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

AMANDA DE BRITO BARBOSA
ANDREZA OLIVEIRA GONÇALVES

LABORATÓRIO DE CIRCUITOS - CODIFICAÇÃO E SIMULAÇÕES

BOA VISTA – RR
2024

AMANDA DE BRITO BARBOSA
ANDREZA OLIVEIRA GONÇALVES

LABORATÓRIO DE CIRCUITOS - CODIFICAÇÃO E SIMULAÇÕES

Relatório Científico apresentado ao Prof. Dr. Herbert Oliveira Rocha, com objetivo de obtenção de nota parcial para aprovação na disciplina DCC 301 - Arquitetura e Organização de Computadores, do Departamento de Ciência da Computação da Universidade Federal de Roraima.

BOA VISTA – RR
2024

1 INTRODUÇÃO

Este relatório documenta o desenvolvimento e a análise de diversos componentes digitais projetados e simulados no *Logisim*, um simulador de circuitos digitais. O objetivo principal é projetar, implementar e validar circuitos fundamentais, detalhando cada etapa de construção e os resultados dos testes realizados.

O trabalho usa a lógica combinacional e sequencial para desenvolver componentes que representam os blocos fundamentais de sistemas digitais. Cada componente será descrito de forma detalhada, abrangendo:

1. **Descrição do Componente**:
   Identificação da funcionalidade do componente, explicando:
   - Pinos de entrada e saída.
   - Operações realizadas internamente (como o funcionamento lógico ou sequencial).

2. **Imagem do Circuito**:
   Capturas do circuito no Logisim, mostrando:
   - Conexões internas entre os elementos.
   - Identificação dos pinos de entrada e saída.

3. **Testes Realizados**:
   Detalhamento das estratégias de teste, incluindo:
   - Análise de estados para circuitos sequenciais.
   - Uso de tabelas verdade para validar lógica combinacional.
   - Configuração de simulações no Logisim, explicando como os estímulos foram aplicados aos pinos de entrada e como as saídas foram observadas.

4. **Descrição dos Testes**:
   Explicação dos cenários de teste e interpretação dos resultados:
   - Casos de teste detalhados (entrada de 0s e 1s para cada combinação possível).
   - Verificação do comportamento em diferentes estados.

2 COMPONENTES

2.1 Flip Flop tipo D e JK

2.1.1 Flip-Flop - D

É um circuito sequencial que armazena um único bit de dado. Ele possui entradas e saídas específicas, e sua funcionalidade é baseada na captura do valor na entrada D durante uma transição do sinal de clock (CLK).

Pinos de entrada:

D (Data): Entrada de dado. O valor presente aqui será armazenado no flip-flop durante a borda ativa do clock.

CLK (Clock): Sinal de controle que sincroniza a captura do dado. Pode ser sensível à borda de subida (↑) ou descida (↓), dependendo do design.

Pinos de saída:

Q: Saída principal que reflete o valor armazenado.

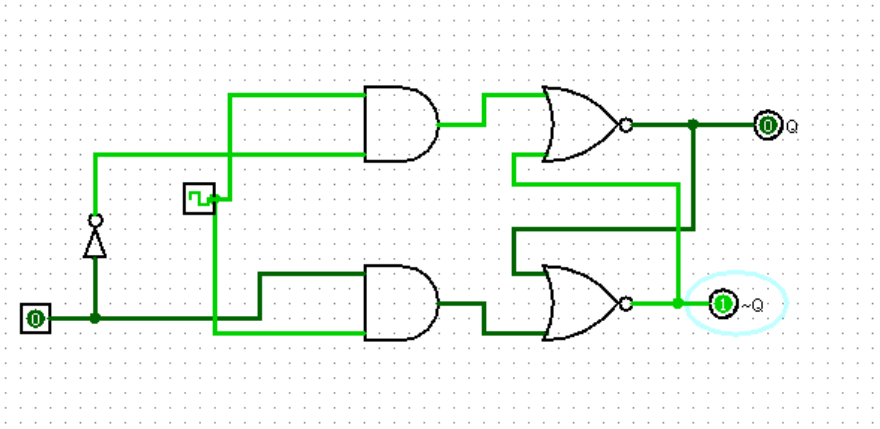
Q': Saída complementar, que é o inverso de Q.

Lógica de funcionamento:

Na borda ativa do clock (CLK), a saída Q assume o valor presente na entrada D.

Fora da borda do clock, Q mantém o valor armazenado anteriormente (estado estável).

Imagem do Circuito:



Teste com Entrada D e Clock CLK:

Com CLK=0, variamos D para verificar que Q não altera.

Com CLK=1, variamos D para observar se Q segue D.

Resultados esperados:

Quando o clock está na borda ativa, Q = D.

Fora da borda ativa, Q mantém o último valor.

Implementação da tabela verdade para verificar estes resultados:

D	~D	Q	~Q
0	0	Na	Na
0	1	0	1
1	0	1	0
1	1	Na	Na

Com apenas uma entrada D e o inversor na segunda entrada, o primeiro e último estado da tabela não existem neste circuito.

2.1.2 Flip Flop JK

É uma evolução do Flip-Flop RS, projetado para eliminar estados indesejados (como a indeterminação presente no Flip-Flop RS). Ele é um circuito sequencial que pode funcionar como um latch, um flip-flop simples, ou um contador, dependendo das entradas.

Pinos de Entrada:

J: Entrada que controla a ativação do estado "SET" (coloca Q=1).

K: Entrada que controla a ativação do estado "RESET" (coloca Q=0).

CLK (Clock): Entrada de controle que sincroniza as operações.

Pinos de Saída:

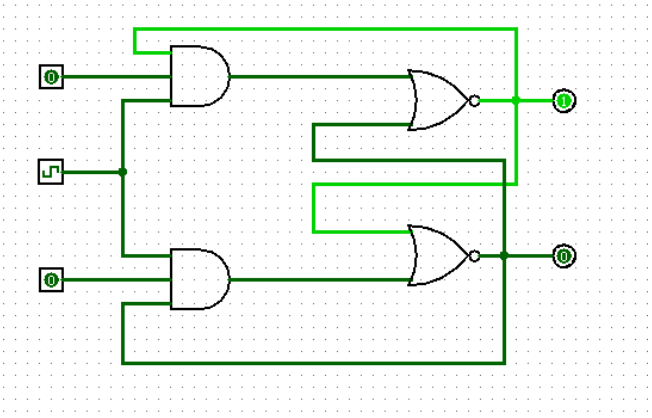
Q: Saída principal do flip-flop.

Q': Saída complementar, sempre inversa de Q.

Lógica de Funcionamento:

O estado das saídas Q e Q′ é definido com base nas entradas J, K e no sinal de clock.

Imagem do Circuito:



Teste de Set (J=1, K=0)
Configuração:
Conectando J=1, K=0 e um pulso no CLK, Q=1, Q′=0.

Teste de Reset (J=0, K=1)
Configuração:
Conectando J=0, K=1 e um pulso no CLK, Q=0, Q′=1.

Implementação da tabela verdade para verificar estes resultados:

J	K	Q	~Q
0	0	Qa	~Qa
0	1	0	1
1	0	1	0
1	1	~Qa	Qa

2.2 Multiplexador de quatro opções de entrada

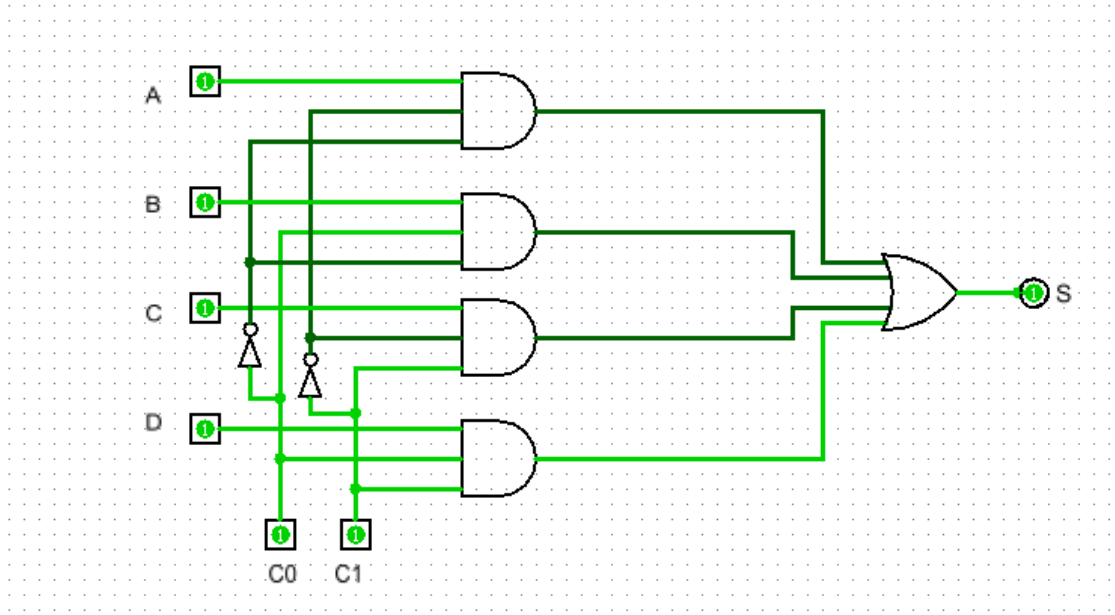
É um circuito combinacional que seleciona uma entre várias entradas de dados digitais e a encaminha para a saída com base em sinais de controle. Um multiplexador de 4 entradas (4:1) possui 4 entradas de dados, 2 entradas de seleção e 1 saída.

Pinos de Entrada:
A, B, C, D: Entradas de dados. Contêm os sinais que podem ser encaminhados à saída.
C0,C1: Entradas de seleção. Determinam qual entrada In será conectada à saída.

Pino de Saída:
S: Saída que reflete o valor da entrada selecionada.

Lógica do circuito: A saída S é determinada pela combinação das entradas C0 e C1.

Imagem do Circuito:



2.3 Porta lógica XOR usando os componentes: AND, NOT, e OR.

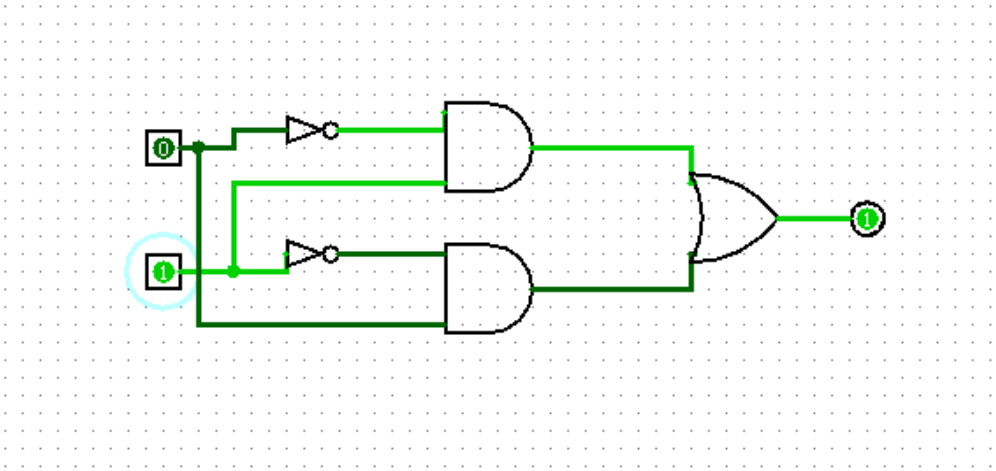
É uma porta lógica que produz uma saída alta (1) apenas quando uma das entradas é alta (1), mas não ambas simultaneamente. É amplamente utilizado em circuitos digitais para operações de paridade, somadores e sistemas de criptografia.

Pinos de Entrada:
A: Primeira entrada lógica.
B: Segunda entrada lógica.

Pino de Saída:
S: Saída que reflete o resultado da operação XOR aplicada às entradas A e B.

Lógica do Circuito:
A saída S é alta (1) se A e B forem diferentes.
A saída S é baixa (0) se A e B forem iguais.

Imagem do Circuito:



S = $AB + \overline{AB}$

Teste de Igualdade (A=B)
Se A=0 e B=0, S=0.
Se A=1 e B=1, S=0.

Teste de Diferença (A≠B)
Se A=0, B=1: Y=1.
Se A=1, B=0: Y=1.

2.4 Somador de 8 bits que recebe um valor inteiro e soma com o valor 4

Este circuito realiza a soma de um número binário de 8 bits, fornecido pelas entradas A0 a A7, com o valor constante 4. Ele utiliza somadores completos para realizar a operação bit a bit, propagando o carry entre os estágios.

Pinos de Entrada:

A0 a A7: Entradas binárias que representam o número inteiro de 8 bits a ser somado com
Cin: Carry de entrada (geralmente configurado como 0 para esta operação).

