese de operações lógicas e algorítmicas de maneira eficiente e estruturada,

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

facilitando a criação de dispositivos eletrônicos avançados. O entendimento desses conceitos é essencial para o aprofundamento em temas relacionados à automação, ao design de circuitos integrados e à informática em geral, proporcionando uma base sólida para a inovação e a resolução de problemas complexos na área de tecnologia.

Para explorar e aprofundar seu conhecimento a respeito de sistemas digitais e máquinas de estado, consulte o Capítulo 3.3, nas páginas 161 a 173, da obra [*Sistemas Digitais: Conceitos e Aplicações*](#). Este capítulo oferece insights valiosos e recursos educacionais relevantes para o entendimento avançado do tema. Boa leitura!

Referências

CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.

IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.

LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.

SOUZA, D. B. da C. et al. **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 2

Memória: Tipos, Interface e Circuitos Integrados

Memória: tipos, interface e circuitos integrados



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Mergulhe nas profundezas da tecnologia com nossa videoaula focada em circuitos integrados de memória! Descubra como estes componentes essenciais armazenam e gerenciam dados, moldando o desempenho de dispositivos eletrônicos e sistemas computacionais. Esta aula não apenas esclarece conceitos básicos, mas também os liga a aplicações reais e tendências de inovação na engenharia. Se você está buscando compreender melhor a infraestrutura digital ou aprimorar suas competências técnicas, esta aula é seu ponto de partida ideal. Não perca a chance de expandir seus horizontes tecnológicos.

Ponto de Partida

No vasto universo da tecnologia da informação, os circuitos integrados de memória são pilares essenciais que sustentam a infraestrutura de nossos dispositivos eletrônicos e sistemas computacionais. Eles armazenam o precioso tesouro de dados e instruções que nossas máquinas precisam para operar de forma inteligente e eficiente. Compreender esses componentes não é apenas uma tarefa para engenheiros e especialistas em TI, mas uma aventura fascinante para qualquer um curioso a respeito de como a tecnologia funciona por dentro.

Ana Beatriz e Carlos Eduardo estão de volta em uma nova missão: explorar o intrincado mundo dos circuitos integrados de memória. Ambos estão representados na Figura 1. Desta vez, Ana enfrenta o desafio de otimizar o sistema de armazenamento de dados de um robô autônomo, enquanto Carlos a guia pelos conceitos de memória RAM e ROM, destacando sua importância para a realização de tarefas complexas e armazenamento de instruções vitais.

Imagine um robô que, ao realizar suas tarefas diárias, enfrenta limitações de velocidade e eficiência devido à gestão ineficaz da memória. Como Ana e Carlos podem aplicar seus conhecimentos de circuitos integrados de memória para superar esses obstáculos? Quais estratégias de design e implementação eles podem utilizar para aprimorar o desempenho do robô? Este cenário nos convida a mergulhar profundamente na ciência por trás da memória dos computadores, desvendando como cada tipo de memória contribui para a execução eficiente de tarefas e o armazenamento seguro de informações. Ao longo desta jornada, você obterá o conhecimento necessário para entender e inovar na área de tecnologia da informação.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Figura 1 | Ana e Carlos trabalhando com memórias em laboratório. Fonte: Cogna IA.

Embarque nesta jornada educativa com Ana e Carlos para desvendar os segredos dos circuitos integrados de memória. Aprofunde seu entendimento da tecnologia moderna e abra portas para aplicações práticas em sua carreira. Com Ana e Carlos, transforme sua interação com a tecnologia.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Vamos Começar!

Tipos de Memórias de Computador

Neste módulo inicial, vamos conhecer os fundamentos das memórias de computador, um componente vital para o entendimento da arquitetura e do funcionamento dos sistemas computacionais. A memória de um computador armazena os dados e instruções que permitem ao computador executar suas funções.

Existem dois tipos principais de memória: memória somente de leitura (ROM – *Read Only Memory*) e memória de acesso aleatório (RAM – *Random Access Memory*). Cada tipo tem características e usos específicos que são cruciais para o desempenho geral de um computador. A Figura 2 ilustra essas duas formas distintas de memória de computador: à esquerda, um módulo DIMM (*Dual Inline Memory Module*), um tipo de memória RAM usada para armazenamento de dados temporário e volátil, essencial para as operações diárias de um computador. À direita, a memória ROM é apresentada: uma memória não volátil que mantém os dados permanentemente, comumente usada para armazenar o firmware ou o software básico que inicializa o sistema.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

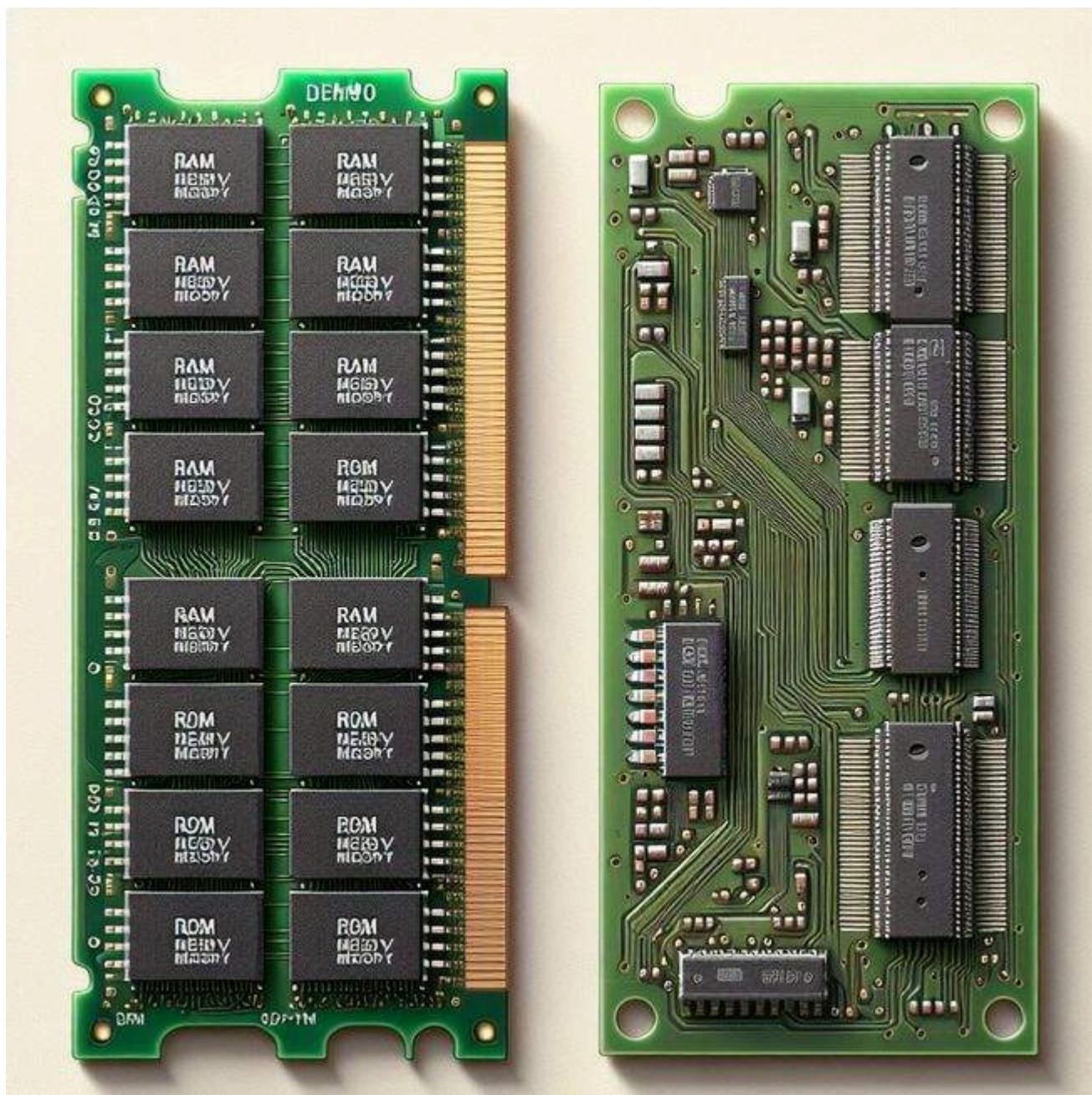


Figura 2 | Ilustração de memória RAM e memória ROM. Fonte: Cogna IA.

A escolha entre ROM e RAM é determinada pelo tipo de dado a ser armazenado e pela necessidade de acesso. A ROM é ideal para armazenar o firmware ou o software básico necessário para inicializar o hardware do computador. Por outro lado, a RAM é crucial para o funcionamento diário do computador, armazenando os dados que estão sendo usados ativamente, o que permite que o processador execute tarefas de forma eficiente.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

As memórias de computador são fundamentais para diversas funções e se apresentam de várias formas, cada uma adequada para diferentes tarefas e operações. A memória ROM, utilizada para armazenamento de longo prazo e imutável, apresenta variantes como PROM, EPROM, EEPROM e Flash, que variam em sua capacidade de serem reescritas e no método pelo qual isso é realizado. Enquanto isso, a RAM, conhecida por seu caráter volátil e por ser essencial na realização de tarefas ativas, tem como principais tipos a SRAM, utilizada para o armazenamento rápido como cache devido à sua eficiência e velocidade, e a DRAM, mais comum e econômica, empregada para a maioria das necessidades de armazenamento temporário devido à sua maior capacidade. Juntas, essas memórias são vitais para o funcionamento de sistemas eletrônicos, da inicialização ao processamento de dados em tempo real. A seguir, você conhece as principais memórias existentes:

ROM: é projetada para armazenar dados permanentemente. Uma vez que os dados são escritos, eles não podem ser modificados ou podem ser alterados apenas por processos específicos. Existem vários tipos de ROM:

- **PROM** (*Programmable Read-Only Memory*): uma vez programada, os dados não podem ser alterados.
- **EPROM** (*Erasable Programmable Read-Only Memory*): permite a regravação de dados pela exposição à luz ultravioleta.
- **EEPROM** (*Electrically Erasable Programmable Read-Only Memory*): possibilita a regravação de dados eletricamente, sem a necessidade de equipamento especial.
- **Flash**: uma forma avançada de EEPROM, com regravação rápida e alta capacidade de armazenamento, comum em dispositivos portáteis.

RAM: é volátil, o que significa que os dados são perdidos quando o computador é desligado. Ela é essencial para armazenar os dados em uso ativo, permitindo acesso rápido pelo processador. Existem dois tipos principais:

- **SRAM** (*Static RAM*): rápida e eficiente, mas mais cara e menos densa, ideal para cache.
- **DRAM** (*Dynamic RAM*): mais lenta que a SRAM, porém mais barata e com maior capacidade, sendo a forma mais comum de RAM.

A SRAM, sendo mais rápida, é geralmente usada como cache para armazenar dados temporários de rápido acesso, enquanto a DRAM é usada para a maioria das necessidades de armazenamento devido ao seu custo menor e maior capacidade. Para ilustrar, considere o uso de SRAM em um smartphone para o armazenamento de dados que requerem acesso rápido e frequente, como o aplicativo da câmera. Já a DRAM pode ser utilizada para executar múltiplos aplicativos simultaneamente, permitindo uma experiência de usuário suave e eficiente.

Ao dominar os conceitos de memória RAM e ROM, percebe-se o impacto direto no desempenho dos sistemas computacionais. Enquanto a RAM facilita o processamento ativo com suas

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

variantes SRAM e DRAM, a ROM garante a permanência dos dados essenciais do sistema. No próximo módulo, vamos explorar como essas memórias influenciam a eficiência operacional dos computadores.

Siga em Frente...

Funcionamento da Memória

À medida que avançamos em nossa compreensão sobre memória, vamos aprofundar os detalhes técnicos que definem a eficácia desses componentes essenciais nos sistemas computacionais e eletrônicos. Este segmento explora as considerações de projeto, a estrutura e organização das memórias, além de detalhar a interface entre o processador e a memória.

A busca por eficiência econômica está na vanguarda da inovação em chips de memória, com o objetivo de aumentar a densidade de armazenamento e diminuir o custo por bit. Este esforço busca um equilíbrio ideal entre custo, capacidade e desempenho. A velocidade de acesso aos dados é melhorada por meio de estratégias internas sofisticadas e técnicas avançadas de endereçamento. A comunicação entre processador e memória é mediada por linhas de dados, linhas de endereço e sinais de controle. A largura e a eficiência dessas linhas são determinantes para a rapidez e a confiabilidade da troca de informações.

A memória principal de um computador é conectada à CPU por meio de três grupos de linhas de sinal: barramento de endereço, barramento de dados e barramento de controle, conforme mostra a Figura 3. Cada um desses barramentos apresenta várias linhas, e sua quantidade varia entre computadores. Esses barramentos são essenciais para permitir que a CPU leia e escreva dados na memória durante a execução de um programa. Quando a CPU executa um programa, ela lê as instruções e os dados relevantes da memória e escreve dados conforme as instruções. Sempre que a CPU escreve um dado na memória, são realizados passos específicos para essa operação.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

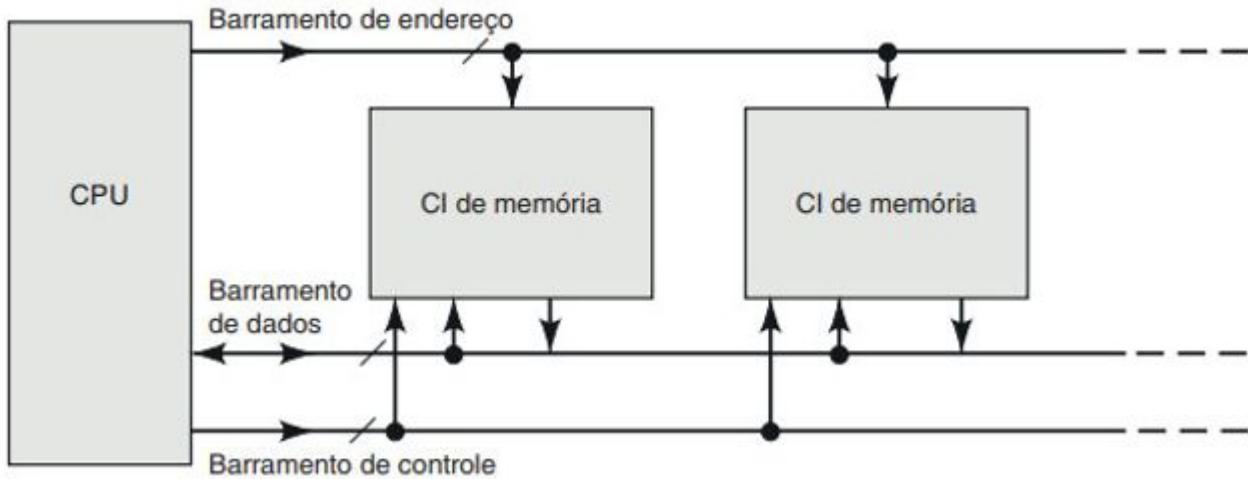


Figura 3 | Barramentos entre memória e CPU. Fonte: Tocci e Widmer (2011, p. 690).

Os barramentos do sistema têm funções específicas:

- Barramento de endereço: unidirecional, envia os endereços da CPU para selecionar posições de memória.
- Barramento de dados: bidirecional, transmite dados entre CPU e memória.
- Barramento de controle: envia sinais de controle da CPU para os dispositivos de memória.

Procedimento de escrita da memória

1. A CPU envia o endereço binário da posição de memória desejada através do barramento de endereço.
2. Um decodificador de endereços ativa a entrada de habilitação do dispositivo de memória.
3. A CPU coloca os dados a serem armazenados no barramento.
4. A CPU ativa os sinais de controle apropriados para a operação de escrita na memória (por exemplo, WR ou R/W).
5. Os CIs de memória decodificam o endereço binário e realizam a operação de armazenamento na posição selecionada.
6. Os dados presentes no barramento são transferidos para a posição de memória selecionada.

Procedimento de leitura da memória

1. A CPU envia o endereço binário da posição de memória desejada através do barramento de endereço.
2. Um decodificador de endereços ativa a entrada de habilitação do dispositivo de memória.
3. A CPU ativa os sinais de controle apropriados para a operação de leitura na memória (por exemplo, RD).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

4. Os CLs de memória decodificam o endereço binário e selecionam a posição para a operação de leitura.
5. Os CLs de memória colocam os dados da posição de memória selecionada no barramento de dados, para transferência à CPU.

Diagrama típico da ROM

Uma ROM típica conta com três conjuntos de sinais: entradas de endereço, de controle e saídas de dados. Por exemplo, uma ROM de 16x8 armazena 16 palavras de 8 bits cada. As saídas de dados são geralmente tristate para permitir a conexão de vários CLs no mesmo barramento. A entrada de controle CS (*Chip Select*) habilita ou desabilita as saídas da ROM. Algumas ROMs têm entradas adicionais de controle para ativar as saídas de dados. Não há uma entrada WE (*Write Enable*), pois a escrita na ROM não é permitida em operações normais. A Figura 4 ilustra a organização de uma ROM 16x8, destacando a correlação entre endereços de entrada e os dados de saída correspondentes, onde cada linha da matriz de endereço acessa uma palavra de 8 bits, demonstrada tanto em binário quanto em hexadecimal.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

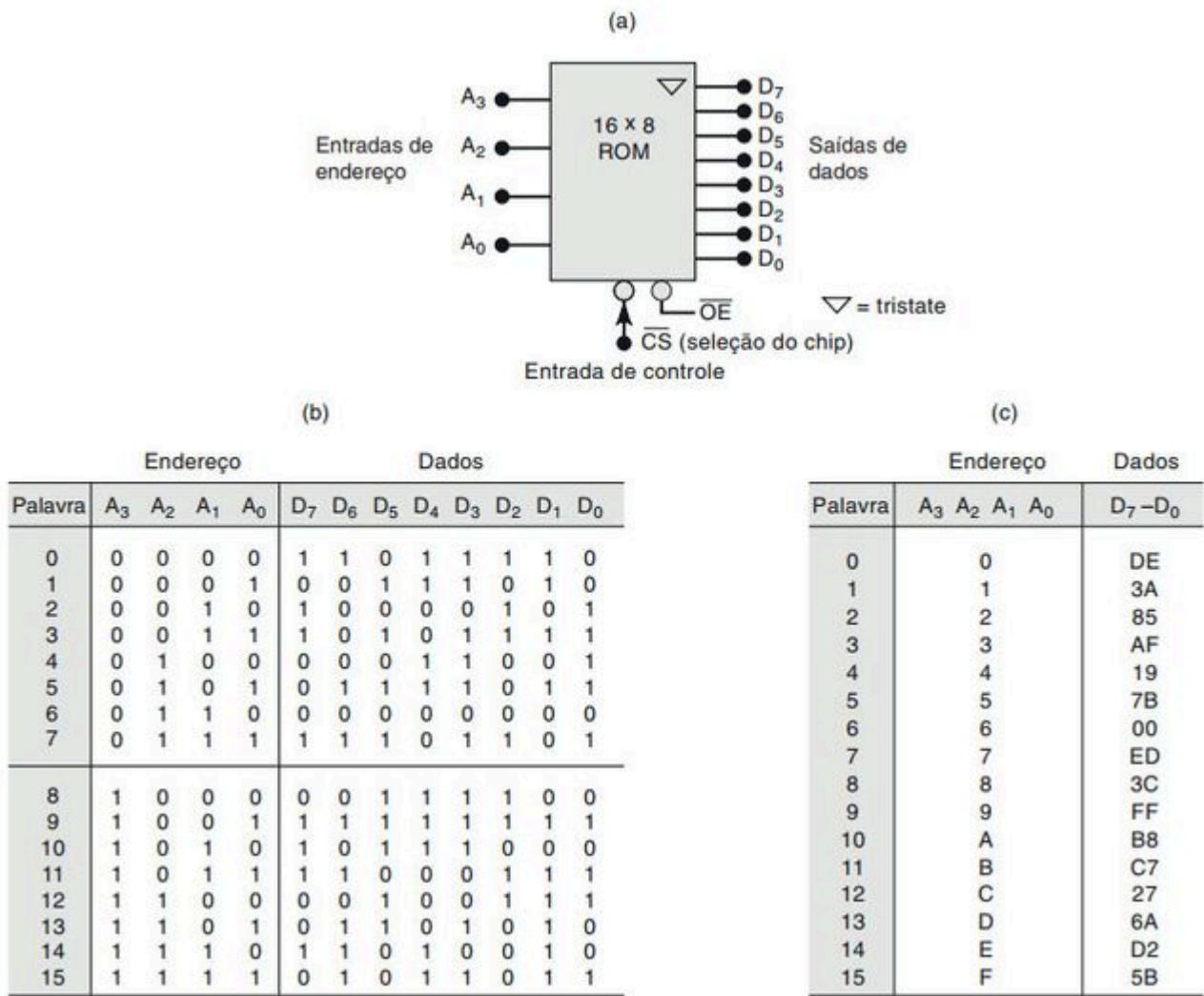


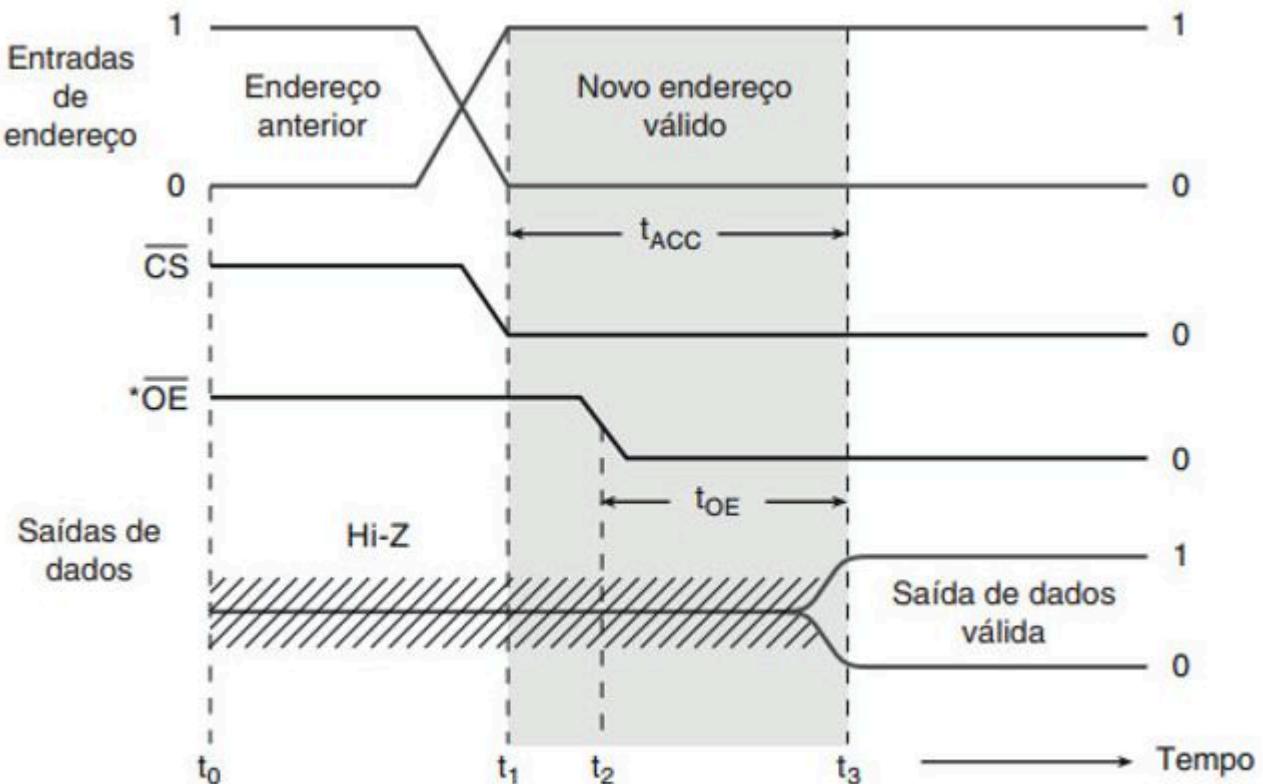
Figura 4 | Organização e mapeamento de dados em uma memória ROM 16x8. Fonte: Tocci e Widmer (2011, p. 692).

O ciclo de barramento e o ciclo de memória são componentes-chave neste processo. Eles delineiam as fases de endereçamento e transferência de dados, sendo cruciais para o desempenho do sistema. A análise de diagramas de tempo para os ciclos de leitura e escrita proporciona uma visão detalhada de como as operações de memória são sincronizadas, ressaltando a importância da precisão no intertravamento dos sinais de controle.

A Figura 5 ilustra o diagrama de temporização de acesso e saída de dados em uma ROM. Durante uma operação de leitura da ROM, há um atraso chamado tempo de acesso (t_{ACC}), que indica a rapidez da ROM. Este atraso é o período entre a aplicação do novo endereço e o momento em que os dados se tornam disponíveis na saída. Na Figura 5, este tempo é mostrado desde que um novo endereço válido é estabelecido (t_1) até quando os dados estão disponíveis na saída (t_3). ROMs modernas, com tecnologias CMOS, têm tempos de acesso variando de 20 a

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

60 ns, mais rápidos que as mais antigas bipolares e NMOS. Outro parâmetro, o tempo de habilitação da saída (t_{OE}), é o atraso entre a ativação da entrada OE e a disponibilidade dos dados válidos, sendo geralmente mais curto que o t_{ACC} .



* t_{OE} é medido a partir do momento em que \overline{CS} e \overline{OE} foram ambos ativados.

Figura 5 | Temporização de acesso e saída de dados em uma ROM. Fonte: Tocci e Widmer (2011, p. 695).

Concluindo, a jornada pelas memórias revela a complexidade e a importância desses componentes. À medida que exploramos o seu funcionamento e interação com o processador, torna-se evidente o papel vital que desempenham na definição da capacidade, velocidade e eficiência dos dispositivos eletrônicos e sistemas computacionais. Este conhecimento aprimora nossa compreensão técnica e destaca a importância da inovação contínua na área de tecnologia da informação.

Vamos Exercitar?

Transformações Tecnológicas

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

A problematização apresentada no início desta jornada educativa desafiou Ana Beatriz e Carlos Eduardo a enfrentar as limitações de velocidade e eficiência em um robô autônomo, causadas por uma gestão ineficaz da memória. Com a exploração dos circuitos integrados de memória, particularmente a distinção entre memória RAM e ROM e suas aplicações, identificamos soluções para otimizar o sistema de armazenamento de dados do robô.

Aplicação dos conteúdos fundamentais:

Otimização da memória RAM: Ana e Carlos decidiram aumentar a capacidade da DRAM no robô para melhorar a multitarefa e o processamento de tarefas em tempo real. Este ajuste permitiu que o robô armazenasse mais dados temporários de operações ativas, melhorando significativamente a velocidade de execução das tarefas.

Eficiência da memória ROM: a escolha por uma memória Flash, um tipo avançado de EEPROM para a ROM, proporcionou uma maneira flexível e eficiente de atualizar o firmware do robô. Isso facilitou a implementação de melhorias no sistema operacional do robô e garantiu a retenção de informações cruciais para o seu funcionamento autônomo, mesmo após desligamentos.

Projeto e implementação estratégica: a compreensão da importância da organização da matriz de memória e das operações básicas de leitura e escrita orientou Ana e Carlos na escolha de componentes de memória mais adequados. Além disso, aplicaram técnicas de endereçamento eficientes para otimizar o acesso aos dados armazenados, reduzindo o tempo de acesso e aumentando a eficiência energética.

Este caso prático ilustra como um entendimento aprofundado dos componentes de memória e suas funcionalidades pode ser aplicado para superar desafios técnicos reais. Contudo, a tecnologia está em constante evolução, e novas soluções emergem regularmente. Você, estudante, está convidado a pensar em alternativas inovadoras que poderiam ser implementadas nesse cenário. Por exemplo, como a incorporação de tecnologias emergentes, como a memória MRAM (*Magnetoresistive Random-Access Memory*), poderia oferecer ainda mais benefícios em termos de velocidade e eficiência energética?

Ao resolver a problematização proposta, aplicamos conhecimento teórico a um problema prático e abrimos caminho para a inovação e a melhoria contínua. Este é o verdadeiro valor da educação tecnológica: equipar-nos com as ferramentas necessárias para não apenas entender o mundo ao nosso redor, mas também para moldá-lo de forma criativa e eficaz.

Saiba mais

Para aprofundar seu entendimento acerca da importância e do funcionamento das memórias em sistemas computacionais, sugerimos a consulta ao material indicado a seguir. Este recurso é um

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

guia completo que aborda diversos tipos de memória, explicando suas funções, aplicações e o papel vital que desempenham na arquitetura de computadores. A leitura é indispensável para aqueles interessados em compreender em detalhes como as memórias afetam o desempenho e a eficiência de sistemas digitais.

Aprofunde seus conhecimentos em memória, seus conceitos e aplicações com o material [Memória: Conceitos e Aplicações – Um Guia Completo](#). Este documento é uma fonte rica de informações que oferece uma visão detalhada da diversidade das memórias, seus mecanismos de funcionamento e a sua contribuição essencial para a eficiência e o desempenho dos sistemas computacionais.

