UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIA E TECNOLOGIA - CCT
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO – DCC
DCC301 - ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES
PROF. HERBERT OLIVEIRA ROCHA

LABORATÓRIO DE CIRCUITOS – CODIFICAÇÃO E SIMULAÇÕES.

Boa Vista - 11/10/2025 - RR

Helthe Magalhães
Tiago Lopes
Rafael da Silva

LABORATÓRIO DE CIRCUITOS – CODIFICAÇÃO E SIMULAÇÕES.

Elaborar um relatório técnico detalhado das soluções, incluindo a metodologia de testes de unidade ou análise de tabela verdade. Todos os artefatos, como o relatório e o código-fonte dos programas, devem ser anexados.

Boa Vista - RR

Relatório Técnico: Implementação de Componentes de Lógica Digital no Logisim

Este relatório técnico detalha a implementação, descrição e testes de diversos componentes de lógica digital criados utilizando o simulador Logisim. Para cada componente, são apresentados sua descrição, a imagem do circuito, a metodologia de teste (análise da tabela verdade) e uma descrição detalhada dos resultados obtidos.

[COMPONENTE 01] Registrador Flip-Flop do tipo D e do tipo JK

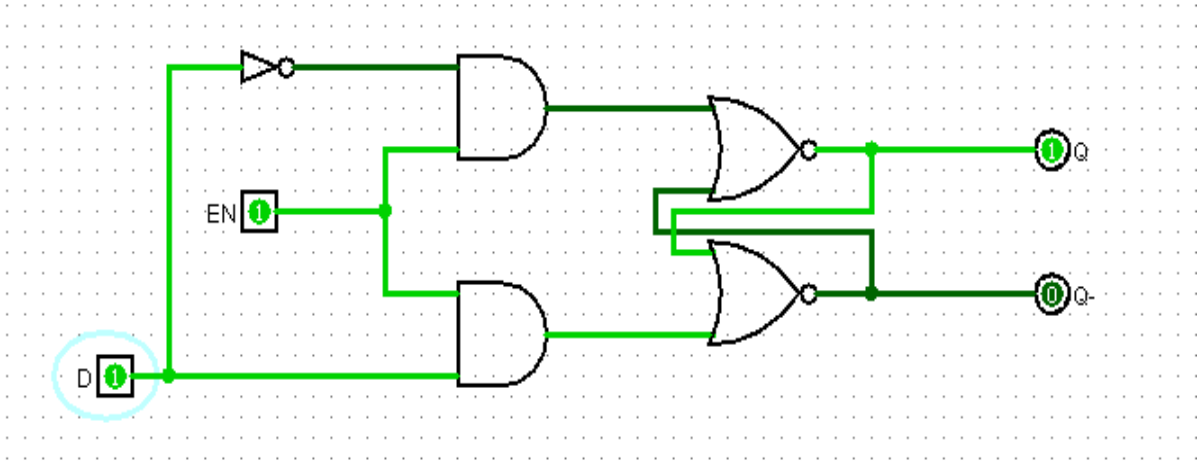
(i) Descrição

Os flip-flops são circuitos sequenciais de um bit, fundamentais para o armazenamento de informações.

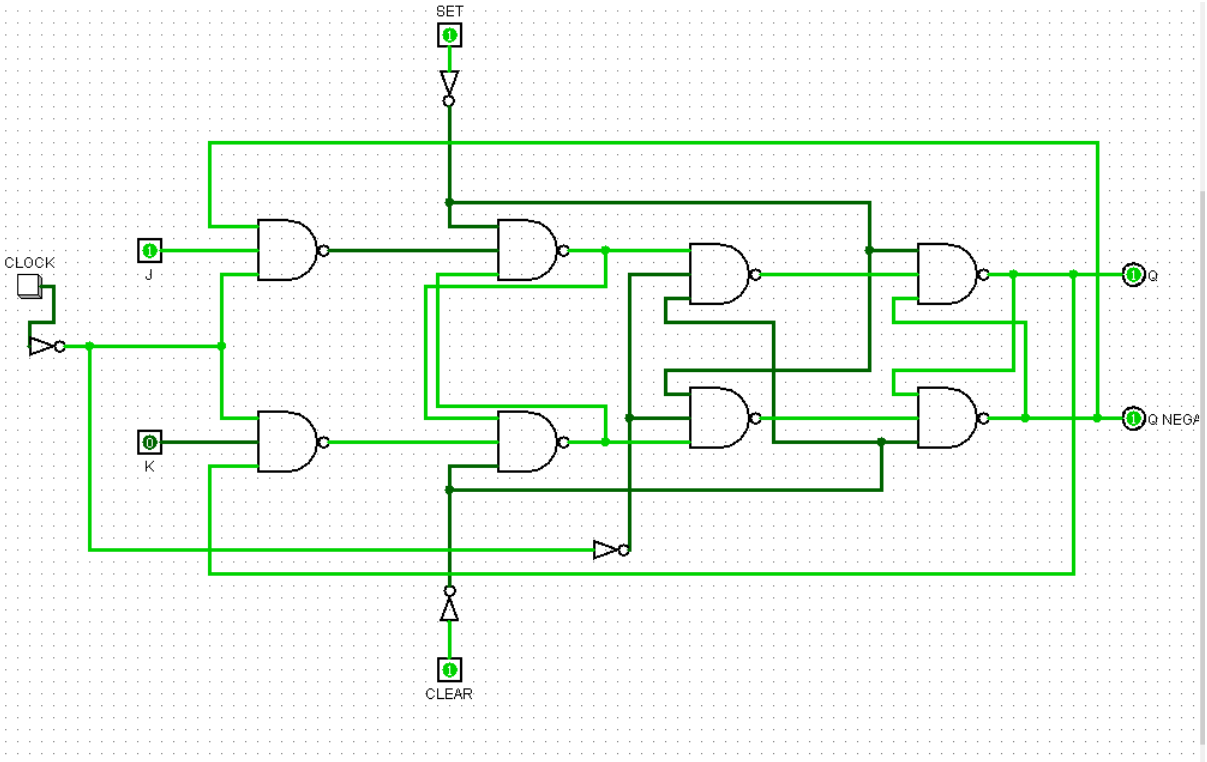
- **Flip-Flop D:** Possui uma entrada de dados (D) e uma entrada de clock (CLK). O valor na entrada D é transferido para a saída Q na borda de subida do clock. A saída permanece inalterada entre os pulsos de clock.
- **Flip-Flop JK:** Considerado um flip-flop universal, possui entradas J e K e uma entrada de clock (CLK). O comportamento depende da combinação das entradas J e K:
    - **J=0, K=0:** Mantém o estado.
    - **J=0, K=1:** Reseta (Q vai para 0).
    - **J=1, K=0:** Seta (Q vai para 1).
    - **J=1, K=1:** Alterna (inverte o estado de Q).