Pinos de Saída:

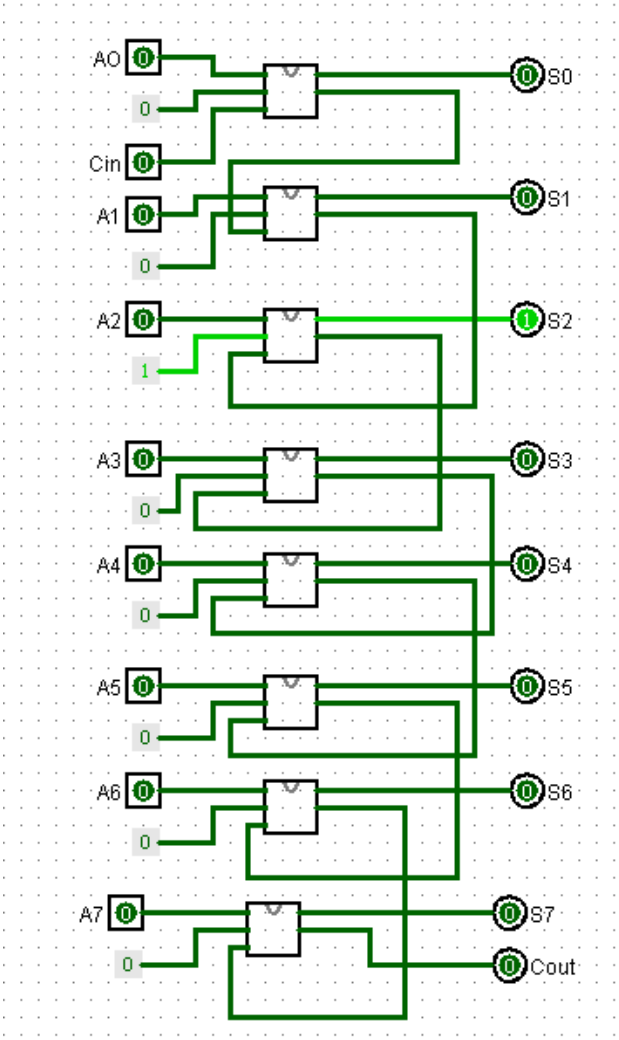
S0 a S7: Saídas binárias que representam o resultado da soma entre o valor fornecido nas entradas e o número 4.
Cout: Carry de saída gerado pela operação de soma, que pode ser usado para extensões ou verificações.

Lógica do Circuito:

1. O número 4 é representado em binário como 00000100.

2. Cada bit do número 4 é somado ao respectivo bit da entrada A0 a A7, utilizando somadores completos.
3. O bit menos significativo de A é somado ao bit menos significativo de 4, e o carry gerado é propagado para o próximo somador.
4. A operação continua até o bit mais significativo, onde o Cout pode indicar um overflow.

Imagem do Circuito:



Teste com Valor Zero:

- Configuração: Configure A0 a A7 como 00000000 e defina Cin=0.
- Saída: 00000100.
- Cout = 0.

Teste com Valor Máximo (Sem Overflow):

- Configuração: Configure A0 a A7 como 0011111 e defina Cin=0.
- Cout=1 (indicando overflow).

Teste com Incremento Básico:

- Configuração: Configure A0 a A7 como 01000000 e defina Cin=0.
- Saída: 00000110.

2.5 Memória ROM de 8 bits

ROM é um acrônimo para Read Only Memory, que significa Memória apenas para leitura. Esse tipo de memória é não-volátil, ou seja, ela não perde os dados com o desligamento da energia. A seguir como ela foi implementada no Logisim.

2.5.1 Componentes Principais e Funcionamento

Os multiplexadores desempenham o papel de selecionar qual registrador ou sinal será enviado para a saída.

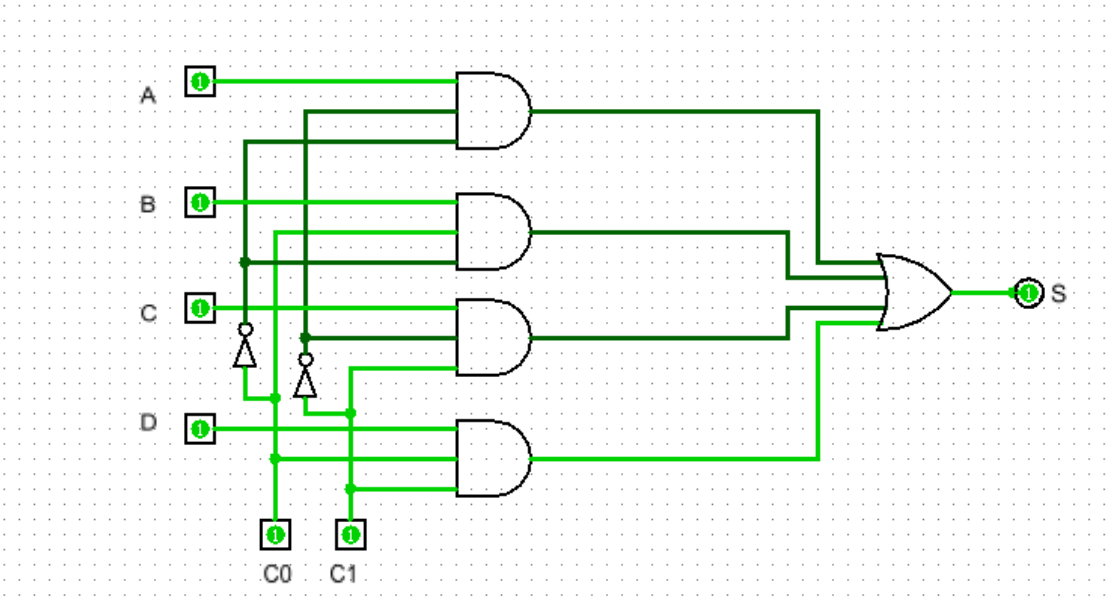
2.5.1.1 Multiplexador 4x1 (1 bit):

Combina quatro entradas de 1 bit para uma única saída.

Load: Conectado a uma porta AND, que combina o sinal de load com as quatro entradas para gerar uma saída única.

Chave: Entrada de 2 bits para selecionar a entrada ativa.

Imagem do Circuito:



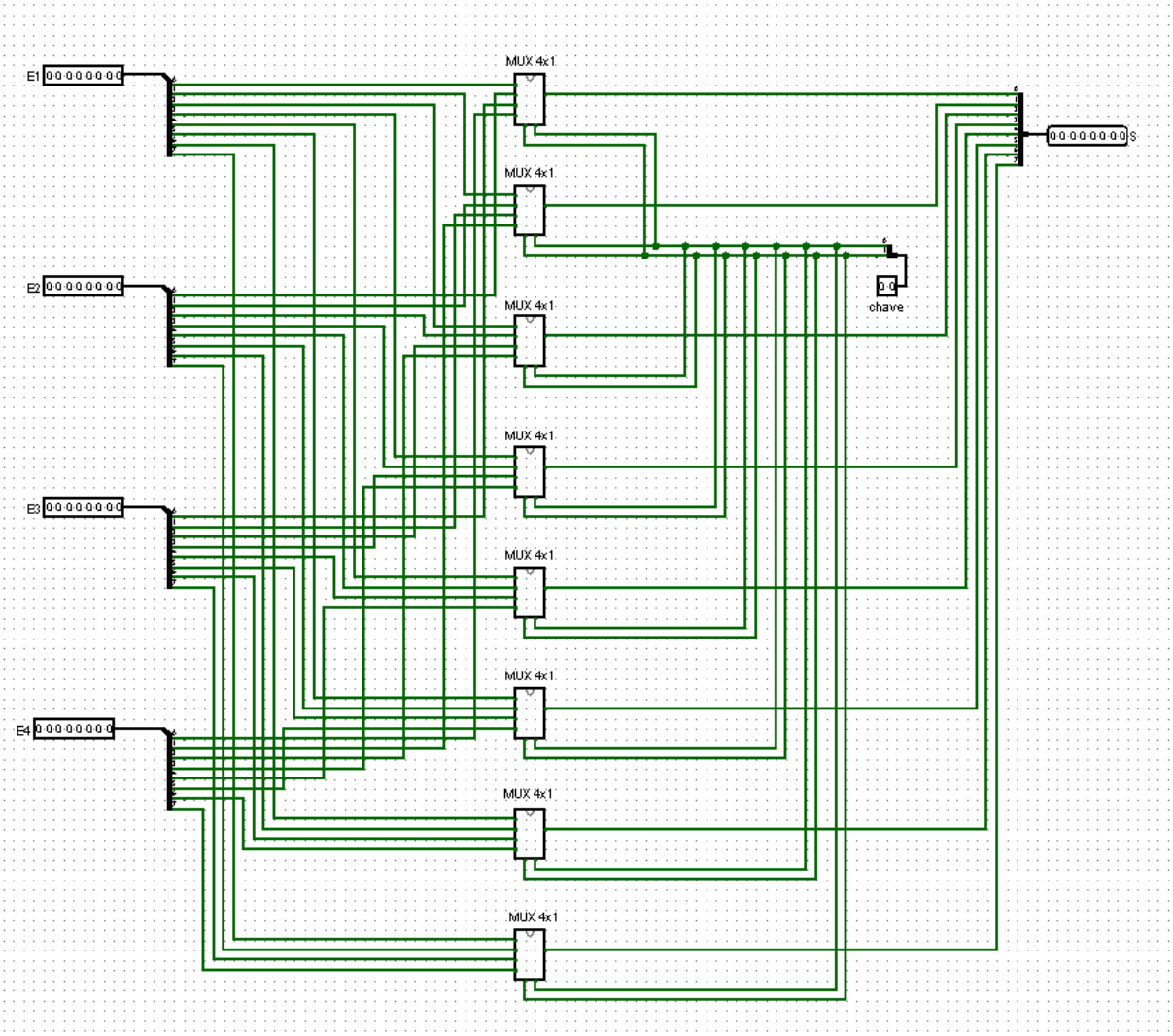
2.5.1.2 Multiplexador 4x1 (8 bits):

Combina quatro entradas de 8 bits para uma única saída.

Load: Conectado a uma porta AND em todas as entradas dos multiplexadores de 4x1 (1 bit), garantindo o controle do fluxo de dados.

Chave: Entrada de 2 bits para selecionar o registrador ativo.

Imagem do Circuito:



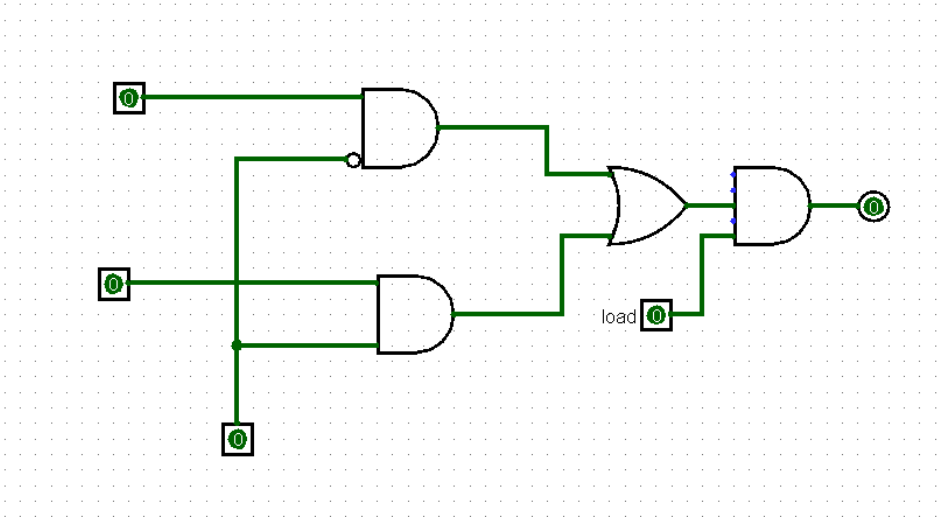
2.5.1.3 Multiplexador 2x1 (1 bit):

Combina duas entradas de 1 bit para uma única saída.

Load: Configurado com portas AND para cada entrada, garantindo que apenas uma esteja ativa.

Chave: Entrada de 1 bit que seleciona qual sinal será passado para a saída.