Referências

- CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. et al. **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.
- TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 3

Processador e Instruções

Processador e instruções



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Adentre o núcleo da computação moderna com nossa videoaula sobre a organização do processador. Descubra a engenharia por trás do processamento de informações e como as instruções do microprocessador Mico se alinham às práticas contemporâneas. Essencial para aspirantes e profissionais de TI, este conteúdo liga teoria à realidade do mercado. Junte-se a nós para uma jornada enriquecedora de aprendizado e aplicação.

Ponto de Partida

Hoje, Ana Beatriz e Carlos Eduardo, nos convidam para um desafio: criar um algoritmo para calcular a média de dois números com o microprocessador Mico, essencial para desbravar a computação moderna. Este exercício nos leva além do básico, explorando otimização e eficiência na programação.

Você tem dois números. O objetivo é simples: calcular a média de forma eficiente, usando instruções do Mico. Este cenário inicial abre portas para entender a interação entre operações simples e o impacto de decisões de design na economia de recursos. As reflexões para o desafio de Ana Beatriz e Carlos Eduardo são: quais instruções do Mico são essenciais para somar e dividir? Como otimizar o algoritmo para rapidez e mínimo uso de recursos?

Prepare-se para ser um engenheiro que decifra códigos e transforma linhas de instruções em soluções práticas e inovadoras. Com Ana Beatriz e Carlos Eduardo guiando sua experiência prática, incentivamos você a pensar como os construtores do amanhã. Este desafio não é apenas um problema a resolver; é uma oportunidade para forjar as habilidades que moldarão sua carreira. Você está pronto para começar? Vamos lá, descobrir o potencial ilimitado do microprocessador Mico!

Vamos Começar!

Organização do Processador

A exploração da organização de processadores permite uma compreensão de como computadores processam informações e executam comandos. Neste contexto, discutiremos a organização do processador por meio do estudo do microprocessador Mico, um modelo didático inspirado no MIPS R4000 e projetado especificamente para fins educativos. O foco será detalhar a estrutura do Mico, seu repertório de instruções e o método de execução dessas instruções, estabelecendo uma base sólida para entender os princípios subjacentes aos processadores contemporâneos.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

O microprocessador Mico revela como o hardware do processador interage com outros componentes do sistema, como memória e dispositivos de entrada/saída. Conforme ilustrado em um diagrama de blocos na Figura 1, a estrutura do Mico engloba uma unidade lógica e aritmética (ULA), um conjunto de registradores e um sistema de controle que dirige a execução das instruções e a comunicação com a memória e os dispositivos periféricos. Central para a organização do Mico é a ideia de que um computador pode ser interpretado de várias maneiras, a depender da perspectiva do observador, seja ele um programador, um arquiteto de processadores ou um engenheiro de hardware. Cada um desses pontos de vista destaca diferentes facetas do processador e do sistema computacional em sua totalidade.

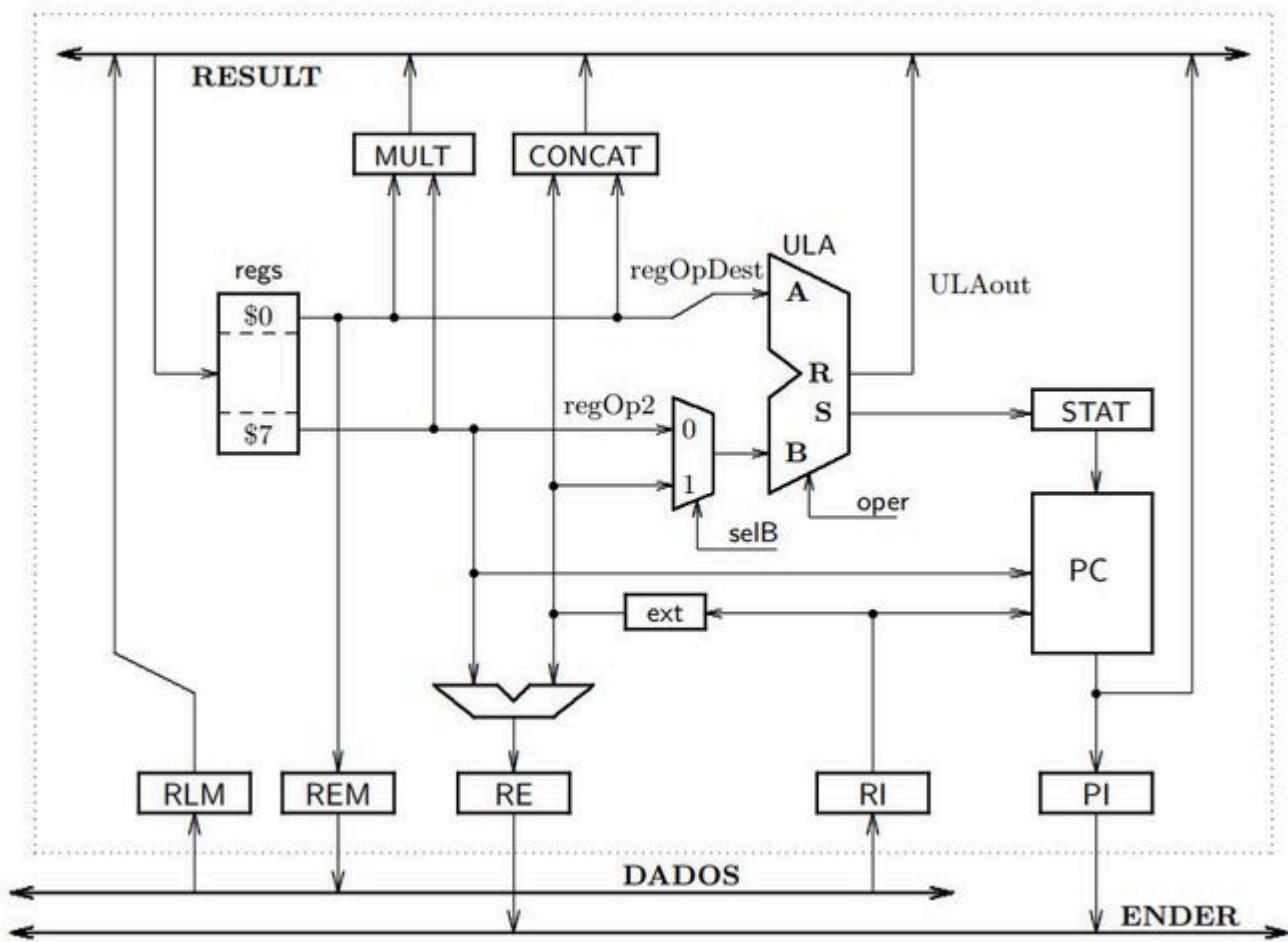


Figura 1 | Organização do Mico em diagrama de blocos. Fonte: Hexsel (2006, p. 78).

Os componentes-chave do Mico incluem:

- **Registradores de uso geral (\$0-\$7):** para operações temporárias e armazenamento.
- **Contador de programa (PC):** indica a próxima instrução a ser executada.
- **Registrador de instrução (RI):** contém a instrução atualmente em execução.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- **Registrador de status (STAT)**: armazena o status da última operação realizada pela ULA.
- **Registradores de interface com memória (RE, RLM, REM)**: facilitam a comunicação com a memória.
- **Blocos funcionais (ULA, MULT, CONCAT)**: responsáveis pelas operações lógicas, aritméticas, multiplicação e concatenação.

O conjunto de instruções funciona como a ponte crítica entre o hardware do processador e o software, estabelecendo o repertório de operações executáveis pelo processador. Para o Mico, esse conjunto é intencionalmente simplificado, promovendo uma compreensão clara dos conceitos essenciais. Abrange categorias fundamentais como operações lógicas e aritméticas, manipulação de dados, controle de fluxo através de saltos e desvios, gerenciamento de funções, além de instruções específicas para entrada/saída e controle, refletindo as necessidades básicas para a execução de programas. A Tabela 1 ilustra exemplos de operações lógicas e aritméticas, exemplificando a abordagem didática adotada.

código	operação	descrição
0	ADD	adição
1	SUB	subtração
2d	LSL	deslocamento para esquerda
3d	LSR	deslocamento para direita
2r	ROL	rotação para esquerda
3r	ROR	rotação para direita
4	NOT	complemento de um
5	AND	conjunção bit a bit
6	XOR	ou-exclusivo bit a bit
7	OR	disjunção bit a bit

Tabela 1 | Operações lógicas e aritmética. Fonte: Hexsel (2006, p. 79).

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

O conjunto de instruções de um processador é definido por três componentes principais: as próprias instruções, os modos de endereçamento, e os registradores visíveis ao programador. Esses elementos trabalham em conjunto para permitir a transformação e o acesso aos dados de maneiras que suportem a execução eficiente de programas.

1. Instruções: são os comandos que o processador pode executar, variando desde operações aritméticas simples, como adição e subtração, até operações mais complexas, como manipulação de bits e controle de fluxo de execução. No caso do Mico, as instruções são categorizadas em operações de lógica e aritmética, movimentação de dados, controle de fluxo, suporte a funções, e operações de entrada/saída.
2. Modos de endereçamento: determinam como os operandos das instruções são especificados. Podem variar desde o uso direto de valores constantes até o uso de complexas expressões aritméticas que envolvem registradores e memória. Esses modos influenciam diretamente a flexibilidade e a eficiência da execução de programas.
3. Registradores: são pequenas unidades de armazenamento dentro do processador que são diretamente acessíveis pelas instruções. Eles proporcionam acesso rápido aos dados mais frequentemente utilizados e suportam a realização de cálculos intermediários e o armazenamento temporário de dados.

À medida que desvendamos a arquitetura do microprocessador Mico, chegamos ao cerne de sua funcionalidade: o conjunto de instruções. Essas instruções são a linguagem por meio da qual o hardware e o software comunicam-se, permitindo a execução de tarefas variadas, desde operações matemáticas simples a complexas sequências de controle de fluxo. Ao compreendermos este repertório de comandos e sua execução, nos aproximamos da capacidade de otimizar e inovar em design de sistemas, além de desenvolver softwares que maximizem a eficácia do processador. A seguir, vamos explorar mais profundamente como essas instruções são executadas pelo Mico, detalhando cada fase do seu ciclo de processamento.

Siga em Frente...

Execução de Instruções

O microprocessador Mico apresenta um conjunto de instruções visando à facilitação do entendimento dos conceitos básicos em arquitetura de computadores. Suas instruções abrangem:

- **Operações de lógica e aritmética:** incluem adição, subtração, operações bit a bit (AND, OR, XOR, NOT), deslocamentos e rotações. Estas operações são fundamentais para a realização de cálculos e manipulação de dados, conforme mostra a Tabela 2.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Instrução	Descrição
CONST \$p,const	carrega constante
ADDI \$p,const	adição de imediato
ADDIC \$p,const	adição de imediato com carry
MUL \$p,\$q	multiplicação
HI \$p,const	concatenação, parte 'alta'
LOW \$p,const	concatenação, parte 'baixa'

Tabela 2 | Instruções aritméticas avançadas. Fonte: Hexsel (2006, p. 80).

- **Movimentação de dados:** envolve o carregamento (*load*) de dados da memória para os registradores, e o armazenamento (*store*) de dados dos registradores para a memória. Estas instruções são vitais para o fluxo de dados dentro do sistema e são apresentadas na Tabela 3.

Instrução	Descrição
LD \$d, desl(\$b)	carregar da memória
ST \$f, desl(\$b)	armazenar na memória

Tabela 3 | Instruções de acesso à memória. Fonte: Hexsel (2006, p. 82).

- **Controle de fluxo:** instruções como saltos (*jumps*) e desvios (*branches*) alteram a sequência linear de execução de instruções, permitindo a implementação de estruturas de controle como loops e condicionais, conforme apresenta a Tabela 4.

Instrução	Descrição
J ender	salto incondicional
Dcd desl	desvio condicional
Dzero ender	desvia se a última operação resultou em zero
Dneg ender	desvia se a última operação produziu número negativo
Dcarry ender	desvia se a última operação produziu vai-um ou carry
Dovfl ender	desvia se a última operação resultou em overflow
DNzero ender	desvia se a última operação não resultou em zero
Dpos ender	desvia se a última operação produziu número não-negativo
DNcarry ender	desvia se a última operação não produziu vai-um ou carry

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

DNovfl ender	desvia se a última operação não resultou em overflow
--------------	--

Tabela 4 | Instruções para alteração no fluxo de controle e desvio condicional. Fonte: Hexsel (2006, p. 83).

- **Suporte a funções:** inclui instruções para a chamada e retorno de funções, facilitando a organização do código e a reutilização de rotinas, conforme mostra a Tabela 5.

Instrução	Descrição
JAL ender	salto e armazena retorno
JR \$r	salto para registrador

Tabela 5 | Instruções de suporte às funções. Fonte: Hexsel (2006, p. 85).

- **Operações de entrada/saída:** permite a comunicação do processador com dispositivos externos, habilitando a entrada de dados para o sistema e a saída de dados para o usuário ou para outros sistemas, de acordo com o que mostra a Tabela 6.

Instrução	Descrição
IN \$r, ender	entrada de periférico
OUT \$r, ender	saída para periférico

Tabela 6 | Instruções de entrada e saída. Fonte: Hexsel (2006, p. 84).

O conjunto de instruções de um processador é fundamental para definir sua capacidade de processamento e eficiência. Uma arquitetura de instruções bem projetada pode minimizar a complexidade do software, melhorar o desempenho do sistema e reduzir o consumo de energia. Além disso, oferece aos programadores uma abstração poderosa que esconde a complexidade do hardware subjacente, permitindo foco na lógica e funcionalidade dos programas.

A execução das instruções é um processo fundamental no funcionamento de qualquer microprocessador. Este processo envolve várias etapas, cada uma crucial para a correta interpretação e execução das instruções pelo processador. Vamos explorar como as instruções são executadas no Mico, desde a busca da instrução na memória até a execução propriamente dita, incluindo as fases de decodificação, execução, acesso à memória e escrita do resultado.

Fases da execução das instruções

A execução de uma instrução no Mico, assim como em muitos processadores modernos, segue um ciclo de execução que pode ser dividido em várias fases: busca, decodificação, execução,

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

acesso à memória (quando necessário), e escrita do resultado. Este ciclo assegura que as instruções sejam processadas de maneira ordenada e eficiente.

1. **Fase de busca:** na fase de busca, o processador utiliza o contador de programa (PC) para localizar a próxima instrução a ser executada na memória. O valor armazenado no PC aponta para o endereço da memória onde a próxima instrução está localizada. Uma vez que a instrução é lida da memória, ela é armazenada no registrador de instrução (RI) do processador.
2. **Fase de decodificação:** durante a decodificação, o processador analisa a instrução armazenada no RI para determinar qual operação deve ser realizada. Isso envolve identificar o opcode da instrução e quaisquer operandos ou modos de endereçamento especificados pela instrução.
3. **Fase de execução:** na fase de execução, o processador realiza a operação especificada pela instrução. Dependendo do tipo de instrução, esta fase pode envolver cálculos aritméticos ou lógicos, manipulação de dados ou outros tipos de operações processadas pela unidade de lógica e aritmética (ULA) ou outros componentes do processador.
4. **Acesso à memória:** para instruções que requerem leitura ou escrita na memória, como instruções de carregamento (*load*) ou armazenamento (*store*), o processador executa as operações de acesso à memória após a execução da operação principal.
5. **Escrita do resultado:** finalmente, o resultado da execução da instrução, seja ele um valor computado pela ULA ou um dado lido da memória, é escrito no destino apropriado, que pode ser um registrador ou um endereço de memória, dependendo da instrução.

Este ciclo de execução permite que o processador execute uma sequência de instruções de maneira ordenada e previsível, garantindo que cada instrução seja corretamente interpretada e processada antes de passar para a próxima. Além disso, o ciclo de execução otimiza o uso dos recursos do processador, assegurando que os componentes do processador, como a ULA e os registradores, sejam utilizados de maneira eficiente.

Em resumo, o conjunto de instruções do Mico, apesar de sua simplicidade, encapsula os principais conceitos encontrados em processadores reais, oferecendo uma base sólida para o entendimento da arquitetura de computadores. Este conhecimento é crucial não apenas para o design de novos processadores e sistemas computacionais, mas também para o desenvolvimento de software eficiente e otimizado.

Vamos Exercitar?

Solução com Processador Mico

No ponto de partida da nossa aula, acompanhamos Ana Beatriz e Carlos Eduardo enfrentando um desafio empolgante: calcular a média de forma eficiente, usando instruções do Mico. Ana

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Beatriz focou a compreensão detalhada das instruções do Mico, enquanto Carlos Eduardo aplicou sua habilidade em lógica para propor uma abordagem eficiente. Juntos, eles delinearam o seguinte algoritmo:

Preparação:

Ana Beatriz sugeriu armazenar *num1* no registrador \$1 e *num2* no registrador \$2, garantindo que os dados estivessem prontamente acessíveis para operações subsequentes.

Passos do algoritmo:

1. Soma dos números:

Carlos Eduardo propôs a utilização do comando:

ADD \$3, \$1, \$2; Soma num1 e num2. Resultado em \$3

Este comando reflete a colaboração da dupla, somando os valores em \$1 e \$2 e armazenando o resultado em \$3.

2. Divisão por dois:

Reconhecendo a ausência de uma instrução de divisão direta, Ana Beatriz lembrou que a operação de deslocamento de bits poderia ser uma alternativa eficaz:

SRL \$4, \$3, 1; Divide o valor em \$3 por 2 via deslocamento à direita

Este insight permitiu simular a divisão por dois, armazenando a média final no registrador \$4.

Resultado e reflexão:

Com a média calculada e armazenada em \$4, Carlos Eduardo pensou na economia de recursos e a eficiência do código, ponderando possíveis otimizações. Ana Beatriz, por outro lado, sugeriu documentar o processo para futuras referências e para facilitar a revisão e o aprendizado de outros estudantes.

A colaboração entre Ana Beatriz e Carlos Eduardo solucionou o problema proposto de maneira eficiente e demonstrou a importância do trabalho em equipe na resolução de desafios técnicos. Sua abordagem conjunta permitiu não só a aplicação prática de conceitos fundamentais de computação, mas também a exploração de estratégias criativas para otimizar o desempenho do algoritmo dentro das capacidades do microprocessador Mico.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Saiba mais

Para enriquecer ainda mais sua compreensão acerca da arquitetura de computadores e aprofundar o conhecimento adquirido com o estudo do microprocessador Mico, sugerimos a leitura do livro [*Sistemas Digitais e Microprocessadores*](#), de R. A. Hexsel. Essa obra, originada do Departamento de Informática da Universidade Federal do Paraná em 2006, é um recurso valioso para todos que desejam explorar os fundamentos e aplicações das memórias em arquiteturas de computadores, além de outros conceitos cruciais na área de sistemas digitais – especialmente nas páginas 75 a 92, com uma análise profunda da organização do processador e as suas instruções, junto com outros temas fundamentais para quem quer compreender ou desenvolver sistemas digitais.

Referências

- CRUZ, E. *et al.* **Sistemas Digitais Reconfiguráveis: FPGA e VHDL.** Rio de Janeiro: Alta Books, 2022.
- HEXSEL, R. A. **Sistemas Digitais e Microprocessadores.** Departamento de Informática, Universidade Federal do Paraná, 2006.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital.** 42. ed. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores.** Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. *et al.* **Sistemas digitais.** Porto Alegre: SER – SAGAH, 2018.
- TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações.** 11. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 4

Memória e Controle

Memória e controle

Este conteúdo é um vídeo!

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Você está convidado a assistir a um vídeo resumo onde mergulhamos na engenharia da interface de memória do Mico, uma peça-chave na arquitetura dos sistemas computacionais modernos. Exploraremos como os dados são transferidos entre processador e memória, desvendando o papel vital dos sinais de controle. Essa compreensão não apenas enriquece o conhecimento técnico, mas também fortalece habilidades aplicáveis no desenvolvimento e análise de sistemas. Junte-se a nós para desbloquear os mistérios das memórias neste fascinante estudo técnico.

Ponto de Partida

Em meio ao avanço constante da tecnologia da informação, somos desafiados a compreender as sutilezas dos sistemas que operam sob nossos dedos. Na interseção da teoria e da prática, encontramos a interface de memória do Mico, um balé de sinais e circuitos que são fundamentais para qualquer entusiasta da tecnologia ou profissional da área. Neste início, nosso foco é o complexo relacionamento entre as linhas de dados e endereços e os sinais de controle que ditam a funcionalidade essencial de leitura e escrita de um computador.

Imagine a Ana Beatriz e o Carlos Eduardo em seu laboratório de inovações, conforme ilustra a Figura 1, estando diante de um desafio: otimizar a interface de memória de um sistema de automação para aumentar a eficiência e reduzir a latência na execução de tarefas. Como podem os conhecimentos sobre os sinais "wr", "bi", "es" e outros ajudá-los a atingir seu objetivo? Eles precisam aplicar e integrar esses conceitos à prática, garantindo que a teoria se transforme em benefícios reais em um ambiente profissional.

Esta aventura pelo conhecimento da memória computacional é tanto um mergulho em diagramas e definições quanto um convite a desvendar os mistérios que permitirão a Ana e Carlos – e a você – inovar e evoluir no campo da tecnologia da informação. Prepare-se para explorar não apenas o "o quê" e o "como", mas também o "porquê" das interfaces de memória. Que comece a jornada de aprendizado e descoberta!

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Figura 1 | Ana e Carlos trabalhando com memórias em laboratório. Fonte: Cogna IA.

Vamos Começar!

Interface com Memória

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

A interface de memória do Mico é composta por 16 linhas de dados, 16 linhas de endereço e um conjunto de sinais de controle. Essas linhas de dados são responsáveis pelo transporte de informações entre o Mico e a memória, tanto para escritas (do Mico para a memória) quanto para leituras e busca de dados (da memória para o Mico). Os sinais de controle asseguram a ordem correta dos processos na interface. A estrutura da interface de memória do Mico é ilustrada na Figura 2.

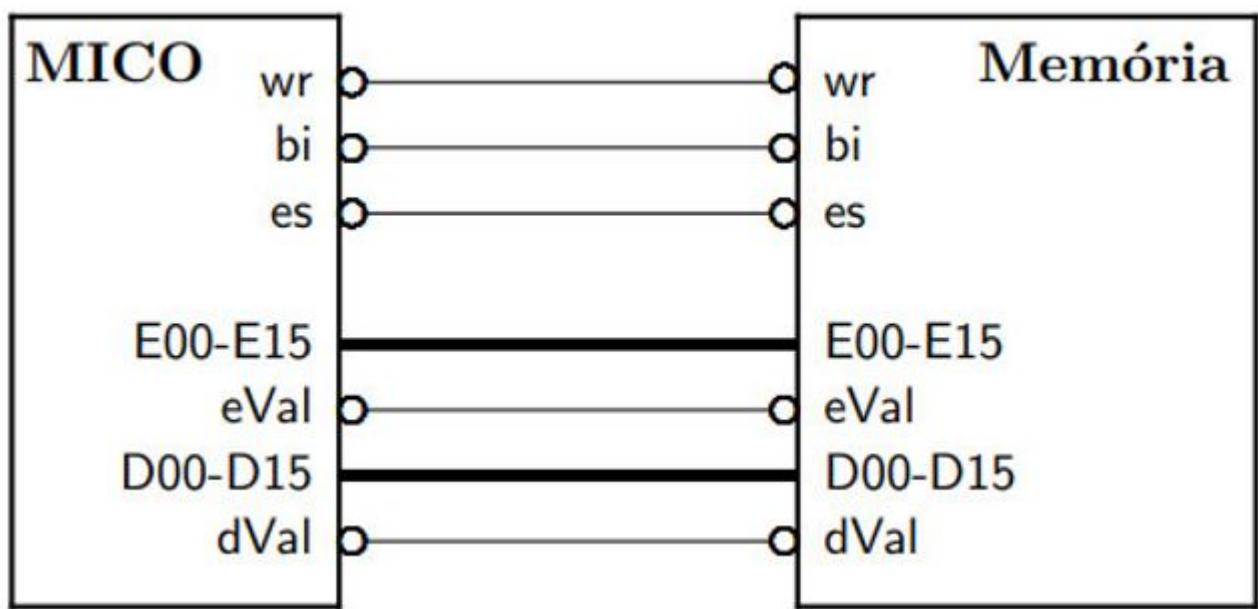


Figura 2 | Interface de memória do Mico. Fonte: Hexsel (2006, p. 93).

As funções específicas dos sinais na interface são detalhadas na Tabela 1.

Sinal	Descrição
wr	Sinal ativo indica que os dados nas linhas D00-D15 devem ser escritos no endereço especificado pelas linhas E00-E15
bi	Permanece ativo durante a operação de busca por uma instrução
es	Ativo durante a execução de instruções de entrada ou saída
E00-E15	Linhas de endereço
eVal	Sinaliza que o endereço presente nas linhas de endereço é válido e pronto para uso pela memória
D00-D15	Linhas de dados

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

dVal	Indica que os dados presentes nas linhas de dados são válidos e prontos para serem utilizados pela memória
------	--

Tabela 1 | Sinais e funções na interface de memória do Mico.

A Figura 3 ilustra como os circuitos de memória se conectam aos barramentos de dados e endereços no Mico, destacando apenas os 8 bits menos significativos de dados e os 13 bits menos significativos de endereço. As conexões de dados são bidirecionais entre o Mico e os chips de memória (cls), permitindo transmissão nos dois sentidos com saídas tristate ativadas pelo sinal "output enable" (oe). Durante um ciclo de busca, a ROM fornece a instrução para o registrador de instruções (ri), enquanto em ciclos de leitura e escrita, a RAM é usada para fornecer dados ao registrador de leitura (rlm) ou receber dados do registrador de escrita (rem).

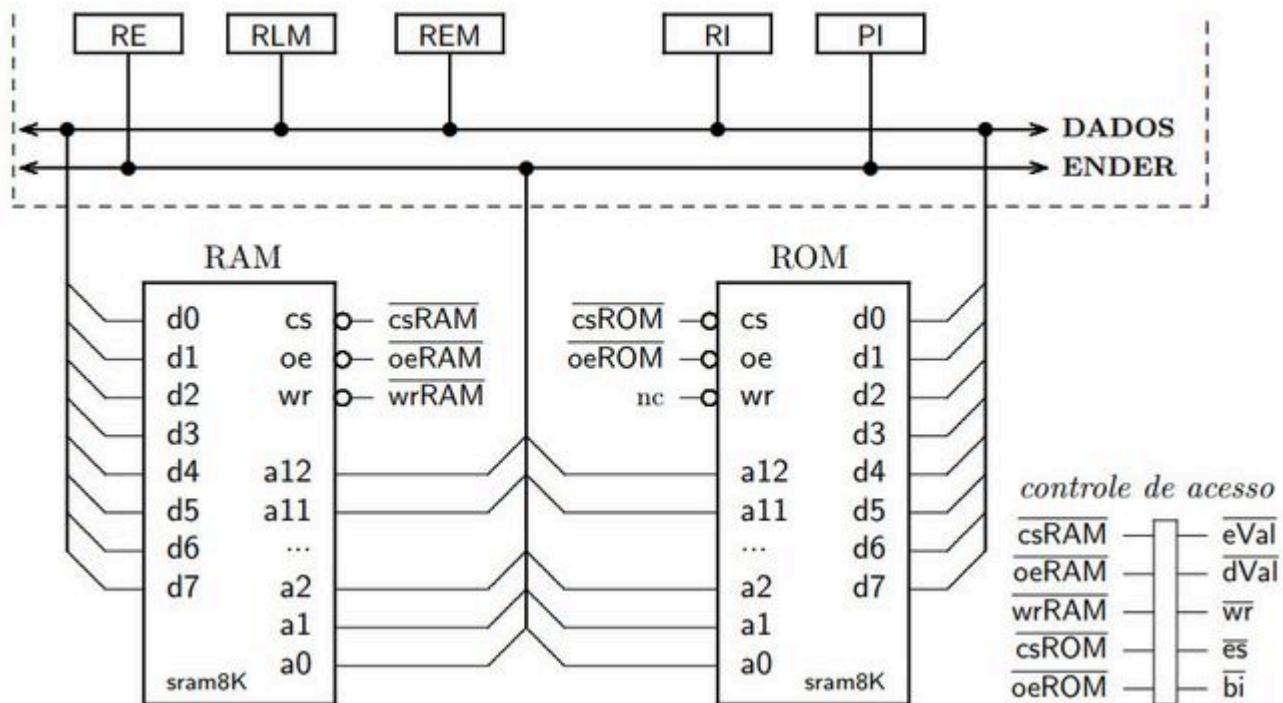


Figura 3 | Conexões com o barramento de memória do Mico. Fonte: Hexsel (2006, p. 94).