(ii) Imagem dos Componentes

- **Flip-Flop D:**



- **Flip-Flop JK:**



(iii) Metodologia de Teste

A validação de ambos os flip-flops foi realizada utilizando a análise da tabela verdade, verificando o comportamento das saídas (Q e ¬Q) em resposta a diferentes combinações das entradas D (ou J e K) e pulsos de clock.

(iv) Descrição dos Testes

- **Flip-Flop D:**
  - **Pinos de entrada:** `D`, `CLK`.
  - **Pinos de saída:** `Q`, `¬Q`.
  - **Teste 1:** `D=1`, pulso de `CLK`. Resultado: `Q=1`, `¬Q=0`. A entrada D foi armazenada corretamente.
  - **Teste 2:** `D=0`, pulso de `CLK`. Resultado: `Q=0`, `¬Q=1`. A entrada D foi armazenada corretamente.
- **Flip-Flop JK:**
  - **Pinos de entrada:** `J`, `K`, `CLK`.
  - **Pinos de saída:** `Q`, `¬Q`.
  - **Teste 1:** `J=0`, `K=0`, pulso de `CLK`. Resultado: O estado anterior de `Q` é mantido.
  - **Teste 2:** `J=0`, `K=1`, pulso de `CLK`. Resultado: `Q=0`, `¬Q=1` (Reset).

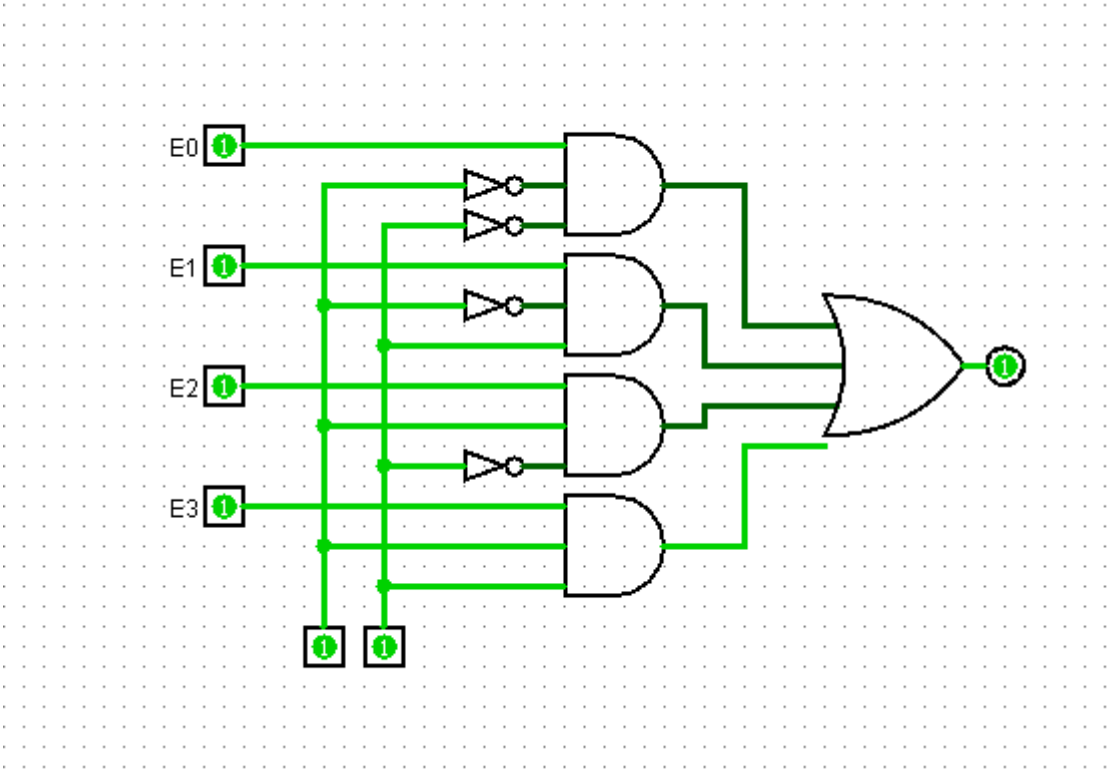
- **Teste 3:** `J=1`, `K=0`, pulso de `CLK`. Resultado: `Q=1`, `¬Q=0` (Set).
- **Teste 4:** `J=1`, `K=1`, pulso de `CLK`. Resultado: `Q` e `¬Q` alternam de estado.

[COMPONENTE 02] Multiplexador de quatro opções de entrada

(i) Descrição

Um multiplexador (MUX) é um seletor de dados que conecta várias entradas a uma única saída. O multiplexador de 4 entradas seleciona uma das quatro entradas (`I0`, `I1`, `I2`, `I3`) com base nas duas entradas de seleção (`S0`, `S1`) e a conecta à saída (`Saída`).

(ii) Imagem do Componente



(iii) Metodologia de Teste

Os testes foram realizados aplicando diferentes combinações nos pinos de seleção (`S0`, `S1`) e observando se a entrada correspondente era direcionada para a saída. A validação foi feita pela análise da tabela verdade.

(iv) Descrição dos Testes

- **Pinos de entrada:** `I0`, `I1`, `I2`, `I3`, `S0`, `S1`.

