UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

Luciano dos Santos
Wesley Silva Araújo

LABORATÓRIO DE CIRCUITOS – CODIFICAÇÃO E SIMULAÇÕES

BOA VISTA-RR
2024

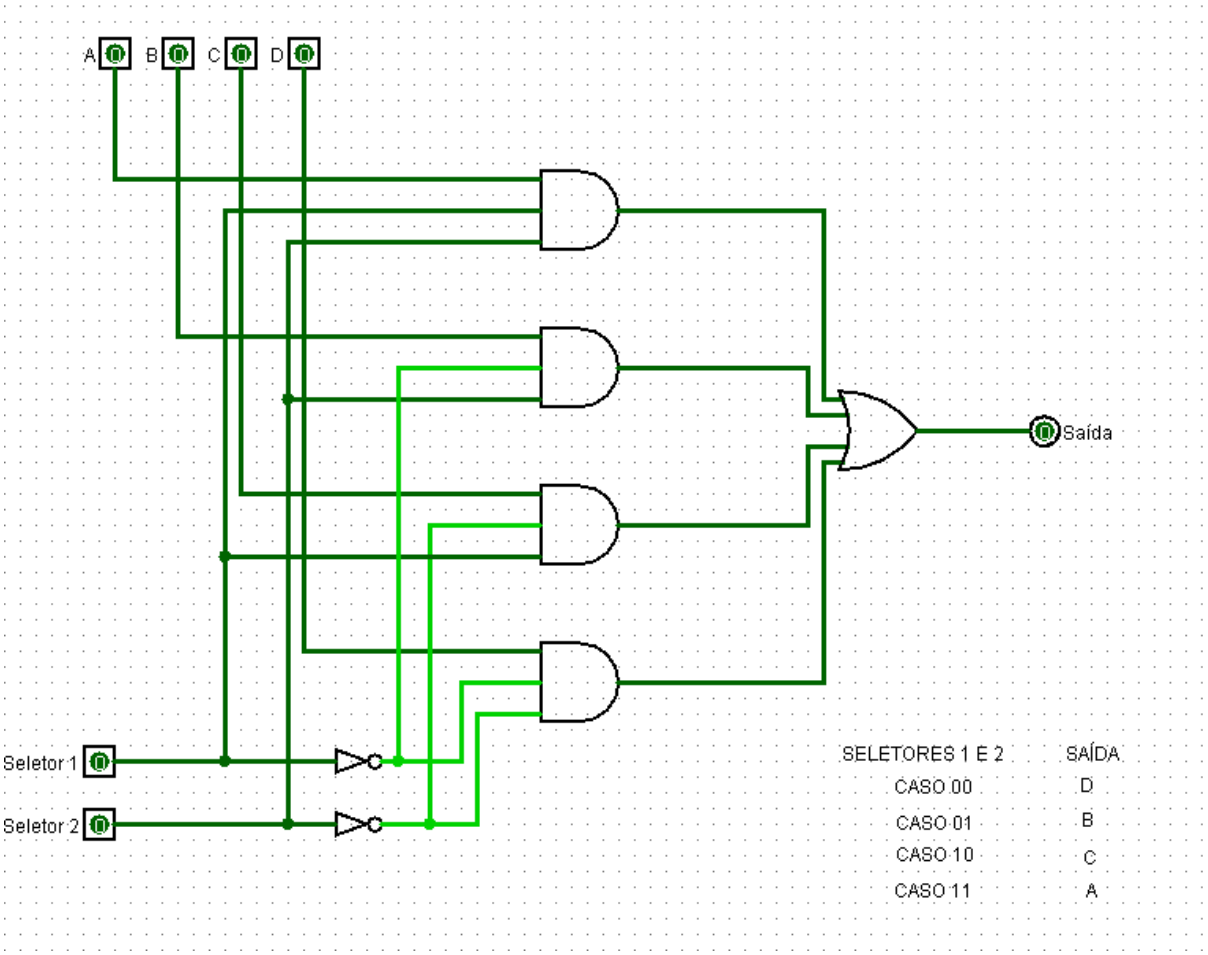
LABORATÓRIO DE CIRCUITOS – CODIFICAÇÃO E SIMULAÇÕES

Relatório Científico apresentado ao Prof. Dr. Herbert Oliveira Rocha como requisito para obtenção de nota parcial na disciplina DCC 301 - Arquitetura e Organização de Computadores, oferecida pelo Departamento de Ciência da Computação da Universidade Federal de Roraima. Os componentes descritos a seguir foram projetados e simulados por meio do software Logisim, conforme apresentado em sala de aula.

BOA VISTA-RR
2024

Multiplexador com 4 opções de entrada:

O multiplexador 4x1 é um circuito digital que funciona como uma chave seletora, escolhendo uma entre quatro entradas (A, B, C e D) para ser encaminhada para a saída (S). Essa seleção é feita através de dois pinos seletores, que atuam como endereços para cada entrada.
Imagine o multiplexador como um interruptor com quatro posições. Cada posição corresponde a uma entrada, e os seletores controlam qual posição está ativa. A saída S sempre refletirá o valor da entrada que estiver selecionada no momento.

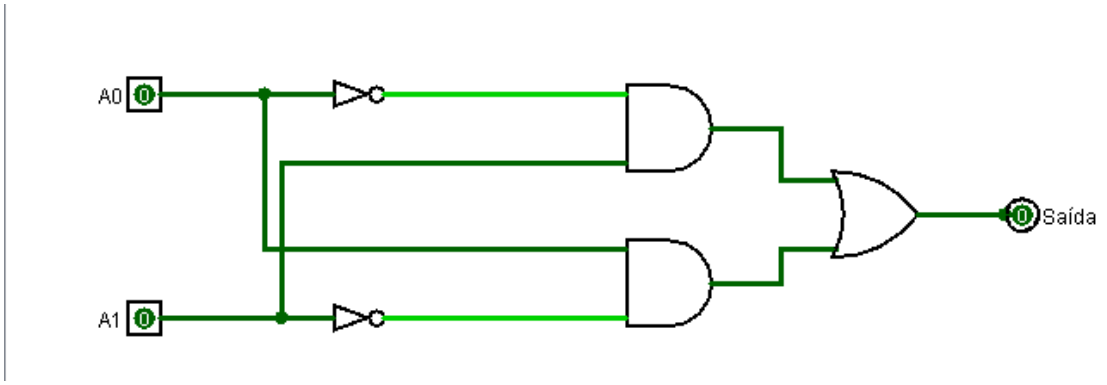


Para entender melhor seu funcionamento, pense em cada entrada conectada a uma porta AND. Os seletores controlam qual dessas portas AND será ativada. Quando uma porta AND é ativada, o valor da entrada correspondente é passado diretamente para a saída S.

Seletor1	Seletor2	A	B	C	D	S
0	0	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	0	0	0	0	0
0	0	0	0	0	0	0
1	0	0	0	0	0	0
1	0	0	0	1	0	1
1	0	0	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

Porta XOR:

Essa é a porta lógica XOR (OU Exclusivo), que utiliza 2 componentes do tipo AND, 1 componente do tipo OR e 2 NOT. O resultado (S) é determinado pelas



entradas A0 e A1. A tabela verdade do XOR é apresentada abaixo:

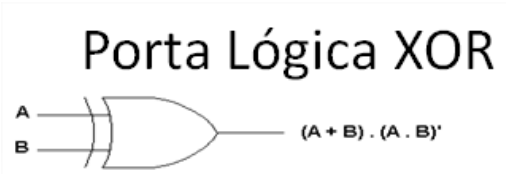


Tabela da Verdade

A	B	S= (A⊕B)
0	0	0
0	1	1
1	0	1
1	1	0

Registrador Flip-Flop do tipo D

Para a construção de um registrador Flip-Flop do tipo D de um bit, será necessário utilizar apenas um Flip-Flop tipo D, o qual é construído com modificação de um Flip-Flop tipo jk mestre e escravo

O Flip- Flop D é um circuito digital usado em sistemas sequenciais para armazenar um único bit de informação. Ele é um dos flip-flops mais simples e amplamente utilizados devido à sua funcionalidade e confiabilidade. As suas características são:

- **Entrada única de dados (D):** O estado armazenado pelo flip-flop é determinado pela entrada DDD no momento em que um sinal de controle (geralmente o clock) é ativado.
- **Sincronização com clock:** O flip-flop do tipo D opera com base em um sinal de clock, transferindo o valor da entrada DDD para a saída QQQ no momento da borda ativa do clock (subida ou descida, dependendo do design).
- **Saída inversa (Q′Q′Q′):** Além da saída normal (QQQ), ele geralmente fornece a saída complementada (Q′Q′Q′).

Logo, percebemos que um registrador de um bit do Flip-Flop tipo D é o próprio Flip-Flop do tipo D. A imagem logo a baixo é uma demostração de um circuito Flip-Flop do tipo D, construído no Logisim, esse circuito usa 8 portas NAND, um Clock, duas entradas e duas saidas.