O controle de acesso ao barramento é realizado por uma combinação de sinais de controle, como csROM, csRAM, oeROM, oeRAM e wrRAM, que são necessários para a operação correta dos circuitos de memória integrados. Estes sinais são fundamentais para garantir que os dados corretos sejam transferidos entre a memória e o processador nos momentos adequados, seguindo os ciclos de memória para busca, leitura e escrita, além de ciclos especiais para atendimento de interrupções e inicialização do processador.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

A operação da memória no Mico envolve várias fases, como o endereçamento, no qual os endereços de memória são transmitidos, e a transferência de dados. Existem três tipos principais de ciclos de memória: busca, leitura e escrita, além de ciclos não relacionados à memória, como os de atendimento a interrupções e inicialização. Processadores mais complexos exigem milhares de ciclos para inicialização, mas o Mico, devido à sua simplicidade, requer um processo menos complexo, embora a simulação de sistemas de memória deva considerar o tempo real de acesso à memória, que é significativamente maior do que o tempo de propagação de uma porta lógica.

O ciclo de busca no Mico envolve a leitura de uma instrução da memória, resultando na instrução armazenada no registrador r_i ao final do ciclo. A leitura e escrita operam de maneira similar: durante a leitura, o endereço efetivo determinado pela instrução resulta na transferência de dados para o registrador de leitura de memória (rlm), mantendo o sinal de escrita (wr) desativado; para a escrita, o conteúdo do registrador de escrita em memória (rem) é transferido para a memória, com o sinal wr ativo. Os ciclos de entrada/saída funcionam de forma comparável aos de leitura e escrita, diferenciando-se pelo uso do sinal es e pela manipulação de dados com periféricos. A Figura 4 mostra o diagrama de tempo de um ciclo de escrita.

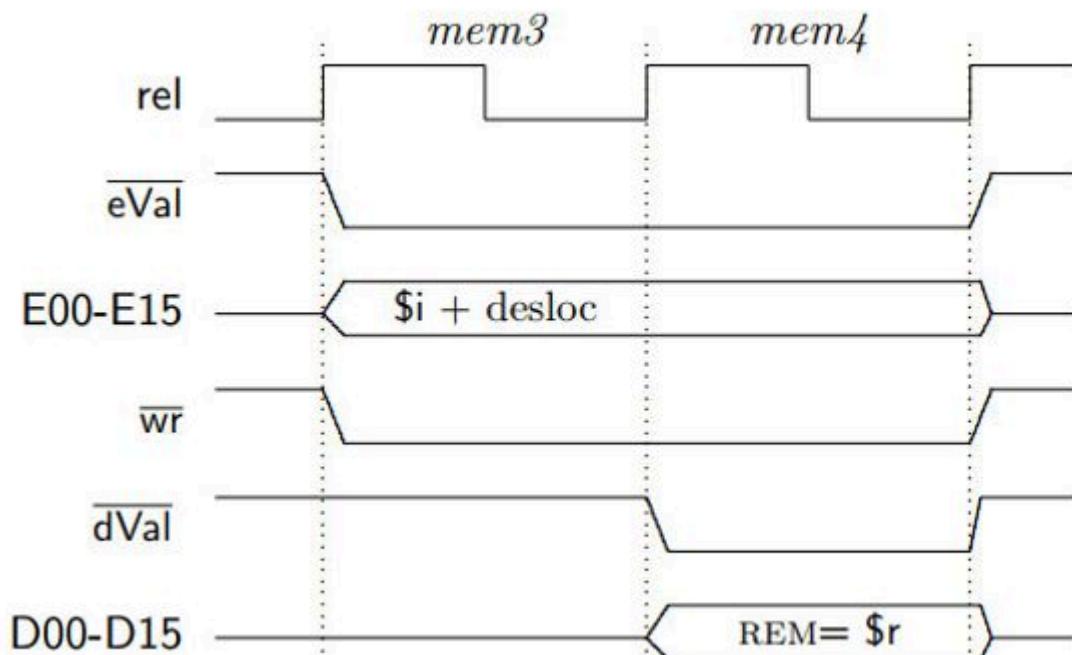


Figura 4 | Ciclo de escrita. Fonte: Hexsel (2006, p. 96).

Além disso, na configuração de memória nos CIs, o sinal RD é ajustado para definir o modo de operação como leitura (quando ativo) ou escrita, em conjunto com os sinais CE (chip enable) e

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

OE (output enable), facilitando o acesso e a manipulação dos dados na memória. As lógicas de ativação desses sinais determinam a leitura ou atualização dos dados na memória, com a seleção dos CI's baseada na validade do endereço e no tipo de ciclo (de código com $bi=0$ para ROM, ou de dados com $bi=1$ para RAM), e a habilitação da saída dos CI's dependendo da validade dos dados para leitura ou escrita.

Siga em Frente...

Círculo de Controle e Periféricos

O círculo de controle do Mico utiliza prefixos específicos em seus sinais de controle para indicar suas funções: "e" para habilitação de escrita em registradores, "s" para controle de saídas de seletores e multiplexadores, "h" para habilitação de saídas tristate, e "c" para outros sinais de controle. Esses sinais podem representar tanto um único bit quanto múltiplos bits, dependendo da função específica que desempenham no circuito. É importante notar que a ativação dos sinais depende do nível lógico especificado pela implementação do sistema.

Na interface com o barramento de memória, os sinais eVal, dVal, bi, e wr são utilizados conforme a instrução em execução, interagindo com registradores específicos como o registrador de endereço de dados (re), o registrador de leitura de memória (rlm), o registrador de escrita em memória (rem), e o endereço da próxima instrução (pi), cada um com seus respectivos sinais de controle para atualização e habilitação de saída tristate. A Figura 5 ilustra como estes componentes são organizados dentro do círculo de controle do Mico, destacando a importância de implementar certos registradores como buffers tristate para otimizar o tempo de resposta, evitando a necessidade de ciclos de relógio adicionais para operações de gravação.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

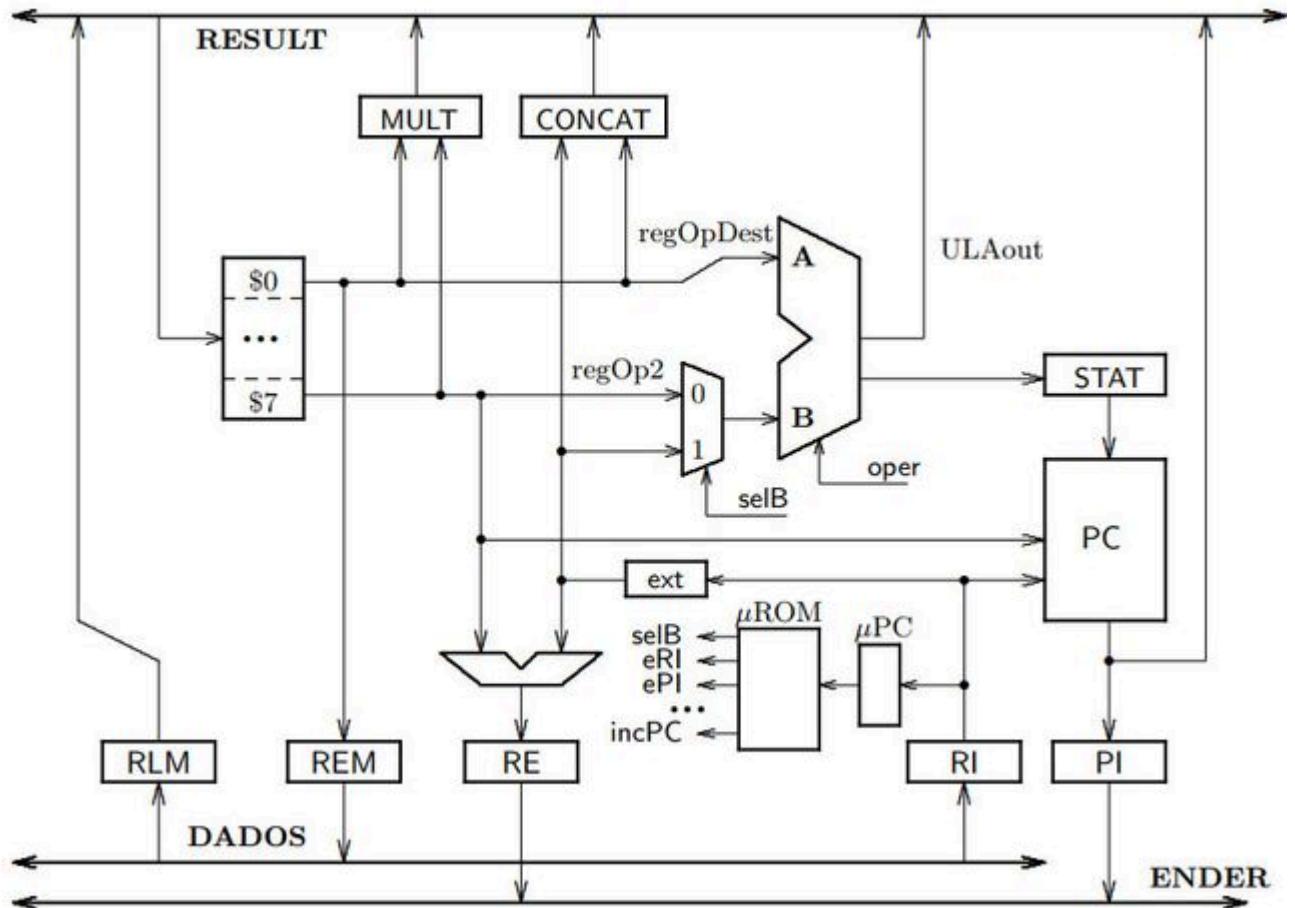


Figura 5 | Esquema do circuito de controle no Mico. Fonte: Hexsel (2006, p. 97).

Os sinais de controle do circuito de dados do Mico são essenciais para garantir o funcionamento adequado do processador, dependendo da implementação específica e da responsabilidade dos projetistas em adequá-los às necessidades de cada projeto. O contador de programa (PC), por exemplo, é um contador de 16 bits que avança com base em condições específicas, influenciadas por sinais de controle como reset, carPC, e habPC, determinando a inicialização do processador, a seleção de instruções e o avanço para a próxima instrução, respectivamente, conforme ilustrado na Tabela 2 e na Figura 6.

Sinal	Efeito no PC	Instrução/Observações
reset	PC := 0	inicialização do processador
carPC	PC := (sPC ? PC/15..12/Lrl/11..0/ : \$r : (PC^+ + ext(RI/7..0/)) : PC)	síncrono; j, jal; jr; desvios; nada acontece
habPC	PC := PC+1	final da busca, síncrono

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Tabela 2 | Atribuição de valores ao contador de programa (PC). Fonte: Hexsel (2006, p. 99).

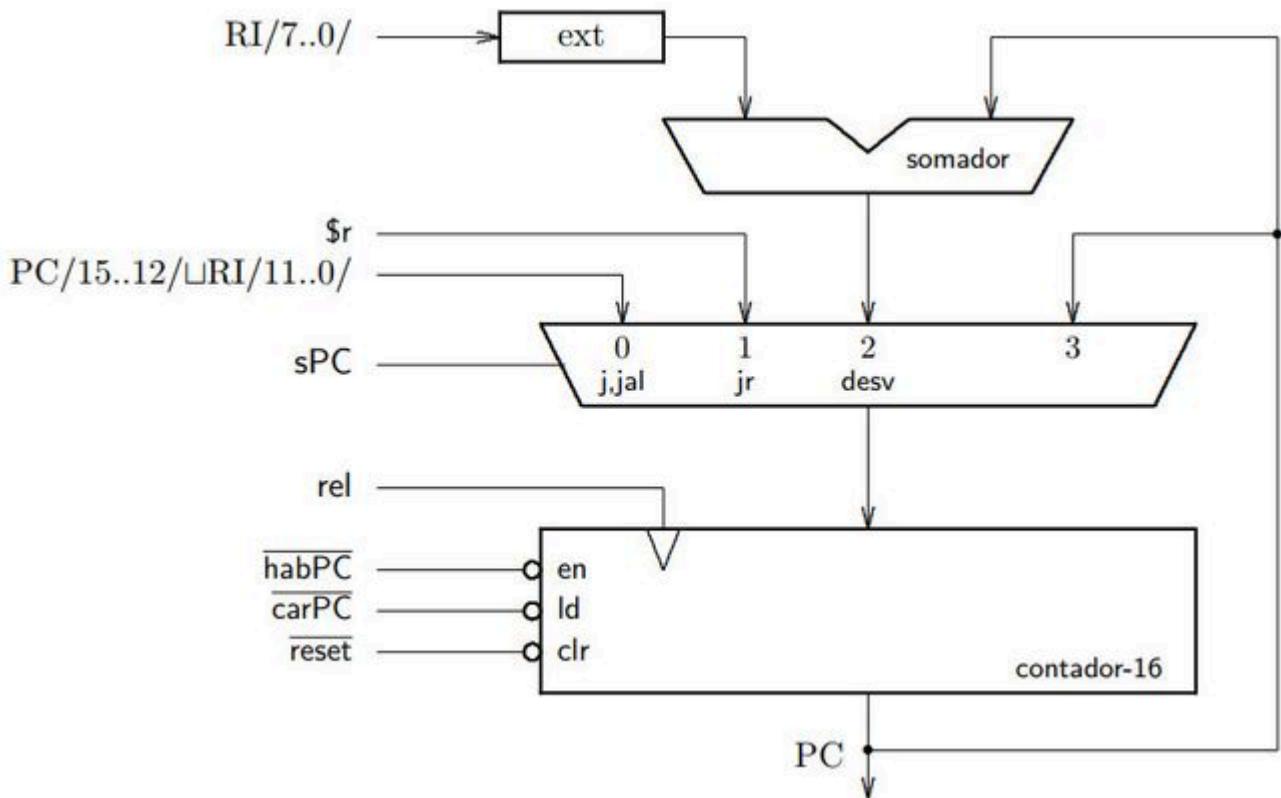


Figura 6 | Contador de programa e circuito para controle de fluxo. Fonte: Hexsel (2006, p. 99).

O circuito também engloba o registrador de instrução (RI), que armazena a instrução atual sendo processada, e o registrador de status (STAT), que mantém o status da última operação da unidade lógica e aritmética (ULA). O bloco de registradores, que contém registradores de 16 bits com funcionalidades específicas para leitura e escrita, é crucial para a execução de instruções, com a lógica de seleção baseada nos sinais recebidos e gerados pelo próprio conjunto de instruções do Mico, destacando a flexibilidade e a complexidade do controle de fluxo e operações dentro do processador.

No Mico, o método de controle microprogramado é usado, no qual os sinais de controle do processador são gerados a partir de uma memória de microprograma. Cada posição dessa memória contém uma microinstrução, e uma microrrotina composta por uma sequência dessas microinstruções implementa cada instrução de máquina. O contador microprogramado (μ pc) percorre a memória de microprograma na ordem correta para executar as instruções. A execução de um programa é um processo de transição entre os estados de um diagrama, com cada estado correspondendo a uma microinstrução e cada transição a uma microrrotina.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

A memória de microprograma do Mico, ou μ rom, é ampla e contém centenas de palavras que são acessadas pelo μ pc. Na implementação prática, várias memórias SRAM8K são utilizadas para simular a μ rom, dividindo-a em blocos associados a cada instrução. Esses blocos são mapeados de acordo com os opcodes de 4 bits, com cada instrução tendo um espaço de até 32 microinstruções para executar as operações necessárias. No fim de cada instrução, a microinstrução de busca é ativada para carregar a próxima instrução, mantendo o fluxo de execução do programa.

O desempenho do Mico pode ser melhorado utilizando a técnica de busca antecipada, que consiste em usar os ciclos ociosos durante as operações da unidade lógica e aritmética (ULA) para buscar a próxima instrução. Isso efetivamente reduz os ciclos gastos por instruções de ULA, otimizando o tempo de execução do programa. A busca antecipada não é aplicável a instruções de desvios e acessos à memória devido às demandas de tempo e uso do barramento de memória. A arquitetura de Princeton, utilizada pelo Mico, emprega um barramento único para busca de instruções e acesso a dados, diferentemente da arquitetura de Harvard, que poderia permitir a busca simultânea de instruções e execução de instruções de Id e st de forma independente.

Existem três métodos principais para alojar endereços a dispositivos periféricos em relação ao processador: mapeamento como memória, mapeamento em um espaço de endereçamento de entrada/saída (E/S) dedicado, e um sistema híbrido que combina os dois. No mapeamento como memória, os periféricos são acessados por meio das instruções de carga e armazenamento (Id e st), o que facilita a programação, pois permite que os periféricos sejam representados por estruturas de dados e manipulados em linguagens de alto nível como C.

Alternativamente, no mapeamento como E/S, os periféricos são acessados por meio de um espaço de endereçamento separado, requerendo instruções especiais de E/S, como in e out, tornando o código mais difícil de portar para outros processadores ou dispositivos semelhantes – isso porque os endereços dos dispositivos precisam ser explicitamente definidos no código. Este método é comumente visto em microprocessadores da Intel (8085, 8086) e Zilog (Z80), nos quais o espaço de endereçamento de E/S é tipicamente menor que o de memória devido ao número reduzido de periféricos e à simplicidade dos modos de endereçamento dessas instruções.

Por fim, o mapeamento híbrido, utilizado em computadores pessoais com processadores mais avançados da Intel a partir do 80486, permite que alguns registradores de periféricos sejam acessados por instruções de E/S, enquanto áreas maiores usadas para transferência de dados, como filas e buffers, são mapeadas como memória. Este método aproveita as vantagens dos dois sistemas anteriores, permitindo a flexibilidade na programação e eficiência no acesso aos diferentes tipos de dispositivos periféricos.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Vamos Exercitar?

Otimização de interface de memória

Na seção "Ponto de Partida", propusemos uma jornada investigativa com Ana Beatriz e Carlos Eduardo, que enfrentavam o desafio de otimizar a interface de memória de um sistema de automação. Eles precisavam aplicar seus conhecimentos sobre os sinais específicos da interface de memória do Mico – “wr”, “bi”, “es” e outros – para aumentar a eficiência e reduzir a latência nas operações do sistema.

Para resolver essa problematização, Ana e Carlos mergulharam nas complexidades da interface de memória do Mico, estudando como as linhas de dados, as linhas de endereço e os sinais de controle interagem para facilitar a comunicação entre o processador e a memória. Eles descobriram que ajustando corretamente os sinais “wr” para escrita, “bi” para busca e “es” para execução de instruções de entrada/saída, poderiam significativamente otimizar o processo de leitura e escrita, melhorando assim a performance do sistema.

Esta exploração teórica e prática não só proporcionou a Ana e Carlos a solução para o desafio em mãos, como também ofereceu a você, estudante, a oportunidade de compreender a aplicação real dos conceitos de interface de memória. Reflita: como você aplicaria esses conhecimentos em um projeto próprio? Quais outras estratégias poderiam ser exploradas para maximizar a eficiência de sistemas baseados em microprocessadores como o Mico? Engaje-se nesta reflexão, e explore possibilidades que transcendam as soluções tradicionais, abrindo caminho para inovações no campo da tecnologia da informação.

Saiba mais

Para explorar mais as interfaces de memória e sua função nos sistemas computacionais, recomendamos o livro [*Sistemas Digitais e Microprocessadores*](#) de R. A. Hexsel. Esta obra, proveniente do Departamento de Informática da Universidade Federal do Paraná, 2006, abrange desde os fundamentos até aplicações avançadas das memórias em arquiteturas de computadores. Aprofundando-se nas páginas 93 a 110, nas quais você encontrará insights detalhados de memória e a dinâmica entre os sinais de controle na interface de memória do Mico, além de outros tópicos essenciais para quem busca entender ou aperfeiçoar sistemas digitais.

Referências

CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

HEXSEL, R. A. **Sistemas Digitais e Microprocessadores**. Departamento de Informática, Universidade Federal do Paraná, 2006.

IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.

LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.

SOUZA, D. B. da C. *et al.* **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Descubra como o conhecimento de microprocessadores se entrelaça com a prática de engenharia na nossa videoaula de encerramento. Conecte cada instrução aprendida às suas aplicações reais, desde conceber algoritmos até otimizar sistemas digitais. Não perca esta oportunidade de transformar teoria em prática, preparando-se para um futuro brilhante na tecnologia da informação. Junte-se a nós e consolide sua jornada de aprendizado.

Ponto de Chegada

Competência em Sistemas Digitais e Microprocessados

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Caro estudante, nas aulas que tratou dos sistemas digitais e microprocessados, você decompôs a complexidade de um microprocessador em suas partes constituintes, abrindo caminho para uma sólida compreensão de sua funcionalidade. Esses conhecimentos são necessários para desenvolver a competência desta unidade, que exige compreender o funcionamento de um microprocessador e ser capaz de utilizar os seus componentes essenciais. A capacidade de identificar, compreender e aplicar o conjunto de instruções do microprocessador Mico evidencia que você não apenas absorveu o conhecimento teórico, mas também está apto a aplicá-lo na prática.

Você se familiarizou com os elementos vitais do processador: os registradores, a unidade lógica e aritmética (ULA), os mecanismos de controle, a interface com a memória, e mais importante, como estas partes interagem por meio de instruções precisas. As instruções como LOAD, STORE, ADD, SUB e JUMP deixaram de ser meros conceitos para se tornarem ferramentas na sua mão, permitindo que você manipule dados, execute algoritmos e resolva problemas com eficiência e inovação.

Neste ponto de chegada, reconhecemos a habilidade adquirida de compreender o microprocessador não como um dispositivo distante e intangível, mas como um ambiente em que cada instrução tem um propósito e cada ciclo de execução contribui para o resultado. Seja calculando médias simples ou executando operações complexas, a competência que você desenvolveu servirá de alicerce para todos os desafios futuros na área de tecnologia digital.

Em posse desse entendimento, olhe para frente com confiança, sabendo que está equipado para aplicar a teoria em cenários reais e inovar no campo dos sistemas digitais e microprocessados.

Parabéns por alcançar este marco. Continue avançando com curiosidade e determinação, pois cada passo à frente é um passo rumo à maestria na engenharia de sistemas digitais.

Até a próxima etapa de sua jornada educativa!

É Hora de Praticar!

Desenvolvimento de um Sistema de Controle para Veículos Autônomos

Imagine que você é um engenheiro de sistemas trabalhando para uma empresa chamada **AutoTech Solutions**. A empresa está desenvolvendo um sistema de controle para veículos autônomos. Este sistema deve ser capaz de processar dados em tempo real de diversos sensores (câmeras, LIDAR, radar) para tomar decisões rápidas e seguras, garantindo a navegação eficiente e a segurança dos passageiros.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Personagens e Empresas Envolvidas

- **Engenheiro de Sistemas (Você):** Responsável pelo design e implementação do sistema de controle.
- **Engenheiro de Software (Paula):** Trabalha no desenvolvimento dos algoritmos de navegação e controle.
- **Gerente de Projeto (Rodrigo):** Coordena as atividades de desenvolvimento e garante a integração dos diferentes componentes do sistema.
- **AutoTech Solutions:** Empresa desenvolvedora de tecnologias para veículos autônomos.

Problema Proposto

O desafio é desenvolver um microprocessador eficiente para o sistema de controle do veículo autônomo que deve ser capaz de:

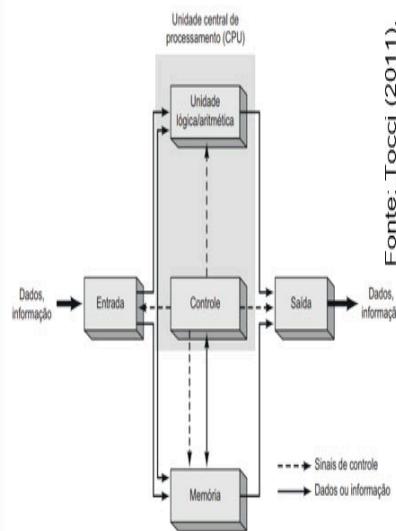
- Processar dados de múltiplos sensores em tempo real.
- Tomar decisões rápidas e precisas para a navegação do veículo.
- Garantir a segurança e a confiabilidade das operações do veículo.
- Como garantir que o microprocessador pode processar dados de múltiplos sensores em tempo real de forma eficiente?
- Quais tipos de memória e circuitos de controle são mais adequados para esta aplicação de veículo autônomo?
- Como assegurar a segurança e a confiabilidade do sistema de controle do veículo autônomo?

Olá estudante, chegamos ao encerramento da unidade!

Vamos realizar a experiência presencial que irá consolidar os conhecimentos adquiridos? É a oportunidade perfeita para aplicar, na prática, o que foi aprendido em sua disciplina. Vamos transformar teoria em vivência e tornar esta etapa ainda mais significativa. Não perca essa chance única de colocar em prática o conhecimento adquirido.

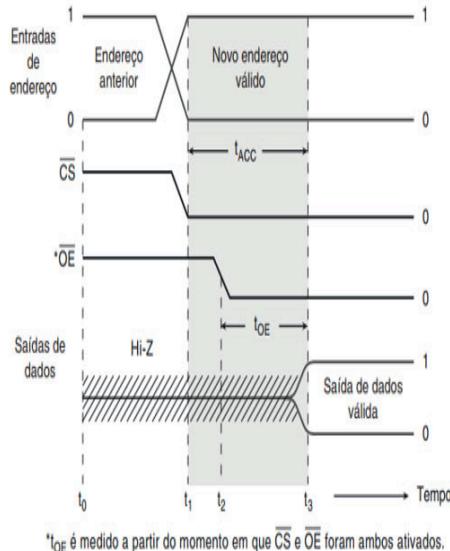
SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Diagrama de uma CPU



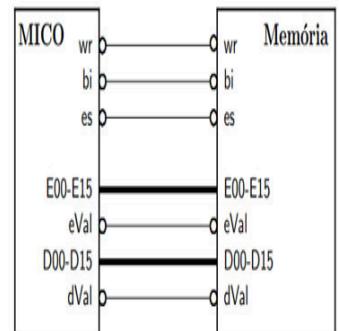
Fonte: Tocci (2011).

Tempo de Acesso ROM

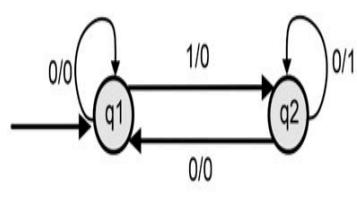


Fonte: Tocci (2011).

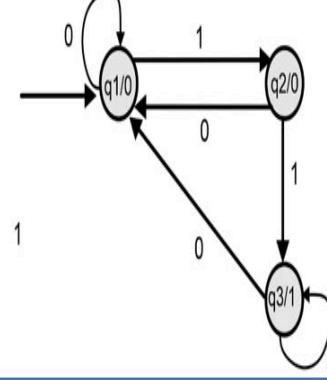
Interface de memória do Micro



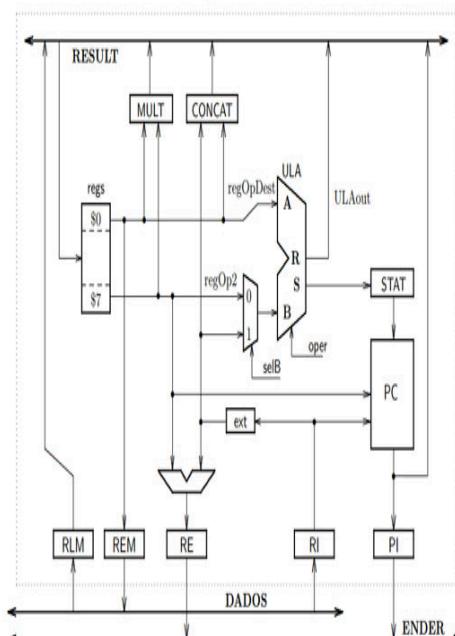
Máquina de Mealy



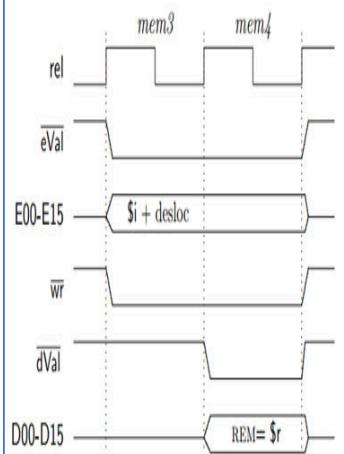
Máquina de Moore



Micro em diagrama de blocos



Ciclo de escrita



SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

SOUZA, Diogo B. da Costa; SANTOS, Sidney C. Bispo dos; MARTON, Italo L. de Alencar et al. **Sistemas digitais**. Porto Alegre: SER - SAGAH, 2018.

CRUZ, Eduardo; GAUDINO, Enzo; ADRIANO, Domingos et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Editora Alta Books, 2022.

HEXSEL, R. A. **Sistemas Digitais e Microprocessadores**. Departamento de Informática, Universidade Federal do Paraná, 2006.

IDOETA, Ivan Valeije; CAPUANO, Francisco Gabriel. **Elementos de eletrônica digital**. 42 ed. São Paulo: Érica, 2019.

TOCCI, R. J.; Widmer, N. S. **Sistemas Digitais - princípios e aplicações**. 11a edição. Rio de Janeiro: LTC - Livros técnicos e científicos, 2011.

