UNIVERSIDADE FEDERAL DE RORAIMA
CURSO DE NÍVEL SUPERIOR DE CIÊNCIA DA COMPUTAÇÃO

BRUNO KHAUAN COSTA ALMEIDA
KAYLON GUTIERRE PERES GONÇALVES

LABORATÓRIO DE CIRCUITO: RELATÓRIO TÉCNICO

Boa vista - RR
2024

BRUNO KHAUAN COSTA ALMEIDA
KAYLON GUTIERRE PERES GONÇALVES

LABORATÓRIO DE CIRCUITO: RELATÓRIO TÉCNICO

Trabalho apresentado à disciplina Arquitetura e Organização de Computadores do Curso
Ciência da Computação da Universidade Federal de Roraima - Campus Paricarana, como requisito parcial para a obtenção de nota.

Orientador: Prof. Herbert Rocha

Boa Vista - RR

2024

Este trabalho apresenta o desenvolvimento de circuitos e relata tecnicamente diversos componentes digitais projetados no Logisim, com foco no entendimento de componentes da arquitetura de um computador, descrição funcional e validação por meio de testes. Foram criados 17 componentes, incluindo registradores Flip-Flop dos tipos D e JK, multiplexadores, portas lógicas compostas, somadores, memórias RAM e ROM, banco de registradores, unidade lógica e aritmética (ULA) de 8 bits, extensores de sinal, decodificadores e detectores de sequência binária e números primos. Cada componente foi documentado detalhadamente, incluindo a descrição dos pinos, feitos apenas usando portas AND, OR, NOT, Contante, Clock e Distribuidor. Ao decorrer do trabalho utilizamos outras ferramentas como o Boole, ferramenta essa que nos auxiliou na criação de alguns circuitos, análises combinacionais, criação de tabelas verdades, utilização do mapa de Karnaugh e sequências a nível de bit

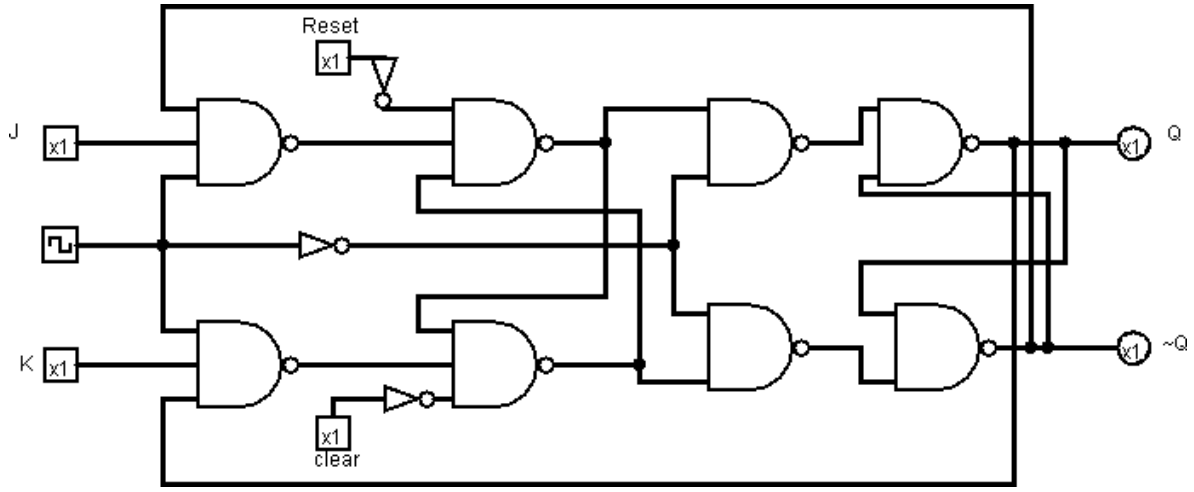
[COMPONENTE 1] – Registrador Flip-Flop do tipo JK e do tipo D.

Flip-Flops são circuitos digitais que armazenam um bit de informação. Ao decorrer do trabalho foram utilizados em contadores, registradores e outras aplicações de memória temporária. A seguir, um relatório técnico de um flip-flop do tipo JK e do tipo D.