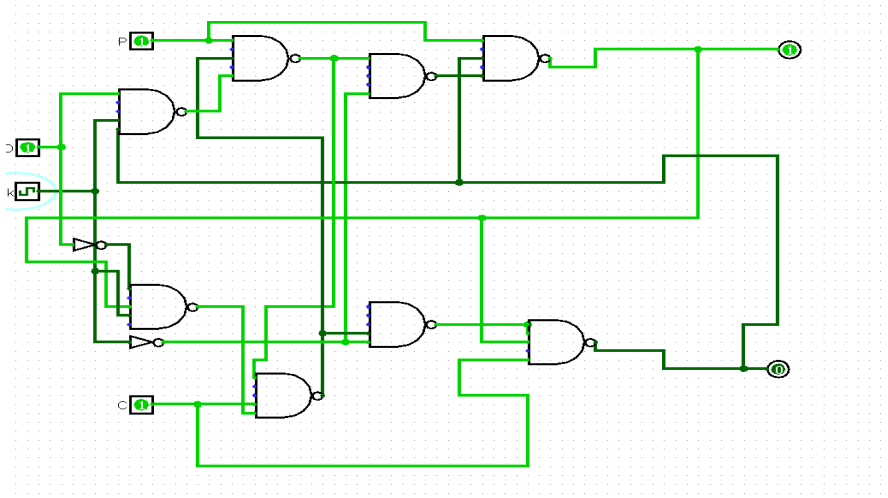
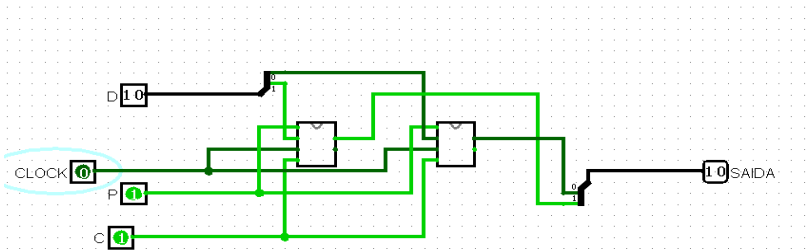


Imagem de um Flip-Flop do tipo D

Tabela Verdade:

Entrada D	Saída Q após o clock
0	0
1	1

Para a construção de um de dois Bits, apenas conectamos uma entrada de dois bits de saída a entradas Dsdois Flip-Flops tipo D.



Registrador Flip-Flop JK

O flip-flop JK é um tipo de circuito digital usado em sistemas sequenciais para armazenar informações em um bit. Ele é uma extensão do flip-flop RS, mas com resolução de condições instáveis. Suas principais características incluem:

Entradas e Saídas:

- **J (Set):** Define a saída como "1".
- **K (Reset):** Define a saída como "0".
- **Clock:** Sincroniza as mudanças do estado.
- **Saída Q e Q' (complemento de Q):** Representam o estado armazenado.

Tabela Verdade:

J	K	Qn(estado atual)	Qn+1próximo estado
0	0	Qn	Sem alteração
0	1	Qn	0
1	0	Qn	1
1	1	Qn	Complemento de Qn

Características:

- **Condição de Memória:** Quando J = 0 e K = 0, o estado permanece inalterado.
- **Set e Reset:** Quando J ou K é ativado exclusivamente, ele seta ou reseta o flip-flop.
- **Toggle:** Quando J = 1 e K = 1, o estado muda para o oposto do atual (complementa Q).

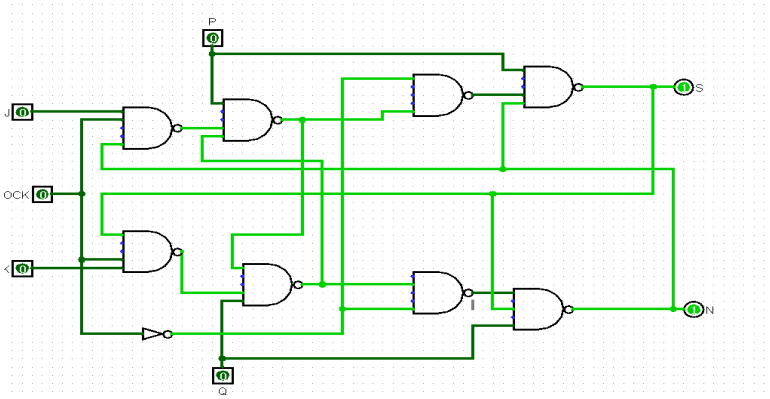
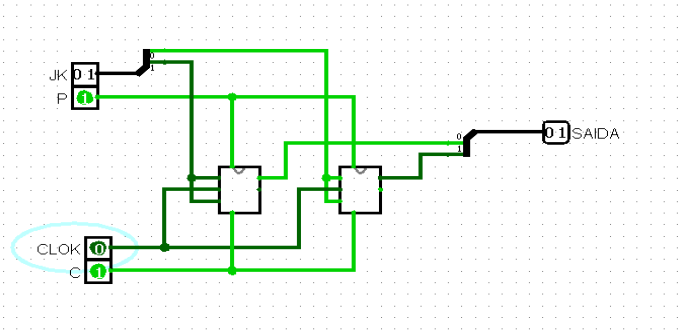


Imagem de um Jk mestre e escravo

O registrador de dois bits tem poucas variações, pois apenas vamos conectar uma entrada com saida de dois bits as entras J e K dos Flip-Flop, com no exemplo:



Flip-Flop Jk Mestre e Escravo

Somador de 8 Bits que recebe 4

Introdução

Este circuito simula um somador de 8 bits que recebe um valor binário inteiro como entrada e soma o valor constante 4. O circuito foi projetado utilizando o software Logisim, com o objetivo de demonstrar o funcionamento básico de uma operação aritmética em hardware digital.

Descrição do Circuito

1. **Entradas**:
    - **A[7:0]**: Representa os 8 bits da entrada principal.
    - **Cin**: Carry in (entrada de transporte), configurado como 0 para esta operação.
2. **Saída**:
    - **S[7:0]**: Saída de 8 bits que representa o resultado da soma A+4A + 4.
    - **Estouro (Carry Out)**: Indica se houve *overflow* (estouro de valor) na operação aritmética.
3. **Operação**:
    - A entrada binária AA é somada ao valor fixo 4 (00000100 em binário).
    - O circuito utiliza um somador completo (*full adder*) interno para realizar a soma bit a bit, levando em conta o *carry* entre os bits.

Funcionamento do Circuito

- O valor de entrada AA é alimentado no somador de 8 bits.

- O somador é projetado para somar diretamente a constante 44, que é conectada às linhas correspondentes de entrada (00000100).
- A saída S[7:0]S[7:0] exibe o resultado da soma.
- O sinal de *estouro* verifica se a operação excedeu os 8 bits disponíveis.