Unidade 3

PROGRAMAÇÃO DE MICROPROCESSADORES

Aula 1

Linguagem de Programação

Linguagem de Programação



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Mergulhe no universo dos microprocessadores e descubra como programá-los para impulsionar inovações práticas! Esta videoaula equipa você com o conhecimento necessário para transformar teorias em aplicações reais. Conecte-se agora e amplie suas habilidades, preparando-se para o futuro da tecnologia. Junte-se a nós e comece a programar o amanhã, hoje!

Ponto de Partida

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

À medida que mergulhamos no universo da inovação tecnológica, os microprocessadores surgem como peças-chave que impulsionam nossa interação cotidiana com a tecnologia. Eles são o coração pulsante por trás de dispositivos que variam de celulares a sistemas avançados de automação residencial. Compreender a essência dos microprocessadores e sua programação é um conhecimento crucial, não só para os engenheiros e desenvolvedores de software, mas também para visionários tecnológicos em potencial.

Nesta aula, Ana Beatriz e Carlos Eduardo serão nossos guias. Juntos, enfrentarão o desafio de programar um microprocessador para automatizar o monitoramento de um jardim comunitário, ilustrado na Figura 1. Ana, com seu pensamento analítico, e Carlos, com sua criatividade técnica, ilustrarão o processo de usar algoritmos e linguagens de programação para criar um sistema eficiente de irrigação automatizada, ajustando a umidade do solo em resposta às variações climáticas detectadas por sensores ambientais.



Figura 1 | Ana e Carlos: parceria em automação de monitoramento de um jardim comunitário. Fonte: elaborada pelo autor com auxílio de ferramenta de inteligência artificial.

Este cenário realista serve como nossa problematização e nos mostra a relevância dos conceitos de algoritmos e linguagens de programação. Ao aplicar essas habilidades na realidade, Ana e Carlos não estão apenas resolvendo um problema imediato; estão também explorando o potencial de suas carreiras futuras em tecnologia. Eles nos demonstram como, com a compreensão correta e aplicação prática, pequenos chips podem gerar grandes impactos na sociedade.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Prepare-se para se juntar a Ana e Carlos nesta jornada educativa. Ao adentrar nos fundamentos da programação de microprocessadores, você também se capacitará a aplicar esses conhecimentos de maneira prática, abrindo caminho para inovações que podem transformar nosso mundo. Juntos, vamos aprender a linguagem secreta dos microprocessadores e a arte de dar vida às máquinas.

Vamos Começar!

Linguagem de Programação

Bem-vindo à jornada pelo mundo fascinante da programação de microprocessadores! Vamos começar explorando dois conceitos fundamentais: algoritmo e linguagem de programação.

Imagine um algoritmo como uma receita de pão de queijo, mas em vez de criar algo delicioso para comer, estamos solucionando problemas ou realizando tarefas. Um algoritmo é uma sequência finita e bem definida de instruções ou passos que visam resolver um problema ou realizar uma tarefa específica. Essas instruções devem ser claras e precisas, de modo que possam ser seguidas por um executor – neste caso, o microprocessador – sem ambiguidade.

Cada passo do algoritmo leva mais perto da solução final. Da mesma forma que uma receita tem etapas que devem ser seguidas em uma certa ordem para que o pão de queijo saia como esperado, um algoritmo tem uma ordem específica de operações que garantem o resultado desejado, conforme ilustrado na Figura 2. Essa sequência lógica é o que permite aos microprocessadores executar desde tarefas simples, como somar dois números, até operações complexas, como controlar o sistema de navegação de um veículo autônomo.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Figura 2 | Algoritmo: uma receita de pão de queijo. Fonte: Freepik.

Se um algoritmo é a receita, então a linguagem de programação é o idioma em que essa receita é escrita. É o meio pelo qual comunicamos ao microprocessador quais tarefas queremos que ele execute. Existem diversas linguagens de programação, cada uma com sua sintaxe (regras de escrita) e semântica (significado) próprias, projetadas para diferentes tipos de tarefas e níveis de abstração.

Uma linguagem de programação permite ao programador especificar de forma precisa quais operações o computador deve realizar e em que sequência. Essas linguagens podem variar desde instruções muito próximas do código de máquina, como Assembly, até linguagens de alto nível, como Python ou C, que abstraem muitos dos detalhes do hardware e permitem focar a lógica do problema.

A Tabela 1 comparativa de linguagens de programação para microprocessadores mostra que o Assembly oferece máximo controle e eficiência, mas com portabilidade limitada e difícil aprendizado, ideal para firmware e sistemas embarcados que exigem manipulação direta do hardware. A linguagem C é um meio termo prático, proporcionando controle e eficiência com melhor portabilidade e usabilidade, comum em sistemas operacionais e hardware embarcado. A

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

linguagem C++ avança na abstração e mantém a eficiência, adicionando orientação a objetos, favorecida em aplicações de alto desempenho como jogos. A linguagem Python se destaca na facilidade de uso e abstração para prototipagem rápida e automação, porém com menor controle do hardware. Java equilibra usabilidade e eficiência, sendo muito portátil, escolhida para aplicações empresariais e desenvolvimento Android. Cada linguagem reflete uma especialização que atende a diferentes necessidades de projetos e ambientes de aplicação dos microprocessadores.

Tabela 1 | Comparativo de linguagens de programação para microprocessadores

Bloco 1

LP	Abstração	Portabilidade	Facilidade
Assembly	Baixo	Baixa	Difícil
C	Médio	Alta	Moderada
C++	Alto	Alta	Moderada
Python	Alto	Alta	Fácil
Java	Alto	Alta	Moderada

Bloco 2

Eficiência	Controle sobre o hardware	Uso comum
Alta	Máximo	Sistemas embarcados, firmware
Alta	Elevado	Sistemas operacionais, sistemas embarcados
Alta	Elevado	Aplicações de desempenho crítico, jogos
Moderada	Baixo	Prototipagem, scripts de automação
Moderada	Baixo	Aplicações empresariais, Android

Fonte: elaborada pelo autor.

Ao programar um microprocessador, escolhemos uma linguagem de programação que nos permita expressar nossos algoritmos de forma eficiente e eficaz, traduzindo-os em instruções que o microprocessador possa entender e executar. A escolha da linguagem muitas vezes depende do problema específico que estamos tentando resolver, das características do dispositivo que estamos programando e de quão próximo do hardware precisamos estar para alcançar o desempenho desejado.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Entender o que são algoritmos e linguagens de programação é essencial para começar a programar microprocessadores. Com esses conceitos em mãos, estamos preparados para explorar mais profundamente como escrever programas que instruam esses dispositivos a realizar tarefas incríveis. À medida que avançamos, lembramos que, no coração de cada dispositivo eletrônico, há um microprocessador esperando para ser guiado por nossas instruções precisas e criativas. Vamos nessa jornada juntos, transformando ideias em realidade por meio da programação!

Siga em Frente...

Operações e Expressões

Dando continuidade ao nosso aprendizado sobre programação de microprocessadores, vamos nos aprofundar em uma das linguagens mais fundamentais e recomendadas para este fim: a linguagem C. Devido à sua portabilidade, eficiência e controle considerável sobre o hardware, além de ser um dos pilares para outras linguagens mais modernas, C é uma escolha sólida para a nossa exploração.

No coração de qualquer linguagem de programação estão as operações que podemos realizar com dados. Em C, operações são majoritariamente divididas em três categorias: aritméticas, relacionais e lógicas, descritas na Tabela 2.

Operações aritméticas: incluem adição (+), subtração (-), multiplicação (*), divisão (/), e o módulo (%) que retorna o resto de uma divisão.

Operações relacionais: usadas para comparar dois valores, resultando em verdadeiro ou falso. Incluem igual a (==), diferente (!=), maior que (>), menor que (<), maior ou igual a (>=), e menor ou igual a (<=).

Operações lógicas: permitem combinar condições e incluem E lógico (&&), OU lógico (||), e NÃO (!).

Tabela 2 | Tabela de operadores em linguagem C

Bloco 1

Categoria	Operador	Descrição	Exemplo
Aritmética	+	Adição	5 + 3
Aritmética	-	Subtração	5 - 3
Aritmética	*	Multiplicação	5 * 3
Aritmética	/	Divisão	5 / 3

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Aritmética	%	Módulo	5 % 3
Relacional	==	Igualdade	5 == 3
Relacional	!=	Diferença	5 != 3
Relacional	>	Maior que	5 > 3
Relacional	<	Menor que	5 < 3
Relacional	>=	Maior ou igual	5 >= 3
Relacional	<=	Menor ou igual	5 <= 3
Lógica	&&	E lógico	(5 > 3) && (3 > 1)
Lógica		OU lógico	(5 < 3) (3 > 1)
Lógica	!	NÃO	!(5 == 3)

Bloco 2

Resultado
8
2
15
1 (em divisão inteira)
2
0 (falso)
1 (verdadeiro)
1 (verdadeiro)
0 (falso)
1 (verdadeiro)
0 (falso)
1 (verdadeiro)
1 (verdadeiro)
1 (verdadeiro)

Fonte: elaborada pelo autor.

Expressões são combinações de valores, variáveis e operadores que, juntos, resultam em um novo valor. Elas são fundamentais para a lógica de qualquer programa em C, usadas para atribuir valores a variáveis, tomar decisões e realizar cálculos. Por exemplo, uma expressão pode ser tão simples quanto $a + b$, ou tão complexa quanto $(a + b) * (c / d) - e$.

Exemplo Prático

Vamos ver um exemplo prático de uma expressão em C que usa tanto operações aritméticas quanto relacionais:

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```
#include <stdio.h>
int main() {
    int a = 5;
    int b = 10;
    int c = a * b; // operação aritmética
    int d = (c > 20) ? 1 : 0; // operação relacional dentro de uma expressão condicional
    printf("Resultado: %d\n", d);
    return 0;
}
```

O algoritmo em C apresentado realiza operações simples com variáveis inteiras, multiplicando os valores de "a" e "b" para atribuir a "c", em seguida, utiliza uma expressão condicional ternária para verificar se "c" é maior que 20, atribuindo 1 a "d" se verdadeiro, caso contrário, atribuindo 0. O programa então imprime o valor de "d", demonstrando o uso de operações aritméticas, relacionais e condicionais em C para executar diferentes tarefas com base em condições específicas. A expressão $(c > 20) ? 1 : 0$ é um exemplo de uma expressão condicional em C, em que o resultado depende da avaliação da condição $c > 20$.

Compreender operações e expressões em C é fundamental para criar programas eficientes e eficazes. Isso se torna ainda mais crucial quando programamos microprocessadores, quando cada instrução pode ter um impacto significativo no desempenho do sistema. À medida que continuamos, encorajamos a prática e a experimentação com esses conceitos, para que se tornem ferramentas naturais no seu arsenal de programação.

Vamos Exercitar?

Programando a Automação de um Jardim Comunitário

Ana Beatriz e Carlos Eduardo estão colaborando em um projeto desafiador e educativo: automatizar o sistema de irrigação de um jardim comunitário. Para isso, eles precisam programar um microprocessador que controle o sistema com base nos dados coletados por sensores de umidade, temperatura e luminosidade. Vamos seguir o algoritmo que eles propuseram:

Início: Carlos verifica os sensores, enquanto Ana inicializa o sistema de monitoramento.

Leitura dos dados: Ana lê os dados atuais de umidade e compara com os limiares definidos.

Decisão: juntos, eles estabelecem as condições: Se a umidade está abaixo do esperado e não há previsão de chuva, o sistema deve agir.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Ação de irrigação: Carlos ativa a válvula de irrigação enquanto Ana monitora os níveis de umidade.

Temporização: Ana determina o tempo necessário para atingir a umidade ideal e Carlos ajusta o temporizador.

Registro: Carlos anota as ações realizadas e Ana analisa os dados para futuras otimizações.

Ciclo de irrigação: eles verificam se ainda está no horário programado para irrigação e, se sim, repetem o ciclo.

Conclusão: fora do horário, Ana desliga o sistema e eles revisam os dados coletados.

Código de Exemplo em C:

```
// Este é um esboço do código que Ana e Carlos desenvolveram para o microprocessador do
// sistema de irrigação.
#include <stdio.h>
// Suponha que estas funções interagem com o hardware real do jardim comunitário
int lerSensorUmidade();
int previsaoTempoChuva();
void abrirValvula();
void fecharValvula();
void registrarLog(int umidade, int acao);
int main() {
    const int umidadeldeal = 500; // Ana ajustou o valor ideal de umidade
    int umidadeAtual, necessidadeDeIrrigacao;

    while (1) {
        umidadeAtual = lerSensorUmidade();
        necessidadeDeIrrigacao = (umidadeAtual < umidadeldeal) && !previsaoTempoChuva();
        if (necessidadeDeIrrigacao) {
            abrirValvula();
            sleep(10); // Carlos definiu que 10 segundos seria o tempo ideal
            fecharValvula();
            registrarLog(umidadeAtual, 1); // Irrigação ativada
        } else {
            registrarLog(umidadeAtual, 0); // Nenhuma ação necessária
        }
        // Descanso entre ciclos de irrigação
        sleep(3600); // Ana pensou em verificar a cada hora
    }
}
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



```
    return 0;  
}
```

O programa em C desenvolvido por Ana Beatriz e Carlos Eduardo incorpora diretamente as lições abordadas na aula que aborda a programação de microprocessadores. As operações aritméticas e relacionais comparam os valores lidos com a umidade ideal definida por Ana, e o resultado dessa comparação determina a ação subsequente, executada por Carlos, que é o controle da válvula de irrigação, marcados em vermelho. O código resume a interação entre o hardware e o software, destacando a necessidade de um entendimento claro dos algoritmos e das capacidades da linguagem de programação para a criação de sistemas automatizados eficazes.

Após testar o sistema básico, Ana e Carlos podem expandir o projeto. Eles podem incluir um sensor de chuva para evitar irrigação desnecessária, uma interface de usuário para permitir ajustes manuais e uma análise de dados mais aprofundada para otimizar o consumo de água. Este exercício realça como a colaboração e a combinação de diferentes habilidades podem levar a soluções inovadoras e sustentáveis.

Saiba mais

Aprofunde seus conhecimentos em programação de microcontroladores com o estudo detalhado da linguagem C, explorando sua aplicação direta no desenvolvimento de sistemas embarcados. Este material complementar oferece uma visão abrangente das técnicas de programação específicas para microcontroladores, unindo teoria e prática. Explore desde a sintaxe da linguagem até o manejo efetivo de hardware, e avance em sua capacidade de criar soluções tecnológicas inovadoras.

Para expandir seu aprendizado na [linguagem C](#) aplicada a microcontroladores, acesse o material completo indicado, das páginas 58 a 66. Ele será um guia valioso para transformar conhecimento em habilidade prática. Boa leitura e exploração!

Referências

- CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. et al. **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. edição. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 2

Estruturas de Decisão e Repetição

Estruturas de Decisão e Repetição



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Domine estruturas condicionais e de repetição em C nesta videoaula focada em microprocessadores. Aprenda sobre if, else, while, do-while e for aplicando-as diretamente em projetos reais. Eleve sua capacidade de solucionar problemas e inicie sua transformação no mundo da automação. Inscreva-se e comece a criar!

Ponto de Partida

Continuando nossa exploração prática dos microprocessadores, Ana Beatriz e Carlos Eduardo nos guiam pelo intrincado mundo das estruturas de decisão condicional. Essenciais para dar inteligência e autonomia aos microprocessadores, essas estruturas permitem analisar dados e executar comandos baseados em condições variáveis. Ao longo desta aula, vamos decifrar como as instruções if, else e switch da linguagem C são convertidas em operações de máquina, capacitando os dispositivos a responder com precisão às mudanças em seu entorno. Estaremos ao lado de Ana e Carlos na continuação da implementação de um sistema de automação para um jardim comunitário, que servirá como exemplo da aplicação destas teorias, ilustrado na Figura 1. Ao aprender a programar com decisões condicionais, não apenas seguimos instruções; nós também projetamos a capacidade de decisão em tecnologia. Prepare-se para transformar ideias em realidade programável e dar o próximo passo na sua jornada tecnológica.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Neste complemento, vamos aprofundar nosso entendimento nas estruturas de decisão condicional, estruturas de repetição condicional e estruturas de decisão determinísticas. Exploraremos como essas técnicas formam a espinha dorsal da programação de microprocessadores, permitindo que sistemas como o nosso projeto de automação do jardim comunitário não apenas respondam a eventos específicos, mas também executem tarefas repetitivas de forma eficiente e tomem decisões previsíveis quando necessário. Com exemplos práticos, como o monitoramento e controle do ambiente do jardim que Ana Beatriz e Carlos Eduardo estão desenvolvendo, ilustraremos como combinar essas estruturas para criar sistemas autônomos. Esse entendimento aprofundado nos preparará para desafios futuros na programação de dispositivos inteligentes, enfatizando a importância da precisão, eficiência e adaptabilidade no mundo da tecnologia.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

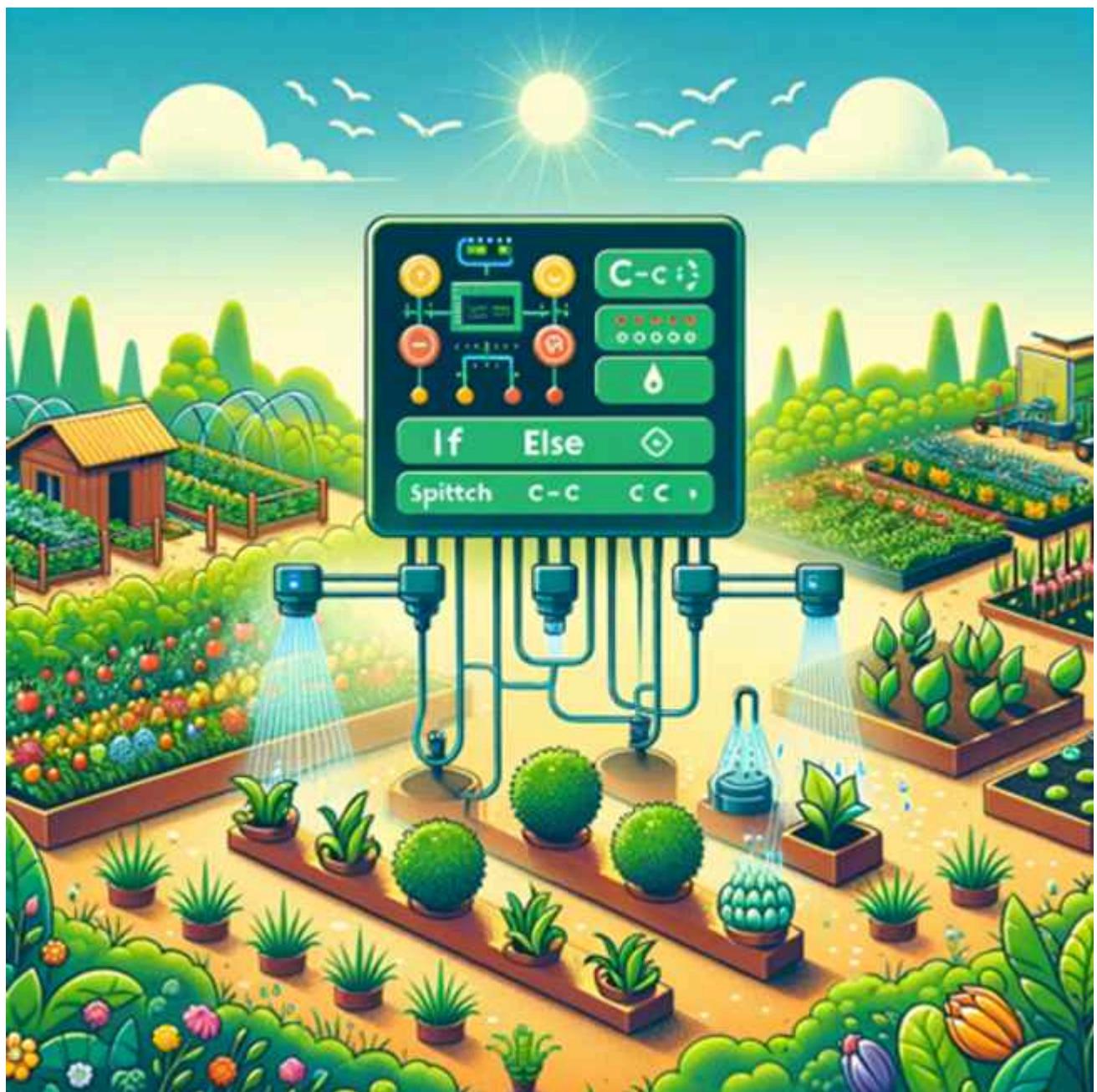


Figura 1 | Ana e Carlos: parceria em automação monitoramento de um jardim comunitário. Fonte: elaborada pelo autor com auxílio de ferramenta de inteligência artificial.

Vamos Começar!

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Estruturas de decisão condicional

As estruturas de decisão como if, else, e switch são fundamentais na programação de microprocessadores. Elas permitem que código escrito em linguagens de alto nível como C seja traduzido pelo compilador em código de máquina, que o microprocessador entende e executa. Em Assembly, isso é feito por meio de comparações e comandos de salto que direcionam o fluxo de execução. Essas operações condicionais são cruciais para que os microprocessadores executem desde operações simples até controle de sistemas complexos.

Estrutura if: avalia uma condição e, se verdadeira, executa um bloco de código.

```
if (condição) {
    // Código a ser executado se a condição for verdadeira
}
```

Estrutura else: vinculada a um if, executa um bloco de código alternativo se a condição inicial for falsa.

```
if (condição) {
    // Código se a condição for verdadeira
} else {
    // Código se a condição for falsa
}
```

Estrutura else if: usada para testar múltiplas condições em sequência.

```
if (condição1) {
    // Código se condição1 for verdadeira
} else if (condição2) {
    // Código se condição2 for verdadeira
} else {
    // Código se nenhuma das condições anteriores for verdadeira
}
```

Estrutura switch: seleciona um bloco de código para executar com base no valor de uma variável.

```
switch (variável) {
    case valor1:
        // Código para valor1
        break;
    case valor2:
        // Código para valor2
        break;
    default:
        // Código se nenhum valor corresponder
}
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Em microprocessadores, a programação em C é compilada para o conjunto de instruções específicas da arquitetura do processador. Instruções de alto nível como if e switch são traduzidas em sequências de instruções de máquina que utilizam comparações e saltos condicionais. Por exemplo, o código C:

```
if (temperatura > 30) {  
    acionaVentilador();  
}
```

Nessa estrutura, será compilada para uma sequência que pode incluir uma instrução de comparação seguida por um salto condicional, que só é seguido se a comparação for verdadeira (isto é, se “temperatura” for de fato maior que “30”).

Esse poder de decisão é essencial para funcionalidades como sistemas de controle e automação, nas quais decisões baseadas em entradas de sensores são frequentes. Por exemplo, em um sistema de monitoramento de temperatura:

```
int temperatura = lerSensorTemperatura();  
if (temperatura > 30) {  
    acionaVentilador();  
} else {  
    desligaVentilador();  
}
```

O código ilustra como um microprocessador pode ser programado para responder automaticamente às mudanças de temperatura, ativando ou desativando um ventilador. O controle de fluxo condicional garante que o microprocessador execute as ações apropriadas baseadas nas condições atuais do ambiente.

Programar microprocessadores, entender e aplicar corretamente as estruturas de decisão condicional é vital para criar sistemas robustos e eficientes que possam reagir a uma ampla gama de condições de entrada e executar ações apropriadas.

Siga em Frente...

Estruturas de Repetição Condicional e Estruturas de Decisão Determinísticas

Estruturas de repetição, ou laços, permitem que microprocessadores executem um conjunto de instruções repetidamente, baseando-se em uma condição. Em C, os laços mais comuns são while, do-while, e for.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Estrutura **while**: executa um bloco de código enquanto a condição for verdadeira.

```
while (condição) {
    // Instruções a serem repetidas
}
```

Estrutura **do-while**: semelhante ao while, mas garante que o bloco de código seja executado pelo menos uma vez.

```
do {
    // Instruções a serem repetidas
} while
```

Estrutura **for**: executa um bloco de código por um número específico de vezes, sendo ideal para quando se sabe antecipadamente o número de repetições.

```
for (inicialização; condição; incremento) {
    // Instruções a serem repetidas
}
```

Essas estruturas são empregadas quando as decisões a serem tomadas são baseadas em condições que sempre produzirão o mesmo resultado sob as mesmas circunstâncias. Elas são frequentemente implementadas em microprocessadores por meio de tabelas de pesquisa ou algoritmos com uma série de condições predeterminadas que não mudam dinamicamente durante a execução.

As estruturas de repetição são essenciais para operações que necessitam de verificação contínua ou processamento iterativo, como monitorar o estado de um sensor ou processar dados coletados. Elas são projetadas para maximizar a eficiência do programa, evitando a necessidade de duplicar código. As decisões determinísticas, por outro lado, são utilizadas em situações em que as respostas do microprocessador precisam ser previsíveis e consistentes, como em rotinas de inicialização ou algoritmos de controle que não requerem entrada externa para tomar decisões.

Imagine um sistema de controle de temperatura que mantém a temperatura de uma sala dentro de um limite predefinido. Uma estrutura de repetição pode ser utilizada para verificar continuamente a temperatura e ajustar o aquecimento ou a refrigeração conforme necessária.

```
int temperaturaAtual;
const int temperaturaDesejada = 22; // Temperatura em graus Celsius
while (1) {
    temperaturaAtual = lerSensorTemperatura();
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```

if (temperaturaAtual < temperaturaDesejada) {
    ligarAquecedor();
} else if (temperaturaAtual > temperaturaDesejada) {
    ligarRefrigerador();
}
// Aguarda um tempo antes de ler a temperatura novamente
esperar(1000);
}

```

Para uma decisão determinística, considere um microcontrolador inicializando um display. A sequência de inicialização é fixa e pode ser executada com uma série de instruções que não mudam.

```

void inicializarDisplay() {
    enviarComando(DISPLAY_LIGAR);
    enviarComando(DISPLAY_CONFIGURAR);
    // Mais comandos de inicialização
}
inicializarDisplay();

```

A compreensão e a aplicação correta dessas estruturas são vitais para o desenvolvimento de sistemas eficientes e confiáveis, capazes de realizar tarefas repetitivas e tomar decisões consistentes. Em nossa jornada de aprendizado, esses conceitos formam a base sobre a qual construímos programas complexos e responsivos que podem operar autonomamente em um vasto leque de aplicações em microprocessadores. A Tabela 1 resume as estruturas de decisão e repetição condicional, fundamentais para a programação de microprocessadores e para a criação de sistemas eficientes e confiáveis.

Tabela 1 | Resumo das estruturas de decisão e repetição condicionalParte superior do formulário

Tipo de Estrutura	Descrição	Exemplo de Uso
if	Avalia uma condição e, se verdadeira, executa um bloco de código.	if (temperatura > 30) { acionaVentilador(); }
else	Vinculado a um if, executa um bloco de código alternativo se a condição inicial for falsa.	if (temperatura > 30) { acionaVentilador(); } else { desligaVentilador(); }
else if	Usado para testar múltiplas condições em sequência.	if (condição1) { // código } else if (condição2) { // código } else { // código }

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

switch	Seleciona um bloco de código para executar com base no valor de uma variável.	switch (variável) { case valor1: // código break; case valor2: // código break; default: // código }
while	Executa um bloco de código enquanto a condição for verdadeira.	while (condição) { // Instruções a serem repetidas }
do-while	Garante que o bloco de código seja executado pelo menos uma vez, depois verifica a condição para repetição.	do { // Instruções a serem repetidas } while (condição);
for	Executa um bloco de código por um número específico de vezes.	for (inicialização; condição; incremento) { // Instruções a serem repetidas }

Fonte: elaborada pelo autor.

Vamos Exercitar?

Programando a Automação de um Jardim Comunitário

Agora continuamos a evoluir o projeto de Ana Beatriz e Carlos Eduardo, aperfeiçoando o sistema de irrigação de um jardim comunitário. As estruturas condicionais e de repetição são elementos centrais, já introduzidos anteriormente, mas agora, com um entendimento mais profundo e aplicado diretamente ao controle de irrigação com base nos dados dos sensores de umidade.