[FLIP-FLOP do tipo JK]

Esse registrador é uma evolução do flip-flop do tipo SR, que busca solucionar o estado proibido (ou condição inválida). Possui algumas características interessantes como o toggle, que é um estado de altertância a depender da borda de sinal do clock.

[CIRCUITO COMPLETO]



Possui duas entras J e K, um clock, duas saídas Q e ~Q, Botão Reset e clear para resetar o circuito.

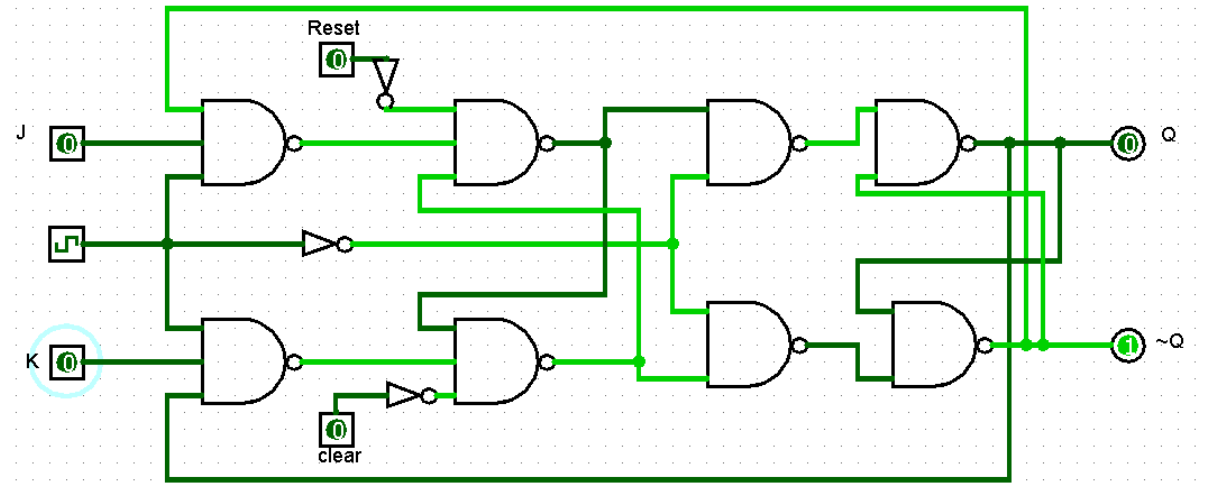
Uma característica importante é que o valor armazenado nas saídas Q e ~Q permanecem quando as entradas J (Jump) e K (Kill) são 0 e alternam quando são 1.

[TABELA VERDADE]

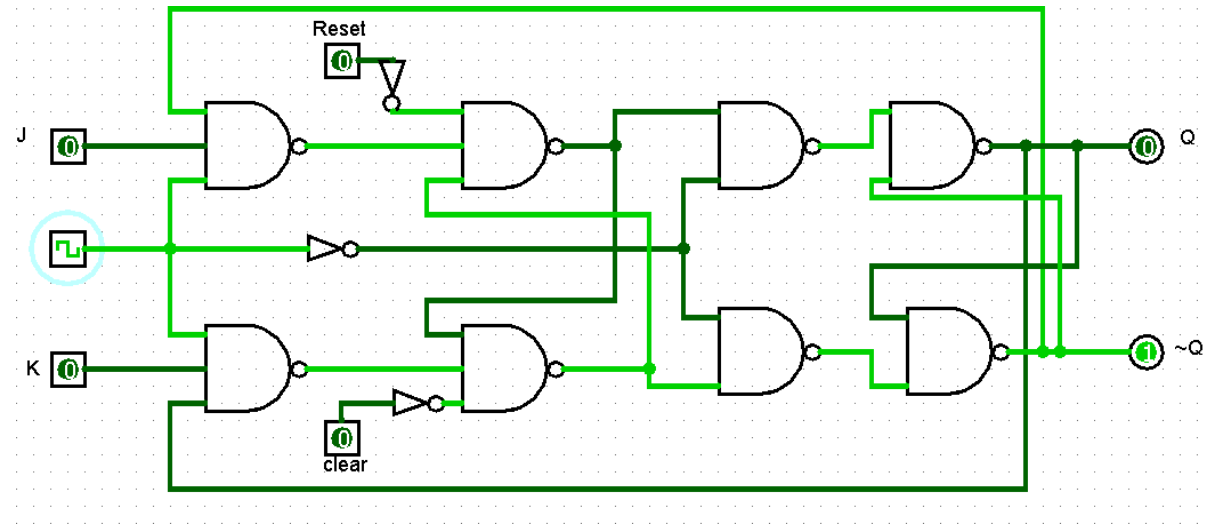
J	K	CLK	Q
0	0	↑	Q_0 (não muda)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q_0}$ (comuta)