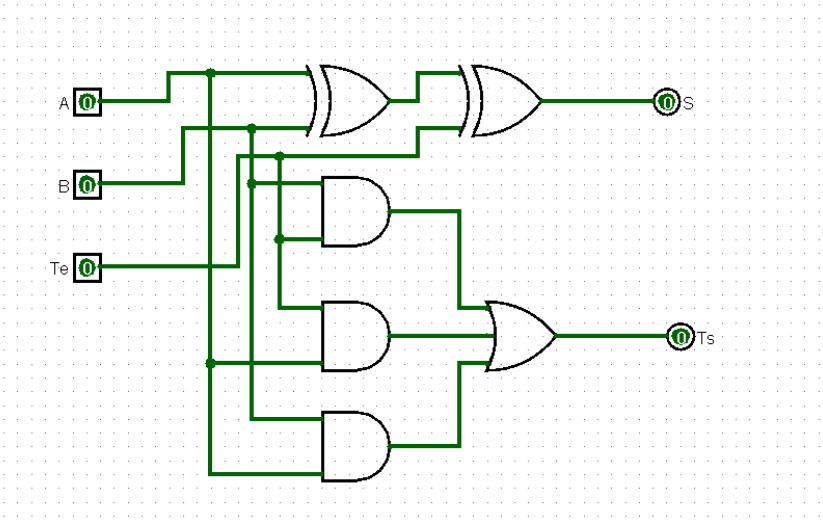
Configuração Específica

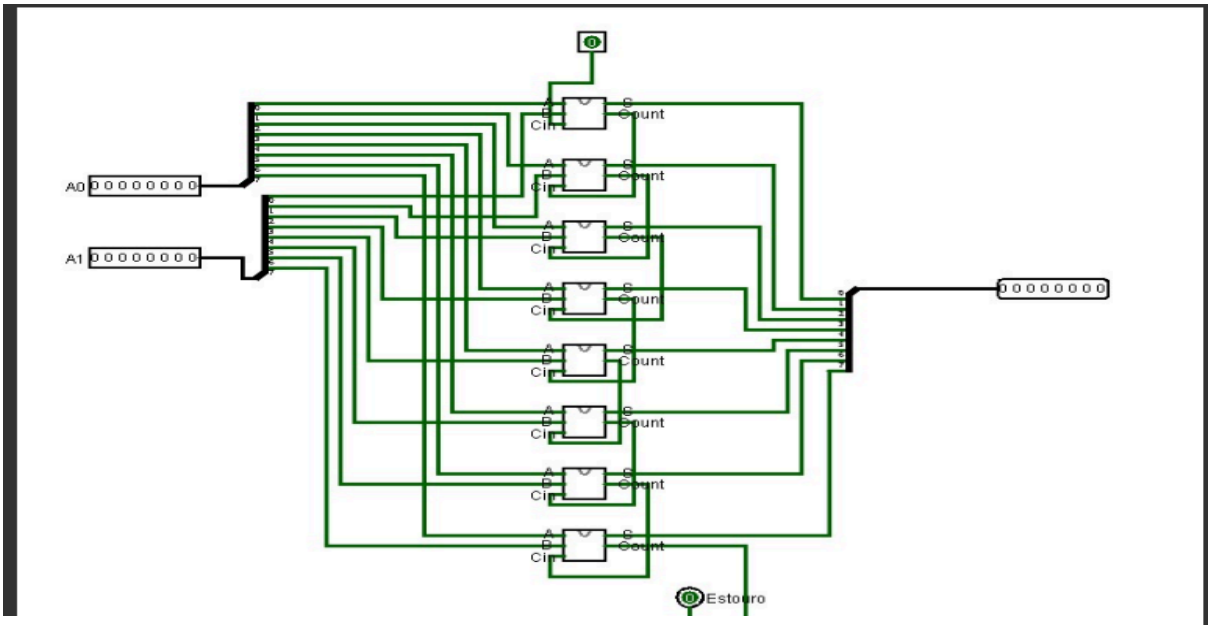
- **Entrada Demonstrada (A[7:0]):** 00000000
- **Soma Efetuada:** 00000000+0000010000000000 + 00000100
- **Saída (S[7:0]):** 00000100
- **Estouro:** 0 (não houve overflow, pois o valor resultante está dentro do intervalo de 8 bits sem sinal).

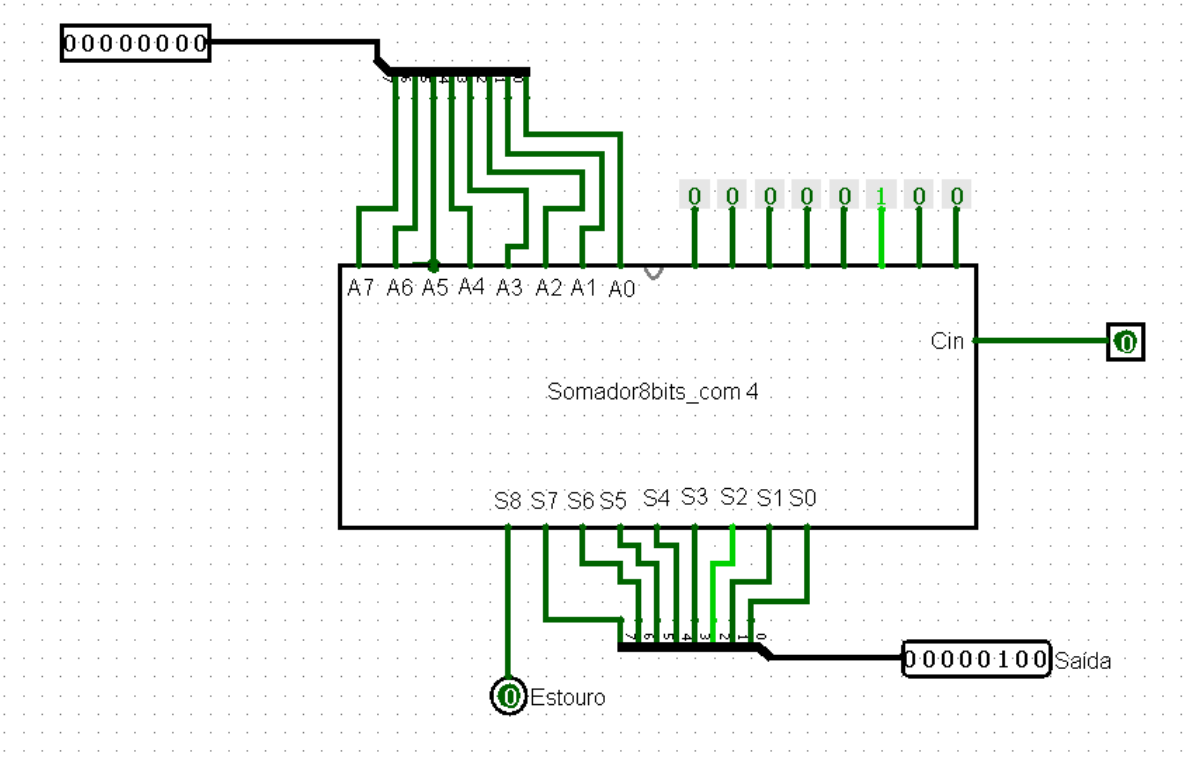
Testes Realizados

1. **Entrada: 00000000**
    - Saída esperada: 00000100
    - Resultado obtido: 00000100
    - Estouro: 0
2. **Entrada: 11111111**
    - Saída esperada: 00000011 (com *carry out* 1 devido ao overflow)
    - Resultado obtido: 00000011
    - Estouro: 1
3. **Entrada: 01111111**
    - Saída esperada: 10000011
    - Resultado obtido: 10000011
    - Estouro: 0

Imagens da montagem do circuito no logisim:







Conclusão

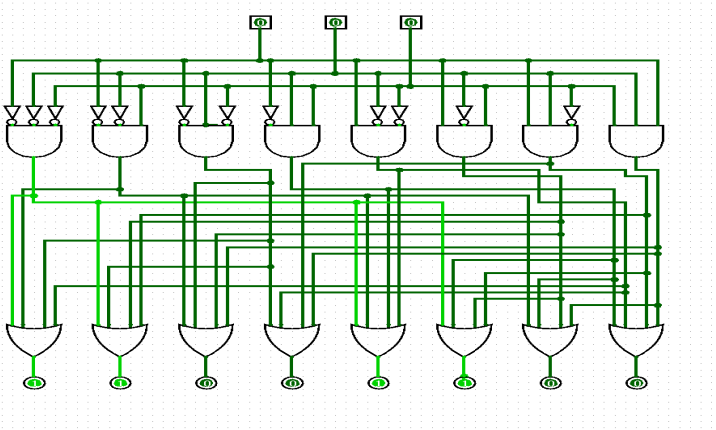
O circuito foi projetado e testado com sucesso. Ele realiza a operação de soma entre uma entrada binária de 8 bits e a constante 4, exibindo o resultado e indicando quando há ocorrência de *overflow*. O somador pode ser aplicado em sistemas digitais para operações aritméticas básicas e demonstra a integração entre lógica combinacional e sistemas de transporte (*carry*).

Memória ROM de 8 bits

Uma Memória de Somente Leitura (ROM) é um tipo de memória não volátil, ou seja, ela mantém os dados armazenados mesmo quando a energia é desligada. A ROM é ideal para armazenar informações que não precisam ser alteradas com frequência, como o código de boot de um computador ou tabelas de referência.

Na construção do circuito, foram usadas 3 entradas que são direcionadas para 8 portas end, depois dessa ligação, as saidas das portas ends são conectadas as entradas de 8 portas or e as saidas das or são saidas de um bit cada.

A



Como pode ser observado, algumas conexões foram negadas antes de chegar na porta end

Tabela de resultados:

Endereço	Saída
000	11001100
001	10101010
010	11110000
011	00001111
100	10011001
101	01100110
110	01010101
111	00110011

a	b	c	x	y	z	u	v	w	s	t
0	0	0	1	1	0	0	1	1	0	0
0	0	1	1	0	1	0	1	0	1	0
0	1	0	1	1	1	1	0	0	0	0
0	1	1	0	0	0	0	1	1	1	1
1	0	0	1	0	0	1	1	0	0	1
1	0	1	0	1	1	0	0	1	1	0
1	1	0	0	1	0	1	0	1	0	1
1	1	1	0	0	1	1	0	0	1	1

Memória RAM de 8 bits.

A ideia central é permitir a leitura e escrita de dados em diferentes endereços de memória, selecionados por um conjunto de linhas de endereço.