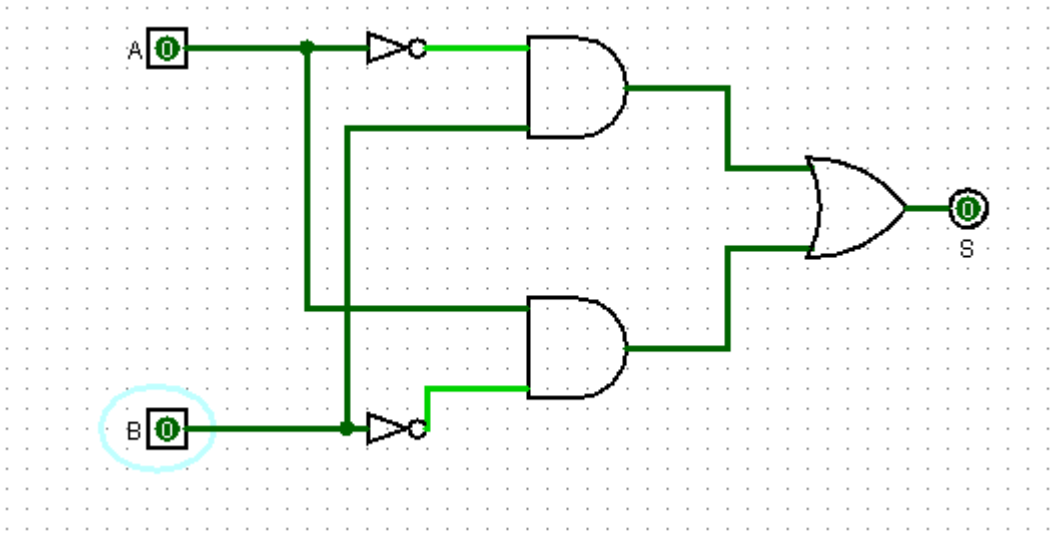
- **Pinos de saída:** `Saída`.
- **Teste 1:** `S1=0`, `S0=0`. Conexão ativa: `I0`. Resultado: `Saída = I0`.
- **Teste 2:** `S1=0`, `S0=1`. Conexão ativa: `I1`. Resultado: `Saída = I1`.
- **Teste 3:** `S1=1`, `S0=0`. Conexão ativa: `I2`. Resultado: `Saída = I2`.
- **Teste 4:** `S1=1`, `S0=1`. Conexão ativa: `I3`. Resultado: `Saída = I3`.

[COMPONENTE 03] Porta lógica XOR usando os componentes: AND, NOT e OR

(i) Descrição

A porta lógica XOR (Exclusive OR) produz uma saída verdadeira (1) se as entradas forem diferentes e uma saída falsa (0) se forem iguais. Sua lógica pode ser implementada com as portas AND, NOT e OR, seguindo a equação booleana: Saı´da=(A·¬B)+(¬A·B).

(ii) Imagem do Componente



(iii) Metodologia de Teste

O componente foi testado com base em sua tabela verdade, validando se a saída era 1 apenas quando as entradas tinham valores diferentes.

(iv) Descrição dos Testes

- **Pinos de entrada:** `A`, `B`.
- **Pinos de saída:** `Saída`.
- **Teste 1:** `A=0`, `B=0`. Resultado: `Saída = 0`.
- **Teste 2:** `A=0`, `B=1`. Resultado: `Saída = 1`.

- **Teste 3:** `A=1`, `B=0`. Resultado: `Saída = 1`.
- **Teste 4:** `A=1`, `B=1`. Resultado: `Saída = 0`.

[COMPONENTE 04] Somador de 8 bits que recebe um valor inteiro e soma com o valor 4

(i) Descrição

Este somador é um circuito combinado que adiciona o valor `4` a uma entrada de 8 bits. Ele utiliza um somador completo (full adder) para cada bit, propagando o "vai um" (carry) para a próxima etapa, com o segundo operando sendo fixo (00000100 em binário).

(ii) Imagem do Componente

(iii) Metodologia de Teste

Foram realizados testes de unidade, com entradas de 8 bits variadas, para verificar se a saída correspondia à soma correta.

(iv) Descrição dos Testes

- **Pinos de entrada:** `Entrada [7:0]`.
- **Pinos de saída:** `Soma [7:0]`.
- **Teste 1:** `Entrada = 0` (00000000). Resultado: `Soma = 4` (00000100).
- **Teste 2:** `Entrada = 12` (00001100). Resultado: `Soma = 16` (00010000).
- **Teste 3:** `Entrada = 250` (11111010). Resultado: `Soma = 254` (11111110).

[COMPONENTE 05] Memória ROM de 8 bits

(i) Descrição

Uma Memória de Somente Leitura (ROM) é um tipo de memória não volátil. A ROM de 8 bits implementada armazena um conjunto de valores pré-definidos que são recuperados com base em um endereço de entrada. O conteúdo da memória é fixo e não pode ser alterado durante a operação.

(ii) Imagem do Componente

(iii) Metodologia de Teste

A validação foi feita através de testes de unidade, onde foram fornecidos diferentes endereços de entrada para verificar se os dados pré-carregados eram lidos corretamente na saída.

(iv) Descrição dos Testes

- **Pinos de entrada:** `Endereço`.
- **Pinos de saída:** `Dados`.
- **Teste 1:** `Endereço = 0`. Conexão ativa: Endereço `0000...000`. Resultado: `Dados = Valor_pré-definido_no_endereço_0`.
- **Teste 2:** `Endereço = 1`. Conexão ativa: Endereço `0000...001`. Resultado: `Dados = Valor_pré-definido_no_endereço_1`.
- **Teste 3:** `Endereço = 2`. Conexão ativa: Endereço `0000...010`. Resultado: `Dados = Valor_pré-definido_no_endereço_2`.

[COMPONENTE 06] Memória RAM de 8 bits

(i) Descrição

Uma Memória de Acesso Aleatório (RAM) é um tipo de memória volátil que permite a leitura e a escrita de dados em qualquer endereço. A RAM de 8 bits pode armazenar e recuperar dados. As entradas `WE` (Write Enable) e `Addr` (Address) controlam a operação: quando `WE=1`, os dados da entrada são escritos no endereço selecionado. Quando `WE=0`, a memória está em modo de leitura e o dado do endereço é lido.

(ii) Imagem do Componente

(iii) Metodologia de Teste

O teste seguiu a metodologia de leitura e escrita. Primeiro, dados foram escritos em endereços específicos com `WE=1`. Em seguida, o `WE` foi desabilitado (`WE=0`) e o mesmo endereço foi acessado para confirmar se os dados escritos foram corretamente recuperados.