Percebe-se que quando J e K estão com bit positivo o efeito toggle entra em vigor comutando os bits de saída para 0 e 1 a depender da borda do clock. Ademais demonstrarei alguns testes.

[TESTES]

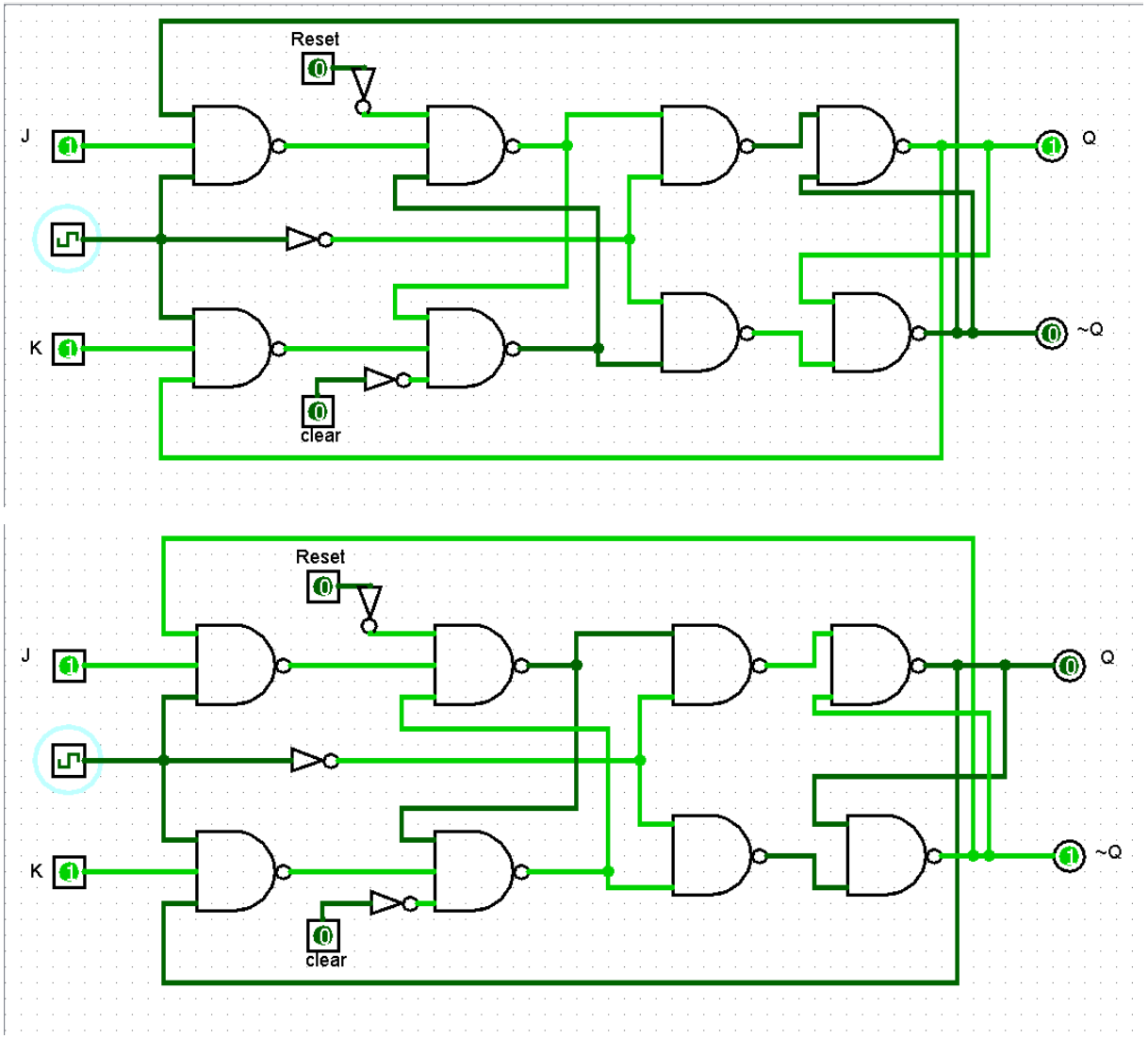


J = 0; K = 0; Q = 0 e clock de borda baixa



Clock de borda alta e o Q permanece com 0, respeitando o primeiro princípio da tabela verdade.