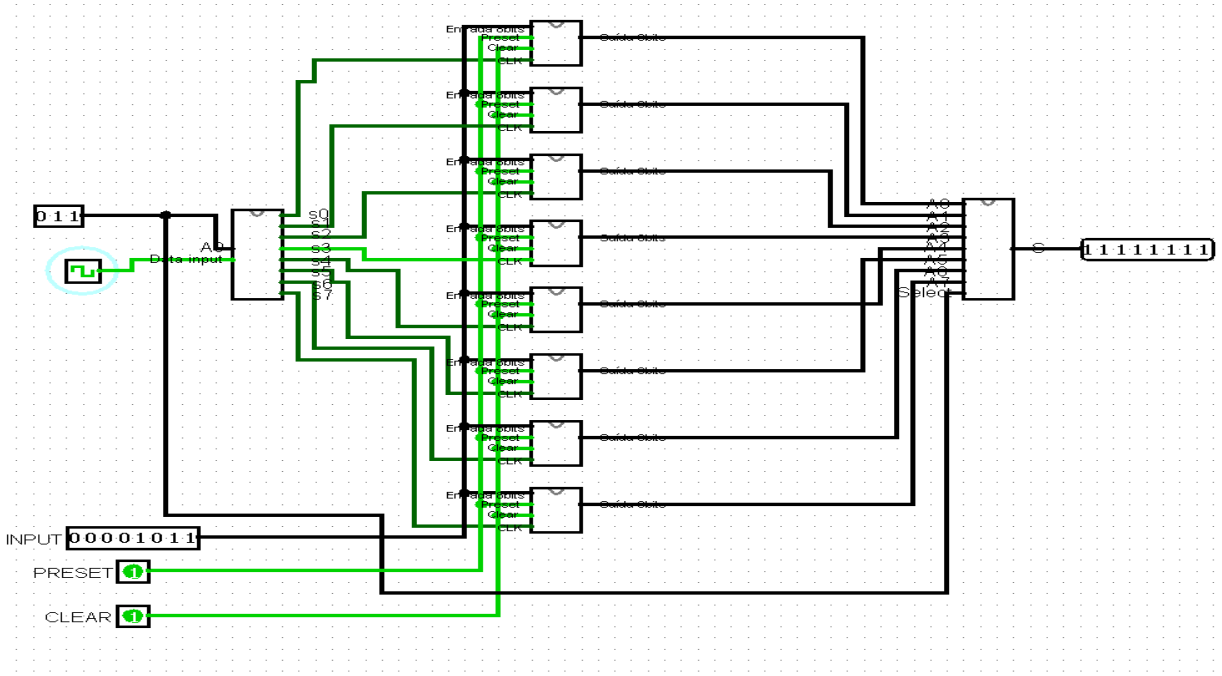
Componentes e Funcionalidades:

- **8 Registradores de 8 bits:** Cada registrador armazena um byte de dados. Eles servem como as células de memória individuais.
- **Demultiplexador 8x1:** Esse componente seleciona um dos 8 registradores com base nas linhas de endereço. A saída do demultiplexador é conectada à entrada de dados do multiplexador.
- **Multiplexador 8x1:** Esse componente seleciona a fonte de dados para escrita na memória. Ele pode receber dados de um barramento de dados externo ou do próprio circuito.
- **Linhas de Endereço:** Um conjunto de linhas determina qual registrador será acessado para leitura ou escrita.
- **Sinais de Controle:** Os sinais de controle, como enable (En), preset e clear, controlam as operações de leitura, escrita e inicialização da memória.

Funcionamento:

1. **Seleção do Endereço:** As linhas de endereço determinam qual dos 8 registradores será acessado. O demultiplexador direciona a saída do registrador selecionado para o multiplexador.
2. **Leitura:** Quando a operação é de leitura, a saída do multiplexador é conectada ao barramento de dados, permitindo que o conteúdo do registrador selecionado seja lido.

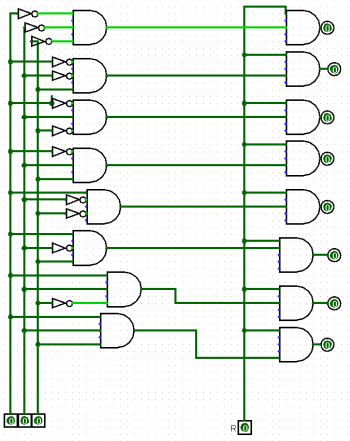
3. **Escrita:** Para escrever um novo valor na memória, o dado a ser escrito é conectado à entrada do multiplexador. O multiplexador direciona esse dado para o registrador selecionado pelo demultiplexador. O sinal de enable e outros sinais de controle podem ser necessários para iniciar a operação de escrita.
4. **Preset e Clear:** Os sinais preset e clear podem ser usados para inicializar os registradores com valores específicos.

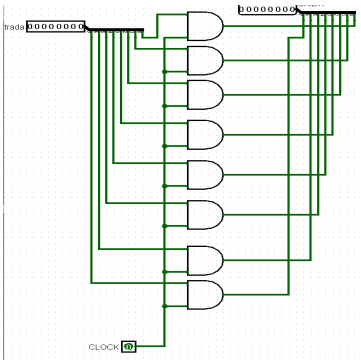


Banco de Registradores de 8 bits

A construção foi feita com um demultiplexador 8 por 1 e 8 registradores. Através do demultiplexador e selecionado qual registrador salvara a informação.

Demultiplexador e registrador, repectivamente:

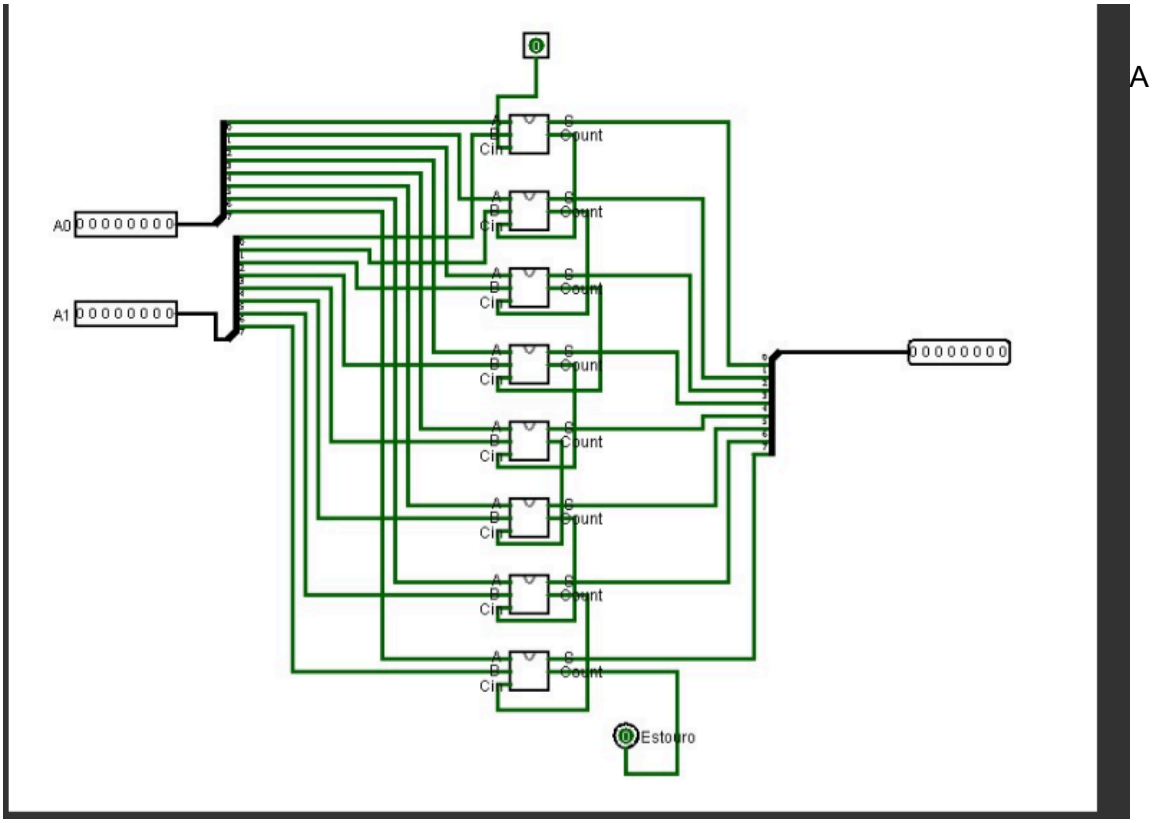




depois das conexões, atribuímos uma saida para cada registrador

Somador de 8 Bits

Introdução



A estrutura básica consiste em uma cadeia de somadores completos de 1 bit, conectados em cascata para formar um somador de 8 bits.

Componentes e Funcionamento

- **Somador Completo de 1 bit:** Cada bloco retangular representa um somador completo. Ele possui três entradas (A, B e Cin - carry in) e duas saídas (S - soma e Cout - carry out). O somador completo realiza a adição de dois bits e do bit de carry proveniente da adição anterior.
- **Conexão em Cascata:** Os somadores completos são conectados em série, de forma que o bit de carry (Cout) de um somador se torna o bit de carry de entrada (Cin) do próximo somador. Essa configuração permite a propagação do carry ao longo de todos os bits, garantindo a adição correta de números de múltiplos bits.
- **Entradas e Saídas:**
    - **A e B:** Representam os dois números binários de 8 bits a serem somados. Cada bit de A é conectado à entrada A de um somador, e cada bit de B à entrada B correspondente.

- **Cin:** É o bit de carry de entrada do somador menos significativo. Geralmente, é inicializado com 0 para adições simples.
- **S:** Representa a soma resultante da operação. Os bits de S formam o número binário de 8 bits que corresponde à soma de A e B.
- **Cout:** É o bit de carry de saída do somador mais significativo. Ele indica se houve um overflow, ou seja, se o resultado da adição excedeu 8 bits.