```
#include <stdio.h>
// Suponha que estas funções interagem com o hardware real do jardim comunitário
int lerSensorUmidade();
int previsaoTempoChuva();
void abrirValvula();
void fecharValvula();
void registrarLog(int umidade, int acao);
int main() {
    const int umidadeldeal = 500; // Ana ajustou o valor ideal de umidade
    int umidadeAtual, necessidadeDeIrrigacao;
    // Estrutura de repetição 'while' que mantém o sistema em constante monitoramento
    while (1) {
        umidadeAtual = lerSensorUmidade(); // Leitura da umidade atual pelo sensor
        // Operação lógica para verificar se é necessário irrigar
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```

necessidadeDelrrigacao = (umidadeAtual < umidadeldeal) && !previsaoTempoChuva();
// Estrutura condicional 'if' que toma a decisão sobre a irrigação
if (necessidadeDelrrigacao) {
    abrirValvula(); // Comando para iniciar a irrigação
    sleep(10); // Temporizador para manter a irrigação por um tempo definido
    fecharValvula(); // Comando para terminar a irrigação
    registrarLog(umidadeAtual, 1); // Registro da ação de irrigação
} else {
    // Registro de que nenhuma ação foi necessária
    registrarLog(umidadeAtual, 0);
}
// Intervalo entre os ciclos de irrigação
sleep(3600); // Ana estipulou um período de espera antes da próxima verificação
}
return 0;
}

```

Neste trecho de código, temos a seguinte mecânica de decisão e repetição:

Estrutura de repetição while: o loop infinito (denotado por while (1)) é o coração do monitoramento contínuo. Ele representa uma estrutura de repetição que permite ao sistema checar periodicamente o estado do jardim sem interrupção.

Estruturas condicionais if e else: dentro do loop, a condição if verifica se a umidade está abaixo do ideal e não há previsão de chuva. Se ambas as condições forem verdadeiras, a válvula de irrigação é aberta, atuando no mundo real, e um registro é criado. Se não for necessária a irrigação, apenas o registro é atualizado com a ação correspondente.

Operações lógicas: a variável necessidadeDelrrigacao utiliza um AND lógico (&&) combinando duas condições: se a umidade atual é menor que a ideal e se não há previsão de chuva. Isso demonstra como as operações lógicas são essenciais para criar condições complexas e precisas.

Função sleep: usada para criar uma pausa na execução do programa, permitindo que o sistema tenha um intervalo definido entre as ações de irrigação e as verificações de umidade, crucial para não sobrecarregar o sistema e simular o comportamento natural do jardim.

Esta seção de exercícios nos permite ver como estruturas condicionais e de repetição funcionam em uníssono para formar a lógica de controle em sistemas automatizados. Além disso, vemos, como ajustes nos condicionais (introduzindo mais variáveis e condições) e nas repetições (alterando a frequência e condições de loop) podem refinar a eficiência e a eficácia do sistema de irrigação.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Saiba mais

Aprofunde seus conhecimentos em programação de microcontroladores com o estudo detalhado da linguagem C, explorando sua aplicação direta no desenvolvimento de sistemas embarcados. Este material complementar oferece uma visão abrangente das técnicas de programação específicas para microcontroladores, unindo teoria e prática. Explore desde a sintaxe da linguagem até o manejo efetivo de hardware, e avance em sua capacidade de criar soluções tecnológicas inovadoras.

Para expandir seu aprendizado na [linguagem C](#) aplicada a microcontroladores, acesse o material completo indicado, das páginas 67 a 77. Ele será um guia valioso para transformar conhecimento em habilidade prática. Boa leitura e exploração!

Referências

- CRUZ, E. *et al.* **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. *et al.* **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.
- TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. edição. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 3

Funções e Recursividade

Funções e Recursividade

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Descubra como as funções em programação de microprocessadores são o alicerce para sistemas inteligentes e automação. Nesta videoaula, vamos mergulhar na prática, transformando teoria em habilidades profissionais aplicáveis. Junte-se a nós na arte de codificar soluções eficientes e inovadoras. Está preparado para impulsionar sua carreira com código que faz a diferença?

Ponto de Partida

À medida que a inovação tecnológica avança, os microprocessadores emergem como catalisadores de nossa interação diária com o mundo digital, agindo como o cérebro de inúmeros dispositivos que permeiam nosso cotidiano. Dominar a programação destas peças fundamentais é uma habilidade essencial para os entusiastas da tecnologia que desejam deixar sua marca no mundo.

Dentro deste panorama tecnológico, encontramos Ana Beatriz e Carlos Eduardo no comando de um projeto instigante: automatizar o monitoramento de um jardim comunitário. Com a analítica aguçada de Ana e a inventividade técnica de Carlos, eles personificam a integração de conhecimentos multidisciplinares na criação de um sistema inteligente de irrigação. Sua missão é programar um microprocessador que não apenas regula a umidade do solo, mas também responde dinamicamente às variações ambientais, garantindo a vitalidade do jardim.

Eles estão diante de um código que interage com o mundo real: lê umidade, prevê chuvas, controla fluxos de água e registra dados. No entanto, a pergunta que paira é: como podem estruturar esse código utilizando funções para otimizar o desempenho e facilitar futuras atualizações? Aqui reside nossa problematização, no ponto em que a teoria encontra a prática e a abstração cede espaço à realidade palpável da programação aplicada.

Este cenário lança luz na importância dos algoritmos e das linguagens de programação, e destaca a habilidade de Ana e Carlos de enxergar além do problema imediato, antevendo o impacto a longo prazo de suas soluções tecnológicas. Com cada função programada, eles tecem o futuro de suas carreiras, demonstrando que o entendimento profundo da programação pode desencadear mudanças significativas em nossa sociedade.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Junte-se a Ana e Carlos nesta jornada educativa, e você tanto aprenderá os fundamentos da programação de microprocessadores quanto se habilitará a transpor esses conhecimentos para a prática, abrindo portas para inovações que podem redefinir nosso futuro. Vamos decifrar juntos a linguagem dos microprocessadores e dominar a arte de animar as máquinas, traduzindo código em crescimento, inovação e, finalmente, em um impacto duradouro.

Vamos Começar!

Funções

Na jornada de programação de microprocessadores, as funções surgem como ferramentas poderosas que aumentam a legibilidade e a eficiência do código. Elas são usadas para executar tarefas específicas e podem ser reutilizadas ao longo do programa, promovendo um desenvolvimento mais estruturado.

Uma função é uma coleção de comandos que executa uma parte do programa. Ela é definida por um nome e pode receber dados externos chamados argumentos. Uma função pode retornar um resultado, facilitando a execução de operações comuns.

Exemplo 1 – Função de soma:

Imagine que frequentemente precisamos somar dois números em nosso programa para um microprocessador. Podemos criar uma função para isso:

```
int soma(int a, int b) {  
    return a + b;  
}
```

Neste exemplo, “soma” é uma função que recebe dois argumentos “int a” e “int b” e retorna a soma deles. Para usar a função, chamamos “soma(3, 4)”, o que retornará “7”.

Exemplo 2 – Função de controle de LED:

Em um microcontrolador, é comum termos que controlar um LED, seja para indicar status ou para alertas. Podemos criar uma função para ligar ou desligar um LED:

```
void controlaLED(int pin, bool estado) {  
    digitalWrite(pin, estado);  
}
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Aqui, “controlaLED” é uma função que não retorna valor algum (“void”). Ela aceita um número de pino “pin” e um valor booleano “estado” que determina se o LED deve estar ligado (“HIGH”) ou desligado (“LOW”). Ao chamar “controlaLED(13, HIGH)”, o LED conectado ao pino 13 será ligado.

Exemplo 3 – Função de leitura de temperatura com filtro:

Suponha que estamos trabalhando com um microprocessador conectado a um sensor de temperatura. Queremos ler a temperatura, mas como os sensores podem ter leituras ruidosas, vamos implementar uma função que faz várias leituras e retorna a média das leituras que estão dentro de um intervalo aceitável (filtro de outliers):

```
float leituraTemperaturaFiltrada(int pinSensor, int numLeituras, float limiteSuperior, float limiteInferior) {
    float somaTemperaturas = 0.0;
    int contagemValida = 0;
    for(int i = 0; i < numLeituras; i++) {
        float leitura = analogRead(pinSensor) * (5.0 / 1023.0) * 100.0; // Conversão da leitura
        analógica para temperatura
        if(leitura >= limiteInferior && leitura <= limiteSuperior) {
            somaTemperaturas += leitura;
            contagemValida++;
        }
    }
    if(contagemValida == 0) {
        return -1; // Retorna -1 se todas as leituras estiverem fora dos limites, indicando erro
    }
    return somaTemperaturas / contagemValida; // Retorna a média das temperaturas válidas
}
```

Neste exemplo complexo, a função leituraTemperaturaFiltrada realiza numLeituras do pinSensor e filtra os valores lendo apenas aqueles que estão dentro de um intervalo definido por limiteInferior e limiteSuperior. As leituras válidas são somadas e a média é calculada, resultando em uma leitura de temperatura mais estável e confiável.

Conceitos-chave a explorar:

Declaração e chamada: como criamos a estrutura de uma função e a invocamos no programa principal.

Parâmetros e argumentos: a diferença entre a definição de variáveis na função e os dados que passamos para ela.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Escopo de variáveis: como as funções interagem com variáveis globais e locais do programa.

Valor de retorno: o processo de retorno de dados para quem chamou a função, se necessário.

Com esses exemplos, começamos a entender como as funções operam e o impacto positivo que têm na organização e modularidade do nosso código. As funções nos ajudam a dividir problemas complexos em partes menores e mais gerenciáveis, facilitando a programação de microprocessadores.

Esses exemplos são comuns na programação de microcontroladores como o Arduino, por exemplo, e devem ajudar a esclarecer o conceito e a prática do uso de funções para quem está aprendendo.

Siga em Frente...

Aprofundando em Funções

Após termos iniciado nossa compreensão sobre funções, é hora de aprofundar nosso conhecimento em dois conceitos avançados que são essenciais para a programação eficaz de microprocessadores: escopo e passagem de parâmetros, e a poderosa técnica da recursividade.

Escopo e passagem de parâmetros

Escopo se refere à visibilidade de variáveis dentro de diferentes partes do nosso programa. Em termos de funções, as variáveis podem ser locais (visíveis apenas dentro da função) ou globais (visíveis em todo o programa). A passagem de parâmetros pode ser feita de duas maneiras: por valor (onde uma cópia do valor é feita) ou por referência (onde a própria variável é passada e pode ser modificada pela função).

Exemplo de escopo e passagem por valor:

```
int globalVar = 5;
void minhaFuncao(int parametroLocal) {
    int localVar = 10;
    parametroLocal += localVar; // Altera apenas a cópia local do parâmetro
}
void setup() {
    minhaFuncao(globalVar);
    // globalVar permanece como 5, pois a função altera apenas a cópia local
}
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Neste exemplo, globalVar é uma variável global acessível por toda a aplicação, enquanto localVar e parametroLocal são locais para minhaFuncao. A passagem por valor significa que as alterações feitas em parametroLocal dentro de minhaFuncao não afetam globalVar.

Exemplo de escopo e passagem por referência:

A passagem por referência permite que funções modifiquem diretamente as variáveis passadas como argumentos, o que pode ser mais eficiente em termos de memória, especialmente com estruturas de dados maiores.

```
void alteraPorReferencia(int &refParametro) {
    refParametro = 20; // Modifica a variável original
}
void setup() {
    int valor = 10;
    alteraPorReferencia(valor);
    // valor agora é 20, pois a função alterou a variável original
}
```

Neste exemplo, a variável valor é passada por referência para a função alteraPorReferencia usando o operador &. Isso significa que qualquer mudança feita em refParametro afeta diretamente “valor” no escopo chamador.

Recursividade

Recursividade é uma técnica em que uma função chama a si mesma para resolver um problema. É essencial que cada chamada recursiva progride em direção a uma condição de parada para evitar uma recursão infinita.

Exemplo de recursividade – fatorial de um número:

```
int factorial(int n) {
    if(n <= 1) return 1; // Condição de parada
    return n * factorial(n - 1); // Chamada recursiva
}
void setup() {
    int resultado = factorial(5);
    // resultado recebe o valor 120, que é 5*4*3*2*1
}
```

A função factorial utiliza recursividade para calcular o fatorial de um número. Ela chama a si mesma com um valor decrescente até atingir a condição de parada, que é quando n é menor ou

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

igual a 1.

Exemplo de recursividade – sequência de Fibonacci:

Recursividade não é apenas para cálculos matemáticos diretos; ela também pode ser usada para criar sequências complexas de forma elegante.

```
int fibonacci(int n) {  
    if(n == 0) return 0; // Condição de parada para n = 0  
    if(n == 1) return 1; // Condição de parada para n = 1  
    return fibonacci(n - 1) + fibonacci(n - 2); // Chamada recursiva  
}  
void setup() {  
    int resultado = fibonacci(5);  
    // resultado é 5, a soma dos dois termos anteriores da sequência (3 e 2)  
}
```

A sequência de Fibonacci é um exemplo clássico em que cada número é a soma dos dois anteriores. As chamadas recursivas continuam até que as condições de parada sejam satisfeitas, que são os casos base nos quais n é 0 ou 1.

Dominar o escopo de variáveis nos ajuda a evitar efeitos colaterais indesejados, enquanto entender a passagem de parâmetros é crucial para manipular dados de maneira eficiente. A recursividade, por sua vez, oferece uma maneira elegante de solucionar certos problemas, mas deve ser usada com cuidado para evitar chamadas infinitas e esgotamento de memória.

Ao aplicar esses conceitos avançados, estamos equipados para seguir em frente em nossa jornada de programação de microprocessadores e para enfrentar desafios de programação mais sofisticados com confiança e criatividade.

Vamos Exercitar?

Uso de Funções

No "Ponto de Partida", exploramos a problematização enfrentada por Ana Beatriz e Carlos Eduardo: a necessidade de programar um microprocessador para otimizar o sistema de irrigação de um jardim comunitário, utilizando a potência das funções para garantir eficácia e adaptabilidade ao código.

Demonstração da Resolução

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Para resolver este desafio, nossos guias adotaram funções que segmentaram o problema em partes menores e mais gerenciáveis. Eles criaram funções específicas para ler a umidade, prever chuva, controlar a irrigação e registrar as ações, cada uma encapsulando uma tarefa específica. Com isso, o código se tornou mais claro e manutenível.

- 1. Função de leitura de umidade:** *lerSensorUmidade()* captura dados do sensor e os retorna para outras partes do programa.
- 2. Função de previsão do tempo:** *previsaoTempoChuva()* utiliza algoritmos de previsão para determinar a probabilidade de chuva, evitando irrigação desnecessária.
- 3. Função de controle de válvula:** *abrirValvula()* e *fecharValvula()* executam as ações físicas baseadas nas necessidades de irrigação.
- 4. Função de registro:** *registrarLog(umidade, acao)* armazena informações cruciais para análises futuras.

Cada função foi criada levando em conta seu escopo e a maneira como os parâmetros seriam passados, decidindo entre passagem por valor ou por referência conforme a necessidade. A recursividade foi empregada na função de previsão do tempo, permitindo que o sistema reavaliasse as condições climáticas em intervalos regulares.

Agora é sua vez de exercitar a mente:

- Como você melhoraria o código de Ana e Carlos para lidar com variações mais extremas de clima, como uma onda de calor ou uma frente fria?
- Poderia a recursividade ser utilizada para otimizar ainda mais o sistema de irrigação?
- Existem outras funções que poderiam ser incorporadas para aumentar a eficiência ou a funcionalidade do sistema?

Considerando o papel vital que as funções desempenham na programação de microprocessadores, pense em como você aplicaria esses conceitos em outro contexto, como o desenvolvimento de um sistema de segurança residencial automatizado. Quais funções você definiria e como elas interagiriam entre si para garantir a segurança e a comodidade dos usuários?

Saiba mais

Aprofunde seus conhecimentos em programação de microcontroladores com o estudo detalhado da linguagem C, explorando sua aplicação direta no desenvolvimento de sistemas embarcados. Este material complementar oferece uma visão abrangente das técnicas de

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

programação específicas para microcontroladores, unindo teoria e prática. Explore desde a sintaxe da linguagem até o manejo efetivo de hardware, e avance em sua capacidade de criar soluções tecnológicas inovadoras.

Para expandir seu aprendizado na [linguagem C](#) aplicada a microcontroladores, acesse o material completo indicado das páginas 88 a 97. Ele será um guia

Referências

- CRUZ, E. *et al.* **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. *et al.* **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.
- TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. edição. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 4

Estruturas de Dados

Estruturas de Dados



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Adentre o universo da programação de microprocessadores e descubra como estruturas de dados fundamentais, como listas, pilhas e filas, são vitais para desenvolver sistemas automatizados inteligentes. Esta videoaula é um convite para conectar teoria e prática,

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

equipando você com habilidades técnicas essenciais para o sucesso profissional. Venha aprender, aplicar e inovar!

Ponto de Partida

Em um mundo que avança rapidamente na automação, Ana Beatriz e Carlos Eduardo embarcam em um projeto prático e inovador: desenvolver um sistema automatizado para o sistema de irrigação de um jardim comunitário. Utilizando um microprocessador, eles pretendem construir um sistema que responda de forma inteligente aos dados coletados por sensores de umidade, temperatura e luminosidade.

Neste processo, a utilização eficiente de listas, pilhas e filas se faz essencial. As listas organizarão as leituras dos sensores, as pilhas auxiliarão no controle das ações do sistema, como o acionamento de válvulas em sequência, e as filas garantirão que todos os eventos sejam processados na ordem correta.

Problematizamos então: como essas estruturas de dados podem ser implementadas para maximizar a eficiência e confiabilidade do sistema de irrigação? Como elas influenciam a tomada de decisão automatizada e o registro de atividades?

Juntos, vamos construir este algoritmo passo a passo, garantindo que cada decisão tomada e cada ação executada seja não apenas adequada, mas otimizada para as necessidades do jardim. Prepare-se para mergulhar na lógica de programação que traz vida à tecnologia, transformando um jardim comunitário em um modelo de sustentabilidade e inovação.

Vamos Começar!

Explorando Listas na Programação de Microprocessadores

Na arte da programação de microprocessadores, entender as estruturas de dados é fundamental. Elas são os ossos do corpo de nossos programas, dando forma e sustentação ao modo como armazenamos e manipulamos dados. Assim, listas, pilhas e filas são estruturas de dados lineares, cada uma com suas características e usos específicos.

Nesta seção, vamos nos concentrar nas listas. Pense em uma lista como uma coleção ordenada de itens, em que cada item tem uma posição. Em um microprocessador, isso poderia ser uma lista de temperaturas a serem monitoradas, ou uma sequência de comandos a serem executados.

Listas na Programação

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Flexibilidade: diferentemente dos arrays, listas são dinâmicas; podemos adicionar ou remover itens sem nos preocuparmos com o tamanho fixo.

Acesso sequencial: em listas, acessamos os elementos sequencialmente. Isso é importante em microprocessadores, nos quais o tempo e a ordem de operações são críticos.

Estrutura de dados versátil: listas podem ser usadas para criar outras estruturas, como pilhas e filas, com operações de inserção e remoção em diferentes extremidades.

Exemplo prático – lista de tarefas

Considere um sistema de gerenciamento de tarefas em um microcontrolador. Poderíamos ter uma lista contendo diferentes operações, cada uma associada a uma prioridade:

```
struct Task {
    int priority;
    void (*execute)(void);
};

// ... inicialização da lista de tarefas ...
void addTask(struct Task newTask) {
    // código para adicionar uma nova tarefa na lista
}
```

Cada tarefa na lista tem uma função associada que o microprocessador pode executar quando necessário, tornando nosso programa modular e fácil de expandir.

Exemplo prático – gerenciamento de alarmes

Imagine um sistema embarcado responsável por monitorar sensores de segurança. Esse sistema pode usar uma lista para armazenar e gerenciar alarmes ativos:

```
typedef struct Alarm {
    int id;
    char* description;
    bool isActive;
} Alarm;

Alarm alarmList[MAX_ALARMS]; // Array estático por simplicidade, mas poderia ser dinâmico
void activateAlarm(int id, char* description) {
    // Encontra um espaço vazio na lista para adicionar um novo alarme
    for (int i = 0; i < MAX_ALARMS; i++) {
        if (!alarmList[i].isActive) {
            alarmList[i].id = id;
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```
    alarmList[i].description = description;
    alarmList[i].isActive = true;
    break;
}
}
```

Neste exemplo, a lista “alarmList” é utilizada para manter um registro dos alarmes. Quando um sensor detecta uma condição que deve ativar um alarme, a função “activateAlarm” é chamada para adicionar esse alarme à lista.

Ao compreender as listas, você ganha uma ferramenta poderosa para a programação de microprocessadores, capaz de implementar estruturas de decisão e repetição de forma eficaz. Vamos aprofundar nosso conhecimento e ver como as listas se encaixam no contexto mais amplo das estruturas de dados.

Siga em Frente...

Pilhas e Filas

Avançando em nossa exploração das estruturas de dados lineares, chegamos às pilhas e filas. Enquanto as listas são versáteis, pilhas e filas oferecem controle especializado sobre a adição e remoção de elementos, cada uma com sua particularidade e contexto de uso em sistemas de microprocessadores.

Pilhas – o princípio LIFO (*last in, first out*)

Pilhas são como uma torre de pratos; você coloca (empilha) um novo prato no topo e o retira (desempilha) também do topo. Isso caracteriza o princípio LIFO.

Exemplo prático – pilha de chamadas de função

```
void funcaoA() {
    // Quando funcaoA é chamada, é colocada no topo da pilha de chamadas
    funcaoB();
    // Após funcaoB terminar, funcaoA é retomada e depois retirada da pilha
}
void funcaoB() {
    // funcaoB agora está no topo da pilha até que sua execução termine
}
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Exemplo adicional – desfazendo comandos em um editor de texto

Considere um microcontrolador operando um pequeno display de texto com funcionalidade de desfazer:

```
#define MAX_COMMANDS 10
char* commandStack[MAX_COMMANDS];
int stackPointer = -1;
void executeCommand(char* command) {
    // Executa o comando e o adiciona à pilha de desfazer
    if (stackPointer < MAX_COMMANDS - 1) {
        stackPointer++;
        commandStack[stackPointer] = command;
    }
    // Executa o comando...
}
void undoCommand() {
    // Desfaz o último comando e o remove da pilha
    if (stackPointer >= 0) {
        // Reverte o comando...
        stackPointer--;
    }
}
```

Neste exemplo, cada comando executado é adicionado à pilha. Se o usuário escolhe desfazer a última ação, o sistema usa a pilha para reverter o comando e remover da história.

Em microprocessadores, pilhas são essenciais para gerenciar chamadas de funções. Cada nova chamada é empilhada, e à medida que as funções são concluídas, elas são desempilhadas.

Filas – o princípio FIFO (*first in, first out*)

Filas funcionam como uma fila de pessoas; a primeira pessoa que chega é a primeira a ser atendida. Isso reflete o princípio FIFO.

Exemplo prático – fila de processamento de eventos:

```
struct Event {
    int type;
    void (*handler)(void);
};
struct Event eventQueue[MAX_EVENTS];
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```

void enqueueEvent(struct Event newEvent) {
    // Adiciona um evento no final da fila
}
void processEvents() {
    // Processa o evento no início da fila
}

```

Exemplo adicional – agendamento de impressão

Imagine um sistema de gerenciamento de impressora no qual múltiplas tarefas de impressão são recebidas:

```

typedef struct {
    int documentId;
    char* documentPath;
} PrintJob;
PrintJob printQueue[MAX_JOBS];
int front = 0;
int rear = -1;
void addPrintJob(PrintJob newJob) {
    // Adiciona um novo trabalho de impressão no final da fila
    if (rear < MAX_JOBS - 1) {
        rear++;
        printQueue[rear] = newJob;
    }
}
void processPrintJobs() {
    // Processa o primeiro trabalho de impressão da fila
    if (front <= rear) {
        PrintJob jobToPrint = printQueue[front];
        printDocument(jobToPrint.documentPath);
        front++;
    }
}

```

Neste cenário, os trabalhos de impressão são adicionados à fila e processados um por um, garantindo que sejam impressos na ordem em que foram recebidos. Para microprocessadores que lidam com múltiplos eventos ou interrupções, filas permitem que cada evento seja processado na ordem de chegada, assegurando uma execução organizada e previsível.

Convidamos você a refletir respeito das situações em que pilhas e filas podem ser aplicadas:

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Considere um sistema de controle de tráfego. Como pilhas e filas podem ajudar na gestão dos sinais e na priorização de caminhos?

Imagine o gerenciamento de memória em um microprocessador. Onde pilhas seriam mais apropriadas do que filas, e vice-versa?

Ao explorar esses conceitos, percebemos a importância de escolher a estrutura de dados certa para cada problema. Aprofunde-se na prática, identifique onde pilhas e filas se encaixam em suas aplicações e desbloqueie um novo nível de eficiência na sua programação.

Vamos Exercitar?

Aplicando Estruturas de Dados no Sistema de Irrigação

No "Ponto de Partida", enfrentamos o desafio de Ana Beatriz e Carlos Eduardo: programar um sistema de irrigação automatizado usando microprocessadores, fundamentado em dados fornecidos por sensores ambientais. A problematização centrou-se no uso eficiente de estruturas de dados – listas, pilhas e filas – para gerenciar e automatizar o sistema de irrigação de forma eficiente e inteligente.

Para alcançar a solução:

Listas foram utilizadas para armazenar leituras de sensores e controlar a sequência de comandos de irrigação, permitindo a adição e remoção dinâmica de itens.

Pilhas ajudaram a gerenciar o estado do sistema, facilitando a execução e o cancelamento de comandos com base no princípio LIFO, ideal para a funcionalidade de desfazer ações.

Filas garantiram que os eventos coletados pelos sensores fossem processados em ordem, sem perder a prioridade dos eventos mais antigos, crucial para manter a ordem de irrigação de acordo com a programação.

Agora é sua vez de aplicar estes conceitos. Imagine que Ana e Carlos querem adicionar uma nova funcionalidade: um sistema de alerta para quando a umidade do solo estiver crítica. Como você usaria pilhas ou filas para implementar essa funcionalidade? E como as listas poderiam ajudar na organização e execução desses alertas?

Além disso, como você modificaria o sistema para que ele lide com múltiplas zonas de irrigação, cada uma com diferentes requisitos de umidade e horários de irrigação? Pense em como as estruturas de dados discutidas podem ser utilizadas para expandir o sistema de forma modular e escalável.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Saiba mais

Aprofunde seu entendimento em estruturas de dados essenciais em programação de microprocessadores com uma imersão nos capítulos 10, 11 e 12 do documento "Estruturas de Dados". Estes capítulos são dedicados às listas, pilhas e filas, respectivamente, e apresentam uma análise minuciosa de cada uma dessas estruturas, seus algoritmos característicos e como elas são aplicadas na programação de sistemas embarcados.

Além de aprofundar o conhecimento teórico, o material oferece exemplos práticos que ilustram a implementação dessas estruturas em linguagem C, facilitando a transição do conhecimento para a aplicação real. Entender esses conceitos é crucial para quem deseja excelência na programação de microcontroladores, permitindo que você desenvolva sistemas mais eficientes e confiáveis.

Para explorar este conteúdo detalhado de listas, pilhas e filas, visite o site indicado a seguir e concentre-se nos capítulos 10 a 12 para um estudo focado e aprofundado.

[Estruturas de Dados](#) – Universidade Federal de Santa Maria.

Referências

- CRUZ, E. *et al.* **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. *et al.* **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.
- TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. edição. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta videoaula de encerramento, unimos todos os conceitos essenciais de programação em C para microprocessadores. Da tomada de decisões condicionais à execução de tarefas repetitivas e o domínio de funções e recursividade, você está prestes a consolidar seu conhecimento em práticas reais de automação. Prepare-se para escrever programas robustos e resolver desafios específicos com o poder do código. Junte-se a nós e solidifique sua jornada no mundo da programação de microprocessadores.

Ponto de Chegada

Durante as aulas, você mergulhou nos princípios da programação de microprocessadores, explorando estruturas de decisão e repetição, além de se aprofundar em funções e recursividade. Esses conhecimentos são cruciais para desenvolver a competência de escrever programas eficientes que controlam dispositivos e respondem a uma vasta gama de entradas, uma habilidade indispensável na era digital.

A competência que você agora possui abre portas para a criação de soluções inovadoras em automação e sistemas embarcados. Compreende a importância das estruturas de dados na organização e execução de tarefas programáticas. Identifica as aplicações práticas dos conceitos aprendidos, como o uso de laços para otimizar operações e a implementação de funções para modularização do código. Conhece também as estratégias para aplicar algoritmos que tomam decisões autônomas com base em informações sensoriais em tempo real.

Ao progredir por esta Unidade, você não apenas seguiu instruções; você projetou capacidade de decisão em tecnologia. Com isso, você está agora capacitado para enfrentar os desafios reais do mundo da programação e automação, tornando-se um elo fundamental entre o código e suas múltiplas aplicações no mundo moderno.

Além disso, a competência que adquiriu vai além do domínio técnico; ela incorpora um pensamento crítico sobre como as soluções tecnológicas impactam nosso ambiente e a sociedade. Analise como as estratégias de programação de microprocessadores podem ser utilizadas para promover a sustentabilidade e melhorar a qualidade de vida. Avalie como otimizar o consumo de energia e recursos através de algoritmos inteligentes, e contribua para projetos de tecnologia que respeitam princípios ecológicos. A habilidade de aplicar esses conceitos em contextos que exigem eficiência e inovação responsável é o que define um especialista na área.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

de sistemas digitais e microprocessados. Continue a construir seu futuro com a confiança de que você pode não apenas interagir com a tecnologia, mas também moldá-la para um amanhã mais promissor.