Imagem do Circuito:

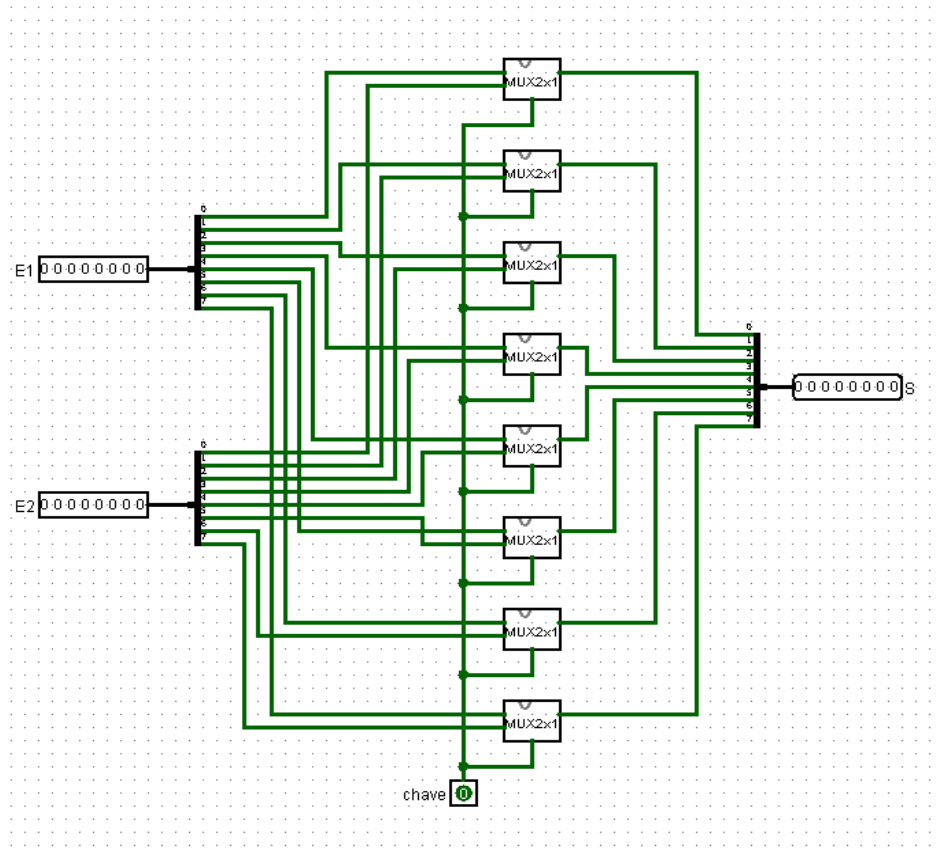


2.5.1.4 Multiplexador 2x1 (8 bit):

Conecta duas entradas de 8 bits e entrega uma saída única.

Chave: Entrada de 1 bit para selecionar a entrada ativa.

Construção: Combinado com o multiplexador de 4x1 (8 bits), recebendo as saídas destes e enviando a saída final para o multiplexador 8x1.

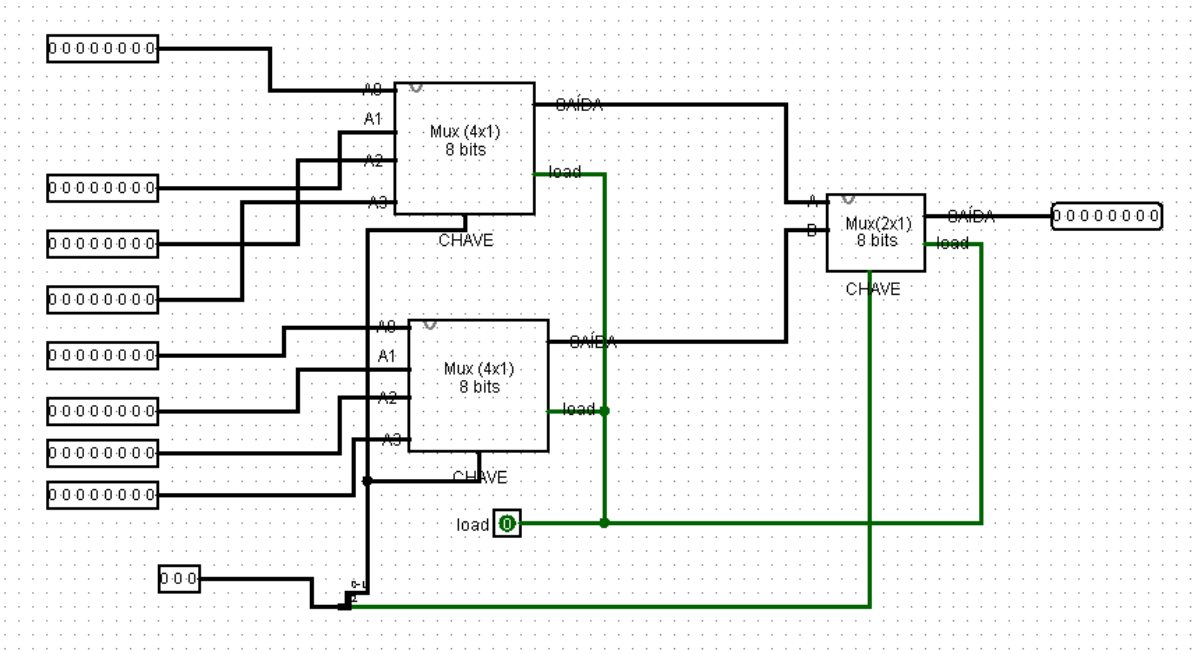


2.5.1.5 Multiplexador 8x1 (8 bit):

Seleciona uma das oito entradas de 8 bits para enviar à saída.

Load: Porta AND controla todas as entradas para garantir que somente o sinal ativo seja enviado à saída.

Chave: Entrada de 3 bits para determinar qual registrador será lido.



Pinos de entrada:

Endereço - Define a posição na memória a ser acessada.

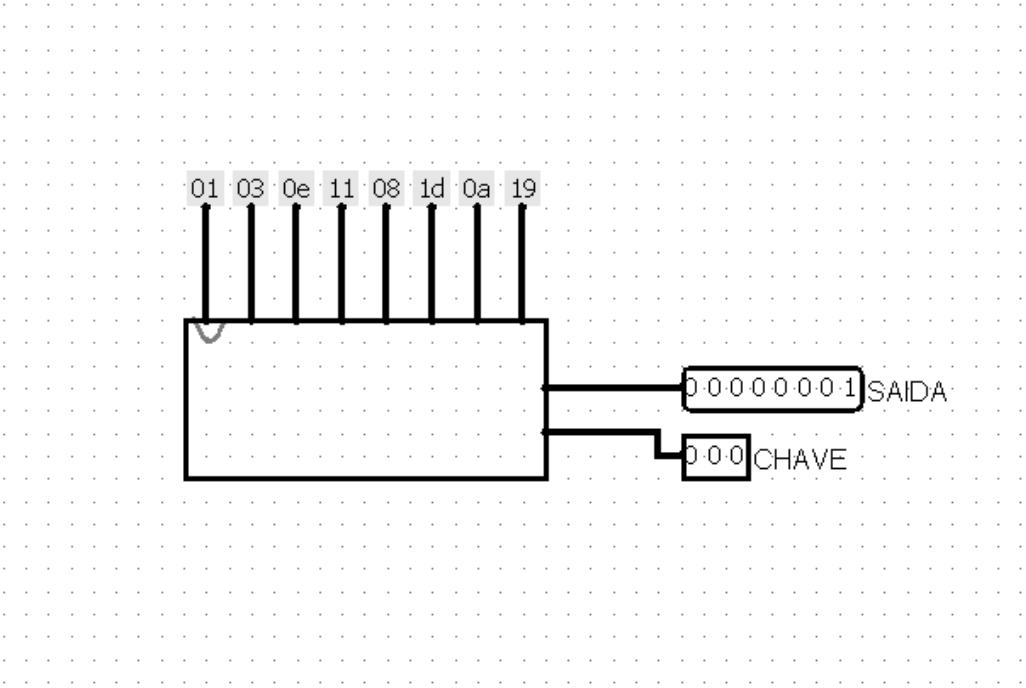
Pinos de saída:

Dados - Representa os dados armazenados no endereço fornecido.

Lógica do Circuito:

O endereço é selecionado por uma entrada que alimenta o multiplexador que lê o conteúdo do endereço e retorna esse dado no output. Ressalta-se que para representar que a memória ROM é somente para leitura, os registradores têm constantes em hexadecimal como input.

Imagem do Circuito:



Testes e Resultados Esperados:

Ao variar os endereços da chave de seleção (0 a 7), pode-se comparar os dados lidos com os valores programados inicialmente.

2.6 Memória RAM de 8 bits

Este circuito implementa uma memória RAM utilizando 8 registradores de 8 bits cada, permitindo operações de leitura e escrita seletiva. A memória é controlada por chaves de seleção, sinais de load, clock e componentes digitais, como multiplexadores e demultiplexadores. O circuito é modular e configurável, promovendo um aprendizado aprofundado sobre arquitetura de memória em sistemas digitais.

2.6.1 Componentes Principais e Funcionamento

2.6.1.1 Registradores de 8 Bits

Cada registrador é responsável por armazenar dados de 8 bits.

Entradas:

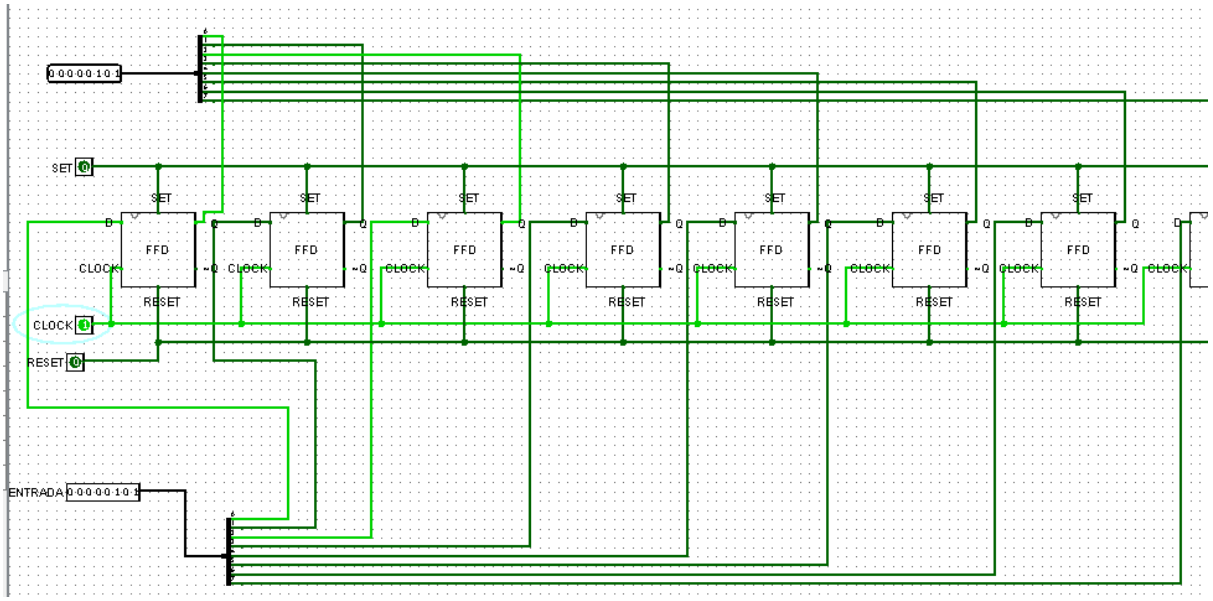
- **Entrada de 8 bits**: Dados a serem armazenados.
- **Set e Reset**: Inicializa ou limpa o estado do registrador.
- **Clock**: Sincroniza as operações de escrita.
- **Load**: Habilita a escrita no registrador.

Saída:

- Dados armazenados no registrador são liberados para o barramento.

Construção:

- Formado por 8 flip-flops D mestre-escravo, onde cada bit passa pelo flip-flop antes de ser armazenado.



2.6.1.2 Demultiplexador de 1 Entrada de 3 bits para 8 Saídas

Direciona o sinal de controle para o registrador selecionado com base nas chaves de seleção.

Entradas:

- **Entrada de 3 bits (seleção)**: Escolhe o registrador a ser ativado.
- **Clock**: Sincroniza o sinal.
- **Load**: Habilita a operação de escrita no registrador.