A título de curiosidade vou demonstrar o efeito Toggle, quando J = 1, K = 1 e quando o clock pulsa as saídas comutam.

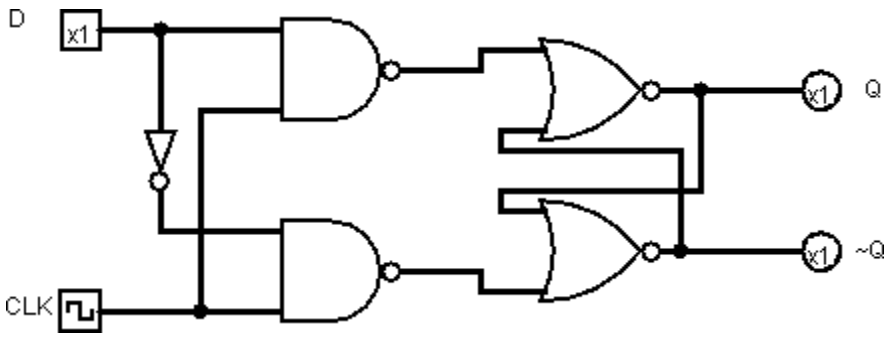


Repare nas saídas Q e ~Q que invertem.

[FLIP-FLOP do tipo D]

Possui apenas uma entrada de dados, o clock e duas saídas. Tem como função armazenar a informação passada na entrada D no momento da borda ativa do clock. É um registrador que não possui estados proibidos.

[CIRCUITO COMPLETO]



[TABELA VERDADE]