É Hora de Praticar!

Imagine que você é um desenvolvedor de software trabalhando para uma empresa chamada **GreenTech Innovations**. A empresa está desenvolvendo um sistema de monitoramento e controle para estufas agrícolas. Este sistema deve ser capaz de monitorar continuamente a temperatura, umidade e níveis de CO₂ dentro da estufa, tomar decisões em tempo real e acionar sistemas de ventilação, aquecimento e irrigação conforme necessário para manter as condições ideais para o crescimento das plantas.

Personagens e empresas envolvidas

- **Desenvolvedor de software (você)**: responsável pela programação do sistema de controle da estufa.
- **Engenheiro de hardware (Miguel)**: cuida da integração dos sensores e atuadores com o microprocessador.
- **Gerente de projeto (Luciana)**: supervisiona o desenvolvimento do projeto e garante que as metas sejam cumpridas.
- **GreenTech Innovations**: empresa especializada em soluções tecnológicas para agricultura.

Problema Proposto

O desafio é desenvolver um sistema baseado em microprocessadores que deve ser capaz de:

- Monitorar continuamente os dados de sensores (temperatura, umidade e níveis de CO₂).
- Tomar decisões automáticas baseadas nos dados coletados.
- Acionar sistemas de ventilação, aquecimento e irrigação conforme necessário para manter as condições ideais dentro da estufa.

Questões Norteadoras

- Como garantir que o sistema de monitoramento e controle responda de forma rápida e eficiente às mudanças nas condições dentro da estufa?
- Quais estruturas de decisão e repetição são mais adequadas para essa aplicação?
- Como integrar diferentes tipos de sensores e atuadores para operar de forma coordenada e eficiente?

Olá estudante, chegamos ao encerramento da unidade!

Vamos realizar a experiência presencial que irá consolidar os conhecimentos adquiridos? É a oportunidade perfeita para aplicar, na prática, o que foi aprendido em sua disciplina. Vamos

Disciplina

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



transformar teoria em vivência e tornar esta etapa ainda mais significativa. Não perca essa chance única de colocar em prática o conhecimento adquirido.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Linguagens de programação para microprocessadores

LP	Abstração	Facilidade	Controle sobre o Hardware	Uso Comum
Assembly	Baixo	Difícil	Máximo	Sistemas embarcados, firmware
C	Médio	Moderada	Elevado	Sistemas operacionais, sistemas embarcados
C++	Alto	Moderada	Elevado	Aplicações de desempenho crítico, jogos
Python	Alto	Fácil	Baixo	Prototipagem, scripts de automação
Java	Alto	Moderada	Baixo	Aplicações empresariais, Android

Estruturas de decisão e repetição condicional

Tipo de Estrutura	Descrição	Exemplo de Uso
if	Avalia uma condição e, se verdadeira, executa um bloco de código.	if (temperatura > 30) { acionaVentilador(); }
else	Vinculado a um if, executa um bloco de código alternativo se a condição inicial for falsa.	if (temperatura > 30) { acionaVentilador(); } else { desligaVentilador(); }
else if	Usado para testar múltiplas condições em sequência.	if (condição1) { // código } else if (condição2) { // código } else { // código }
switch	Seleciona um bloco de código para executar com base no valor de uma variável.	switch (variável) { case valor1: // código break; case valor2: // código break; default: // código }
while	Executa um bloco de código enquanto a condição for verdadeira.	while (condição) { // Instruções a serem repetidas }
do-while	Garante que o bloco de código seja executado pelo menos uma vez, depois verifica a condição para repetição.	do { // Instruções a serem repetidas } while (condição);
for	Executa um bloco de código por um número específico de vezes.	for (inicialização; condição; incremento) { // Instruções a serem repetidas }

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Funções, recursividade e passagem de parâmetros

Característica	Funções	Recursividade	Escopo e Passagem de Parâmetros
Definição	Blocos de código reutilizáveis que executam tarefas específicas.	Uma função que chama a si mesma para resolver um problema. Resolver problemas complexos que podem ser divididos em casos menores semelhantes.	Determina a visibilidade das variáveis e como os dados são passados às funções.
Uso	Simplificar e estruturar o código para reuso e clareza.		Organizar o código e evitar efeitos colaterais indesejados ao manipular dados.
Exemplos	<code>int soma(int a, int b); void controlaLED(int pin, bool estado).</code>	<code>Fatorial: int factorial(int n); Fibonacci: int fibonacci(int n).</code>	Passagem por valor: <code>void minhaFuncao(int parametroLocal);</code> Passagem por referência: <code>void alteraPorReferencia(int &refParametro).</code>

Listas, pilhas e filas

Característica	Listas	Pilhas	Filas
Definição	Coleções de elementos ordenados acessíveis por posição.	Coleções de elementos com acesso LIFO (último a entrar, primeiro a sair).	Coleções de elementos com acesso FIFO (primeiro a entrar, primeiro a sair).
Uso	Armazenar dados com uma ordem sequencial.	Armazenar dados onde o último elemento adicionado é o primeiro a ser removido.	Armazenar dados onde os elementos são processados na ordem em que são adicionados.
Exemplos	Lista de tarefas, lista encadeada de números ou objetos.	Histórico de navegação, chamadas de função (stack trace).	Fila de impressão, fila de mensagens em sistemas de comunicação.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

SOUZA, Diogo B. da Costa; SANTOS, Sidney C. Bispo dos; MARTON, Italo L. de Alencar et al. **Sistemas digitais**. Porto Alegre: SER - SAGAH, 2018.

CRUZ, Eduardo; GAUDINO, Enzo; ADRIANO, Domingos et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Editora Alta Books, 2022.

HEXSEL, R. A. **Sistemas Digitais e Microprocessadores**. Departamento de Informática, Universidade Federal do Paraná, 2006.

IDOETA, Ivan Valeije; CAPUANO, Francisco Gabriel. **Elementos de eletrônica digital**. 42 ed. São Paulo: Érica, 2019.

TOCCI, R. J.; Widmer, N. S. **Sistemas Digitais - princípios e aplicações**. 11a edição. Rio de Janeiro: LTC - Livros técnicos e científicos, 2011.

Unidade 4

INTERFACES E MODELAGEM

Aula 1

Linguagem de Descrição de Hardware - VHDL/VHSIC

Linguagem de descrição de hardware – VHDL/VHSIC



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Explore o mundo do VHDL nesta videoaula completa, em que abordamos desde a sintaxe até a aplicação prática em design de circuitos digitais. Descubra como o VHDL facilita a modelagem e simulação de sistemas eletrônicos, preparando-o para desafios reais de engenharia. Participe e aprimore suas habilidades, preparando-se para inovações no campo do design eletrônico. Não perca esta oportunidade de aprofundar seus conhecimentos e aplicá-los profissionalmente!

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Ponto de Partida

Na nossa jornada pelo estudo do VHDL, uma linguagem fundamental para modelar e simular sistemas eletrônicos digitais, Ana Beatriz, uma estudante de engenharia de software, encontra-se às vésperas de um projeto de conclusão de curso que pode revolucionar seu entendimento de design de hardware. Ela decide explorar como o VHDL tem sido usado para superar desafios técnicos na criação de sistemas mais robustos e eficientes.

Durante seu projeto, Ana enfrenta uma dificuldade: como aplicar os conceitos teóricos do VHDL para melhorar a eficiência e a eficácia de sistemas reais? Para resolver esse enigma, Ana busca a orientação de Carlos Eduardo, um engenheiro experiente, para entender melhor a aplicação prática do VHDL. Juntos, eles se deparam com um desafio específico: otimizar um sistema de comunicação FPGA que apresenta problemas de sincronização e eficiência de dados. A questão central que se apresenta é: como podem utilizar o VHDL para modelar uma solução que resolva esses problemas técnicos e introduza melhorias substanciais no desempenho do sistema?

Este desafio serve como um convite a você, estudante, para mergulhar nas possibilidades que o VHDL oferece no campo do design de hardware. Como Ana e Carlos, você tem a oportunidade de aplicar o conhecimento teórico em cenários reais, enfrentando e superando obstáculos práticos com criatividade e inovação. Encorajamos você a abordar cada problema como uma chance de aprender e crescer, utilizando o VHDL não apenas como uma ferramenta de simulação, mas como um facilitador de soluções inovadoras em tecnologia.

Vamos Começar!

Conceitos Básicos de VHDL

O VHDL (VHSIC *Hardware Description Language*) é uma linguagem desenvolvida para descrever a estrutura e o comportamento de sistemas eletrônicos digitais. Diferentemente de uma linguagem de programação, o VHDL permite modelar especificações precisas de hardware, tendo sido criado inicialmente para suprir as necessidades do Departamento de Defesa dos Estados Unidos.

Surgido na década de 1980, o VHDL foi uma das primeiras linguagens de descrição de hardware a ser padronizada pelo IEEE. É relevante destacar a evolução dessa linguagem ao longo dos anos, enfatizando sua importância contínua no design de sistemas eletrônicos.

As entidades (*entities*) e as arquiteturas (*architectures*) são componentes fundamentais em qualquer descrição VHDL. As entidades servem como interfaces externas que definem entradas e saídas, enquanto as arquiteturas detalham o comportamento interno do sistema modelado.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Estrutura do VHDL

Ao iniciar qualquer projeto em VHDL, é necessário declarar as bibliotecas padrão, como IEEE, e utilizar pacotes como std_logic_1164. Estes pacotes fornecem tipos de dados essenciais e são necessários para a modelagem eficaz do hardware. Essas declarações ajudam a estabelecer um padrão de design, facilitando a interoperabilidade e a reutilização de código entre diferentes projetos e plataformas. Ao manter um conjunto comum de tipos e definições, garante-se que componentes e sistemas distintos possam interagir sem incompatibilidades.

A entidade é uma das estruturas fundamentais em VHDL. Ela define a interface de um componente do sistema, especificando suas entradas e saídas, conhecidas como portas (port). Por exemplo, em uma entidade, pode-se definir portas para sinais de entrada, saída ou bidirecionais, configurando como o componente se comunica com o restante do sistema ou com o ambiente externo. A declaração da entidade encapsula essas interfaces, facilitando a compreensão e o uso do componente em contextos mais amplos.

Estrutura da entidade:

```
ENTITY nome_do_circuito IS
  PORT(
    entrada1 : IN STD_LOGIC;
    saída1 : OUT STD_LOGIC
  );
END nome_do_circuito;
```

A arquitetura detalha como a entidade opera internamente, descrevendo a funcionalidade do componente com uma ou mais definições.

Estrutura da arquitetura:

```
ARCHITECTURE comportamento OF nome_do_circuito IS
BEGIN
  -- Definição da lógica
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

saída1 <= entrada1;

END comportamento;

A arquitetura permite o uso de diferentes abordagens de modelagem, como comportamental, que foca o comportamento do sistema; estrutural, que detalha a conexão entre subcomponentes; e de fluxo de dados, que descreve como os dados movem-se através do componente. A escolha da abordagem depende das necessidades específicas do design e do nível de abstração desejado, proporcionando flexibilidade e poder ao designer para expressar o funcionamento interno do hardware de forma clara e eficiente.

Para ilustrar o uso prático do VHDL, vamos examinar um exemplo de um circuito simples usando flip-flops e portas lógicas AND e NAND, representando o código VHDL nos três formatos: **comportamental, estrutural** e de **fluxo de dados**. Vamos usar um flip-flop do tipo D com um circuito de entrada que usa portas AND e NAND para demonstrar diferentes estilos de modelagem em VHDL.

Exemplo de código VHDL comportamental

Este exemplo ilustra um circuito simples de flip-flop do tipo D, no qual o estado do flip-flop muda de acordo com a entrada ‘d’ a cada borda de subida do clock (‘clk’), desde que o sinal de ‘reset’ não esteja ativo. Quando o ‘reset’ está ativo (alto), o flip-flop é resetado para ‘0’, independentemente das outras condições. Este estilo é chamado comportamental porque foca em descrever o comportamento do flip-flop em resposta às suas entradas de controle.

LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

ENTITY ff_circuit IS

PORT (

clk : IN STD_LOGIC;

reset : IN STD_LOGIC;

d : IN STD_LOGIC;

q : OUT STD_LOGIC

);

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

END ff_circuit;

ARCHITECTURE behavioral OF ff_circuit IS

BEGIN

PROCESS (clk, reset)

BEGIN

IF reset = '1' THEN

q <= '0';

ELSIF rising_edge(clk) THEN

q <= d;

END IF;

END PROCESS;

END behavioral;

Exemplo de código VHDL estrutural

Este estilo detalha a conexão entre subcomponentes (como portas lógicas e outros flip-flops). Aqui, definimos as portas AND e NAND e as usamos para criar um circuito que controla o sinal de entrada de um flip-flop D. Os sinais ‘input1’ e ‘input2’ são processados pelas portas lógicas AND e NAND. O resultado dessas operações é usado para determinar o sinal ‘d’, que por sua vez controla o estado do flip-flop do tipo D. Este estilo é chamado estrutural porque enfatiza a conexão física e a organização dos subcomponentes, em vez de descrever apenas o comportamento.

LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

ENTITY ff_circuit IS

POR T(

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```

clk  : IN STD_LOGIC;
reset : IN STD_LOGIC;
input1 : IN STD_LOGIC;
input2 : IN STD_LOGIC;
q    : OUT STD_LOGIC
);
END ff_circuit;

ARCHITECTURE structural OF ff_circuit IS

SIGNAL and_output, nand_output, d : STD_LOGIC;

BEGIN

and_gate: ENTITY work.and2 PORT MAP (input1, input2, and_output);
nand_gate: ENTITY work.nand2 PORT MAP (input1, input2, nand_output);
d <= and_output AND nand_output;

ff_d: PROCESS (clk, reset)
BEGIN

IF reset = '1' THEN
  q <= '0';
ELSIF rising_edge(clk) THEN
  q <= d;
END IF;
END PROCESS;

```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

END structural;

Exemplo de código VHDL de fluxo de dados

Este estilo foca a maneira como os dados se movem através do componente. Não descreve explicitamente a lógica interna, mas as operações realizadas nos sinais. O estilo de fluxo de dados é usado para expressar a funcionalidade do circuito, focando as transformações dos dados por meio das operações lógicas. O sinal '**d**' é determinado pela combinação das operações **and** e **nand** sobre '**input1**' e '**input2**'. Especificamente, '**d**' será alto apenas se ambas as entradas forem diferentes uma da outra. A saída '**q**' do flip-flop do tipo D é atualizada na borda de subida do clock '**clk**' se o '**reset**' não estiver ativo. Este estilo é útil para visualizar como os dados são manipulados e propagados dentro do circuito, facilitando a compreensão de operações complexas de processamento de sinal.

LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

ENTITY ff_circuit IS

PORT(

clk : IN STD_LOGIC;

reset : IN STD_LOGIC;

input1 : IN STD_LOGIC;

input2 : IN STD_LOGIC;

q : OUT STD_LOGIC

);

END ff_circuit;

ARCHITECTURE dataflow OF ff_circuit IS

SIGNAL d : STD_LOGIC;

BEGIN

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```
d <= (input1 AND input2) AND NOT (input1 AND input2);
```

```
PROCESS (clk, reset)
```

```
BEGIN
```

```
IF reset = '1' THEN
```

```
    q <= '0';
```

```
ELSIF rising_edge(clk) THEN
```

```
    q <= d;
```

```
END IF;
```

```
END PROCESS;
```

```
END dataflow;
```

Cada um desses exemplos serve para ilustrar diferentes abordagens na modelagem de hardware usando VHDL, refletindo a flexibilidade e o poder da linguagem para descrever sistemas digitais de forma eficiente e adaptada às necessidades específicas de cada projeto.

Siga em Frente...

Fundamentos de Sistemas Digitais

Após explorar os conceitos básicos e a estrutura do VHDL, é essencial compreender as regras sintáticas que governam a escrita de códigos eficazes e sem erros em VHDL. A sintaxe do VHDL, semelhante a muitas linguagens de programação, requer precisão e atenção aos detalhes para garantir que o hardware descrito funcione conforme esperado.

Regras de sintaxe e estilo

A sintaxe do VHDL é rigorosa e inclui algumas regras essenciais que facilitam a leitura e manutenção do código:

- VHDL é uma linguagem que não diferencia maiúsculas de minúsculas para identificadores e palavras-chave, mas é uma boa prática usar maiúsculas para palavras-chave e tipos

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

definidos pelo usuário para melhor legibilidade.

- Comentários em VHDL são iniciados com -- e continuam até o final da linha. Eles são essenciais para explicar a lógica e as decisões de design no código.
- Ponto e vírgula: cada declaração deve terminar com um ponto e vírgula, indicando o fim da instrução.

Uso de declarações condicionais e laços

Em VHDL, as estruturas de controle como condições IF-ELSE e laços FOR e WHILE são usadas dentro de processos para definir comportamentos complexos.

IF-ELSE:

IF condição THEN

-- *instruções se verdadeiro*

ELSE

-- *instruções se falso*

END IF;

FOR LOOP:

FOR i IN 0 TO 10 LOOP

-- *repetir até que i seja 10*

END LOOP;

O entendimento e a aplicação correta das regras sintáticas em VHDL são essenciais para:

- Evitar erros de sintaxe: códigos que não seguem as regras sintáticas corretamente não serão compilados, resultando em atrasos e potenciais falhas no projeto.
- Garantir comportamentos corretos: uma sintaxe correta assegura que o comportamento simulado e sintetizado do hardware será conforme o esperado, reduzindo o risco de falhas no design.
- Facilitar a manutenção e revisão de código: códigos bem estruturados e com sintaxe clara são mais fáceis de entender, revisar e manter por outros desenvolvedores ou em revisões futuras do projeto.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- Conhecer profundamente a sintaxe VHDL e aplicá-la corretamente facilita o desenvolvimento de sistemas digitais complexos, minimizando erros e melhorando a eficiência do processo de design eletrônico.

Implementação de arrays e sinais

O uso de arrays e sinais em VHDL é fundamental para manipular conjuntos de dados e representar as interconexões em um sistema digital. Arrays permitem agrupar elementos de dados semelhantes, facilitando a gestão de grandes volumes de informação, enquanto sinais são usados para modelar as conexões físicas entre componentes lógicos:

```
TYPE vetor IS ARRAY (0 TO 9) OF std_logic;
```

```
SIGNAL sinal_vetor : vetor;
```

Importância das bibliotecas e pacotes

Bibliotecas e pacotes em VHDL contêm definições e implementações que podem ser reutilizadas em vários projetos. Utilizar essas ferramentas não só economiza tempo, mas também assegura consistência e confiabilidade em designs de hardware. Por exemplo, o pacote IEEE.std_logic_1164 é amplamente usado para definir tipos de dados e operações em lógica multivalorada, essencial para modelar o comportamento de circuitos digitais:

```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;
```

O Quadro 1 tem a descrição dos pacotes mais comuns. Esses pacotes são fundamentais para trabalhar com diferentes aspectos de dados e operações em VHDL, ajudando a modelar comportamentos de hardware mais precisos e eficientes. Incorporar bibliotecas padrão e criar pacotes personalizados são práticas recomendadas para manter a modularidade e a escalabilidade do código VHDL.

Pacote	Descrição
ieee.std_logic_1164	Fornece definições para o tipo de dados std_logic e std_logic_vector, que são essenciais para modelar sinais digitais em VHDL.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

<code>ieee.numeric_std</code>	Oferece recursos para aritmética de números inteiros e ponto fixo, incluindo definições de tipos e operações aritméticas.
<code>ieee.std_logic_unsigned</code>	Fornece operações aritméticas para <code>std_logic_vector</code> como se fossem números sem sinal, permitindo cálculos mais intuitivos em certos contextos.
<code>ieee.std_logic_arith</code>	Semelhante ao <code>std_logic_unsigned</code> , mas adiciona suporte para operações aritméticas com sinais de <code>std_logic_vector</code> .
<code>ieee.std_logic_signed</code>	Permite que <code>std_logic_vector</code> seja tratado como números assinados, facilitando a realização de operações aritméticas com sinais.
<code>ieee.synopsys</code>	Inclui funções e procedimentos para facilitar a síntese de projetos, com foco em compatibilidade e otimização de desempenho.

Quadro 1 | Pacotes comuns em VHDL e suas aplicações. Fonte: elaborado pelo autor. Parte superior do formulário

Como uma linguagem de descrição de hardware, o VHDL permite aos engenheiros modelar e simular precisamente o comportamento de sistemas digitais antes da fabricação física. Isso é especialmente vital em aplicações que envolvem FPGAs (*Field-Programmable Gate Arrays*) e ASICs (*Application-Specific Integrated Circuits*), em que a precisão e eficácia do design podem significativamente impactar o desempenho e a funcionalidade do produto final. O entendimento e a aplicação competente dos conceitos básicos de VHDL são fundamentais para qualquer engenheiro eletrônico ou de hardware, fornecendo as ferramentas necessárias para inovar e otimizar no campo do design eletrônico.

Vamos Exercitar?

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Aplicação do VHDL

No início desta aula, abordamos a complexidade do VHDL e como ele permite modelar sistemas eletrônicos com precisão antes de sua fabricação física. Ana Beatriz, em colaboração com Carlos Eduardo, enfrentou o desafio de aplicar o VHDL para otimizar um sistema de comunicação FPGA que sofria com problemas de sincronização e eficiência de dados.

A resolução dessa problemática começou com Ana e Carlos revisando os conceitos fundamentais do VHDL, especialmente as estruturas de controle e a definição de entidades e arquiteturas, utilizando a linguagem VHDL:

- Modelagem da entidade:** Ana e Carlos definiram uma entidade específica para o módulo de comunicação, detalhando todas as entradas e saídas necessárias. Eles usaram a sintaxe VHDL para criar portas claras e bem definidas, o que facilitou a integração com outros módulos do sistema.
- Desenvolvimento da arquitetura:** na arquitetura, eles implementaram um novo algoritmo de sincronização usando a estrutura de controle “IF-ELSE” e loops “FOR”, que permitiram um controle mais refinado sobre o fluxo de dados. O algoritmo foi projetado para identificar e corrigir desfasamentos em tempo real, aumentando significativamente a eficiência do sistema.
- Simulação e testes:** com o VHDL, foram realizadas simulações detalhadas para testar a nova arquitetura. Essas simulações ajudaram a identificar falhas antecipadamente e ajustar o design antes da implementação final.
- Implementação e validação:** após ajustes baseados nos resultados dos testes, a solução foi implementada no sistema de comunicação FPGA. O impacto foi imediatamente notável, com melhorias na sincronização e na eficiência dos dados.

Esse processo resolveu os problemas imediatos e abriu caminhos para futuras inovações no design de sistemas digitais. Encorajamos você a pensar em como técnicas semelhantes podem ser aplicadas em outros contextos ou problemas no campo da eletrônica digital. Quais outros desafios tecnológicos poderiam ser abordados com uma abordagem semelhante usando VHDL? Como você adaptaria o processo de Ana e Carlos para atender a necessidades específicas de outros projetos?

Este exercício de aplicação prática reforça o entendimento do VHDL e estimula a inovação contínua, seguindo o exemplo de Ana e Carlos ao enfrentar e superar desafios técnicos com soluções criativas e eficazes.

Saiba mais

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Para complementar seu estudo da linguagem VHDL e suas aplicações no design de circuitos digitais, recomendamos a seguinte apostila, que oferece uma abordagem prática e detalhada:

[**"Apostila com Códigos de Programas Demonstrativos usando a linguagem VHDL para Circuitos Digitais"**](#), por Alexandre Santos de la Vega (Universidade Federal Fluminense).

Esta apostila contém uma série de exemplos práticos e códigos demonstrativos em VHDL, tornando-a um recurso valioso para entender melhor a aplicação da linguagem na modelagem e simulação de circuitos digitais. O material abrange desde conceitos básicos até tópicos mais avançados, com foco prático que se estende da página 3 até a página 19.

Referências

CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.

IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.

LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.

SOUZA, D. B. da C. et al. **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

Aula 2

Circuitos Combinacionais: Construção Básica

Circuitos combinacionais: construção básica



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Descubra a essência dos elementos de controle e paridade em circuitos combinacionais nesta videoaula focada na aplicação prática desses conceitos fundamentais na engenharia eletrônica.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



Você aprenderá como pequenas alterações nos latches e elementos de paridade podem impactar significativamente a confiabilidade dos sistemas digitais, desde redes de fibra óptica até sistemas de segurança digital. Acompanhe Ana e Carlos enquanto eles exploram soluções de design para desafios reais enfrentados por engenheiros todos os dias. Participe desta exploração prática e aprofunde seu conhecimento em circuitos combinacionais para melhorar suas habilidades profissionais.

Ponto de Partida

Imagine um mundo em que cada bit de informação é crucial, e um simples erro pode causar falhas catastróficas em sistemas críticos como comunicações aeroespaciais, transações financeiras ou infraestruturas de TI. No campo da eletrônica digital, a integridade e precisão dos dados não são apenas desejáveis, são absolutamente essenciais. É aqui que os elementos de controle e paridade em circuitos combinacionais entram em jogo, garantindo que cada pedaço de informação transmitido ou processado seja exato e confiável.

Na sala de aula, Ana Beatriz se depara com um desafio intrigante: como garantir que os sistemas digitais verifiquem seus próprios erros e os corrija automaticamente? Este problema não é apenas acadêmico, mas uma questão real que profissionais como Carlos Eduardo, um engenheiro sênior e mentor de Ana, enfrentam diariamente. Carlos introduz Ana aos conceitos de "latches" e portas controladas, fundamentais para controlar o fluxo de dados, e "elementos de paridade", essenciais para a detecção de erros.

Durante uma discussão em classe, Carlos apresenta a Ana um cenário: um sistema de comunicação via satélite que falha periodicamente devido a erros não detectados durante a transmissão de dados. Ele desafia Ana a aplicar o que aprendeu sobre elementos de paridade para desenvolver uma solução que possa reduzir a taxa de erros, garantindo a confiabilidade das comunicações.

Ana aceita o desafio entusiasmada. Ela sabe que resolver este problema aprofundará seu entendimento dos princípios fundamentais de circuitos combinacionais, e terá um impacto direto em sua futura carreira como engenheira eletrônica. Além disso, as soluções desenvolvidas podem ser aplicadas em uma ampla gama de tecnologias, desde redes de fibra óptica até sistemas de segurança digital.

Carlos e Ana exploram juntos exemplos de código VHDL para latches e elementos de paridade, analisando como pequenas alterações nos circuitos podem levar a melhorias significativas na detecção e correção de erros. Essa abordagem prática motiva Ana e prepara todos os alunos para enfrentar desafios semelhantes em seus futuros ambientes de trabalho.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Vamos Começar!

Elementos de controle e paridade em circuitos combinacionais

Os elementos de controle em circuitos combinacionais são componentes essenciais que direcionam o fluxo de dados dentro de um circuito digital, assegurando que a saída correta seja produzida em resposta às várias combinações de entrada. Eles são amplamente utilizados em projetos de circuitos para realizar funções específicas como seleção de dados, encaminhamento de sinal e controle de operações lógicas.

Elementos de controle

Os latches e portas controladas são utilizados para controlar quando um sinal é permitido passar através de um ponto no circuito. Eles funcionam como uma chave que pode ser aberta ou fechada com base em um sinal de controle. Um exemplo de código VHDL de um latch é descrito a seguir:

```
ENTITY latch_control IS
  PORT (
    data : IN STD_LOGIC;
    enable : IN STD_LOGIC;
    q : OUT STD_LOGIC
  );
END latch_control;

ARCHITECTURE behavior OF latch_control IS
BEGIN
  PROCESS (enable, data)
  BEGIN
    IF enable = '1' THEN
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

q <= data;

END IF;

END PROCESS;

END behavior;

As portas controladas por tempo, que realizam operações lógicas somente em determinados momentos, são baseadas em um sinal de controle de tempo ou clock, conforme o exemplo a seguir.

ENTITY and_gate_controlled IS

POR T(

a, b, clk : IN STD_LOGIC;

y : OUT STD_LOGIC

);

END and_gate_controlled;

ARCHITECTURE behavior OF and_gate_controlled IS

BEGIN

y <= a AND b WHEN clk = '1' ELSE '0';

END behavior;

Elementos de paridade

A paridade é uma técnica de verificação de erros utilizada para garantir a integridade de dados em sistemas de comunicação e armazenamento. É configurada de duas formas principais:

Paridade par: a soma total de bits '1', incluindo o bit de paridade, deve ser par.

Paridade ímpar: a soma total de bits '1', incluindo o bit de paridade, deve ser ímpar.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Os elementos de paridade, como geradores e verificadores, são implementados em VHDL utilizando operações lógicas simples, principalmente o operador XOR, devido às suas propriedades algébricas adequadas para essa função.

1. Gerador de paridade

O gerador de paridade calcula e adiciona um bit de paridade aos dados de forma que a soma total de bits '1' seja conforme a configuração desejada (par ou ímpar).