Funcionamento Detalhado

1. **Entrada de Dados:** Os números binários A e B são aplicados às entradas correspondentes de cada somador completo.
2. **Adição de Bits:** Cada somador completo realiza a adição de um par de bits (um de A e um de B) e do bit de carry proveniente do somador anterior.
3. **Propagação do Carry:** O bit de carry (Cout) gerado por cada somador é propagado para o próximo somador, permitindo que a adição se propague por todos os bits.
4. **Saída:** A soma final é obtida nos bits de saída S de cada somador. O bit de carry de saída (Cout) indica se houve um overflow.

Vantagens da Arquitetura

- **Modularidade:** A estrutura modular do somador de 8 bits permite a fácil expansão para números de maior quantidade de bits.
- **Simplicidade:** Cada somador completo realiza uma operação simples e bem definida, facilitando a compreensão e a implementação do circuito.
- **Flexibilidade:** O somador pode ser utilizado em diversas aplicações que envolvam a adição de números binários.

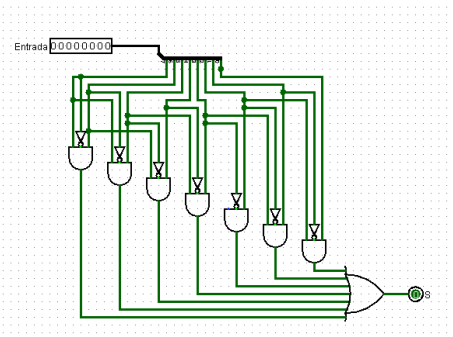
Considerações Adicionais

- **Tempo de Propagação:** O tempo necessário para que a adição seja completada depende do tempo de propagação dos sinais através dos somadores completos e das conexões entre eles.
- **Otimizações:** Existem diversas técnicas para otimizar o desempenho do somador, como o uso de somadores mais rápidos ou a redução do número de portas lógicas.
- **Aplicações:** Somadores de 8 bits são utilizados em uma ampla variedade de circuitos digitais, como unidades lógicas aritméticas (ULAs), processadores e sistemas de controle.

Conclusão

O somador de 8 bits apresentado é um exemplo clássico de circuito digital combinacional. Sua estrutura modular e funcionamento simples o tornam uma ferramenta fundamental para a construção de sistemas digitais mais complexos. A compreensão do funcionamento desse circuito é essencial para estudantes e profissionais da área de eletrônica digital

Detector de Sequência Binária "101"



Introdução

O circuito digital na imagem, implementado no software Logisim, tem como objetivo principal detectar a ocorrência da sequência binária "101" em um fluxo de entrada de bits. Essa funcionalidade é fundamental em diversas aplicações da eletrônica digital, como processamento de sinais, comunicação e controle.

Funcionamento do Circuito

Componentes:

- **Portas Lógicas:** O circuito é composto por portas lógicas básicas, como AND e OR, que realizam operações lógicas sobre os sinais de entrada.
- **Flip-Flops:** Os flip-flops são elementos de memória que armazenam o estado atual do circuito, permitindo a detecção de sequências de bits ao longo do tempo.

Funcionamento:

1. **Entrada de Dados:** Os bits de entrada são aplicados sequencialmente às portas lógicas.
2. **Detecção de Subsequências:** O circuito é projetado para detectar as subsequências "1" e "01" da sequência "101". Cada porta AND detecta a ocorrência de uma dessas subsequências.
3. **Armazenamento de Estado:** Os flip-flops armazenam o estado atual da detecção, indicando se a subsequência anterior foi encontrada.
4. **Detecção Final:** A porta OR final combina as saídas das portas AND e dos flip-flops, gerando um sinal de saída alto (1) quando a sequência completa "101" é detectada.

Análise Detalhada

Funcionamento dos Flip-Flops:

- **Flip-Flop 1:** Armazena a informação se o último bit de entrada foi "1".
- **Flip-Flop 2:** Armazena a informação se os dois últimos bits de entrada foram "10".

Detecção da Sequência:

- **Primeira Subsequência ("1"):** A porta AND 1 detecta quando o bit de entrada atual é "1".
- **Segunda Subsequência ("01"):** A porta AND 2 detecta quando o bit de entrada atual é "0" e o bit anterior (armazenado no flip-flop 1) era "1".
- **Sequência Completa ("101"):** A porta OR final gera um sinal de saída alto quando a porta AND 1 está ativa (bit atual é "1") e o flip-flop 2 está ativo (os dois bits anteriores eram "10").

Vantagens e Limitações

Vantagens:

- **Simplicidade:** O circuito utiliza um número limitado de componentes e é relativamente fácil de entender.
- **Flexibilidade:** Pode ser adaptado para detectar outras sequências binárias, ajustando as conexões entre as portas lógicas e os flip-flops.

- **Velocidade:** A velocidade de detecção depende da velocidade dos componentes utilizados.

Limitações:

- **Detecção de uma única sequência:** O circuito é projetado para detectar apenas a sequência "101". Para detectar outras sequências, seria necessário modificar a lógica.
- **Sensibilidade a ruídos:** O circuito pode ser sensível a ruídos na entrada, o que pode levar a falsas detecções.
- **Complexidade para sequências mais longas:** A complexidade do circuito aumenta com o comprimento da sequência a ser detectada.

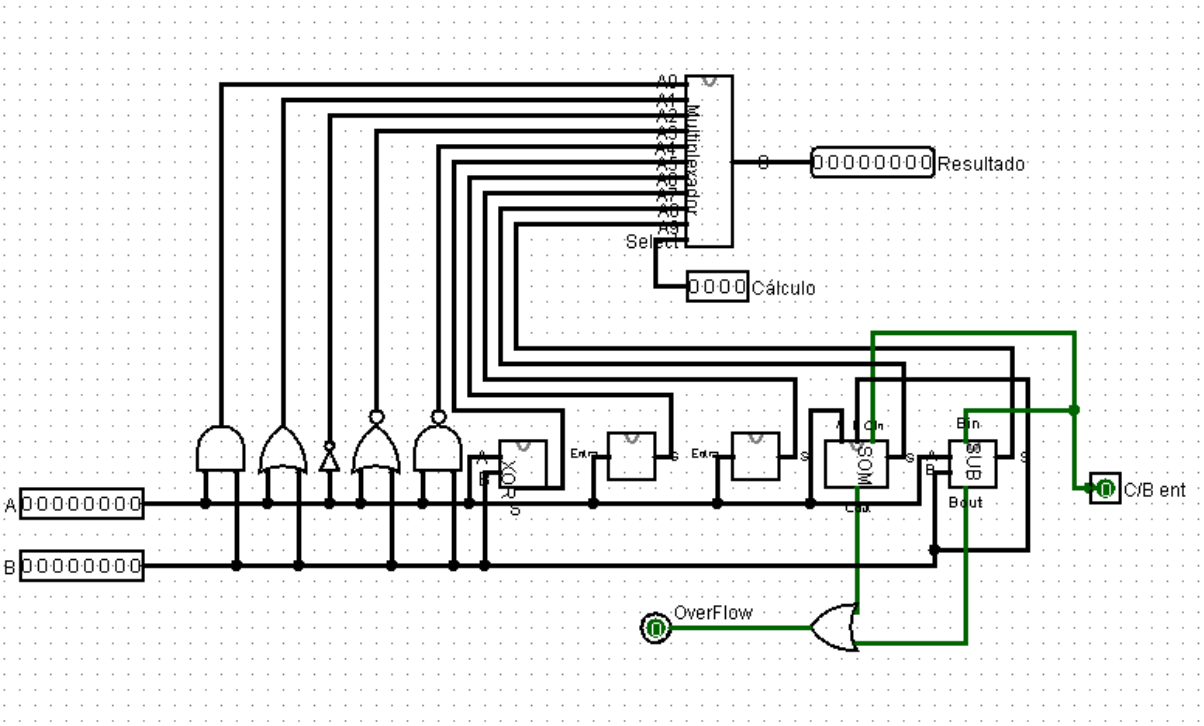
Aplicações

- **Sincronização:** Detectar padrões específicos em um fluxo de dados para sincronizar dispositivos.
- **Comunicação:** Identificar códigos de início e fim de pacotes de dados.
- **Processamento de sinais:** Detectar transições ou padrões específicos em sinais digitais.
- **Controle:** Implementar sistemas de controle baseados em sequências de eventos.

Conclusões

O detector de sequência binária "101" apresentado é um exemplo simples e eficaz de circuito digital combinacional e sequencial. Sua compreensão é fundamental para o projeto de sistemas digitais mais complexos.

Unidade Lógica Aritmética (ULA) de 8 bits



Introdução

A imagem apresenta uma implementação de uma Unidade Lógica Aritmética (ULA) de 8 bits, projetada para realizar diversas operações aritméticas e lógicas sobre dois operandos de 8 bits. A ULA é um componente fundamental em processadores e outros circuitos digitais, sendo responsável por executar as instruções aritméticas e lógicas de um programa.