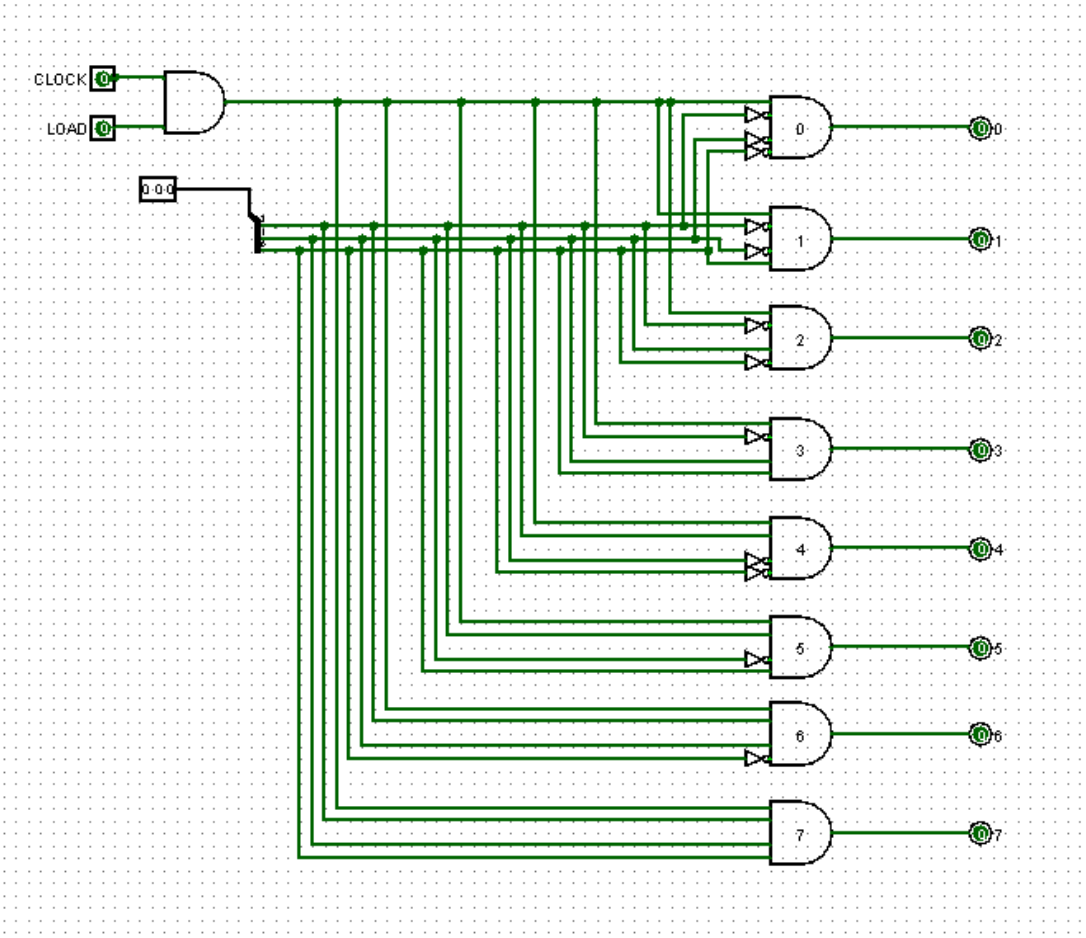
Saídas:

- Cada uma das 8 saídas do demultiplexador corresponde a um registrador.

Construção:

- Uma porta AND combina o sinal de load com o clock, garantindo que a ativação só ocorra em registradores específicos.
- As 3 entradas de seleção definem as 8 combinações possíveis para ativar cada saída.

Imagem do Circuito:



2.6.1.3 Multiplexadores

Os multiplexadores desempenham o papel de selecionar qual registrador ou sinal será enviado para a saída. Imagens dos circuitos já presentes neste repositório.

2.6.1.3.1 Multiplexador 4x1 (1 bit):

Combina quatro entradas de 1 bit para uma única saída.

Load: Conectado a uma porta AND, que combina o sinal de load com as quatro entradas para gerar uma saída única.

Chave: Entrada de 2 bits para selecionar a entrada ativa.

2.6.1.3.2 Multiplexador 4x1 (8 bits):

Combina quatro entradas de 8 bits para uma única saída.

Load: Conectado a uma porta AND em todas as entradas dos multiplexadores de 4x1 (1 bit), garantindo o controle do fluxo de dados.

Chave: Entrada de 2 bits para selecionar o registrador ativo.

2.6.1.3.3 Multiplexador 2x1 (1 bit):

Combina duas entradas de 1 bit para uma única saída.

Load: Configurado com portas AND para cada entrada, garantindo que apenas uma esteja ativa.
Chave: Entrada de 1 bit que seleciona qual sinal será passado para a saída.

2.6.1.3.4 Multiplexador 2x1 (8 bit):
Conecta duas entradas de 8 bits e entrega uma saída única.
Chave: Entrada de 1 bit para selecionar a entrada ativa.
Construção: Combinado com o multiplexador de 4x1 (8 bits), recebendo as saídas destes e enviando a saída final para o multiplexador 8x1.

2.6.1.3.4 Multiplexador 8x1 (8 bit):
Seleciona uma das oito entradas de 8 bits para enviar à saída.
Load: Porta AND controla todas as entradas para garantir que somente o sinal ativo seja enviado à saída.
Chave: Entrada de 3 bits para determinar qual registrador será lido.

2.6.2 Operação do Circuito

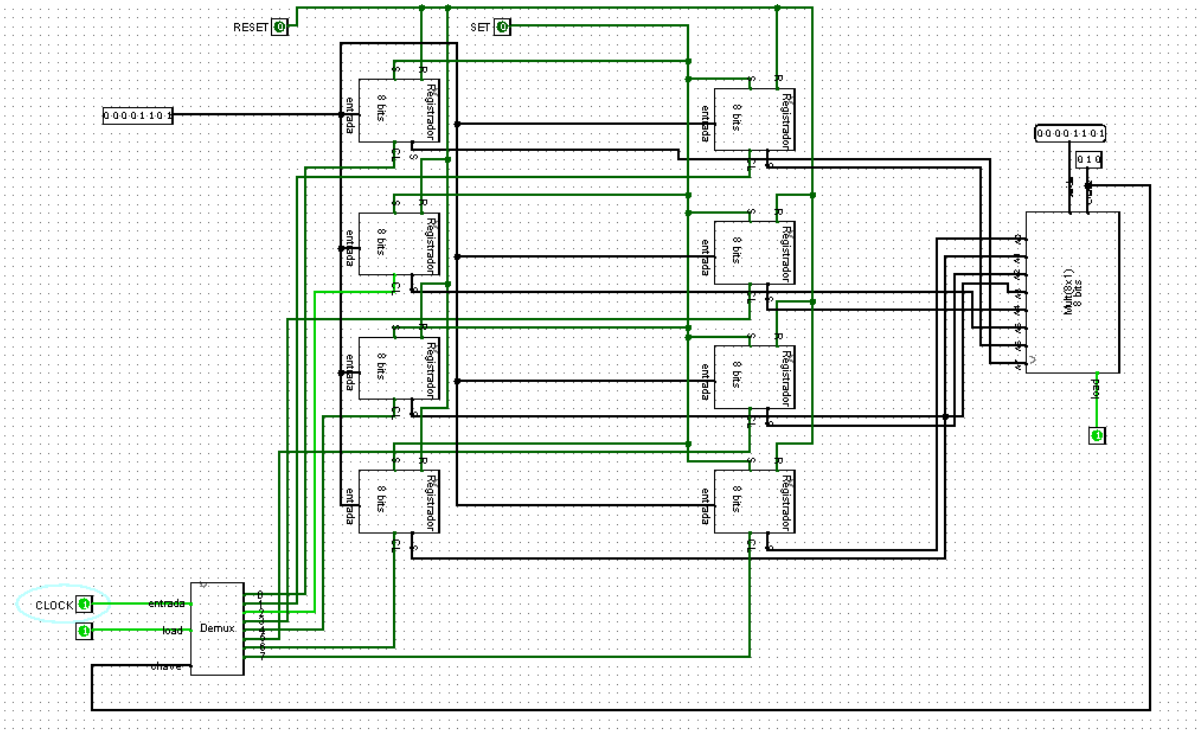
Escrita na Memória

1. Configure a chave de seleção no demultiplexador para selecionar o registrador desejado.
2. Insira os dados na entrada de 8 bits.
3. Ative o load e o clock.
4. O dado será armazenado no registrador selecionado.

Leitura da Memória

1. Configure a chave de seleção no multiplexador 8x1 para apontar ao registrador desejado.
2. Mantenha o load no estado 1.
3. A saída exibirá o conteúdo do registrador selecionado.

6.1.3 Imagem do Circuito:



Testes Realizados

Os testes foram realizados ao preencher com uma entrada qualquer de 8 bits, escolher um registrador com a chave e ativar o load e clock, que consequentemente o escreve. Em seguida, ativando o load do multiplexador, pode-se realizar a leitura dos dados registrados no registrador indicado pela chave correspondente.

2.7 Banco de Registradores de 8 bits

Um banco de registradores é uma estrutura fundamental em sistemas digitais, permitindo o armazenamento e a leitura de múltiplos dados em registradores individuais. Ele é utilizado em unidades de processamento para operações rápidas de leitura e escrita de dados. Este trabalho apresenta a implementação de um banco de registradores com 8 registradores de 8 bits cada, utilizando multiplexadores e demultiplexadores para controlar as operações de leitura e escrita.

2.7.1 Componentes Utilizados:

- **Registradores**: 8 registradores, cada um com 8 bits.
- **Demultiplexador**: Para direcionar o dado de entrada ao registrador correto.
- **Multiplexadores**: Dois multiplexadores de 8 entradas de 8 bits com uma saída de 8 bits.
- **Entrada de 8 bits**: Dados a serem armazenados.
- **Set e Reset**: Inicializa ou limpa o estado do registrador.
- **Clock**: Sincroniza as operações de escrita.

- **Load**: Habilita a escrita no registrador.

Os mesmos componentes utilizados aqui são os da memória RAM. No entanto, será adicionado um multiplexador de 8 entradas de 8 bits com uma única saída. A chave de seleção desse multiplexador não estará conectada à chave inicial do demultiplexador. Assim, os componentes descritos nos itens 2.6.1.1 a 2.6.1.3.4 da memória RAM permanecem os mesmos. No final, as saídas serão interligadas para formar uma única saída de leitura, só que com chaves de seleção que podem se coincidir ou não.

Entradas:

- Dados de entrada: 8 bits.
- Seleção de registrador para escrita (3 bits).
- Seleção de registradores para leitura (2 conjuntos de 3 bits).
- Sinal de Load: Controla a habilitação da escrita.

Saídas:

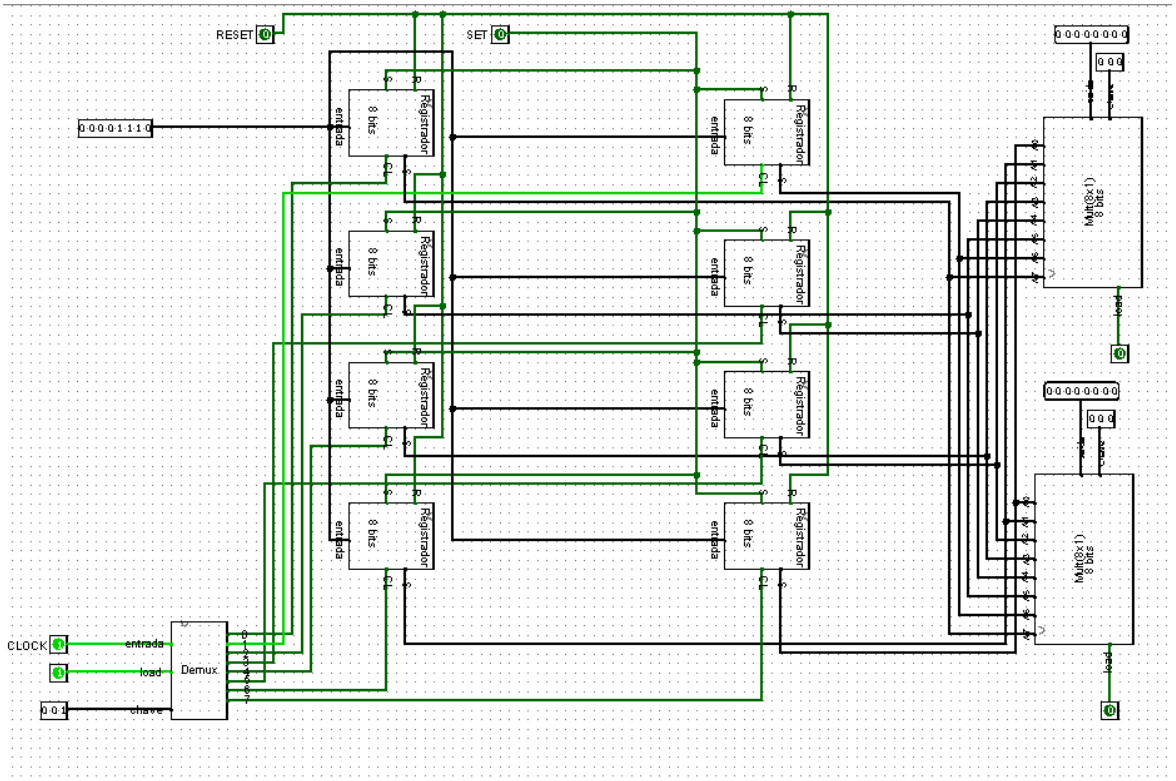
- Duas saídas de 8 bits para leitura dos registradores.