Exemplo de gerador de paridade par em VHDL:

ENTITY parity_generator IS

PORT (

data : IN STD_LOGIC_VECTOR(7 DOWNTO 0);

parity_bit : OUT STD_LOGIC

);

END parity_generator;

ARCHITECTURE behavior OF parity_generator IS

BEGIN

-- Gera paridade par fazendo XOR de todos os bits de dados

parity_bit <= data(0) XOR data(1) XOR data(2) XOR data(3) XOR

data(4) XOR data(5) XOR data(6) XOR data(7);

END behavior;

2. Verificador de paridade

O verificador de paridade avalia os dados recebidos, incluindo o bit de paridade, para determinar se ocorreu algum erro durante a transmissão.

Exemplo de verificador de paridade par em VHDL:

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```
ENTITY parity_checker IS
  PORT (
    data : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    parity_bit : IN STD_LOGIC;
    error : OUT STD_LOGIC
  );
END parity_checker;
```

```
ARCHITECTURE behavior OF parity_checker IS
BEGIN
  -- Verifica a paridade comparando o XOR de todos os bits, incluindo o de paridade
  error <= NOT((data(0) XOR data(1) XOR data(2) XOR data(3) XOR
                data(4) XOR data(5) XOR data(6) XOR data(7) XOR
                parity_bit) = '0');
END behavior;
```

3. Aplicações

Os elementos de paridade são essenciais em várias aplicações, especialmente naquelas em que a precisão dos dados é crítica:

Sistemas de comunicação: para detectar erros em dados transmitidos.

Memórias de computador: para verificar a integridade dos dados armazenados e acessados.

FPGAs e dispositivos lógicos programáveis: incorporados diretamente para garantir a confiabilidade e robustez dos sistemas digitais.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Esses elementos são simples de implementar e podem reduzir significativamente a taxa de erros em dados transmitidos ou armazenados, garantindo a integridade e a precisão necessárias para operações críticas.

Siga em Frente...

Elementos de igualdade em circuitos combinacionais

Os elementos de igualdade são usados para comparar se dois conjuntos de dados são iguais. Esses elementos são fundamentais em operações de controle, roteamento de dados e verificações de segurança em que decisões são tomadas com base na comparação de dados.

Comparadores de igualdade

Em VHDL, um comparador simples pode ser criado usando operadores lógicos para comparar cada bit correspondente de dois vetores de entrada.

1. Comparador de 1 bit

Um comparador de 1 bit verifica se dois sinais individuais são iguais. Isso pode ser simplesmente implementado usando um operador XNOR, que retorna '1' se ambos os sinais de entrada forem iguais.

ENTITY compare1bit IS

PORT (

A, B : IN STD_LOGIC;

Equal : OUT STD_LOGIC

);

END compare1bit;

ARCHITECTURE Behavior OF compare1bit IS

BEGIN

Equal <= A XNOR B;

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

END Behavior;

2. Comparador de N Bits

Comparadores de múltiplos bits são usados para verificar a igualdade entre vetores de bits. A igualdade é verdadeira apenas se todos os bits correspondentes entre os dois vetores são iguais.

ENTITY compareNbit IS

PORT (

A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);

Equal : OUT STD_LOGIC

);

END compareNbit;

ARCHITECTURE Behavior OF compareNbit IS

BEGIN

Equal <= '1' WHEN A = B ELSE '0';

END Behavior;

3. Aplicação de elementos de igualdade

Os elementos de igualdade são amplamente utilizados em diversas aplicações de hardware, como:

- Sistemas de memória: para verificar endereços ou dados durante operações de leitura e escrita.
- Redes de comutação: para rotear pacotes com base na comparação de endereços de destino.
- Processadores: para operações de desvio condicional, em que a execução de instruções subsequentes depende do resultado de comparações.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

4. Elementos de igualdade avançados

Em aplicações mais complexas, comparadores podem ser combinados com outras lógicas para formar unidades de processamento mais avançadas como parte de ULAs (unidades lógicas e aritméticas) em microprocessadores. Estes comparadores podem incluir lógica adicional para lidar com sinais de carry, overflow e outros indicadores de estado.

A seguir é mostrado um exemplo de comparador em uma ULA:

ENTITY alu IS

PORT (

A, B : IN STD_LOGIC_VECTOR(7 DOWNTO 0);

OpCode : IN STD_LOGIC_VECTOR(2 DOWNTO 0);

Result : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);

Flags : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) -- Flags for zero, carry, etc.

);

END alu;

ARCHITECTURE Behavior OF alu IS

BEGIN

PROCESS (A, B, OpCode)

BEGIN

CASE OpCode IS

WHEN "000" => -- Add A and B

Result <= A + B;

WHEN "001" => -- Subtract A from B

Result <= A - B;

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

WHEN "010" => -- Check equality

Result(0) <= '1' WHEN A = B ELSE '0';

-- Other operations

END CASE;

END PROCESS;

END Behavior;

Este é um trecho de código VHDL que define uma entidade (ENTITY) e sua arquitetura de comportamento (ARCHITECTURE) para uma unidade lógico-aritmética (ULA).

ENTIDADE (ENTITY):

- Declara as entradas e saídas do módulo ULA.
- As entradas (A, B e OpCode) são definidas como STD_LOGIC_VECTOR(7 DOWNTO 0), o que significa que são vetores de 8 bits.
- As saídas (Result e Flags) também são STD_LOGIC_VECTOR(7 DOWNTO 0) e STD_LOGIC_VECTOR(3 DOWNTO 0), respectivamente.

ARQUITETURA Comportamento (ARCHITECTURE Behavior):

- Esta seção define o comportamento da ULA.
- Dentro do bloco PROCESS, a ULA realiza diferentes operações com base no valor de OpCode.
- O comando CASE avalia o valor de OpCode e executa a operação correspondente. Por exemplo:
 - Se OpCode for "000", realiza-se a adição (Result <= A + B;).
 - Se OpCode for "001", realiza-se a subtração (Result <= A - B;).
 - Se OpCode for "010", verifica-se a igualdade entre A e B, definindo o bit menos significativo de Result conforme necessário.

Outras operações podem ser adicionadas no comando CASE para diferentes valores de OpCode. Os elementos de igualdade são fundamentais em uma vasta gama de aplicações eletrônicas, proporcionando a base para operações de comparação e decisão em muitos sistemas digitais modernos.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Vamos Exercitar?

Aplicando elementos de controle e paridade

No início da aula, destacamos o desafio enfrentado por Ana Beatriz sob a orientação de Carlos Eduardo: desenvolver um sistema capaz de autoverificar e corrigir erros em circuitos combinacionais. Utilizando os conceitos de elementos de controle e paridade, eles buscaram garantir a integridade e precisão dos dados em sistemas digitais críticos.

Para resolver esse desafio, Ana e Carlos decidiram focar duas frentes principais: a implementação de um circuito com latches controlados para gerenciar o fluxo de dados e a aplicação de elementos de paridade para detecção e correção de erros.

Implementação e simulação de um latch controlado

Objetivo: demonstrar como um latch controlado pode ser usado para permitir ou bloquear o fluxo de dados com base em sinais de controle, crucial em sistemas onde precisão e timing são essenciais.

Tarefa realizada: Ana programou o latch usando VHDL, simulando o circuito para verificar sua resposta sob condições de controle variáveis. Eles confirmaram que o latch só permitia a passagem de dados quando o sinal de controle estava ativo, bloqueando eficazmente quando necessário.

Criação de um gerador e verificador de paridade

Objetivo: estabelecer um método de verificação de erros para garantir que os dados transmitidos não sofram alterações inadvertidas ou corrupção.

Tarefa realizada: Carlos guiou Ana na implementação do gerador de paridade par, no qual adicionaram um bit de paridade aos dados transmitidos. Seguidamente, desenvolveram um verificador de paridade para testar a integridade dos dados recebidos.

Simulação: eles simularam a transmissão de dados com e sem erros, observando a eficácia do verificador em identificar discrepâncias.

Discussão e reflexão

Após a implementação e validação dos circuitos, Ana e Carlos discutiram as implicações de seus achados:

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- Ana percebeu a importância crítica de elementos de controle e paridade em sistemas de comunicação e como pequenos erros podem ser magnificados em sistemas complexos.
- Carlos destacou a relevância de tais sistemas em ambientes reais, como na comunicação satelital e nas redes de fibra óptica, em que a integridade dos dados é fundamental.

A experiência prática ajudou Ana a entender não só a teoria por trás dos circuitos combinacionais, mas também como aplicar esses conceitos para resolver problemas reais. Carlos incentivou Ana a pensar em outras aplicações dos conceitos, como em sistemas de segurança digital e monitoramento de saúde, nos quais a precisão e a confiabilidade são igualmente críticas.

Saiba mais

Para complementar seu estudo da linguagem VHDL e suas aplicações no design de circuitos digitais, recomendamos a seguinte apostila, que oferece uma abordagem prática e detalhada:

[**"Apostila com Códigos de Programas Demonstrativos usando a linguagem VHDL para Circuitos Digitais"**](#) por Alexandre Santos de la Vega (Universidade Federal Fluminense).

Esta apostila contém uma série de exemplos práticos e códigos demonstrativos em VHDL, tornando-a um recurso valioso para entender melhor a aplicação da linguagem na modelagem e simulação de circuitos digitais. O material abrange desde conceitos básicos até tópicos mais avançados, com foco prático que se estende da página 21 até a página 52.

Referências

- CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. et al. **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

Aula 3

Circuitos Combinacionais: Blocos Funcionais

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Circuitos Combinacionais: Blocos Funcionais



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Mergulhe no estudo de multiplexadores, demultiplexadores e decodificadores através de VHDL nesta videoaula. Aprenda como esses componentes-chave gerenciam fluxos de dados em sistemas digitais, aumentando a eficiência e segurança das comunicações. Desde conceitos teóricos até simulações práticas, esta aula é essencial para quem busca aprofundar habilidades em eletrônica digital e VHDL. Junte-se a nós para transformar seu conhecimento em soluções práticas.

Ponto de Partida

No dinâmico campo da eletrônica digital, os multiplexadores, demultiplexadores e decodificadores desempenham papéis cruciais ao otimizar o fluxo de dados através de circuitos complexos. Essenciais em tecnologias como telecomunicações e processamento de dados, esses componentes são vitais para a gestão eficiente da largura de banda e direcionamento de sinais em sistemas eletrônicos.

Ana Beatriz, uma estudante avançada de engenharia eletrônica, está trabalhando em seu projeto final, que envolve a simulação de um sistema de comunicação para um satélite utilizando VHDL. O desafio é projetar um sistema usando multiplexadores e demultiplexadores para gerenciar eficientemente a transmissão de dados entre a Terra e o satélite, garantindo que o sinal correto seja enviado por meio do canal apropriado sem interferências. Sob a supervisão de Carlos Eduardo, um engenheiro experiente e seu mentor, Ana precisa desenvolver e simular o código VHDL que implemente eficazmente esta seleção, minimize os erros de transmissão e otimize o uso do espectro de comunicação.

Acompanhe Ana e Carlos enquanto eles exploram a implementação prática de multiplexadores e demultiplexadores em VHDL nesta videoaula. Esta sessão reforçará seu entendimento da sintaxe e funcionalidades do VHDL e demonstrará como esses dispositivos são implementados em cenários do mundo real. Engaje-se neste estudo profundo para aprimorar suas habilidades de projeto e simulação de circuitos digitais, fundamentais para qualquer engenheiro eletrônico que deseja inovar no campo das telecomunicações.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Vamos Começar!

Multiplexadores, demultiplexadores e decodificadores

Os multiplexadores e demultiplexadores são componentes fundamentais em circuitos digitais, utilizados para gerenciar o fluxo de dados dentro de um sistema eletrônico. Essas estruturas são descritas a seguir usando VHDL.

- **Multiplexador (MUX)**

Um multiplexador é um dispositivo que seleciona uma entre várias entradas de dados e a transmite para uma única linha de saída. O sinal de seleção, geralmente um código binário, determina qual entrada será conectada à saída. Em sistemas digitais, os MUXs são essenciais para otimizar o uso de recursos limitados, como linhas de comunicação.

```

ENTITY mux IS
  PORT(
    data0, data1, data2, data3 : IN STD_LOGIC; -- Entradas
    select : IN STD_LOGIC_VECTOR(1 DOWNTO 0); -- Sinal de seleção
    output : OUT STD_LOGIC           -- Saída
  );
END mux;

ARCHITECTURE behavior OF mux IS
BEGIN
  WITH select SELECT
    output <= data0 WHEN "00",
      data1 WHEN "01",
      data2 WHEN "10",
      data3 WHEN OTHERS;
END behavior;

```

A entidade mux é a definição da interface do circuito. Ela descreve as entradas e saídas do multiplexador:

data0, data1, data2, data3 : IN STD_LOGIC; – Estas são as entradas de dados do MUX. Cada uma delas pode receber um valor de lógica digital (*STD_LOGIC*), que é tipicamente 0 ou 1.

select: IN STD_LOGIC_VECTOR(1 DOWNTO 0); – Este é o sinal de seleção, um vetor de lógica de dois bits que decide qual das entradas de dados será passada para a saída. Os bits são contados de 1 para 0.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

output: OUT STD_LOGIC; – Esta é a saída do MUX, e o sinal selecionado das entradas é enviado.

A arquitetura *behavior* define como o MUX funciona internamente, ou seja, como ele processa as entradas para produzir uma saída:

WITH select SELECT – Esta construção é usada para fazer uma seleção baseada no valor do vetor *select*.

output <= data0 WHEN "00", data1 WHEN "01", data2 WHEN "10", data3 WHEN OTHERS; – Define a lógica de seleção:

- Quando *select* é "00", *output* será igual a *data0*.
- Quando *select* é "01", *output* será igual a *data1*.
- Quando *select* é "10", *output* será igual a *data2*.

Para qualquer outro valor de *select (OTHERS)*, *output* será igual a *data3*.

Essencialmente, este código VHDL cria um circuito em que uma entre quatro entradas é selecionada para ser transmitida à saída única, com base nos valores de dois bits de entrada *select*. Esse tipo de dispositivo é fundamental em sistemas digitais nos quais o número de linhas de comunicação é limitado, pois permite que múltiplos sinais compartilhem uma única linha de comunicação.

- **Demultiplexador (DEMUX)**

Um demultiplexador realiza a função inversa de um multiplexador, distribuindo uma única entrada de dados para uma das várias saídas, baseado no sinal de seleção. Os DEMUXs são usados para direcionar informações de uma fonte para diferentes destinos.

```
ENTITY demux IS
  PORT(
    input : IN STD_LOGIC;          -- Entrada
    select : IN STD_LOGIC_VECTOR(1 DOWNTO 0);-- Sinal de seleção
    data0, data1, data2, data3 : OUT STD_LOGIC -- Saídas
  );
END demux;
```

```
ARCHITECTURE behavior OF demux IS
BEGIN
  CASE select IS
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```

WHEN "00" =>
    data0 <= input;
    data1 <= '0';
    data2 <= '0';
    data3 <= '0';
WHEN "01" =>
    data0 <= '0';
    data1 <= input;
    data2 <= '0';
    data3 <= '0';
WHEN "10" =>
    data0 <= '0';
    data1 <= '0';
    data2 <= input;
    data3 <= '0';
WHEN OTHERS =>
    data0 <= '0';
    data1 <= '0';
    data2 <= '0';
    data3 <= input;
END CASE;
END behavior;

```

A entidade *demux* é a definição da interface do circuito. Ela descreve a entrada e as saídas do demultiplexador:

input: IN STD_LOGIC; – Esta é a única entrada de dados do *DEMUX*.

select: IN STD_LOGIC_VECTOR(1 DOWNTO 0); – Este é o sinal de seleção, um vetor de dois bits que decide qual das saídas receberá o sinal de entrada.

data0, data1, data2, data3: OUT STD_LOGIC; – Estas são as saídas do *DEMUX*. Uma destas saídas será selecionada para receber o valor da entrada, enquanto as outras saídas serão configuradas para zero.

A arquitetura *behavior* define como o *DEMUX* opera internamente, especificando como a entrada é distribuída para as saídas com base no valor de *select*.

A instrução *CASE select IS* inicia a definição de várias condições, dependendo do valor do vetor *select*.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

WHEN "00" => – Se *select* é "00", a entrada é enviada para *data0* e todas as outras saídas (*data1*, *data2*, *data3*) são configuradas para zero ('0').

WHEN "01" => – Se *select* é "01", a entrada é enviada para *data1* e as outras saídas são configuradas para zero.

WHEN "10" => – Se *select* é "10", a entrada é enviada para *data2* e as outras saídas são configuradas para zero.

WHEN OTHERS => – Para qualquer outro valor de *select* (como "11"), a entrada é enviada para *data3* e as outras saídas são configuradas para zero.

Essencialmente, este código VHDL implementa um circuito no qual um sinal de entrada pode ser direcionado para uma entre quatro saídas, com base nos valores de dois bits de entrada *select*. Esse tipo de dispositivo é crucial em sistemas digitais para direcionar informações de uma única fonte para diferentes destinos, como múltiplos dispositivos ou processos.

- **Decodificador**

Os decodificadores são dispositivos que realizam a função oposta aos codificadores, convertendo códigos de entrada binários em sinais de saída múltiplos. São amplamente usados em aplicações que exigem seleção de linha ou identificação de estados específicos.

Exemplo de um decodificador 2 para 4:

```

ENTITY decoder2to4 IS
PORT(
    input : IN STD_LOGIC_VECTOR(1 DOWNTO 0); -- Entrada binária
    enable : IN STD_LOGIC; -- Habilitação
    outputs : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) -- Saídas
);
END decoder2to4;

ARCHITECTURE behavior OF decoder2to4 IS
BEGIN
    IF enable = '1' THEN
        CASE input IS
            WHEN "00" => outputs <= "0001";
            WHEN "01" => outputs <= "0010";
            WHEN "10" => outputs <= "0100";
            WHEN "11" => outputs <= "1000";
            WHEN OTHERS => outputs <= "0000";
        END CASE;
    END IF;
END behavior;

```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```

END CASE;
ELSE
  outputs <= (OTHERS => '0');
END IF;
END behavior;

```

O código VHDL apresentado define um decodificador de 2 para 4, que é um tipo de circuito digital usado para converter um código binário de entrada em múltiplos sinais de saída, em que apenas uma das saídas é ativada por vez. Este tipo de dispositivo é útil para aplicações que requerem seleção de linha ou identificação de estados específicos. Vamos examinar cada parte do código:

A entidade *decoder2to4* define a interface do decodificador:

input : IN STD_LOGIC_VECTOR(1 DOWNTO 0); – Esta é a entrada binária que será decodificada. Consiste em um vetor de dois bits.

enable : IN STD_LOGIC; – Um sinal de habilitação que controla se o decodificador está ativo ('1') ou inativo ('0').

outputs : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); – Estas são as saídas do decodificador, um vetor de quatro bits em que cada bit representa uma linha de saída diferente.

A arquitetura *behavior* descreve como o decodificador processa as entradas para gerar as saídas:

A condição *IF enable = '1' THEN* verifica se o decodificador está habilitado. Se estiver, a lógica de decodificação é processada; caso contrário, todas as saídas são configuradas para zero.

A instrução *CASE input IS* define como a entrada é decodificada em saídas:

WHEN "00" => outputs <= "0001"; – Se a entrada é "00", a primeira saída (posição mais à direita) é ativada (1), enquanto todas as outras saídas são desativadas (0).

WHEN "01" => outputs <= "0010"; – Se a entrada é "01", a segunda saída é ativada.

WHEN "10" => outputs <= "0100"; – Se a entrada é "10", a terceira saída é ativada.

WHEN "11" => outputs <= "1000"; – Se a entrada é "11", a quarta saída é ativada.

WHEN OTHERS => outputs <= "0000"; – Para qualquer outra entrada não especificada, todas as saídas são desativadas.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Se o sinal *enable* não for '1' (ou seja, é '0'), então *outputs* \leq (*OTHERS* \Rightarrow '0'); define todas as saídas para zero, independentemente da entrada.

Esse decodificador é útil em muitas aplicações de hardware, como seleção de múltiplas linhas de memória ou dispositivos de entrada/saída, em que um sinal específico precisa ser roteado para diferentes componentes de um sistema baseado na entrada fornecida.

Siga em Frente...

Somadores

Os somadores são componentes essenciais em sistemas digitais, responsáveis por realizar a soma de números binários. Eles são usados em uma variedade de aplicações que vão desde operações aritméticas simples em calculadoras até operações complexas em processadores e sistemas de controle.

Existem principalmente dois tipos de somadores em circuitos digitais:

- **Somador de meio bit (*half adder*):**

O somador de meio bit é a forma mais simples de somador, capaz de adicionar dois bits singulares e produzir uma soma e um bit de *carry out*.

Ele é composto por uma porta *XOR* para a soma e uma porta *AND* para o *carry out*.

```
ENTITY half_adder IS
  PORT(
    A, B : IN STD_LOGIC; -- Entradas de bit único
    Sum, CarryOut : OUT STD_LOGIC -- Soma e Carry out
  );
END half_adder;

ARCHITECTURE behavior OF half_adder IS
BEGIN
  Sum <= A XOR B; -- Soma dos bits
  CarryOut <= A AND B; -- Carry out é verdadeiro se ambos os bits são 1
END behavior;
```

O código VHDL descrito define um somador de meio bit (*half adder*), que é um tipo básico de circuito somador utilizado em sistemas digitais. Este dispositivo realiza a soma de dois bits

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

singulares, gerando um resultado de soma e um bit de *carry out*. Vejamos a estrutura e funcionamento do código:

A entidade *half_adder* especifica as entradas e saídas do somador:

A, B : IN STD_LOGIC; – São as entradas do somador, cada uma recebendo um bit.

Sum, CarryOut : OUT STD_LOGIC; – *Sum* é a saída que representa a soma dos dois bits, e *CarryOut* é o bit de *carry out*, que indica se houve um overflow da soma (isto é, ambos os bits são 1).

A arquitetura *behavior* define como os bits são somados:

Sum <= A XOR B; – A operação *XOR* é usada para calcular a soma dos dois bits. O resultado será 1 se os bits forem diferentes (0 e 1 ou 1 e 0) e 0 se os bits forem iguais (0 e 0 ou 1 e 1).

CarryOut <= A AND B; – A operação *AND* é utilizada para determinar o *carry out*. Este bit será 1 apenas se ambos os bits de entrada são 1, indicando um overflow da soma para o próximo bit mais significativo.

Este somador de meio bit é essencial em sistemas digitais para construir somadores mais complexos, como somadores completos e somadores em série, que são usados para somar números binários de múltiplos bits.

- Somador completo (*full adder*):

Um somador completo adiciona três bits: dois bits de dados e um *carry in*, que é o *carry out* de uma adição anterior. Isso permite a construção de somadores que podem adicionar números multi-bit. Ele usa duas portas *XOR*, duas portas *AND* e uma porta *OR* para calcular a soma e o *carry out*.

```
ENTITY full_adder IS
  PORT(
    A, B, CarryIn : IN STD_LOGIC; -- Entradas
    Sum, CarryOut : OUT STD_LOGIC -- Saída de soma e carry out
  );
END full_adder;
```

```
ARCHITECTURE behavior OF full_adder IS
BEGIN
  -- Primeiro nível de soma e carry
  Sum <= A XOR B XOR CarryIn;
```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

-- *Carry out* é verdadeiro se pelo menos dois dos três inputs são 1
 $CarryOut \leq (A \text{ AND } B) \text{ OR } (B \text{ AND } CarryIn) \text{ OR } (A \text{ AND } CarryIn);$
END behavior;

O código VHDL descrito define um somador completo (*full adder*), um circuito que adiciona três bits: dois bits de entrada de dados e um bit de entrada conhecido como "*carry in*". Este bit de *carry in* geralmente vem do *carry out* de uma adição anterior, o que permite a construção de somadores que podem lidar com números binários de múltiplos bits. Veja a seguir uma visão geral do código:

A entidade *full_adder* descreve as entradas e saídas do somador completo:

A, B, CarryIn : IN STD_LOGIC; – Estas são as três entradas do somador: dois bits de dados (A e B) e um bit de *carry in* (*CarryIn*).

Sum, CarryOut : OUT STD_LOGIC; – *Sum* é a saída que fornece o resultado da soma dos três bits de entrada, e *CarryOut* é o bit que indica se há um overflow da soma que precisa ser passado para o próximo bit mais significativo na adição de números de múltiplos bits.

A arquitetura *behavior* descreve como os bits são somados e como o *carry out* é gerado:

Sum <= A XOR B XOR CarryIn; – Utiliza duas operações *XOR* para calcular a soma dos três bits de entrada. Esta operação garante que cada bit é contado corretamente para a soma, em que um número ímpar de 1s resultará em uma soma de 1.

CarryOut <= (A AND B) OR (B AND CarryIn) OR (A AND CarryIn); – Esta linha de código determina o *carry out*. Ele será verdadeiro (1) se pelo menos dois dos três inputs são 1, o que é realizado por meio de operações *AND* para cada par de entradas seguido de uma operação *OR* para juntar esses resultados.

O somador completo é essencial em circuitos digitais para a adição de números *multi-bit*, em que múltiplos somadores completos são encadeados juntos, permitindo que o *carry out* de uma posição de bit seja passado como *carry in* para a posição de bit seguinte.

- **Aplicações de somadores**

Os somadores são usados em uma variedade de aplicações digitais, incluindo:

- Unidades aritméticas e lógicas (ULAs): em microprocessadores, nos quais eles executam adições e outras operações aritméticas.
- Circuitos de contagem: em *timers* e contadores digitais.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- Operações de hardware para gráficos e processamento de sinais: para cálculos rápidos necessários em processamento de imagem e áudio.

Os somadores formam a base para operações aritméticas em muitos dispositivos eletrônicos e sistemas computacionais. A compreensão de sua operação e aplicação é fundamental para o design de sistemas eletrônicos eficientes e para a implementação de funções mais complexas em sistemas digitais avançados.

Os decodificadores são dispositivos que realizam a função oposta aos codificadores, convertendo códigos de entrada binários em sinais de saída múltiplos. São amplamente usados em aplicações que exigem seleção de linha ou identificação de estados específicos.

Vamos Exercitar?

Otimização de sistemas de comunicação via satélite

Ana Beatriz enfrenta o desafio de desenvolver um sistema de comunicação via satélite mais eficiente em seu estágio, sob a mentoria de Carlos Eduardo. A tarefa é utilizar multiplexadores e demultiplexadores para otimizar a alocação de largura de banda e minimizar os erros de transmissão, aplicando seus conhecimentos de VHDL para implementar a solução.

Para resolver este problema, Ana seguiu os seguintes passos:

- **Implementação do multiplexador em VHDL:**
 - Objetivo: gerenciar eficientemente o tráfego de entrada no satélite, selecionando o canal apropriado com base no sinal de seleção.
 - Ação: Ana projetou o multiplexador em VHDL, que permite selecionar dinamicamente entre várias entradas de dados usando um sinal de seleção binário. O código VHDL foi testado para garantir que apenas a entrada correta fosse escolhida e transmitida para a saída.
- **Desenvolvimento do demultiplexador em VHDL:**
 - Objetivo: distribuir o sinal de entrada para múltiplos canais de saída no satélite, dependendo do sinal de seleção.
 - Ação: ela codificou e simulou um demultiplexador que direcionava a entrada para a saída adequada, garantindo que os outros canais permanecessem inativos. Este componente é crucial para evitar a sobreposição de sinais e garantir a integridade dos dados transmitidos.
- Objetivo: distribuir o sinal de entrada para múltiplos canais de saída no satélite, dependendo do sinal de seleção.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- Ação: ela codificou e simulou um demultiplexador que direcionava a entrada para a saída adequada, garantindo que os outros canais permanecessem inativos. Este componente é crucial para evitar a sobreposição de sinais e garantir a integridade dos dados transmitidos.

Ana e Carlos realizaram uma série de simulações para validar o desempenho dos circuitos sob várias condições de operação. Eles testaram diferentes cenários de sinal de seleção para garantir que os sistemas multiplexador e demultiplexador funcionassem conforme o esperado, sem perda ou erro de dados.

A solução desenvolvida por Ana provou ser eficaz na melhoria da eficiência da comunicação do satélite, atendendo aos objetivos do projeto. Além disso, Ana aprendeu como pequenas mudanças na gestão de sinais podem ter um impacto significativo na performance dos sistemas de comunicação. Este projeto não só a ajudou a entender melhor os conceitos de multiplexação e demultiplexação como também a preparou para enfrentar desafios semelhantes em sua futura carreira profissional.

Ana e Carlos discutiram futuras melhorias e aplicações dos circuitos, considerando expandir a aplicação para redes de comunicação terrestres, nas quais a eficiência da largura de banda é igualmente crítica. Esta experiência abriu novas áreas de investigação e desenvolvimento para Ana, incentivando-a a explorar ainda mais o potencial do VHDL e da eletrônica digital em suas futuras iniciativas profissionais.

Saiba mais

Para complementar seu estudo da linguagem VHDL e suas aplicações no design de circuitos digitais, recomendamos a seguinte apostila, que oferece uma abordagem prática e detalhada:

["Apostila com Códigos de Programas Demonstrativos usando a linguagem VHDL para Circuitos Digitais"](#) por Alexandre Santos de la Vega (Universidade Federal Fluminense).