Componentes e Funcionalidades

A ULA apresentada é composta por diversos componentes interconectados, cada um com uma função específica:

- **Somadores e Subtratores:** Realizam as operações de adição e subtração entre os operandos de 8 bits.
- **Multiplexadores:** Selecionam os dados a serem processados com base em um sinal de controle.
- **Portas Lógicas:** Realizam operações lógicas como AND, OR, NOT e XOR.
- **Flip-flops:** Armazenam o resultado das operações e controlam o fluxo de dados.
- **Decodificadores:** Convertem um código binário em um sinal de controle para selecionar diferentes operações.

Funcionamento Geral

A ULA funciona da seguinte forma:

1. **Entrada de Dados:** Dois operandos de 8 bits (A e B) são fornecidos como entrada para a ULA.
2. **Seleção da Operação:** Um conjunto de sinais de controle determina a operação a ser realizada (adição, subtração, etc.). Esses sinais são geralmente gerados por um decodificador, que interpreta um código de operação.
3. **Processamento:** Os multiplexadores direcionam os operandos para os componentes responsáveis pela operação selecionada. Por exemplo, para uma adição, os operandos são enviados para os somadores.
4. **Saída:** O resultado da operação é gerado e pode ser armazenado em um registrador ou utilizado como entrada para outras partes do circuito.

Análise Detalhada

Componentes-chave:

- **Somador de 8 bits:** Realiza a adição de dois números binários de 8 bits.
- **Subtrator de 8 bits:** Realiza a subtração de dois números binários de 8 bits, geralmente implementando a subtração através da adição do complemento de dois.
- **Multiplexador de 8 bits:** Seleciona qual dos dois operandos será utilizado como entrada para uma determinada operação.
- **Decodificador:** Interpreta o código de operação e gera os sinais de controle necessários para selecionar os multiplexadores e os componentes aritméticos.

Operações Possíveis:

A ULA pode realizar diversas operações, como:

- **Adição:** Soma os dois operandos.
- **Subtração:** Subtrai o segundo operando do primeiro.
- **Complemento de dois:** Calcula o complemento de dois de um número.
- **Operações lógicas:** AND, OR, XOR, NOT.
- **Incremento:** Adiciona 1 ao operando.
- **Decremento:** Subtrai 1 do operando.

Sinais de Controle:

Os sinais de controle determinam qual operação será realizada. Eles podem ser gerados por um microcontrolador ou por um circuito de controle específico.

Aplicações

A ULA é um componente fundamental em diversos sistemas digitais, incluindo:

- **Microprocessadores:** Realizam as operações aritméticas e lógicas básicas.
- **Microcontroladores:** Controlam dispositivos eletrônicos, realizando cálculos e tomadas de decisão.
- **FPGAs:** Podem ser configuradas para implementar diversas funções, incluindo ULAs.
- **Co-processadores:** Aceleram operações matemáticas em sistemas computacionais.

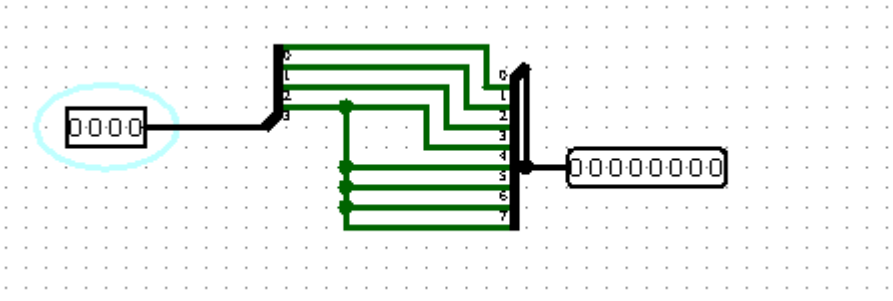
Considerações Adicionais

- **Otimizações:** Existem diversas técnicas para otimizar o desempenho da ULA, como o uso de somadores mais rápidos, a redução do número de portas lógicas e a implementação de pipelines.
- **Extensões:** A ULA pode ser expandida para suportar números de maior precisão, operações mais complexas e outras funcionalidades, como multiplicação e divisão.
- **Implementação:** A ULA pode ser implementada utilizando diferentes tecnologias, como TTL, CMOS, FPGA, etc.

Conclusões

A ULA é um componente essencial em qualquer sistema digital que necessite realizar operações aritméticas e lógicas. A sua complexidade e funcionalidade podem variar de acordo com as necessidades da aplicação. A compreensão do funcionamento da ULA é fundamental para o desenvolvimento de sistemas digitais mais complexos.

Extensor de Sinal de 4 Bits para 8 Bits



Introdução

Um extensor de sinal de 4 bits para 8 bits é um circuito digital projetado para aumentar a largura de um sinal binário de 4 bits para 8 bits. Essa operação é comum em sistemas digitais onde é necessário adaptar a largura de dados entre diferentes componentes.

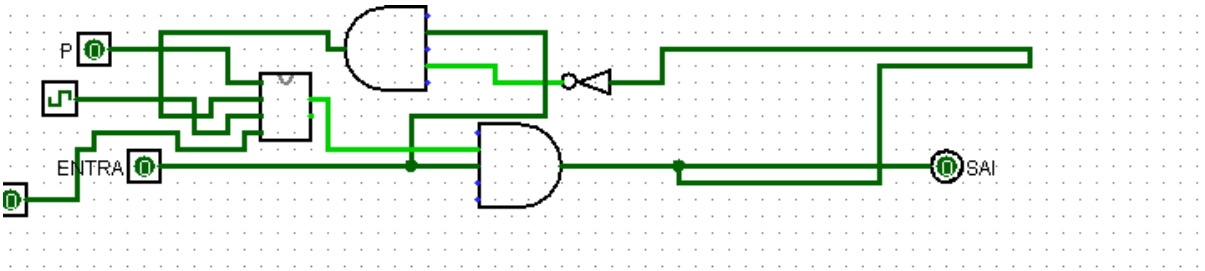
Funcionamento Básico

A principal função de um extensor de sinal é replicar os bits mais significativos do sinal de entrada para os bits mais significativos da saída, enquanto os bits menos significativos da saída são preenchidos com zeros. Essa operação é conhecida como extensão com zeros.

Para fazer ele foram usados um input de quatro bits e um output de 8 bits e utilizamos dois distribuidores de bits.

Maquina de estados

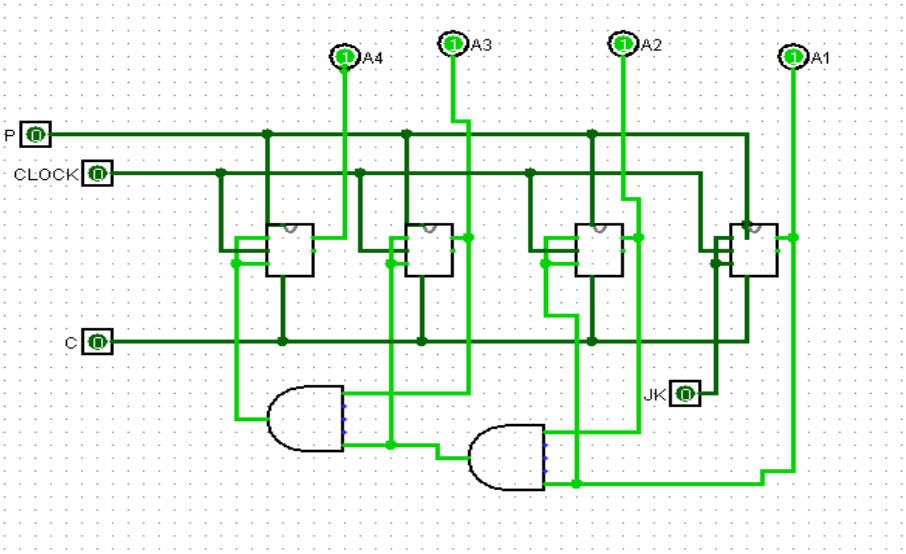
Ela tem uma entrada que passa em uma porta And e a saida invertida passa por outra porta And, nas duas portas Ands presentes tem uma conexão com a entrada. A saida da And que recebe a entrada e a saida são salvas em um flip flop jk.



Sua variação o corre quando a entrada é um e a saida é um ou zero

P	ENTRA	C	SAI
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	!!

Contador Síncrono de 4 Bits com Flip-Flop JK no Logisim



Introdução

Contadores síncronos são circuitos sequenciais digitais utilizados para contar pulsos de um sinal de clock, e o flip-flop JK oferece flexibilidade na implementação de diferentes tipos de contagem.

Objetivo

O objetivo principal deste trabalho é:

- **Implementar:** Um contador síncrono de 4 bits utilizando flip-flops JK no software Logisim.
- **Simular:** O funcionamento do contador para verificar sua funcionalidade e comportamento.
- **Analisar:** O circuito implementado, identificando suas características e limitações.
- **Documentar:** Os resultados obtidos através de diagramas, tabelas de verdade e explicações detalhadas.