Funcionamento do Circuito:

- **Escrita em Registradores**:
    - A entrada de 3 bits (chave de seleção do registrador) é usada para ativar uma linha específica no demultiplexador.
    - O sinal de Load habilita a escrita no registrador selecionado.
    - Apenas quando a chave do demultiplexador coincide com a entrada de controle e o Load está ativado, os dados na entrada são gravados no registrador correspondente.

- **Leitura dos Registradores**:
    - Dois multiplexadores de 8 entradas de 8 bits são utilizados para leitura simultânea de dois registradores diferentes.
    - Cada multiplexador possui uma chave de 3 bits que seleciona o registrador a ser lido.
    - As saídas dos multiplexadores representam os dados dos registradores selecionados.

Imagem do Circuito:



Testes Realizados

Os testes foram realizados em um simulador digital (ex.: Logisim) para validar as operações de leitura e escrita no banco de registradores.

1. **Teste de Escrita**:
    - **Entradas**:
        - Dados: 11001100.
        - Chave de escrita: 010 (registrador 2).
        - Load: 111.
    - **Saída**: O registrador 2 armazena 11001100.

2. **Teste de Leitura Simultânea**:
    - **Entradas**:
        - Chave de leitura 1: 010 (registrador 2).
        - Chave de leitura 2: 011 (registrador 3).
    - **Saídas**:
        - Leitura 1: 11001100 (registrador 2).
        - Leitura 2: Valor armazenado no registrador 3.

3. **Teste de Operação Independente**:

- **Entradas**:
  - Chave de escrita: 010 (registrador 2).
  - Dados de escrita: 10101010.
  - Chave de leitura 1: 001 (registrador 1).
  - Chave de leitura 2: 100 (registrador 4).
  - Load: 000.
- **Saídas**:
  - Registrador 2 não é modificado (Load = 0).
  - Registrador 1 e 4 retornam seus valores.

2.8 Somador de 8 bits

O somador de 8 bits é um circuito combinacional utilizado para realizar a adição de dois números binários de 8 bits, gerando uma saída também de 8 bits e um Carry Out. Ele é construído utilizando dois somadores de 4 bits conectados em cascata, permitindo que o Carry gerado pelos bits menos significativos seja propagado para a adição dos bits mais significativos. Este circuito tem aplicação em unidades aritméticas e lógicas (ULAs) de sistemas digitais.

Entradas:

- Entrada 1(8 bits): Primeira entrada binária.
- Entrada 2(8 bits): Segunda entrada binária.
- Carry In (Cin): Definido como constante **0**.

Saídas:

- Saída[7:0] (8 bits): Resultado da soma binária.
- Carry Out (Cout): Indica o "vai-um" gerado pela soma.

Divisão em Módulos:

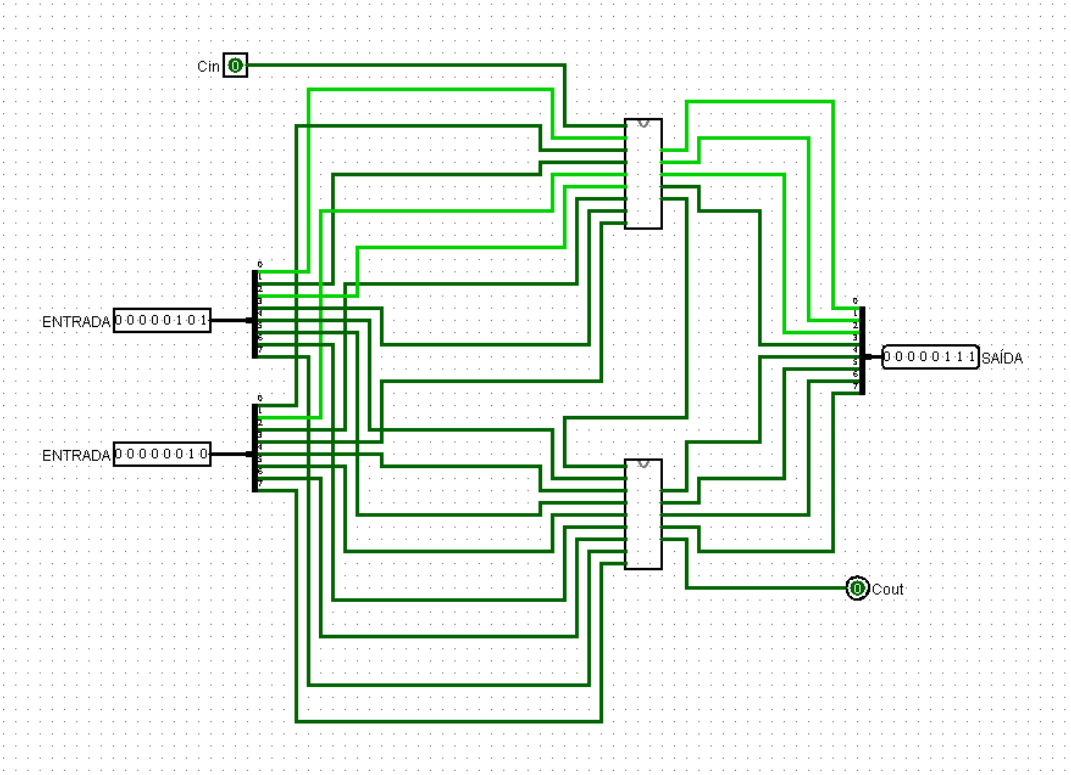
O circuito é dividido em dois somadores de 4 bits:

- **Primeiro Somador**: Soma os 4 bits menos significativos (Entrada 1[3:0] + Entrada 2[3:0] + Cin).
- **Segundo Somador**: Soma os 4 bits mais significativos (Entrada 1[7:4] + Entrada 2[7:4] + Cout do primeiro somador).

Conexões Lógicas:

- O Carry Out do primeiro somador (Cout_1) é conectado como Carry In do segundo somador.
- As saídas de ambos os somadores são combinadas para formar o resultado final.

Imagem do Circuito



Testes Realizados

Para validar o funcionamento do somador de 8 bits, foram realizados os seguintes testes em ambiente de simulação (ex.: Logisim):

Teste 1:

- **Entradas**:
  Entrada 1=00000001 , Entrada 2=00000001, Cin=0.
- **Saída Esperada**:
  Saída=00000010, Cout=0.

Teste 2:

- **Entradas**:
  Entrada 1=11111111, Entrada 2=00000001, Cin=0.
- **Saída Esperada**:
  Saída=00000000, Cout=1.

2.9 Detector de sequência binária para identificar a sequência "101"

Com base na questão 13 do contador síncrono descrito, podemos implementar um detector de sequência binária para identificar a sequência "101" em um fluxo de dados de entrada. O objetivo desse circuito é monitorar a sequência de entrada e ativar a saída (gerar um "1") apenas quando a sequência "101" for detectada. Nos demais casos, a saída permanece em "0".

Descrição do Circuito

- **Base do Circuito**:
    - Utilizamos o **contador síncrono** já descrito, com suas três saídas principais (**A**, **B**, e **C**) representando o estado do contador.
    - As saídas do contador são conectadas a uma **porta lógica AND**, que detecta a sequência desejada.
- **Detecção da Sequência**:
    - A sequência "101" será detectada quando as saídas do contador tiverem o seguinte estado:
        - **A = 1**, **B = 0**, **C = 1**.
    - Para isso, utilizamos uma **porta lógica AND** que combina:
        - **A** diretamente (pois precisa ser 1).
        - **B** invertido (para garantir que seja 0, usamos uma porta NOT).
        - **C** diretamente (pois precisa ser 1).
- **Tabela Verdade**:
    - A tabela verdade abaixo mostra os estados do contador e a saída do detector:

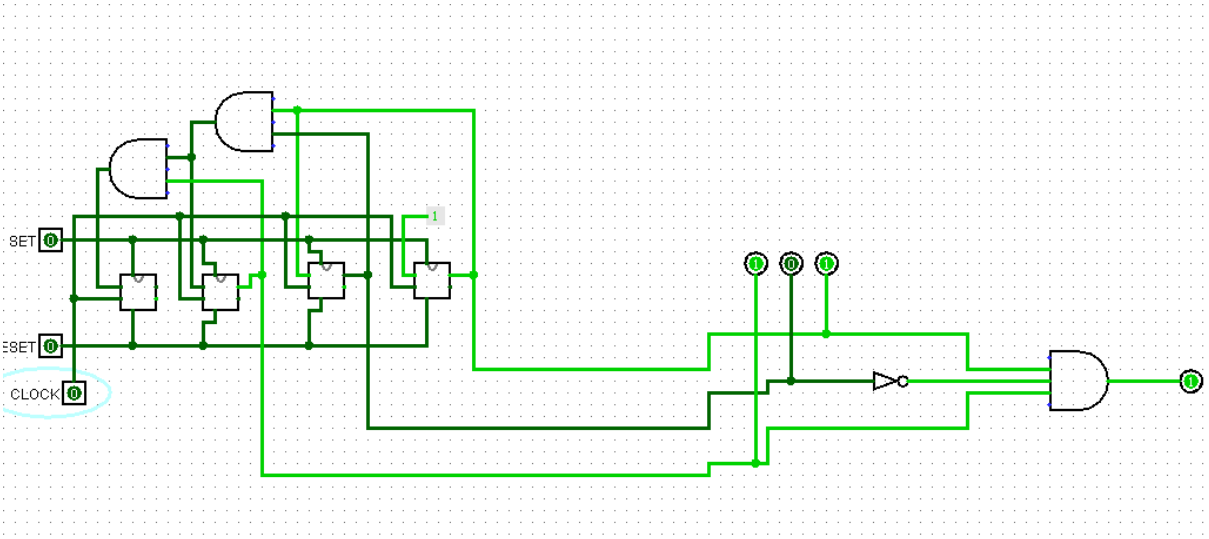
A	B	C	SAÍDA
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0

1	1	1	0

- A saída é **1** somente quando a sequência **ABC = 101** for detectada.

4. **Funcionamento do Circuito**:
   - O contador síncrono altera seus estados com base no Clock.
   - A cada pulso, os valores de **A**, **B**, e **C** são atualizados.
   - A porta lógica AND avalia continuamente os estados do contador. Quando o estado é "101", a saída do detector será "1".
   - Em qualquer outro estado, a saída será "0".

Imagem do Circuito



2.10 Unidade Lógica e Aritmética de 8 bits

Unidade Lógica e Aritmética de 8 bits no Logisim realiza operações lógicas e aritméticas de forma combinacional.

1. Operações Lógicas

- **AND**: Porta AND entre os 8 bits de A e B.
- **OR**: Porta OR entre os 8 bits de A e B.
- **NOT**: Inverte cada bit de A (sem B).
- **NOR**: Porta NOR entre os 8 bits de A e B.
- **NAND**: Porta NAND entre os 8 bits de A e B.
- **XOR**: Porta XOR entre os 8 bits de A e B.

2. Operações de Deslocamento

- **SHIFT LEFT (2 bits)**: Multiplica A por 4 deslocando os bits à esquerda.
- **SHIFT RIGHT (2 bits)**: Divide A por 4 deslocando os bits à direita.