Esta apostila contém uma série de exemplos práticos e códigos demonstrativos em VHDL, tornando-a um recurso valioso para entender melhor a aplicação da linguagem na modelagem e simulação de circuitos digitais. O material abrange desde conceitos básicos até tópicos mais avançados, com foco prático que se estende da página 55 até a página 67 e da página 75 até a página 78.

Referências

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.

IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.

LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.

SOUZA, D. B. da C. et al. **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

Aula 4

Circuitos Sequenciais

Circuitos Sequenciais



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Explore o coração dos sistemas digitais com esta videoaula que trata de flip-flops e registradores, modelados em VHDL. Desde fundamentos teóricos até aplicações práticas, aprenderemos como esta linguagem de descrição de hardware transforma designs digitais em realidade. Ideal para estudantes e profissionais de engenharia eletrônica, este curso é um convite para dominar o VHDL e aprimorar suas habilidades de simulação e design de circuitos. Junte-se a nós para moldar o futuro da tecnologia digital.

Ponto de Partida

No universo da engenharia eletrônica, a capacidade de projetar circuitos que executem operações lógicas e retenham estados é crucial. Essa capacidade é viabilizada por meio dos circuitos sequenciais, como flip-flops e registradores, fundamentais para a criação de memórias e sistemas de controle automático. O VHDL, como linguagem de descrição de hardware, desempenha um papel essencial ao permitir que engenheiros como nós projetem e simulem esses circuitos com precisão antes de qualquer implementação física.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Ana Beatriz, uma engenheira em formação, está desenvolvendo um contador digital para seu projeto de fim de curso e precisa garantir que seu design seja não apenas funcional, mas também resiliente a falhas. Carlos Eduardo, seu mentor, sugere a utilização de flip-flops do tipo D, programados em VHDL, para garantir a estabilidade do sistema. No entanto, Ana encontra dificuldades em modelar o comportamento desejado dos flip-flops no VHDL, enfrentando problemas específicos na sincronização dos estados e na manipulação de dados durante as transições de estado.

Essa aula explora como os flip-flops e registradores são modelados em VHDL e como esses modelos são essenciais para a construção de sistemas digitais robustos e confiáveis.

Discutiremos as diferenças entre os tipos de flip-flops e mostraremos exemplos práticos de como implementá-los no VHDL para diferentes aplicações. Prepare-se para mergulhar na aplicação prática do VHDL e descubra como essa linguagem pode transformar conceitos teóricos em componentes funcionais de sistemas digitais.

Vamos Começar!

Flip-flops e registradores

Os circuitos sequenciais são fundamentais na engenharia eletrônica, pois diferem dos circuitos combinacionais ao incorporar memória, permitindo que o sistema digital retenha estados. Essa característica é essencial para a criação de máquinas de estados, sistemas de memória e uma vasta gama de outras aplicações digitais que requerem mais do que simples operações lógicas instantâneas. Os flip-flops e os registradores formam a base desses circuitos ao fornecerem capacidades de armazenamento temporário e permanente de dados.

Flip-flops

Os flip-flops são dispositivos binários que mantêm um estado até serem alterados por um input externo. Eles são a base para o armazenamento de dados em sistemas digitais e são usados para estabilizar e sincronizar as mudanças de dados. Os tipos mais comuns de flip-flops são descritos a seguir.

- **Flip-Flop SR (Set-Reset):** este é um dos tipos mais simples, utilizando sinais de *set* (*S*) e *reset* (*R*) para controlar o estado armazenado. É fundamental em aplicações nas quais é necessário um controle claro de ligar/desligar.
- **Flip-Flop D (Data):** amplamente utilizado em sistemas digitais para armazenar dados, ele captura o valor na entrada *D* no flanco de subida do sinal de *clock* e o mantém até a próxima mudança de *clock*.
- **Flip-Flop T (Toggle):** também conhecido como flip-flop de disparo, alterna seu estado em resposta a cada pulso de *clock*, e é útil em contadores binários.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- **Flip-Flop JK:** uma evolução dos tipos *SR* e *T*, permitindo mais controle sobre o estado armazenado com entradas que determinam não apenas o estado, mas também as condições de mudança.

O trecho de código VHDL apresentado a seguir define uma entidade chamada *d_flip_flop* e sua arquitetura de comportamento.

```
rquitetura de comportamento.
ENTITY d_flip_flop IS
  PORT(
    D, Clock : IN STD_LOGIC;
    Q : OUT STD_LOGIC
  );
END d_flip_flop;
ARCHITECTURE behavior OF d_flip_flop IS
BEGIN
  PROCESS (Clock)
  BEGIN
    IF rising_edge(Clock) THEN
      Q <= D;
    END IF;
  END PROCESS;
END behavior;
```

Veja uma explicação detalhada do código:

A entidade *d_flip_flop* é definida com a seguinte interface:

- **PORT:** a seção *PORT* lista as entradas e saídas do flip-flop tipo D.
- **D:** uma entrada digital (*IN STD_LOGIC*). Este é o dado que será capturado e armazenado pelo flip-flop na borda de subida do sinal de clock.
- **Clock:** uma entrada digital (*IN STD_LOGIC*) que serve como o clock do flip-flop. As mudanças de estado do flip-flop são desencadeadas pelas bordas de subida desse sinal de clock.
- **Q:** uma saída digital (*OUT STD_LOGIC*). Esta é a saída do flip-flop, que reflete o valor armazenado dentro dele, isto é, a entrada D após uma borda de subida do clock.

A arquitetura chamada *behavior* define o comportamento funcional do flip-flop tipo D:

- **PROCESS (Clock):** este bloco de processo executa as operações baseadas na entrada clock. O processo é sensível apenas à entrada de clock, o que significa que o código

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

dentro do bloco *PROCESS* é avaliado somente quando há uma mudança no estado do clock.

- **IF rising_edge(Clock) THEN:** esta instrução condicional verifica se o sinal de clock está em uma borda de subida — a transição de '0' para '1'. O flip-flop tipo *D* captura o valor de entrada *D* apenas nesse momento específico.
- **Q <= D:** se a condição for verdadeira (borda de subida do clock), a saída *Q* é atualizada para refletir o valor da entrada *D*. Isso significa que o valor de *D* no momento da borda de subida do clock é armazenado e mantido em *Q* até a próxima borda de subida do clock.

Este design de flip-flop *D* é um exemplo clássico de um circuito de armazenamento temporário ou de "trava" (*latch*) que é usado amplamente em sistemas digitais para armazenar dados binários. Ele é essencial em aplicações que requerem sincronização e armazenamento temporário de valores de bits, como em sistemas de memória, registradores e muitas outras aplicações lógicas sequenciais.

Registradores

Os registradores são conjuntos de flip-flops usados conjuntamente para armazenar múltiplos bits de dados. Eles são usados em microprocessadores para armazenar endereços, instruções e dados intermediários essenciais durante o processamento computacional. Um registrador pode armazenar dados que são atualizados em cada pulso de clock, permitindo a manipulação e armazenamento temporário de informações em operações de computador. O código VHDL fornecido a seguir descreve um registrador digital de 4 bits que opera com base em um sinal de clock.

```

ENTITY register IS
PORT(
  Data : IN STD_LOGIC_VECTOR(3 DOWNTO 0); -- Entrada de dados de 4 bits
  Clock : IN STD_LOGIC;
  Output : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) -- Saída de dados de 4 bits
);
END register;
ARCHITECTURE behavior OF register IS
BEGIN
  PROCESS (Clock)
  BEGIN
    IF rising_edge(Clock) THEN
      Output <= Data;
    END IF;
  END PROCESS;
END behavior;

```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Vamos detalhar cada seção do código para entender seu funcionamento e aplicação:

A definição da *ENTITY* nomeada register especifica a interface do registrador:

- **Data:** esta é a entrada do registrador e é definida como um vetor lógico de 4 bits (*STD_LOGIC_VECTOR(3 DOWNTO 0)*). Os dados que chegam nesta porta são armazenados no registrador na borda de subida do sinal de clock.
- **Clock:** este é o sinal de *clock* (*IN STD_LOGIC*), que controla quando os dados são lidos e armazenados no registrador.
- **Output:** esta é a saída do registrador, também um vetor lógico de 4 bits (*STD_LOGIC_VECTOR(3 DOWNTO 0)*). Ela mostra o estado atual dos dados armazenados no registrador.

A *ARCHITECTURE* chamada *behavior* descreve o comportamento do registrador:

- **PROCESS (Clock):** este bloco de processo é sensível apenas à entrada de clock. Isso significa que as instruções dentro do processo são executadas apenas em resposta a mudanças no sinal de clock, mais especificamente, na borda de subida do clock.
- **IF rising_edge(Clock) THEN:** a função *rising_edge(Clock)* é uma condição que verifica se o clock está em uma borda de subida (transição de '0' para '1'). Este é o momento em que as operações de atualização de dados são realizadas no registrador.
- **Output <= Data:** quando detectada uma borda de subida do clock, o registrador captura e armazena os valores presentes na entrada Data. Em seguida, esses valores são imediatamente refletidos na saída *output*. Isso garante que o estado do registrador seja atualizado em cada ciclo de clock, mantendo a sincronia com outras operações do sistema que dependem do mesmo sinal de clock.

Aplicações de flip-flops e registradores

- Memória temporária usada em CPUs e GPUs em que operações complexas requerem acesso rápido e temporário a grandes volumes de dados.
- Contadores e divisores de frequência usados para medir o tempo e dividir frequências em sistemas eletrônicos, de relógios a sistemas de comunicação.
- *Debouncing* de botões fundamental em interfaces de usuário, em que estabilizam sinais elétricos para evitar leituras múltiplas causadas por ruídos.

A compreensão e aplicação de flip-flops e registradores são importantes para o desenvolvimento de tecnologias digitais avançadas. Eles permitem que engenheiros e projetistas criem sistemas que não apenas respondem a estímulos externos, mas também mantêm um registro de eventos anteriores, facilitando o desenvolvimento de dispositivos mais inteligentes e responsivos. Estudar esses componentes oferece uma base sólida para explorar ainda mais a eletrônica digital e suas aplicações práticas em tecnologia moderna.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Siga em Frente...

Máquinas de Estados

As máquinas de estados, ou autômatos finitos, são modelos computacionais que executam sequências de operações baseadas no estado atual do sistema e nas entradas recebidas. Esses dispositivos são essenciais para o controle de processos em sistemas digitais, desde interfaces de usuário interativas até sistemas embarcados complexos. Em essência, uma máquina de estados pode ser vista como um diagrama que descreve os estados possíveis de um sistema e as transições entre esses estados em resposta a eventos específicos.

Existem dois tipos principais de máquinas de estados utilizadas em sistemas digitais:

Máquinas de estados de Mealy: as saídas são determinadas tanto pelo estado atual quanto pelas entradas. Isso permite reações mais rápidas às entradas, pois as saídas podem mudar assim que as entradas mudam, sem necessidade de transição de estado.

Máquinas de estados de Moore: neste tipo, as saídas são determinadas exclusivamente pelo estado atual. As mudanças de estado ocorrem com base nas entradas, mas a saída é uma função direta do estado.

A implementação de uma máquina de estados em VHDL requer a definição dos estados, das transições de estado e das saídas associadas a cada estado. Utiliza-se frequentemente a estrutura de case para lidar com as transições e as saídas com base no estado atual e nas entradas.

Exemplo de máquina de estados de Mealy em VHDL:

A seguir é mostrado um exemplo em VHDL de uma máquina de Mealy. Neste caso, a máquina vai mudar a saída baseada diretamente na entrada recebida e no estado atual. Ela vai gerar uma saída '1' sempre que a entrada mudar de '0' para '1', e '0' em todos os outros casos.

```

ENTITY simple_mealy IS
  PORT(
    clock : IN STD_LOGIC;
    reset : IN STD_LOGIC;
    input : IN STD_LOGIC; -- Entrada binária
    output : OUT STD_LOGIC -- Saída '1' se a entrada muda de '0' para '1'
  );
END simple_mealy;

ARCHITECTURE behavior OF simple_mealy IS

```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```

SIGNAL last_input : STD_LOGIC := '0'; -- Armazena o último estado da entrada
BEGIN
    -- Processo para atualizar a saída e o último estado da entrada
    PROCESS (clock, reset)
    BEGIN
        IF reset = '1' THEN
            last_input <= '0'; -- Reseta o último estado da entrada
            output <= '0'; -- Reseta a saída
        ELSIF rising_edge(clock) THEN
            IF last_input = '0' AND input = '1' THEN
                output <= '1'; -- Muda a saída para '1' se a entrada mudou de '0' para '1'
            ELSE
                output <= '0';
            END IF;
            last_input <= input; -- Atualiza o último estado da entrada
        END IF;
    END PROCESS;
END behavior;

```

Vamos detalhar as principais partes do código para entender seu funcionamento e aplicação:

- A máquina tem entradas para clock e reset, além de um sinal de entrada binário, e uma saída que indica se ocorreu uma mudança de '0' para '1'.
- Sinal `last_input` armazena o estado anterior da entrada para detecção de mudança.
- Processo: controlado pelo reset e clock. Se o reset for ativado, ele reseta a saída e o estado anterior da entrada. Na borda de subida do clock, ele verifica se a entrada mudou de '0' para '1'. Se sim, a saída é '1', caso contrário, é '0'. O estado anterior da entrada é então atualizado para o estado atual.

Exemplo de máquina de estados de Moore em VHDL:

Para transformar o exemplo de uma máquina de Mealy em uma máquina de Moore, precisamos separar a lógica de determinação da saída da lógica de transição de estados. Na máquina de Moore, a saída é determinada exclusivamente pelo estado atual, e não diretamente pela entrada como em Mealy. Vamos adaptar o exemplo anterior para seguir este princípio.

```

ENTITY simple_moore IS
PORT(
    clock : IN STD_LOGIC;
    reset : IN STD_LOGIC;
    input : IN STD_LOGIC; -- Entrada binária
    output : OUT STD_LOGIC -- Saída '1' apenas se no estado detectado

```

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

```

);
END simple_moore;
ARCHITECTURE behavior OF simple_moore IS
  TYPE state_type IS (wait_for_change, detected_change);
  SIGNAL current_state, next_state : state_type;
BEGIN
  -- Processo de Transição de Estado
  PROCESS (clock, reset)
  BEGIN
    IF reset = '1' THEN
      current_state <= wait_for_change;
    ELSIF rising_edge(clock) THEN
      current_state <= next_state;
    END IF;
  END PROCESS;
  -- Processo de Definição de Próximo Estado
  PROCESS (current_state, input)
  BEGIN
    CASE current_state IS
      WHEN wait_for_change =>
        IF input = '1' THEN
          next_state <= detected_change;
        ELSE
          next_state <= wait_for_change;
        END IF;
      WHEN detected_change =>
        next_state <= wait_for_change; -- Volta ao estado de espera independentemente da
    entra
    END CASE;
  END PROCESS;
  -- Lógica de Saída
  PROCESS (current_state)
  BEGIN
    CASE current_state IS
      WHEN wait_for_change =>
        output <= '0';
      WHEN detected_change =>
        output <= '1';
    END CASE;
  END PROCESS;
END behavior;

```

Vamos detalhar as principais partes do código para entender seu funcionamento e aplicação:

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- A máquina tem entradas para clock, reset, um sinal de entrada binário e uma saída.
- *state_type* define dois estados – *wait_for_change*, em que a máquina aguarda por uma mudança na entrada, e *detected_change*, ponto em que a mudança foi detectada.
- Transição de estado: se *reset* está ativado, a máquina retorna para o estado *wait_for_change*. Em cada borda de subida do clock, o estado atual muda para o *next_state* calculado anteriormente.
- Definição de próximo estado: dependendo do estado atual e da entrada, define o próximo estado. Se no estado *wait_for_change* e a entrada é '1', transita para *detected_change*. Caso contrário, permanece no mesmo estado ou retorna ao *wait_for_change* se o estado foi *detected_change*.
- Lógica de saída: a saída é definida exclusivamente pelo estado atual. No estado *detected_change*, a saída é '1'. Em *wait_for_change*, é '0'.

Este exemplo de máquina de Moore pode ser usado em aplicações que exigem uma resposta clara e estável aos estados detectados, como sistemas de alarme ou indicadores de status, em que a mudança precisa ser notificada e mantida claramente até o próximo ciclo de processamento. Ao contrário de Mealy, a saída muda apenas na transição entre os estados, tornando as respostas menos suscetíveis a ruídos momentâneos nas entradas.

Aplicações de máquinas de estados

- Controle de sistemas embarcados: utilizadas para gerenciar o funcionamento de dispositivos eletrônicos, como máquinas de lavar, fornos micro-ondas e sistemas de segurança.
- Protocolos de comunicação: essenciais na implementação de protocolos em que diversas condições precisam ser verificadas e tratadas sequencialmente.
- Jogos e interfaces de usuário: implementam a lógica de controle que responde às ações do usuário, alterando estados visuais ou de jogo.

A compreensão e implementação de máquinas de estados são fundamentais para engenheiros e projetistas que desejam criar sistemas digitais interativos e responsivos. A habilidade de modelar comportamentos complexos como estados e transições permite a criação de sistemas mais organizados e eficientes, capazes de responder dinamicamente a uma ampla gama de condições operacionais e entradas de usuários.

Vamos Exercitar?

Resolvendo desafios com VHDL

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

O desafio enfrentado por Ana Beatriz, sob a orientação de Carlos Eduardo, consiste em projetar um sistema de cronometragem que precisava ser robusto e resiliente a falhas, utilizando flip-flops em sua estrutura. Carlos ajudou Ana a entender as propriedades dos flip-flops tipo D usando VHDL para garantir a integridade dos dados durante as transições de estado e falhas de energia. Juntos, eles modificaram o design original para incluir um mecanismo que mantém o estado do flip-flop durante interrupções de energia, utilizando VHDL para simular e validar a solução. As seguintes ações foram tomadas:

- **Implementação do bloco VHDL:** Ana implementou um bloco VHDL que adiciona capacidade de retenção de estado ao flip-flop D. Ela adicionou um sinal de reset controlado por uma entrada que indica a condição de energia. Se a energia estiver estável, o flip-flop mantém seu estado; se houver falha, ele mantém o último estado conhecido até que a energia seja restaurada.
- **Testes e simulações:** Carlos ajudou Ana a configurar um ambiente de teste em um simulador VHDL para criar diferentes cenários de falha de energia. Eles verificaram se o estado do flip-flop era mantido corretamente e como o sistema reagia quando a energia era restaurada.

Discussindo o impacto deste trabalho, Ana percebeu como um design cuidadoso pode aumentar significativamente a confiabilidade dos sistemas digitais em ambientes industriais, onde máquinas precisam de alta disponibilidade e resiliência. Carlos enfatizou a importância de testes rigorosos e de pensar proativamente em cenários de falha durante a fase de design.

Ao trabalhar juntos, Ana e Carlos não só resolveram o problema técnico com o uso de VHDL, mas também prepararam Ana para pensar como uma engenheira que antecipa e resolve problemas antes que eles ocorram em operações críticas. Esta atividade serve como um exemplo valioso de como conhecimentos teóricos podem ser aplicados para desenvolver soluções práticas e eficazes em engenharia eletrônica.

Saiba mais

Para complementar seu estudo de linguagem VHDL e suas aplicações no design de circuitos digitais, recomendamos a seguinte apostila, que oferece uma abordagem prática e detalhada:

["Apostila com Códigos de Programas Demonstrativos usando a linguagem VHDL para Circuitos Digitais"](#) por Alexandre Santos de la Vega (Universidade Federal Fluminense).

Esta apostila contém uma série de exemplos práticos e códigos demonstrativos em VHDL, tornando-a um recurso valioso para entender melhor a aplicação da linguagem na modelagem e simulação de circuitos digitais. O material abrange desde conceitos básicos até tópicos mais avançados, com foco prático que se estende da página 105 até a página 121.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Referências

- RUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL.** Rio de Janeiro: Alta Books, 2022.
- IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital.** 42. ed. São Paulo: Érica, 2019.
- LENZ, M. L.; TORRES, F. E. **Microprocessadores.** Porto Alegre: SAGAH, 2019.
- SOUZA, D. B. da C. et al. **Sistemas digitais.** Porto Alegre: SER – SAGAH, 2018.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta videoaula de encerramento, vamos revisar e integrar todo o conhecimento adquirido a respeito de circuitos digitais e ad linguagem VHDL. Desde o entendimento de circuitos combinacionais e seus blocos funcionais, como multiplexadores e decodificadores, até a prática com circuitos sequenciais que empregam flip-flops e registradores, esta aula é uma oportunidade para consolidar sua compreensão e habilidades em design de hardware digital. Participe conosco nesta última etapa da unidade e assegure-se de que está pronto para não apenas entender, mas também moldar o futuro da tecnologia digital.

Ponto de Chegada

Programação VHDL

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

Durante as aulas você estudou os conceitos do complexo mundo dos circuitos combinacionais e sequenciais, e a linguagem VHDL. Esta unidade consolidou seu entendimento acerca da teoria dos circuitos digitais e o capacitou com habilidades práticas para modelar e simular esses sistemas.

Você mergulhou nos conceitos básicos da linguagem VHDL, uma ferramenta poderosa para a descrição de hardware digital. Você conheceu a estrutura e as regras sintáticas essenciais que são a base para criar designs de circuitos precisos e eficientes. Esta habilidade é crucial para desenvolver projetos que requerem alta confiabilidade e desempenho, como encontrados em sistemas de comunicação e processadores.

Também foram abordados os conceitos de construção e aplicação de circuitos combinacionais, e você explorou os blocos construtivos como multiplexadores, demultiplexadores, decodificadores e somadores. Esses elementos são vitais para a implementação de lógica complexa em sistemas digitais, permitindo a execução eficiente de operações lógicas e aritméticas fundamentais que formam a base para a computação moderna.

Por fim, você colocou a teoria em prática com circuitos sequenciais, usando flip-flops, registradores e máquinas de estados. Essa experiência prática demonstrou como os dados são armazenados, transferidos e manipulados dentro de dispositivos eletrônicos, equipando você com o conhecimento para projetar sistemas que não apenas processam informações rapidamente, mas também mantêm a integridade dos dados ao longo do tempo.

Com estas habilidades, você está agora mais preparado para enfrentar desafios no campo da eletrônica e da computação. Seu conhecimento aprofundado em VHDL e circuitos digitais abre caminhos para inovações em design de hardware e desenvolvimento de novas tecnologias. À medida que avança, continue a aplicar e expandir seu conhecimento, explorando como cada componente pode ser otimizado para criar soluções mais eficientes e sustentáveis em tecnologia.

É Hora de Praticar!

Você é um estudante avançado de engenharia eletrônica, e está desenvolvendo um sistema de controle para uma linha de produção automatizada em uma fábrica de componentes eletrônicos. O desafio principal é garantir que os diferentes módulos da linha de produção operem de maneira sincronizada e eficiente, evitando erros de temporização que poderiam resultar em defeitos nos produtos finais. Para isso, Ana precisa implementar um sistema de controle robusto que utilize flip-flops e registradores para armazenar e sincronizar os dados de operação de cada módulo. Você precisa desenvolver um sistema digital utilizando VHDL que permita:

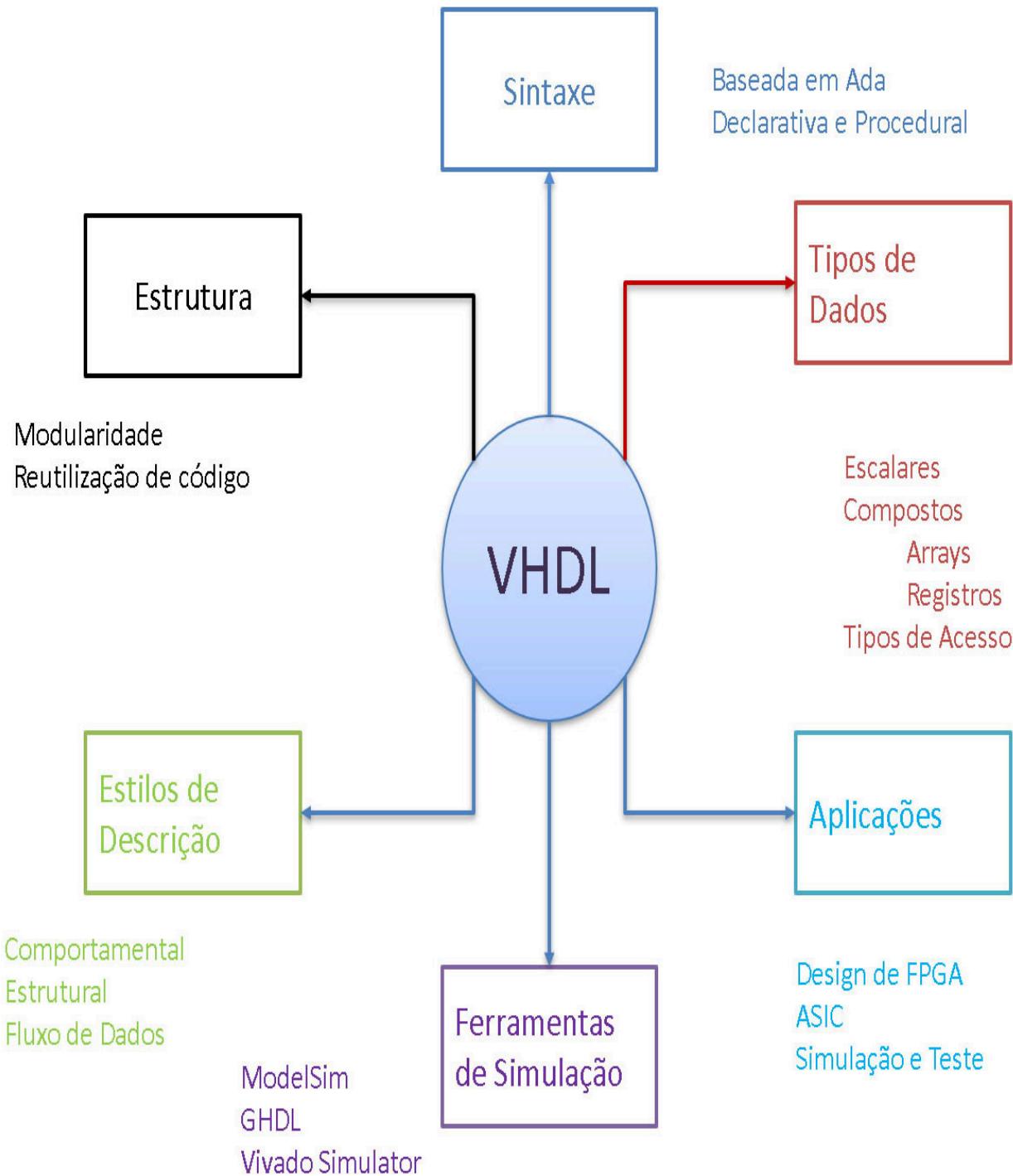
SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES

- **Armazenar e sincronizar dados:** Utilizar registradores para armazenar temporariamente os dados operacionais de cada módulo da linha de produção.
- **Controle de Estados:** Implementar máquinas de estados para gerenciar a operação de cada módulo e garantir a transição correta entre estados.
- **Sincronização de Temporização:** Garantir que todos os módulos operem em sincronia utilizando flip-flops para estabilizar e sincronizar os sinais de temporização.
- Como garantir que os dados armazenados nos registradores sejam atualizados de maneira síncrona com o *clock* da linha de produção?
- Qual a melhor maneira de implementar uma máquina de estados em VHDL para controlar a operação dos módulos da linha de produção?
- Como os flip-flops podem ser utilizados para garantir a estabilidade e a sincronização dos sinais de temporização?

Olá estudante, chegamos ao encerramento da unidade!

Vamos realizar a experiência presencial que irá consolidar os conhecimentos adquiridos? É a oportunidade perfeita para aplicar, na prática, o que foi aprendido em sua disciplina. Vamos transformar teoria em vivência e tornar esta etapa ainda mais significativa. Não perca essa chance única de colocar em prática o conhecimento adquirido.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



CRUZ, E. et al. **Sistemas Digitais Reconfiguráveis: FPGA e VHDL**. Rio de Janeiro: Alta Books, 2022.
IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital**. 42. ed. São Paulo: Érica, 2019.

SISTEMAS DIGITAIS E MICROPROCESSADOS - PRINCÍPIOS E APLICAÇÕES



HEXSEL, R. A. **Sistemas Digitais e Microprocessadores**. Departamento de Informática, Universidade Federal do Paraná, 2006.

LENZ, M. L.; TORRES, F. E. **Microprocessadores**. Porto Alegre: SAGAH, 2019.

SOUZA, D. B. da C. *et al.* **Sistemas digitais**. Porto Alegre: SER – SAGAH, 2018.

TOCCI, R. J.; WIDMER, N. S. **Sistemas Digitais – princípios e aplicações**. 11. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2011.