(iv) Descrição dos Testes

- **Pinos de entrada:** `WE`, `Addr`, `Dados de Entrada`.
- **Pinos de saída:** `Dados de Saída`.
- **Teste de Escrita:** `WE=1`, `Addr=2`, `Dados de Entrada=150` (10010110). O valor 150 é escrito no endereço 2.

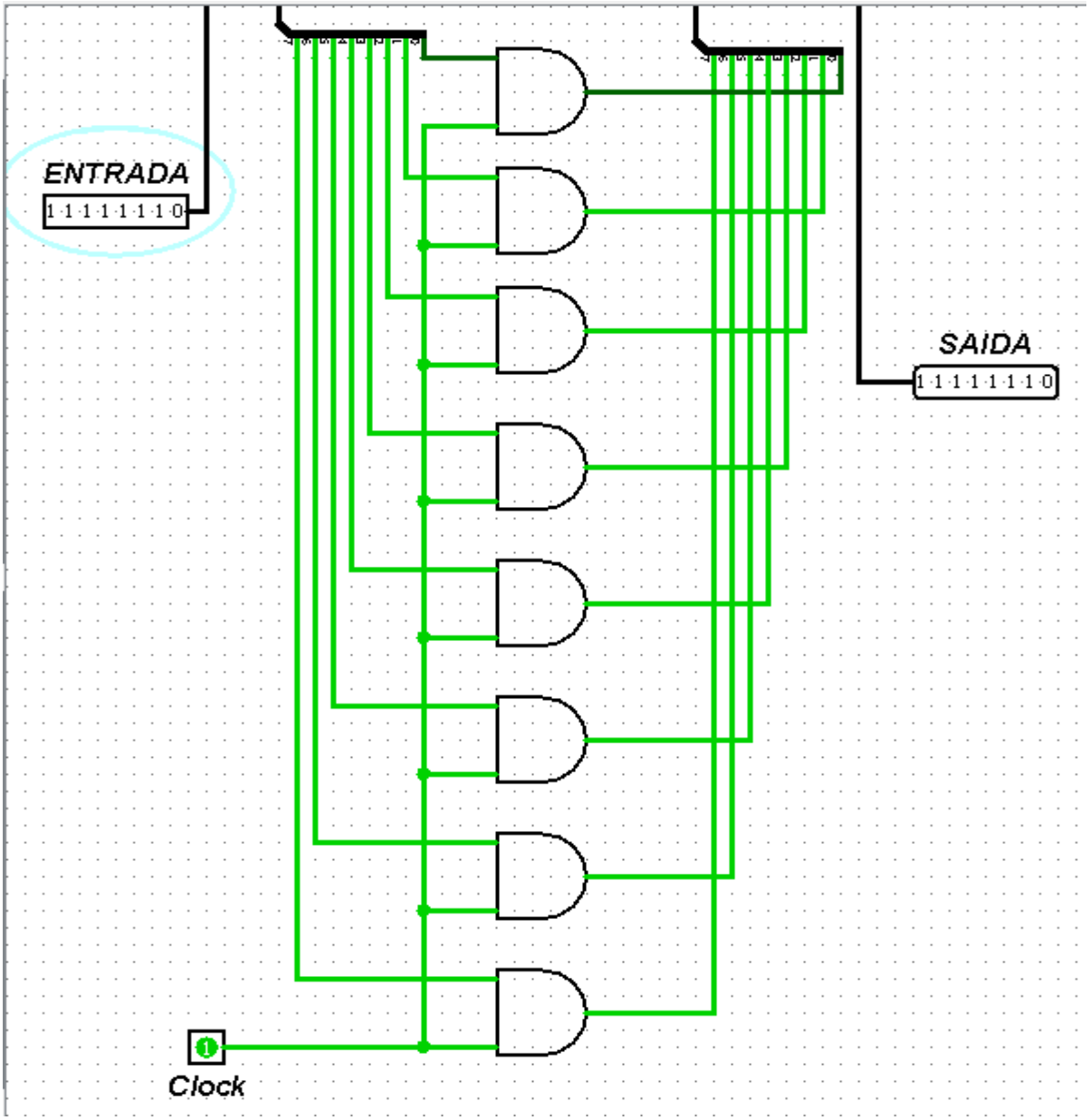
- **Teste de Leitura:** `WE=0`, `Addr=2`. Conexão ativa: Endereço 2. Resultado: `Dados de Saída = 150`.

[COMPONENTE 07] Banco de Registradores de 8 bits

(i) Descrição

Um Banco de Registradores é um conjunto de registradores de armazenamento de dados, tipicamente utilizados na CPU. Este componente é um banco de registradores de 8 bits, permitindo que a CPU armazene dados temporariamente. Ele possui entradas de dados, um pino de seleção de registrador e saídas de dados.

(ii) Imagem do Componente



(iii) Metodologia de Teste

O teste foi realizado simulando operações de leitura e escrita. Foram salvos dados em diferentes registradores e, em seguida, verificou-se se os dados foram lidos corretamente.

(iv) Descrição dos Testes

- **Pinos de entrada:** `Entrada de Dados [7:0]`, `Número do Registrador [2:0]`, `Habilitar Escrita`.
- **Pinos de saída:** `Saída de Dados [7:0]`.
- **Teste de Escrita:** `Habilitar Escrita=1`, `Número do Registrador=1`, `Entrada de Dados=10`. O valor 10 foi salvo no registrador 1.
- **Teste de Leitura:** `Habilitar Escrita=0`, `Número do Registrador=1`. Conexão ativa: Registrador 1. Resultado: `Saída de Dados = 10`.

[COMPONENTE 08] Somador de 8 bits

(i) Descrição

Este somador é um circuito combinacional que adiciona dois números de 8 bits. Ele é construído com 8 somadores completos (full adders), que somam os bits individuais e propagam o bit de "vai um" (carry-out) para o próximo somador. A entrada `Cin` (Carry-in) e a saída `Cout` (Carry-out) completam o circuito.

(ii) Imagem do Componente

(iii) Metodologia de Teste

A validação foi realizada com testes de unidade, somando diferentes pares de números de 8 bits e verificando se o resultado era o esperado.