Materiais e Métodos

Software: Logisim

Componentes:

- 4 Flip-flops JK
- Portas lógicas (AND, OR, NOT)
- Conectores

Metodologia:

1. **Projeto:** Desenhar o circuito no Logisim, conectando os flip-flops JK e as portas lógicas de acordo com a configuração desejada do contador.
2. **Simulação:** Simular o circuito para verificar seu funcionamento, observando as mudanças de estado dos flip-flops JK em resposta aos pulsos de clock.

3. **Análise:** Analisar os resultados da simulação, comparando-os com a tabela verdade esperada para o contador.
4. **Documentação:** Elaborar um relatório detalhado, incluindo diagramas, tabelas de verdade e explicações sobre o funcionamento do circuito.

Análise dos Resultados:

- **Funcionamento:** O contador funciona corretamente, incrementando seu valor de 0 a 15 a cada pulso de clock. As saídas A1, A2, A3 e A4 representam os bits menos significativo, segundo menos significativo, terceiro menos significativo e mais significativo, respectivamente.
- **Limitações:** O contador implementado é um contador ascendente. Para implementar um contador descendente ou com outras funcionalidades, seria necessário modificar a lógica das entradas J e K dos flip-flops.
- **Flexibilidade:** O flip-flop JK oferece flexibilidade na implementação de diferentes tipos de contagem, permitindo a criação de contadores com diferentes sequências de contagem.

Discussão

A implementação de um contador síncrono de 4 bits utilizando flip-flops JK no Logisim é uma atividade fundamental para a compreensão de circuitos sequenciais digitais. A utilização de flip-flops JK permite criar contadores com diferentes características e funcionalidades, dependendo da configuração das entradas J e K.

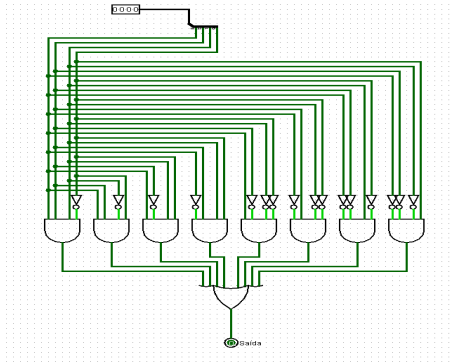
Considerações Finais:

- **Otimização:** É possível otimizar o circuito utilizando técnicas de minimização de portas lógicas, reduzindo o número de componentes e o consumo de energia.
- **Extensões:** O contador pode ser expandido para contar em diferentes bases numéricas (binário, octal, hexadecimal) e implementar diferentes tipos de contagem (ascendente, descendente, com carga).
- **Aplicações:** Contadores síncronos são utilizados em diversas aplicações, como em sistemas de temporização, divisores de frequência, medidores digitais e muitos outros.

Conclusões

A análise dos resultados mostrou que o circuito funciona corretamente, incrementando seu valor a cada pulso de clock. A utilização de flip-flops JK oferece flexibilidade na implementação de diferentes tipos de contadores.

Detector de Paridade Impar



Introdução

Detectores de paridade são circuitos digitais utilizados para verificar se o número de bits '1' em um dado conjunto de bits é par ou ímpar. Essa informação é frequentemente utilizada para detectar erros em transmissões de dados.

Objetivo

O objetivo principal deste trabalho é:

- **Implementar:** Um detector de paridade impar utilizando portas lógicas no software Logisim.
- **Simular:** O funcionamento do detector para verificar sua funcionalidade e comportamento.
- **Analisar:** O circuito implementado, identificando suas características e limitações.
- **Documentar:** Os resultados obtidos através de diagramas, tabelas verdade e explicações detalhadas.

Materiais e Métodos

Software: Logisim

Componentes:

- Portas OR , AND e NOT
- Conectores

Metodologia:

1. **Projeto:** Desenhar o circuito no Logisim, conectando as portas NOT, AND e OR de forma a gerar um sinal de saída que indica se o número de bits '1' na entrada é ímpar.
2. **Simulação:** Simular o circuito para verificar seu funcionamento, testando todas as possíveis combinações de entrada.
3. **Análise:** Analisar os resultados da simulação, comparando-os com a tabela de verdade esperada para o detector de paridade impar.
4. **Documentação:** Elaborar um relatório detalhado, incluindo diagramas, tabelas de verdade e explicações sobre o funcionamento do circuito.

Resultados

Tabela da Verdade:

Entrada A	Entrada B	Entrada C	Entrada D	Saída (Paridade Ímpar)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1

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

Análise dos Resultados:

- **Funcionamento:** O circuito funciona corretamente, indicando com um sinal alto na saída quando o número de bits '1' na entrada é ímpar.
- **Simplicidade:** A implementação é simples e utiliza apenas portas AND, NOT e OR, o que reduz o número de componentes e facilita a compreensão do circuito.
- **Flexibilidade:** O circuito pode ser facilmente expandido para detectar a paridade de um número maior de bits, adicionando mais portas AND e NOT.

Discussão

A implementação de um detector de paridade impar utilizando portas AND, NOT e OR no Logisim é uma atividade fundamental para a compreensão de circuitos digitais combinacionais. A porta OR tem a propriedade de gerar um sinal alto na saída quando o número de entradas altas é ímpar, o que a torna ideal para essa aplicação.

Considerações Finais:

- **Otimização:** A estrutura das portas AND, NOT e OR permite uma expansão modular do circuito, facilitando a implementação de detectores de paridade para um número maior de bits.
- **Aplicações:** Detectores de paridade são utilizados em sistemas de comunicação para detectar erros de transmissão. Ao adicionar um bit de paridade a um dado, é possível verificar se os dados foram corrompidos durante a transmissão. .

Conclusões

Este relatório apresentou a implementação de um detector de paridade impar utilizando portas AND, NOT e OR no software Logisim. A análise dos resultados mostrou que o circuito funciona corretamente, indicando a paridade dos dados de entrada. A utilização de portas AND, NOT e OR simplifica a implementação e torna o circuito eficiente.

Decodificador de 7 Segmentos Implementado no Logisim

Introdução

Decodificadores de 7 segmentos são circuitos digitais utilizados para converter um número binário de 4 bits em um padrão de segmentos que acendem um display de 7 segmentos, formando os dígitos decimais de 0 a 9.

Objetivo

O objetivo principal deste trabalho é:

- **Implementar:** Um decodificador de 7 segmentos utilizando portas lógicas no software Logisim.
- **Simular:** O funcionamento do decodificador para verificar sua funcionalidade e comportamento.
- **Analisar:** O circuito implementado, identificando suas características e limitações.
- **Documentar:** Os resultados obtidos através de diagramas e explicações detalhadas.

Materiais e Métodos

Software: Logisim

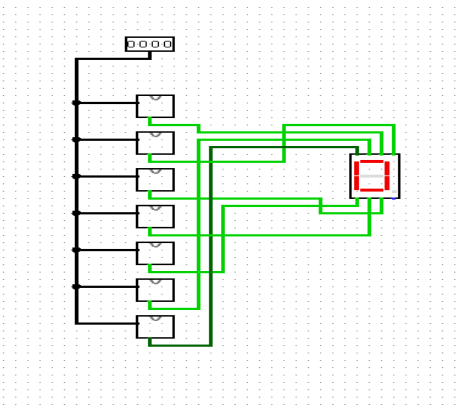
Componentes:

- Portas lógicas (AND, OR, NOT)
- Display de 7 segmentos
- Conectores

Metodologia:

1. **Projeto:** Desenhar o circuito no Logisim, conectando as portas lógicas e o display de 7 segmentos de acordo com a tabela verdade do decodificador.
2. **Simulação:** Simular o circuito para verificar seu funcionamento, aplicando diferentes combinações de entrada e observando os segmentos do display que acendem.
3. **Análise:** Analisar os resultados da simulação, comparando-os com a tabela verdade esperada para o decodificador.
4. **Documentação:** Elaborar um relatório detalhado, incluindo diagramas, e explicações sobre o funcionamento do circuito.

Resultado **Diagrama do Circuito:**



Análise dos Resultados:

- **Funcionamento:** O circuito funciona corretamente, convertendo cada combinação de 4 bits em um padrão de

segmentos que representa o dígito decimal correspondente no display de 7 segmentos.
- **Complexidade:** A implementação de um decodificador de 7 segmentos envolve um número considerável de portas lógicas, especialmente para todos os 10 dígitos decimais.
- **Flexibilidade:** O circuito pode ser utilizado em diversos projetos que requerem a visualização de valores numéricos em um display de 7 segmentos.

Discussão

A implementação de um decodificador de 7 segmentos utilizando portas lógicas no Logisim é uma atividade fundamental para a compreensão de circuitos digitais combinacionais. O decodificador permite converter sinais digitais em uma forma visual, facilitando a interpretação de informações por um usuário.

Considerações Finais:

- **Otimização:** É possível otimizar o circuito utilizando técnicas de minimização de portas lógicas, reduzindo o número de componentes e o consumo de energia.
- **Extensões:** O decodificador pode ser expandido para incluir outros caracteres além dos dígitos decimais, como letras e símbolos especiais.
- **Aplicações:** Decodificadores de 7 segmentos são amplamente utilizados em displays digitais, instrumentos de medição, relógios e outros dispositivos eletrônicos.