D	Qf
0	0
1	1

OU

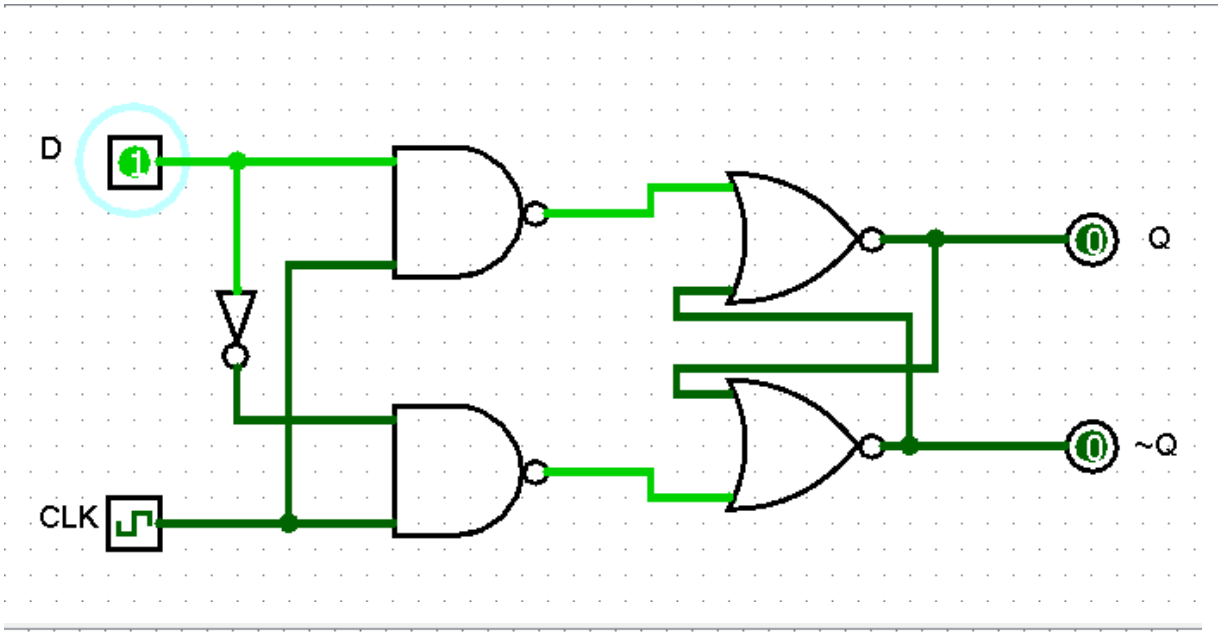
Q	D	Q(T+1)
0	0	0
0	1	1
1	0	0
1	1	1

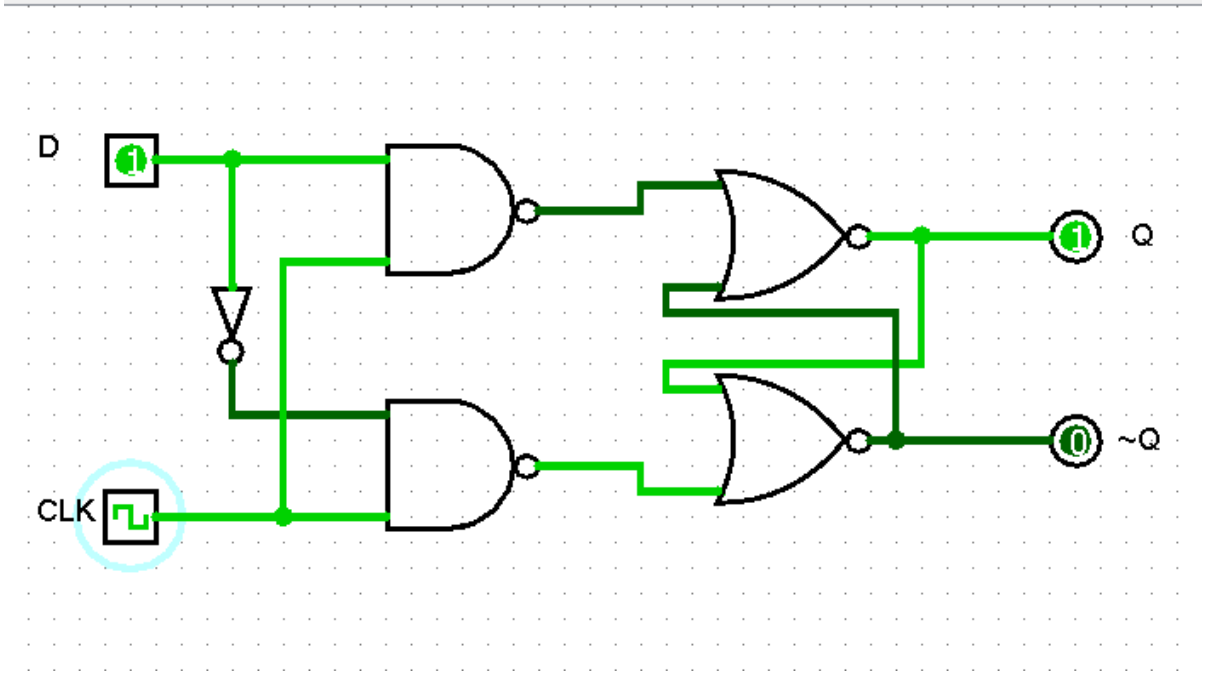
É simples e direto:

D = 0: A saída será **0** no próximo pulso de clock;

D = 1: A saída será **1** no próximo pulso de clock.

[TESTES]