(iv) Descrição dos Testes

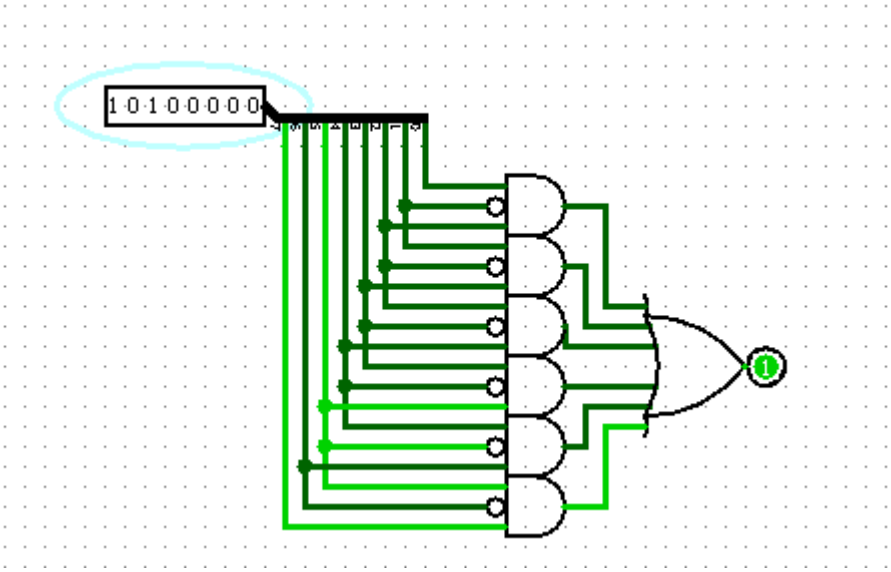
- **Pinos de entrada:** `A [7:0]`, `B [7:0]`, `Cin`.
- **Pinos de saída:** `Soma [7:0]`, `Cout`.
- **Teste 1:** `A=10` (00001010), `B=5` (00000101). Resultado: `Soma = 15` (00001111), `Cout=0`.
- **Teste 2:** `A=200` (11001000), `B=100` (01100100). Resultado: `Soma = 44` (00101100), `Cout=1`.

[COMPONENTE 09] Detector de sequência binária "101"

(i) Descrição

Este circuito sequencial detecta a ocorrência da sequência binária "101" em um fluxo de entrada de bits. O circuito é construído utilizando flip-flops para armazenar os estados e portas lógicas para determinar o próximo estado e a saída. Quando a sequência `101` é completamente recebida, a saída é ativada (1).

(ii) Imagem do Componente



(iii) Metodologia de Teste

O teste seguiu a análise de um fluxo de entrada de bits, observando a saída do circuito em cada pulso de clock. A validação foi feita pela verificação da tabela de estados do circuito.

(iv) Descrição dos Testes

- **Pinos de entrada:** `Entrada`, `CLK`.
- **Pinos de saída:** `Saída`.
- **Sequência de entrada:** `...0, 1, 0, 1, 0...`
- **Teste 1:** Entrada `0`, `CLK` pulsa. Saída permanece `0`.
- **Teste 2:** Entrada `1`, `CLK` pulsa. Saída permanece `0`.
- **Teste 3:** Entrada `0`, `CLK` pulsa. Saída permanece `0`.
- **Teste 4:** Entrada `1`, `CLK` pulsa. A sequência `101` é completada. Resultado: `Saída = 1`.

[COMPONENTE 10] ULA de 8 bits

(i) Descrição

Uma Unidade Lógica e Aritmética (ULA) é um bloco fundamental de um processador. Esta ULA de 8 bits executa diversas operações lógicas e aritméticas em duas entradas de 8 bits (`A`, `B`) com base em um código de operação de 4 bits (`S [3:0]`). As operações implementadas são: AND, OR, NOT, NOR, NAND, XOR, SHIFT de 2 bits à esquerda, SHIFT de 2 bits à direita, soma e subtração.

(ii) Imagem do Componente

(iii) Metodologia de Teste

O teste foi realizado em modo de teste de unidade, aplicando-se diferentes códigos de operação (`S`) e entradas (`A`, `B`) e verificando se a saída (`Resultado`) era a esperada para cada operação.

(iv) Descrição dos Testes

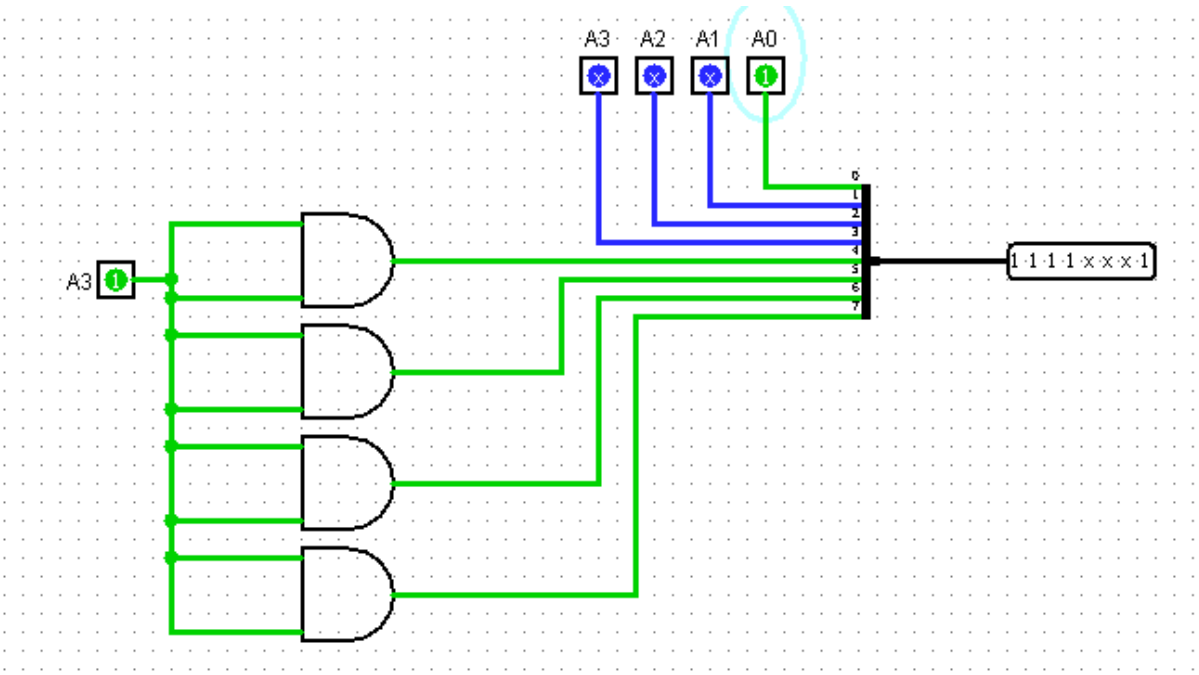
- **Pinos de entrada:** `A [7:0]`, `B [7:0]`, `S [3:0]`.
- **Pinos de saída:** `Resultado [7:0]`.
- **Teste de Soma:** `S=1010`, `A=5`, `B=10`. Resultado: `Resultado = 15`.
- **Teste de AND:** `S=0000`, `A=10` (1010), `B=6` (0110). Resultado: `Resultado = 2` (0010).
- **Teste de SHIFT Esquerda:** `S=0110`, `A=4` (00000100). Resultado: `Resultado = 16` (00010000).