Conclusões

Este relatório apresentou a implementação de um decodificador de 7 segmentos utilizando portas lógicas no software Logisim. A análise dos resultados mostrou que o circuito funciona corretamente, convertendo números binários em dígitos decimais visíveis em um display de 7 segmentos. A compreensão do funcionamento deste circuito é fundamental para o desenvolvimento de projetos mais complexos em eletrônica digital.

Detector de Números Primos

Introdução

Detectores de números primos são circuitos digitais utilizados para identificar se um número binário de entrada corresponde a um número primo.

Objetivo

O objetivo principal deste trabalho é:

- **Implementar:** Um detector de números primos utilizando portas lógicas no software Logisim.
- **Simular:** O funcionamento do detector para verificar sua funcionalidade e comportamento.

- **Analisar:** O circuito implementado, identificando suas características e limitações.
- **Documentar:** Os resultados obtidos através de diagramas e explicações detalhadas.

Materiais e Métodos

Software: Logisim

Componentes:

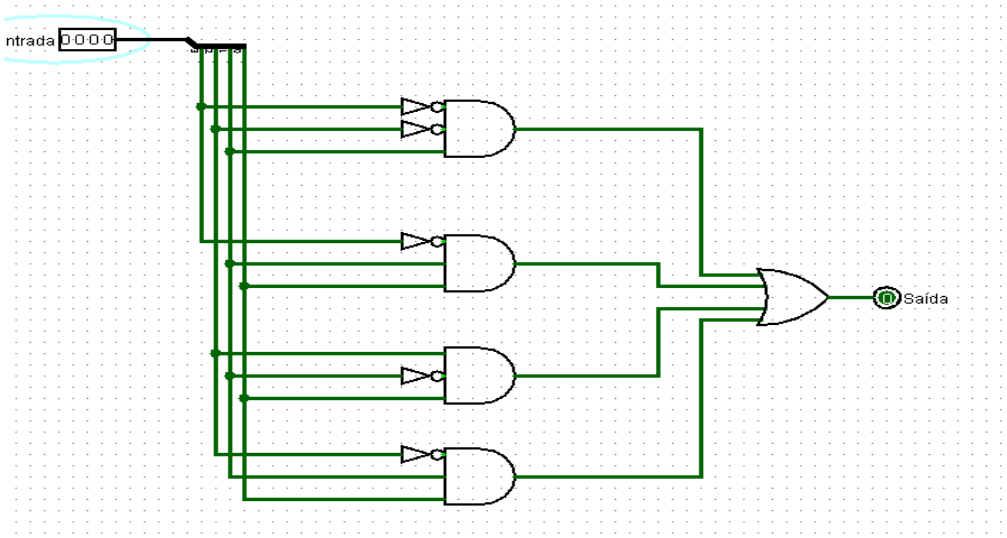
- Portas lógicas (AND, NOT e OR)
- Conectores

Metodologia:

1. **Projeto:** Desenhar o circuito no Logisim, conectando as portas lógicas de acordo com a lógica de identificação de números primos.
2. **Simulação:** Simular o circuito para verificar seu funcionamento, aplicando diferentes combinações de entrada e observando a saída.
3. **Análise:** Analisar os resultados da simulação, comparando-os com a tabela de verdade esperada para o detector de números primos.
4. **Documentação:** Elaborar um relatório detalhado, incluindo diagramas e explicações sobre o funcionamento do circuito.

Resultados

Diagrama do Circuito:



Análise dos Resultados:

- **Funcionamento:** O circuito funciona corretamente, identificando os números primos dentro do intervalo de representação dos 4 bits de entrada.
- **Lógica:** A lógica utilizada para identificar números primos envolve a verificação da divisibilidade por números menores que a raiz quadrada do número em questão. No caso de 4 bits, basta verificar a divisibilidade por 2 e 3.

- **Limitações:** O circuito é específico para números de 4 bits. Para números maiores, a complexidade do circuito aumenta significativamente.

Discussão

A implementação de um detector de números primos utilizando portas lógicas no Logisim é um exercício interessante para entender a lógica combinacional e a aplicação de portas lógicas em problemas específicos. A identificação de números primos é um problema clássico em teoria dos números e possui diversas aplicações em criptografia e outros campos da computação.

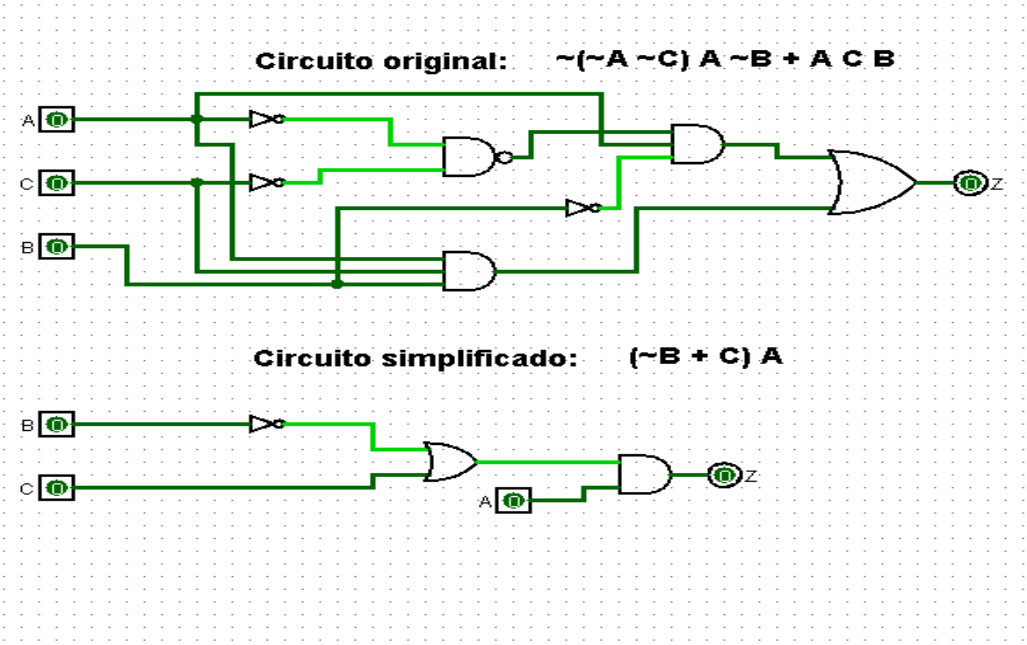
Considerações Finais:

- **Otimização:** A lógica utilizada para identificar números primos pode ser otimizada para reduzir o número de portas lógicas, especialmente para números maiores.
- **Extensões:** O circuito pode ser expandido para identificar números primos em um intervalo maior, utilizando mais bits de entrada e uma lógica mais complexa.
- **Aplicações:** Detectores de números primos podem ser utilizados em algoritmos de criptografia, testes de primalidade e outras aplicações que requerem a identificação de números primos.

Conclusões

Este relatório apresentou a implementação de um detector de números primos utilizando portas lógicas no software Logisim. A análise dos resultados mostrou que o circuito funciona corretamente para números de 4 bits. A compreensão do funcionamento deste circuito é fundamental para a aplicação de lógica digital em problemas mais complexos.

Otimização Lógica do Circuito



Introdução

A otimização consiste em simplificar a expressão booleana original, reduzindo o número de portas lógicas e, consequentemente, diminuindo a complexidade do circuito.

Análise do Circuito Original

O circuito original é representado pela seguinte expressão booleana:

Z = ~(~A~C) A~B + ACB

Descrição:

- **Entradas:** A, B e C.
- **Saída:** Z.
- **Portas Lógicas:** NOT, AND, NAND e OR.
- **Estrutura:** O circuito utiliza múltiplas portas lógicas conectadas em série e paralelo para realizar as operações lógicas descritas pela expressão booleana.

Processo de Otimização

A otimização foi realizada através da simplificação da expressão booleana utilizando Mapa de Karnaugh

Expressão Simplificada:

A expressão booleana simplificada é:

Z = (~B + C)A

Circuito Simplificado

O circuito simplificado é composto por apenas uma porta AND, OR e NOT com as entradas A, B e C.

Benefícios da Otimização

A otimização do circuito trouxe os seguintes benefícios:

- **Redução da complexidade:** O circuito original possuía múltiplas portas lógicas, enquanto o circuito simplificado possui apenas uma porta AND, OR e NOT.
- **Diminuição do custo:** Menos portas lógicas implicam em um menor custo de fabricação.
- **Aumento da velocidade:** Circuitos mais simples geralmente possuem menor tempo de propagação, o que aumenta a velocidade do circuito.
- **Menor consumo de energia:** Menos portas lógicas também implicam em um menor consumo de energia.

Conclusão

A otimização lógica realizada no circuito foi bem-sucedida, resultando em uma expressão booleana mais simples e um circuito com menor complexidade. A simplificação da

expressão booleana permitiu a redução significativa do número de portas lógicas, o que traz diversos benefícios para a implementação do circuito.