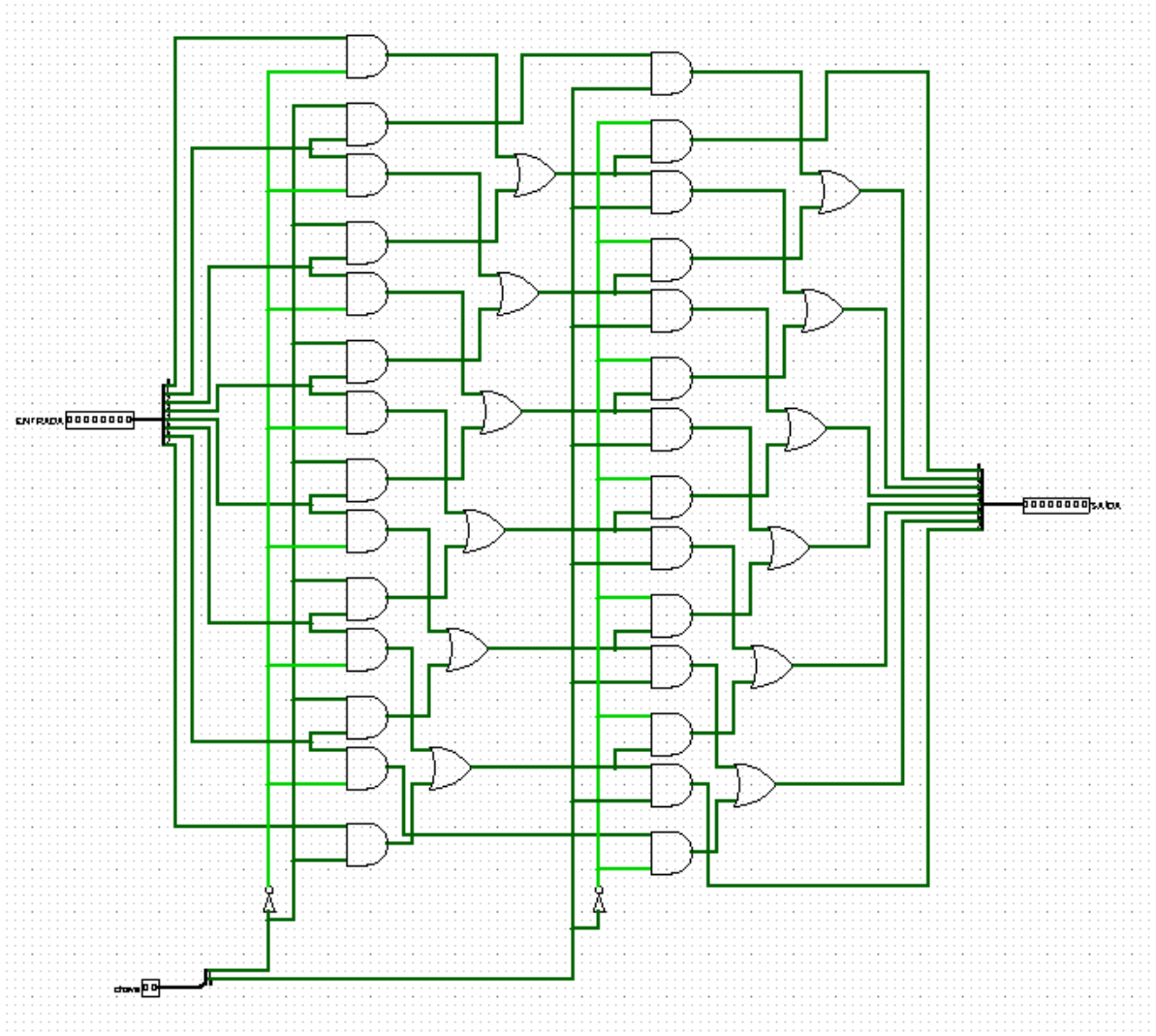
3. Operações Aritméticas

- **Soma**.
- **Subtração**.

Componentes Utilizados (8 bits):

1. Multiplexador 16x1.
2. Shift 2 bits à esquerda.
3. Shift 2 bits à direita.
4. Somador.
5. Subtrador.

Imagem do componente 3.



Lógica do Circuito:

- O MUX de 16 entradas seleciona uma das operações com base no controle (4 bits).
- Cada linha do MUX recebe uma operação específica.

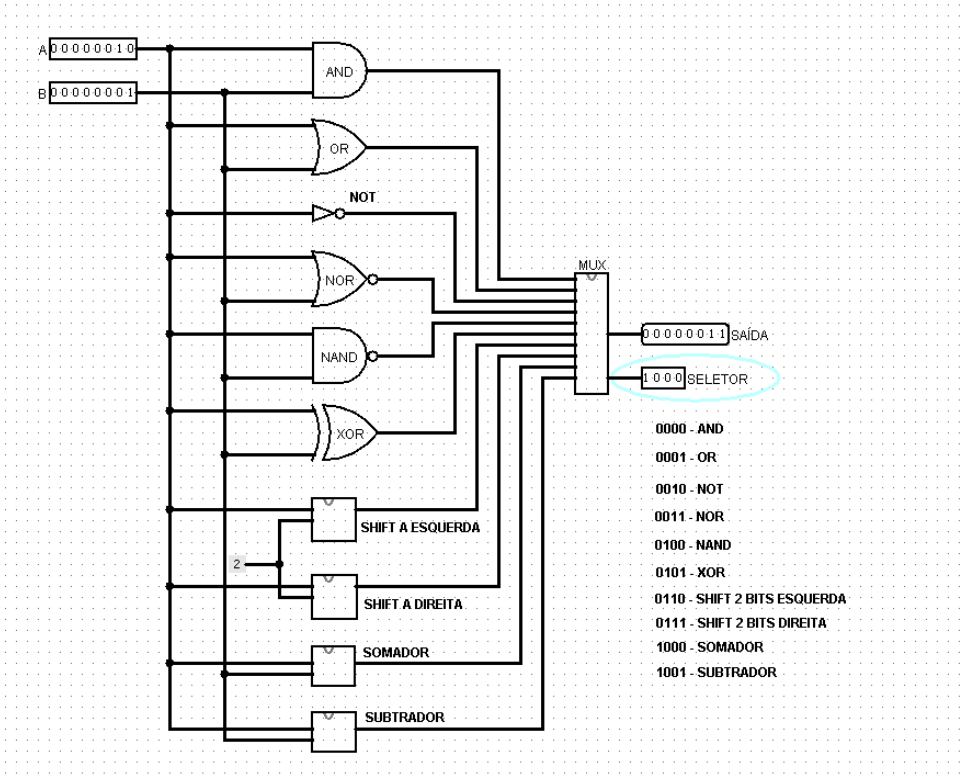
Entradas:

- Dois inputs de 8 bits para A e B.
- Um input de 4 bits para o controle.

Saídas:

- Resultado da operação (8 bits).
- Carry Out, Zero, e Overflow (1 bit cada).

Imagem do Circuito:



Teste de operações lógicas

- **AND**: A = 11001100 e B = 10101010 ⇒ Resultado = 10001000.
- **OR**: A = 11001100, B = 10101010 ⇒ Resultado = 11101110.
- **NOT**: A = 11001100 ⇒ Resultado = 00110011.
- **NOR**: A = 01010111, B = 00110010 ⇒ Resultado = 10001000.
- **NAND**: A = 00000100, B = 00010010 ⇒ Resultado = 11101111.
- **XOR**: A = 00001010, B = 00011011 ⇒ Resultado = 00010001.

Teste de operações de deslocamento

- **SHIFT LEFT (2 bits)**: A = 00001010 ⇒ Resultado = 00101000.
- **SHIFT RIGHT (2 bits)**: A = 00001010 ⇒ Resultado = 00000010.

Testes de operações aritméticas

- **Soma**: A = 00001010, B = 00000101 ⇒ Resultado = 00001111.
- **Subtração**: A = 00001010, B = 00000111 ⇒ Resultado = 00000011.

2.11 Extensor de sinal de 4 bits para 8 bits

Um extensor de sinal é utilizado em sistemas digitais para converter uma palavra de menor largura para outra de maior largura, sem alterar o valor numérico da informação representada. Este circuito realiza a extensão de um número de 4 bits para 8 bits, com a possibilidade de extensão por sinal ou por zero**.**

Componentes Utilizados:

Entradas:

- Entrada de 4 bits: Representa o valor binário original.
- Controle de Extensão (1 bit): Determina o tipo de extensão (sinal ou zero).

Distribuidores:

- Um distribuidor para dividir os 4 bits de entrada.
- Outro distribuidor para combinar os 8 bits de saída.

Conexões Lógicas: Implementadas de acordo com o tipo de extensão.

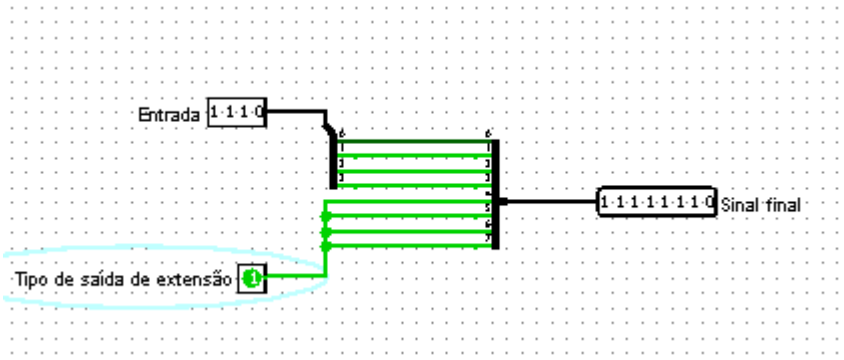
Funcionamento do Circuito:

- A entrada de 4 bits é mapeada para os 4 bits menos significativos da saída de 8 bits.
- Os 4 bits mais significativos (MSB) da saída dependem do controle:
    - Se o controle for 1 (extensão por sinal), os MSBs serão preenchidos com o bit mais significativo (MSB) da entrada de 4 bits.
    - Se o controle for 0 (extensão por zero), os bits mais significativos serão preenchidos com zeros.

Descrição do Circuito

1. **Entradas e Saídas**:
    - **Entradas**:

        - A[3:0]: Número de 4 bits a ser estendido.
        - C: Controle de extensão (1 para extensão por sinal, 0 para extensão por zero).
    - **Saída**:
        - B[7:0]: Número de 8 bits estendido.
2. **Extensão por Sinal**:
    - Para C=1:
        - B[7:4]=A[3] (o MSB da entrada é replicado nos MSBs da saída).
        - B[3:0]=A[3:0].
3. **Extensão por Zero**:
    - Para C=0:
        - B[7:4]=0000.
        - B[3:0]=A[3:0].
4. **Exemplo de Funcionamento**:
    - Entrada:
        - A=1010.
        - C=1.
    - Saída: B=11111010.
    - Entrada:
        - A=10.
        - C=0 (extensão por zero).
    - Saída: B=00001010.



2.12 Circuito de Máquina de Estados - Detector de Sequência Binária "101"

Uma máquina de estados é um circuito digital projetado para realizar operações sequenciais com base em entradas e estados anteriores. Este trabalho apresenta a implementação de uma máquina de estados que detecta a sequência binária "101" em um fluxo de entrada,

utilizando flip-flops D mestre-escravo, portas lógicas e sinais de controle. A saída é ativada (nível lógico alto) somente quando a sequência "101" é detectada.

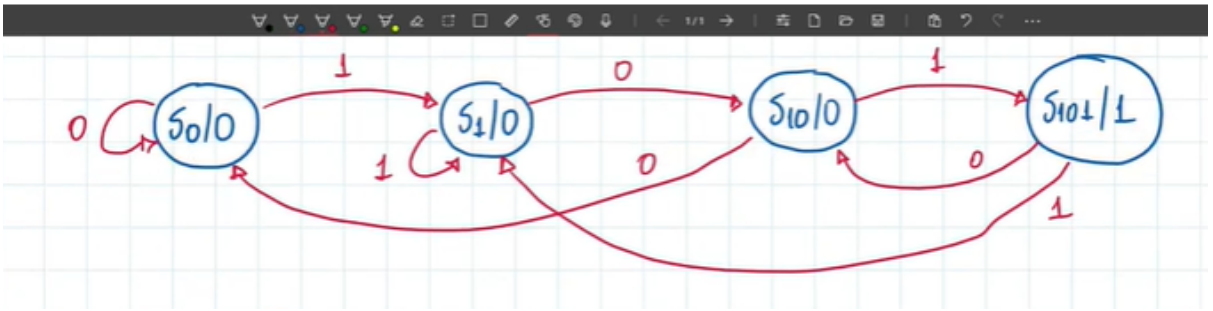
Componentes Utilizados:

- **Flip-Flops D Mestre-Escravo**: Dois flip-flops para armazenar os estados da máquina.
- **Portas Lógicas**: Três portas AND, uma porta OR e inversores para implementar as transições de estado.
- **Sinais de Controle**: Clock, Set e Reset para sincronização e inicialização.

Funcionamento do Circuito:

- O circuito é composto por dois flip-flops D que representam as variáveis de estado E1 e E0.
- As transições de estado são controladas pelas portas lógicas, que recebem a entrada X, os estados atuais (E1 e E0) e as negações desses estados.
- A saída YYY será ativada apenas quando os estados indicarem a sequência "101".

A seguir, o diagrama que corresponde a essa máquina de estados:



Quando o estado é **S0** e o bit de entrada é **0**, o sistema permanece no estado **S0**. Caso receba o bit **1**, o sistema irá para o estado **S1**. No estado **S1**, se o bit recebido for **0**, o sistema avança para um próximo estado, pois a sequência buscada é **101** e, neste ponto, já temos **10**. Entretanto, se o bit recebido for **1**, o sistema permanece em **S1**, pois isso indica o início da sequência desejada, formando **S10**.

Ao receber outro **1**, o sistema completa a sequência **101** e avança para o estado **S101**, onde a saída será **1**. Nos demais estados anteriores, a saída permanece em **0**, mesmo quando há mudanças de estado.

Se o sistema estiver no estado **S101** e receber o bit **1**, ele retornará ao estado **S1**, já que o primeiro bit da nova sequência foi recebido. Contudo, se o bit recebido for **0**, o sistema volta ao estado **S0**, reiniciando a busca pela sequência desejada.