[COMPONENTE 11] Extensor de sinal de 4 bits para 8 bits

(i) Descrição

Um extensor de sinal é usado para aumentar a largura de um barramento de dados. Este componente recebe uma entrada de 4 bits e a estende para 8 bits, replicando o bit mais significativo (sign extension). Isso é comumente usado em arquiteturas de computadores para tratar números de 4 bits como se fossem de 8 bits, preservando o sinal (positivo ou negativo).

(ii) Imagem do Componente



(iii) Metodologia de Teste

A validação foi feita aplicando entradas de 4 bits e observando se o sinal era corretamente estendido para 8 bits. A análise da tabela verdade simplificada para alguns casos foi utilizada.

(iv) Descrição dos Testes

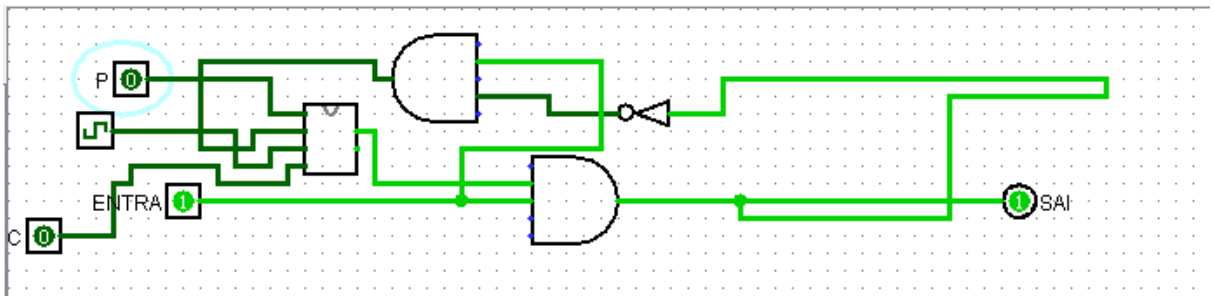
- **Pinos de entrada:** `Entrada [3:0]`.
- **Pinos de saída:** `Saída [7:0]`.
- **Teste 1:** `Entrada = 0101` (5, positivo). Resultado: `Saída = 00000101` (5, positivo).
- **Teste 2:** `Entrada = 1011` (-5, negativo em complemento de 2). Resultado: `Saída = 11111011` (-5, negativo em complemento de 2). O bit de sinal (`1`) foi replicado.

[COMPONENTE 12] Máquina de estados utilizando portas lógicas

(i) Descrição

Uma máquina de estados finita é um modelo matemático que descreve o comportamento de um sistema sequencial com base em seus estados e transições. Esta implementação utiliza flip-flops (para armazenar o estado atual) e portas lógicas combinacionais (para determinar o próximo estado e a saída com base no estado atual e na entrada).

(ii) Imagem do Componente



(iii) Metodologia de Teste

O circuito foi testado seguindo uma tabela de estados e transições pré-definida. A validação foi feita injetando entradas e observando a transição de estado e a saída para cada pulso de clock.

(iv) Descrição dos Testes

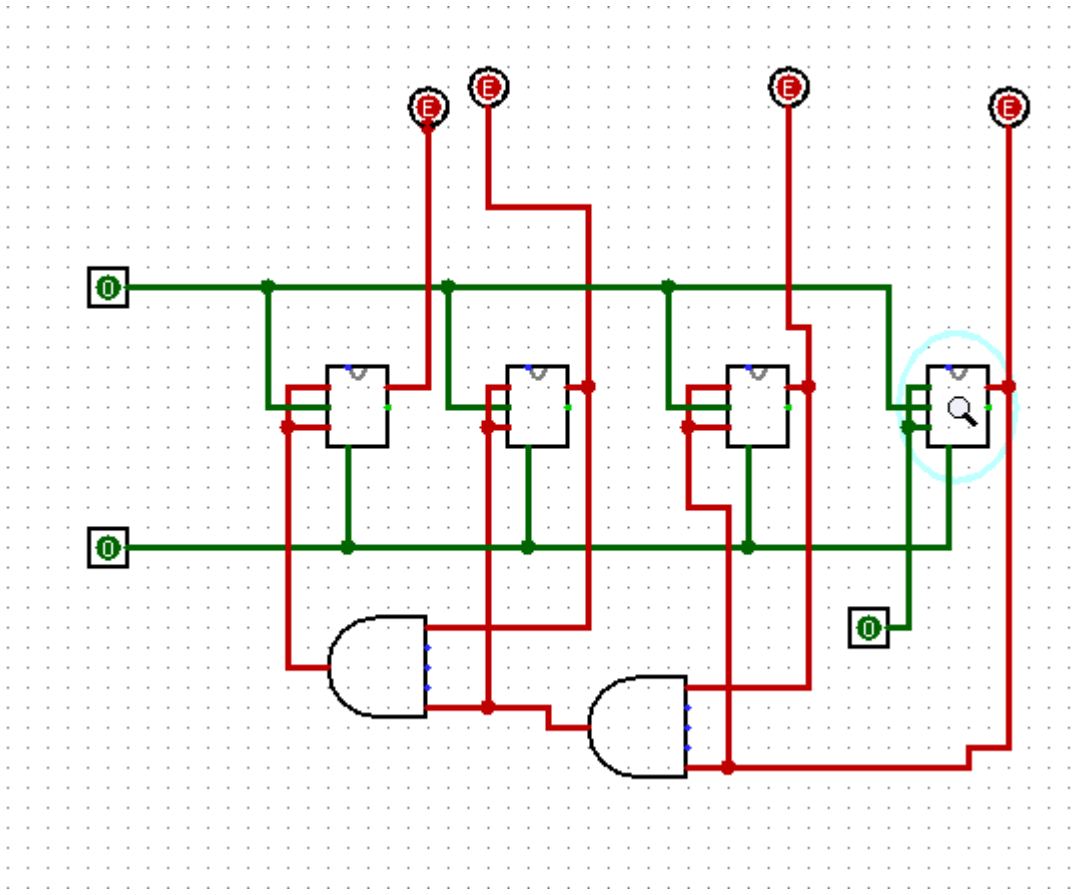
- **Pinos de entrada:** `Entrada`, `CLK`.
- **Pinos de saída:** `Saída`.
- **Teste:**
  - **Estado Inicial:** `S0`. `Entrada=1`. O circuito transita para o estado `S1`. `Saída=0`.
  - **Estado Atual:** `S1`. `Entrada=0`. O circuito transita para o estado `S2`. `Saída=0`.
  - **Estado Atual:** `S2`. `Entrada=1`. O circuito transita para o estado `S0`. `Saída=1`.