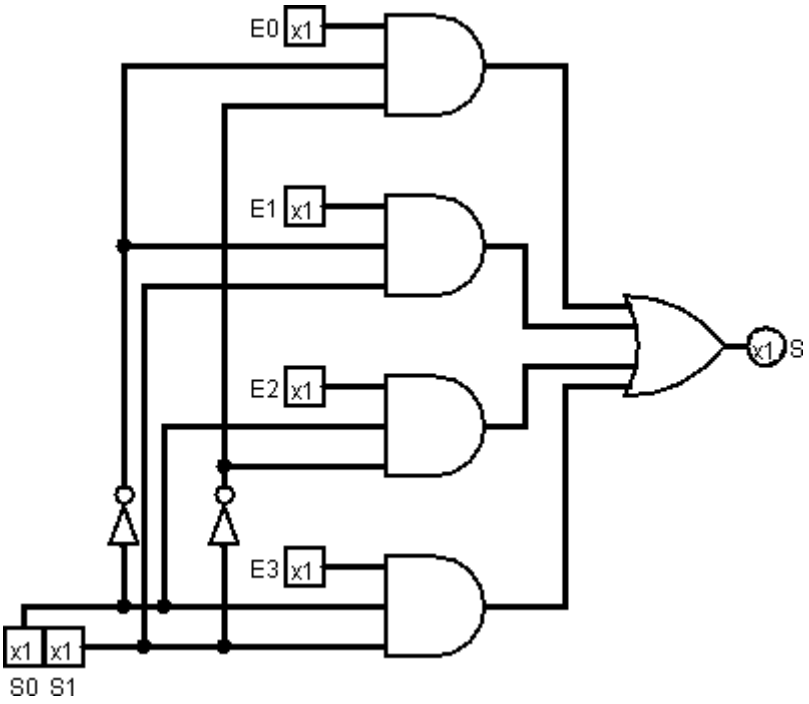




[COMPONENTE 02]. Multiplexador de quatro opções de entrada.

Um multiplexador, conhecido como MUX, é um circuito combinacional que funciona como um seletor de dados, seleciona uma entre várias entradas de dados e a direciona para a saída única, com base nos sinais de controle. Possui 4 entradas de dados (E0, E1, E2, E3), duas entradas de seleção (S1 e S2) e uma saída (S).
Basicamente a entrada de seleção determina qual entrada de dados é transmitida para a saída.

[CIRCUITO COMPLETO]



S = E0 ~S1 ~S2 + E1 ~S1 S2 + E2 S1 ~S2 + E3 S1 S2

As entradas de seleção funcionam da seguinte maneira:
S0 = 0 e S1 = 0 entrada selecionada -> E0;
S0 = 0 e S1 = 1 entrada selecionada -> E1;
S0 = 1 e S1 = 0 entrada selecionada -> E2;
S0 = 1 e S1 = 1 entrada selecionada -> E3.

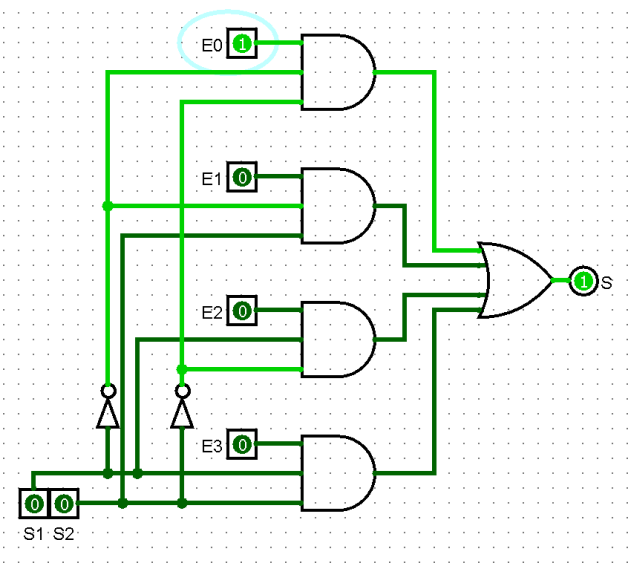
Assim eles selecionam a porta que vai emitir sinal.

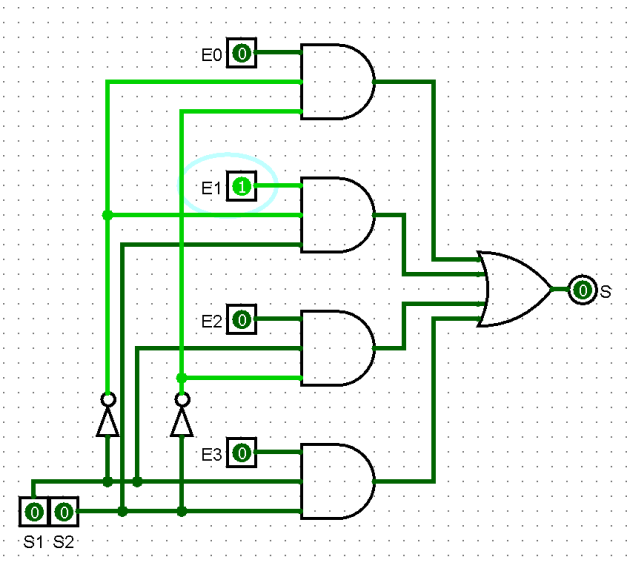
[TABELA VERDADE]