Entrada:

- X: Sinal binário de entrada.
- Sinais de controle: Clock, Set e Reset.

Saída:

- Y: Ativada (1) apenas quando a sequência "101" for detectada.

Estados e Transições:

- Estados representados por E1 e E0, armazenados nos flip-flops D.
- As transições são determinadas pela entrada X e pelos estados atuais.

Portas Lógicas:

- Primeira porta AND:
  - Entradas: X, E1 e ¬E0.
- Segunda porta AND:
  - Entradas: ¬X , E0.
- Porta OR:
  - Entradas: Saídas das duas primeiras portas AND.
  - Saída: Ligada à entrada D1 do flip-flop que define E1.
- Entrada D0: Ligada diretamente à entrada X.
- Porta AND final:
  - Entradas: E1 e E0.
  - Saída: Y, indicando a detecção da sequência.

Inicialização:

- O sinal de Reset é usado para inicializar os flip-flops (E1 e E0=0).

Tabela de Transições de Estado

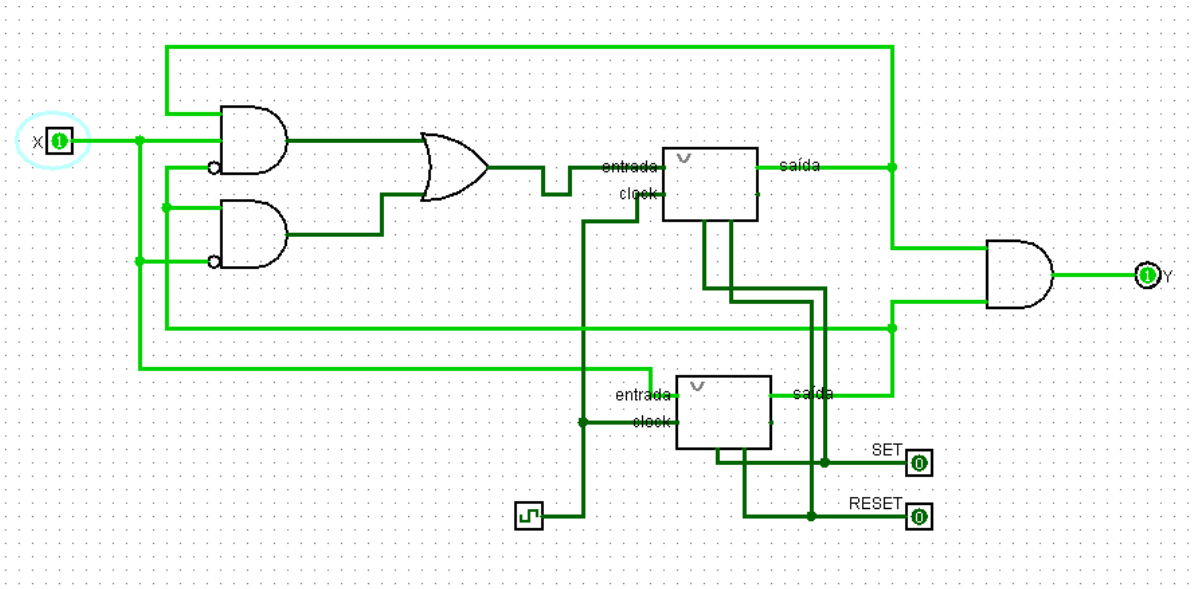
Estado Atual	**Estado Atual**	**Entrada**	**Estado Prox.**	**Estado Prox.**	**Saída**
E1	**E0**	**X**	**E1+**	**E0+**	**Y**
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0

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

Testes Realizados

Os testes foram realizados em um simulador digital para validar as transições de estado e o comportamento da saída Y.

1. **Teste de Detecção da Sequência "101"**:
    - **Entradas**: X={1,0,1}**Estados Resultantes**:
        - E1=0, E0=1.
        - E1=1, E0=0.
        - E1=1, E0=1.
    - **Saída**: Y=1 após a sequência ser detectada.
2. **Teste com Sequências Diferentes**:
    - Entradas que não formam "101" resultaram em Y=0, como esperado.



2.13 Contador síncrono

O circuito descrito é um contador síncrono projetado para contar valores em binário de 0 a 7, utilizando quatro Flip-Flops do tipo T. Ele é amplamente utilizado em sistemas digitais

para operações sequenciais, como contagem de eventos, controle de estado e temporização.

Componentes Utilizados:

1. **Flip-Flop T**

- Quatro Flip-Flops T são usados, cada um responsável por armazenar e alternar os bits do contador.
- Entradas: T, Set, Reset, e Clock.
- Saídas: Q (estado atual) e ~Q (inverso do estado atual).

2. **Portas Lógicas AND:**

- Conectadas às saídas dos Flip-Flops T, combinando os sinais necessários para garantir que a contagem ocorra de forma sincronizada.
- Garantem a propagação correta do pulso de Clock para os Flip-Flops subsequentes.

3. **Clock**

- Sinal de entrada/ que sincroniza todas as mudanças de estado no circuito.
- Cada Flip-Flop alterna seu estado com base na entrada do Clock e nas condições das portas AND.

4. **Entradas Set e Reset**

- Set: Permite definir manualmente os Flip-Flops para 1.
- Reset: Reseta os Flip-Flops para 0, reiniciando o contador.

Funcionamento do Circuito:

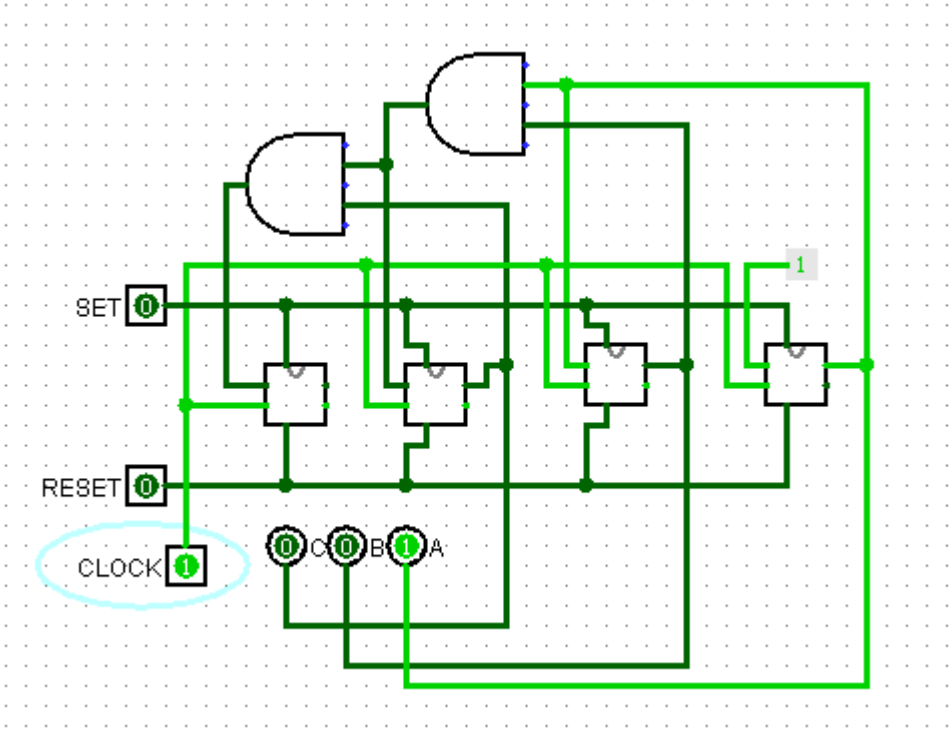
1. **Configuração Inicial**:
    - No início, todos os Flip-Flops estão em 0 (estado resetado).
    - A entrada do primeiro Flip-Flop é conectada a uma constante de valor 1, para garantir que ele alterne a cada pulso do Clock.
2. **Propagação do Sinal**:
    - O Flip-Flop T altera seu estado quando sua entrada T é 1 e ocorre um pulso de Clock.
    - As saídas dos Flip-Flops T são usadas como entradas para as portas AND, que determinam quando os Flip-Flops subsequentes devem alternar.
3. **Contagem**:
    - A cada pulso de Clock, o primeiro Flip-Flop T alterna.
    - As saídas dos Flip-Flops anteriores (em conjunto com as portas AND) determinam se os Flip-Flops subsequentes também devem alternar.
    - O resultado é uma contagem binária sincronizada, que varia de 000 a 111 (ou 001 a 111, dependendo da configuração inicial).
4. **Ciclo Completo**:

- Após atingir o valor máximo (111 em binário), o contador retorna ao estado inicial (000) e o ciclo recomeça.

Especificação das Saídas:

- **Q1**: Saída do primeiro Flip-Flop, representa o bit menos significativo (LSB).
- **Q2**: Saída do segundo Flip-Flop.
- **Q3**: Saída do terceiro Flip-Flop.
- **Q4**: Saída do quarto Flip-Flop, representa o bit mais significativo (MSB).

Imagem do Circuito

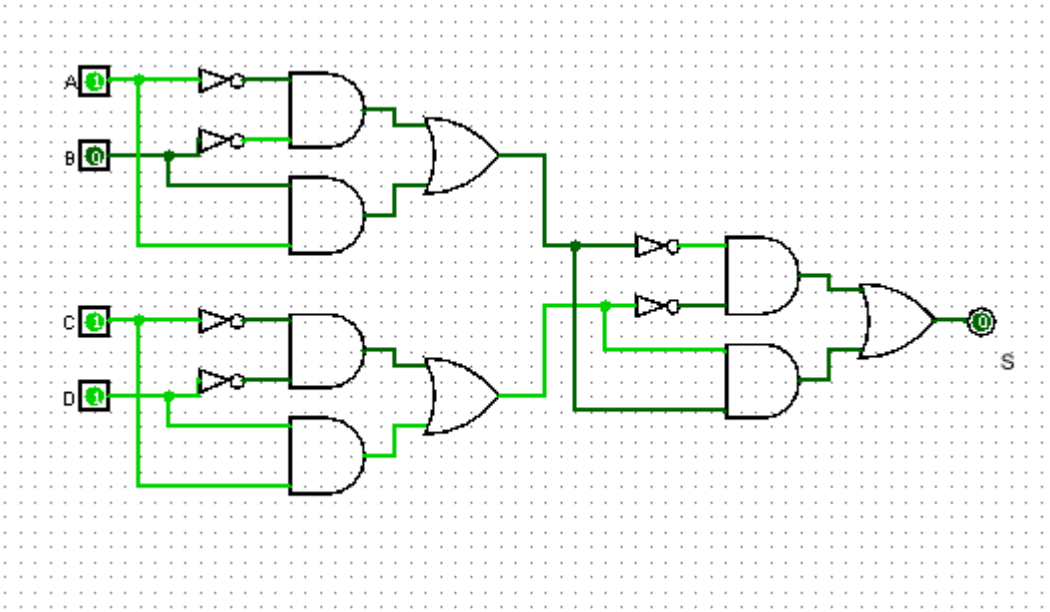


Testes do Circuito

1. **Teste de Incremento**:
   - Configure o circuito com todos os Flip-Flops em 0.
   - Aplique pulsos de Clock e verifique as saídas, que devem seguir a sequência binária (000, 001, 010, ..., 111).
2. **Teste de Reset**:
   - Aplique o sinal de Reset enquanto o contador está em operação.
   - Todas as saídas devem retornar a 0 imediatamente.
3. **Teste de Set**:
   - Aplique o sinal de Set enquanto o contador está em operação.
   - Todas as saídas devem ser configuradas para 1.

2.14 Detector de paridade ímpar (entrada com número ímpar de 1s)

É um circuito combinacional que verifica se o número total de bits com valor 1 em uma entrada binária é ímpar. Se for ímpar, a saída do detector será 0; caso contrário, será 1.