[COMPONENTE 13] Contador Síncrono

(i) Descrição

Um contador síncrono é um circuito sequencial onde todos os flip-flops são acionados pelo mesmo pulso de clock, garantindo que as transições de estado ocorram simultaneamente. Este contador conta em uma sequência binária pré-definida. A lógica é implementada com portas lógicas que controlam as entradas dos flip-flops para alcançar a contagem desejada.

(ii) Imagem do Componente



(iii) Metodologia de Teste

O teste foi feito verificando a sequência de contagem na saída do circuito a cada pulso de clock.

(iv) Descrição dos Testes

- **Pinos de entrada:** `CLK`, `Reset`.
- **Pinos de saída:** `Q [n:0]`.
- **Teste de Contagem:**
  - **Inicial:** `Reset=1` (Reseta o contador). Resultado: `Q=0000`.
  - `CLK` pulsa 1 vez. Resultado: `Q=0001`.
  - `CLK` pulsa 2 vezes. Resultado: `Q=0010`.
  - `CLK` pulsa 3 vezes. Resultado: `Q=0011`.
  - ...e assim por diante.

[COMPONENTE 14] Detector de paridade ímpar

(i) Descrição

O detector de paridade ímpar é um circuito combinacional que verifica se a entrada binária possui um número ímpar de bits '1'. Se o número de uns for ímpar, a saída é '1', caso contrário, a saída é '0'. A lógica é implementada combinando portas AND, OR e NOT.

(ii) Imagem do Componente

(iii) Metodologia de Teste

O teste foi realizado com a análise da tabela verdade, aplicando todas as combinações de entrada e verificando se a saída correspondia à paridade ímpar.

(iv) Descrição dos Testes

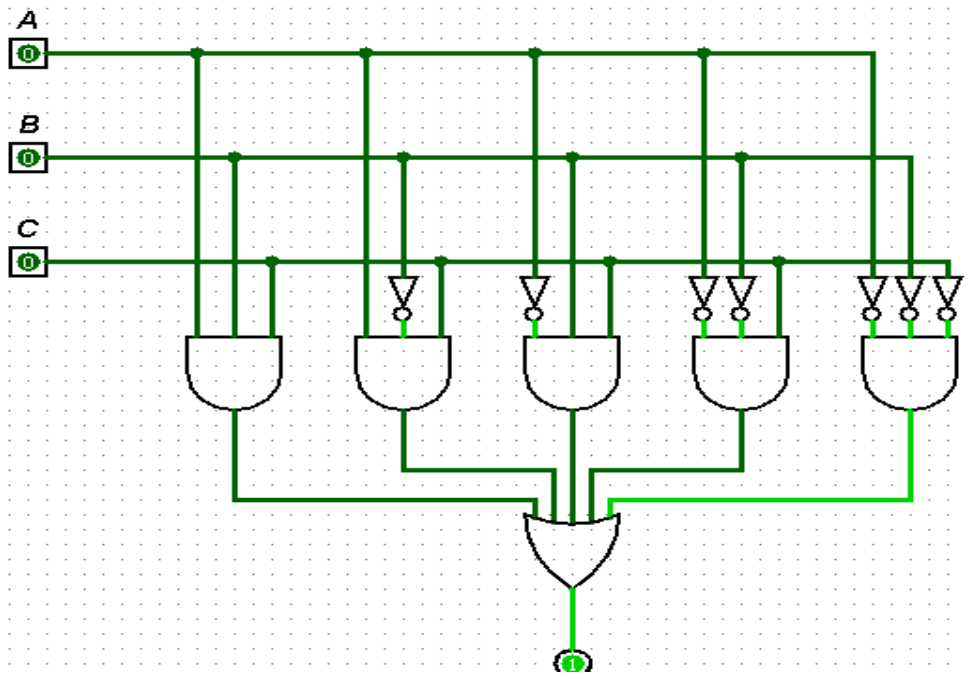
- **Pinos de entrada:** `A`, `B`, `C`, `D`.
- **Pinos de saída:** `Saída`.
- **Teste 1:** `A=0`, `B=0`, `C=1`, `D=0` (Um `1`). Resultado: `Saída = 1`.
- **Teste 2:** `A=1`, `B=1`, `C=1`, `D=0` (Três `1s`). Resultado: `Saída = 1`.
- **Teste 3:** `A=1`, `B=1`, `C=0`, `D=0` (Dois `1s`). Resultado: `Saída = 0`.
- **Teste 4:** `A=0`, `B=0`, `C=0`, `D=0` (Zero `1s`). Resultado: `Saída = 0`.

[COMPONENTE 15] Otimização lógica utilizando mapas de Karnaugh

(i) Descrição

Mapas de Karnaugh (K-map) são uma ferramenta gráfica para simplificar expressões lógicas booleanas. O problema a ser otimizado é a expressão $F(A,B,C)=\neg A\neg B\neg C+\neg AB\neg C+AB\neg C$. A implementação no Logisim reflete o circuito simplificado obtido pelo K-map.