Testes do Circuito

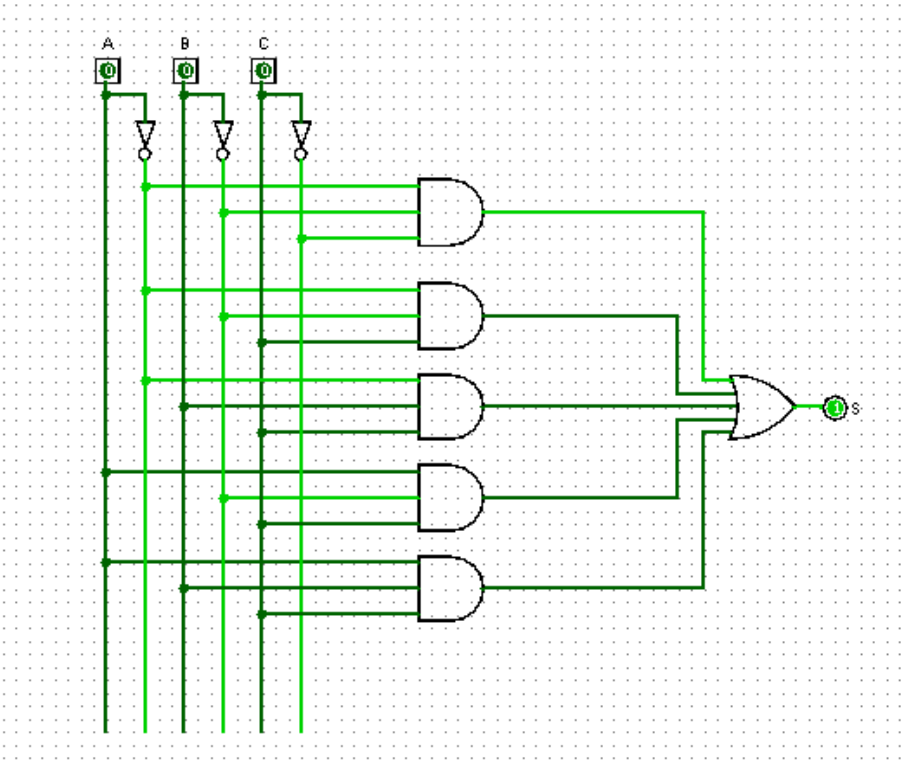
Testando todas as combinações possíveis de entradas (2^4 = 16), obtemos a seguinte tabela verdade:

A	B	C	D	Número de 1s	Saída (S)
0	0	0	0	0 (par)	1
0	0	0	1	1 (ímpar)	0
0	0	1	0	1 (ímpar)	0
0	0	1	1	2 (par)	1
0	1	0	0	1 (ímpar)	0
0	1	0	1	2 (par)	1
…	…	…	…	…	…
1	1	0	1	3 (ímpar)	0
1	1	1	0	3 (ímpar)	0
1	1	1	1	4 (par)	1

Os testes confirmaram que o detector funciona corretamente para todas as combinações de entradas. A saída S é 1 apenas quando o número de bits 1 é par, de acordo com a tabela verdade. Significa que a entrada em questão precisa de um bit 1 para ter o número de bits ímpares.

2.15 Otimização lógica utilizando mapas de Karnaugh

Um circuito foi implementado com 3 entradas e 5 portas lógicas tendo como expressão S = $\overline{A}\overline{B}\overline{C}$ + $\overline{A}\overline{B}C$ + $\overline{A}B\overline{C}$ + $A\overline{B}C$ + ABC. Circuito implementado no *Logisim:*



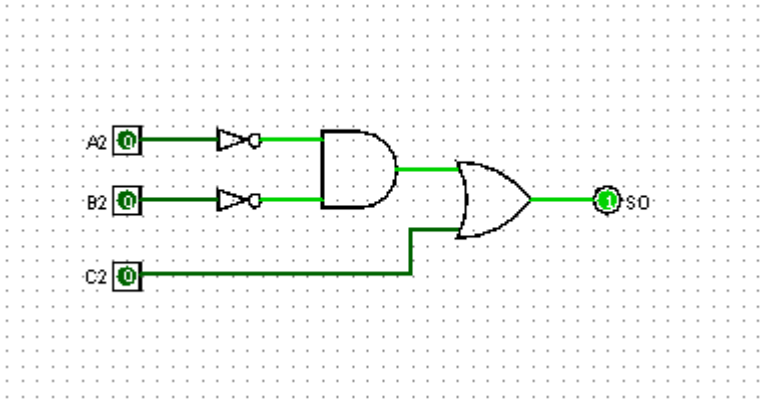
Otimização com mapa de Karnaugh

S = $\overline{A}\overline{B}\overline{C}$ + $\overline{A}\overline{B}C$ + $\overline{A}B\overline{C}$ + $A\overline{B}C$ + ABC

	A		**~A**	
B	0	1	1	0
~B	0	1	1	1
	~C	**C**		**~C**

Circuito otimizado:

S0 = $\overline{AB} + C$



S0 = $\overline{AB} + C$, $\Rightarrow S \equiv S0$.

2.16 Decodificador de 7 Segmentos

Converte valores binários de 4 bits para números de 0 a 9 em sinais que controlam os segmentos de um display de 7 segmentos. O display contém 7 LEDs, organizados em formato de "8", e os segmentos são ativados de acordo com o número binário de entrada.

Entradas:

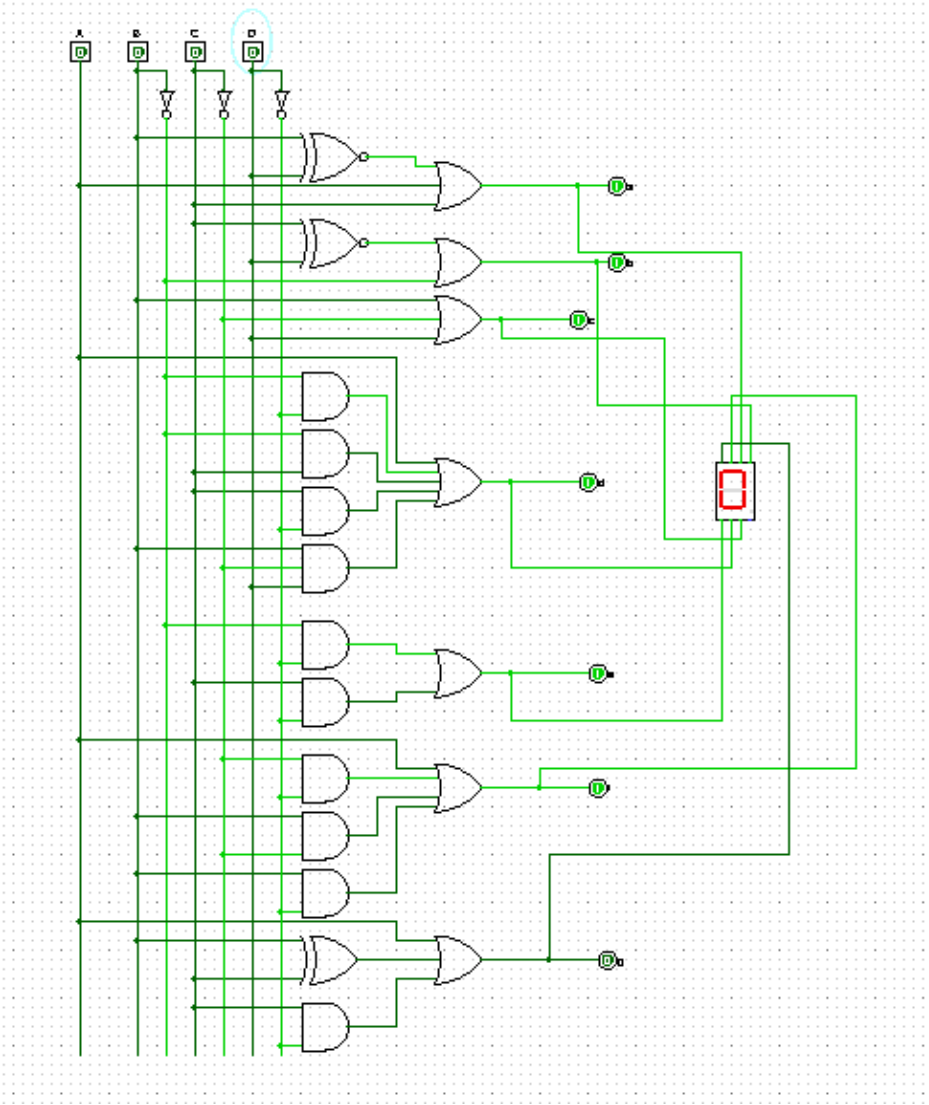
4 bits para representar números de 0 a 9.

Saída:

7 sinais (a,b,c,d,e,f,g) que ativam os LEDs do display e escrevem o número.

Cada segmento (a,b,c,d,e,f,g) é controlado por uma expressão lógica, derivada da tabela verdade e simplificada com mapas de Karnaugh.

Imagem do Circuito:



Testes do circuito

Ao inserir quatro inputs de 1 bit para as entradas, serão ligados os segmentos correspondentes de acordo com a tabela verdade.

Tabela verdade

ABCD	a	b	c	d	e	f	g	Número
0000	1	1	1	1	1	1	0	0
0001	0	1	1	0	0	0	0	1
0010	1	1	0	1	1	0	1	2
0011	1	1	1	1	0	0	1	3
0100	0	1	1	0	0	1	1	4
0101	1	0	1	1	0	1	1	5

0110	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**6**
0111	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**7**
1000	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**8**
1001	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**9**

2.17 Detector de Número Primo

O objetivo é criar um circuito que detecte se uma entrada binária de 4 bits (A, B, C, D) representa um número primo. O circuito utiliza portas lógicas e simplificação com mapas de Karnaugh para reduzir a complexidade.

Tabela verdade

A	B	C	D	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
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	0
1	1	1	1	0

Simplificação:

A expressão final após simplificação com Karnaugh é:

x = ¬A¬BC + ¬A¬BD + ¬BDC + ABD¬C

Implementação no Circuito

1. **Entradas**: 4 bits (A, B, C, D).

2. **Portas AND.**

- **AND1:** ~A~BC
- **AND2:** ~AD~B
- **AND3:** ~BDC
- **AND4:** AB~CD

3. **Portas OR.**

- As saídas das 4 portas AND são combinadas com uma porta OR para gerar a saída final.

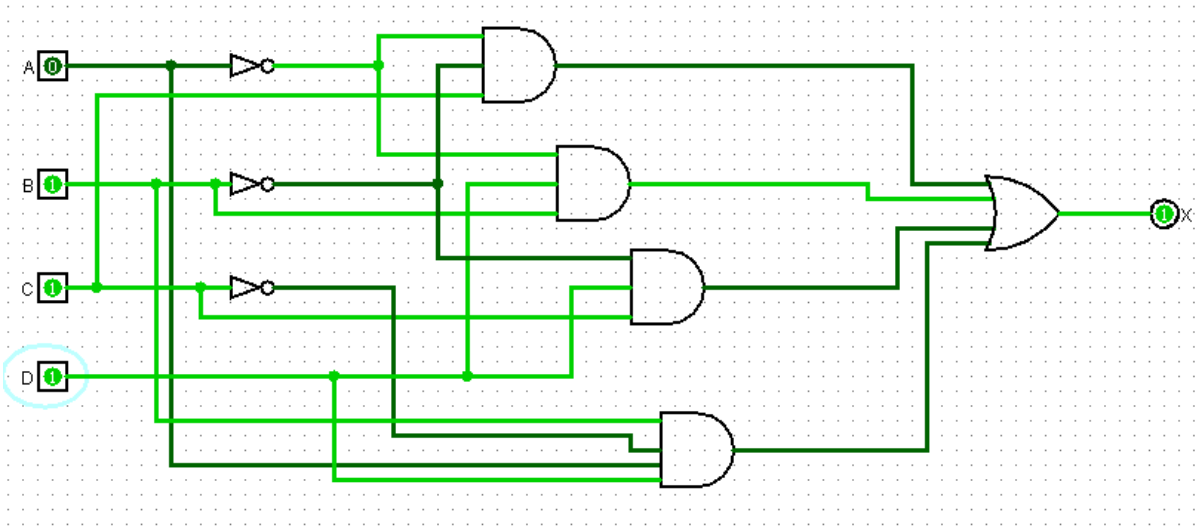
Circuito Lógico

Inversores (NOT): Para gerar a negação das entradas A, B, C e D.

Portas AND: Implementam os termos simplificados.

Porta OR:Combina os resultados das portas AND para obter a saída.

Imagem do Circuito



Funcionamento do Circuito

- Quando a entrada binária representa um número primo (2, 3, 5, 7, 11, 13), a saída será 1.
- Para todas as outras entradas, a saída será 0.

3 REFERÊNCIAS

Idoeta, Ivan Valeije. **Elementos de Eletrônica Digital**; Contribuição de Francisco Capuano. 40. edição. São Paulo: Editora Érica LTDA., 2008.

Stallings, William. **Arquitetura e Organização de Computadores**; Contribuição de Peter Zeno. 10. edição. São Paulo: Pearson Education do Brasil, 2017.

Dr. Vieira, Marcelo. **SEL 0414 - Sistemas Digitais: Aula 2 - Paridade**. Disponível em: https://edisciplinas.usp.br/pluginfile.php/8145048/mod_resource/content/3/Aula%202%20-%20Paridade.pdf. Acesso em 06 de dezembro de 2024.

Vídeos que auxiliaram no processo de criação dos circuitos:

- https://youtu.be/6sBkmAlJlvs?si=Uy1JUHxK1VVrpAEo
- https://youtu.be/IAXOvGVvxxU?si=u5sRogsPkuDnJ3ve