(ii) Imagem do Componente



(iii) Metodologia de Teste

O circuito foi testado usando a tabela verdade da expressão simplificada e comparando-a com a expressão original para garantir que a otimização não alterou o comportamento.

(iv) Descrição dos Testes

- **Pinos de entrada:** `A`, `B`, `C`.
- **Pinos de saída:** `Saída`.
- **Expressão simplificada:** F(A,B,C)=¬B¬C+AB
- **Teste 1:** `A=0`, `B=0`, `C=0`. `Saída` esperada: 1. Resultado: `Saída=1`.
- **Teste 2:** `A=0`, `B=1`, `C=0`. `Saída` esperada: 1. Resultado: `Saída=1`.
- **Teste 3:** `A=1`, `B=1`, `C=0`. `Saída` esperada: 1. Resultado: `Saída=1`.
- **Teste 4:** `A=1`, `B=0`, `C=1`. `Saída` esperada: 0. Resultado: `Saída=0`.

[COMPONENTE 16] Decodificador de 7 Segmentos

(i) Descrição

Um decodificador de 7 segmentos converte uma entrada binária (neste caso, de 4 bits, BCD) para os sinais que acionam os 7 segmentos de um display, permitindo a exibição de caracteres hexadecimais (0-9, A-F). A lógica de cada segmento é implementada com base na tabela verdade que mapeia cada combinação de entrada para o estado (ligado/desligado) do segmento correspondente.

(ii) Imagem do Componente

(iii) Metodologia de Teste

O circuito foi testado em modo de teste de unidade, com entradas de 4 bits de 0 a 15, verificando se os segmentos de saída (`a` a `g`) acendiam corretamente para formar o dígito hexadecimal correspondente.

(iv) Descrição dos Testes

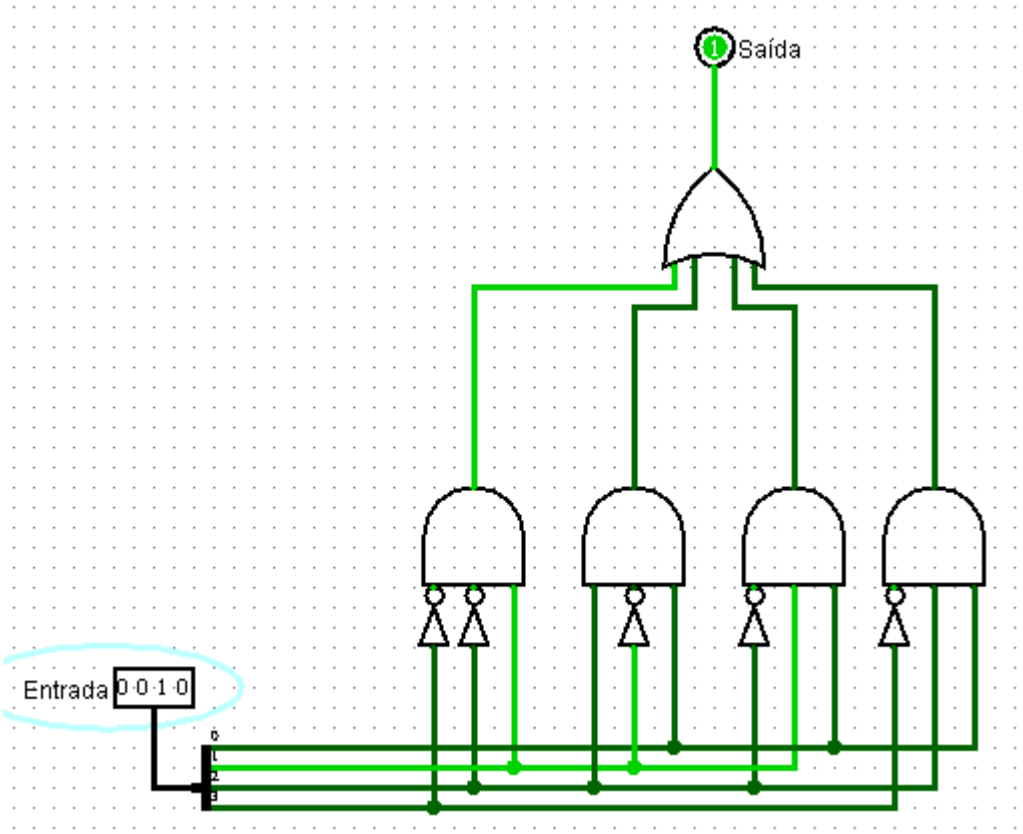
- **Pinos de entrada:** `Entrada [3:0]`.
- **Pinos de saída:** `a`, `b`, `c`, `d`, `e`, `f`, `g`.
- **Teste 1:** `Entrada=0000` (0). Conexões ativas: `a, b, c, d, e, f`. Resultado: Display mostra `0`.
- **Teste 2:** `Entrada=0111` (7). Conexões ativas: `a, b, c`. Resultado: Display mostra `7`.
- **Teste 3:** `Entrada=1010` (A). Conexões ativas: `a, b, c, e, f, g`. Resultado: Display mostra `A`.

[COMPONENTE 17] Detector de Número Primo

(i) Descrição

Este circuito combinacional de 4 bits detecta se a entrada binária representa um número primo (2, 3, 5, 7, 11, 13). A lógica para a saída é `1` apenas quando a entrada é um número primo. A expressão booleana resultante foi simplificada usando um mapa de Karnaugh para otimizar o circuito.

(ii) Imagem do Componente



(iii) Metodologia de Teste

A validação foi realizada com a análise da tabela verdade para todos os 16 valores de entrada (0 a 15), confirmando que a saída era `1` somente para os números primos.

(iv) Descrição dos Testes

- **Pinos de entrada:** `Entrada [3:0]`.
- **Pinos de saída:** `Saída`.
- **Teste 1:** `Entrada=0010` (2). Resultado: `Saída = 1` (2 é primo).
- **Teste 2:** `Entrada=0011` (3). Resultado: `Saída = 1` (3 é primo).
- **Teste 3:** `Entrada=0100` (4). Resultado: `Saída = 0` (4 não é primo).
- **Teste 4:** `Entrada=1101` (13). Resultado: `Saída = 1` (13 é primo).
- **Teste 5:** `Entrada=1111` (15). Resultado: `Saída = 0` (15 não é primo).