S_0	S_1	Saída
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

[TESTES]

No teste a seguir, eu quero que a entrada de dado E0 emita sinal para a saída, impossibilitando qualquer outra entrada de emitir sinal positivo. Portanto:

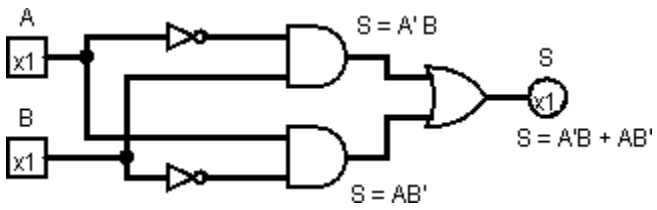




[COMPONENTE 03]. Porta lógica XOR usando os componentes: AND, NOT, e OR.

Porta XOR utilizando exclusivamente as portas lógicas **AND**, **NOT**, e **OR.** Essa operação lógica binária produz saída 1 quando seus valores de entrada são invertidos.

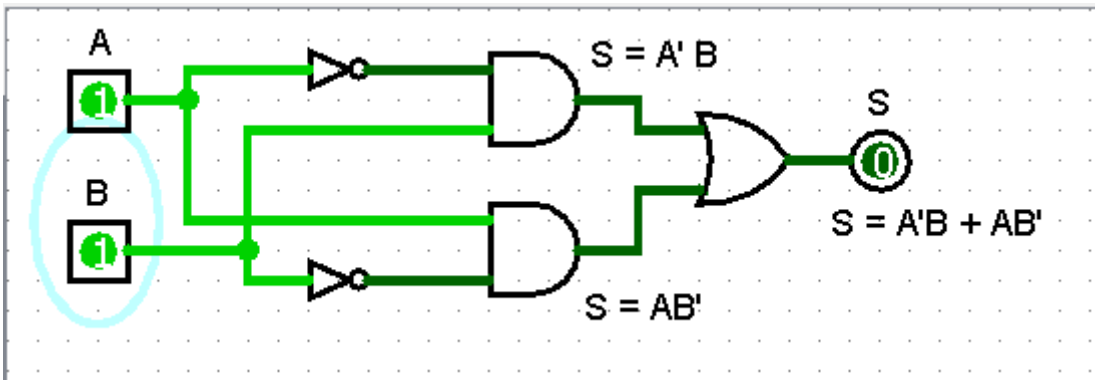
[CIRCUITO COMPLETO]

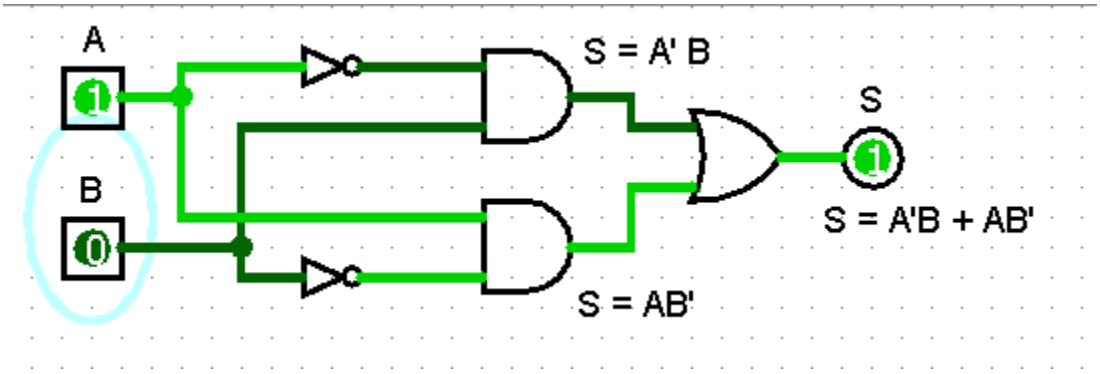


[TABELA VERDADE]

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

[TESTES]

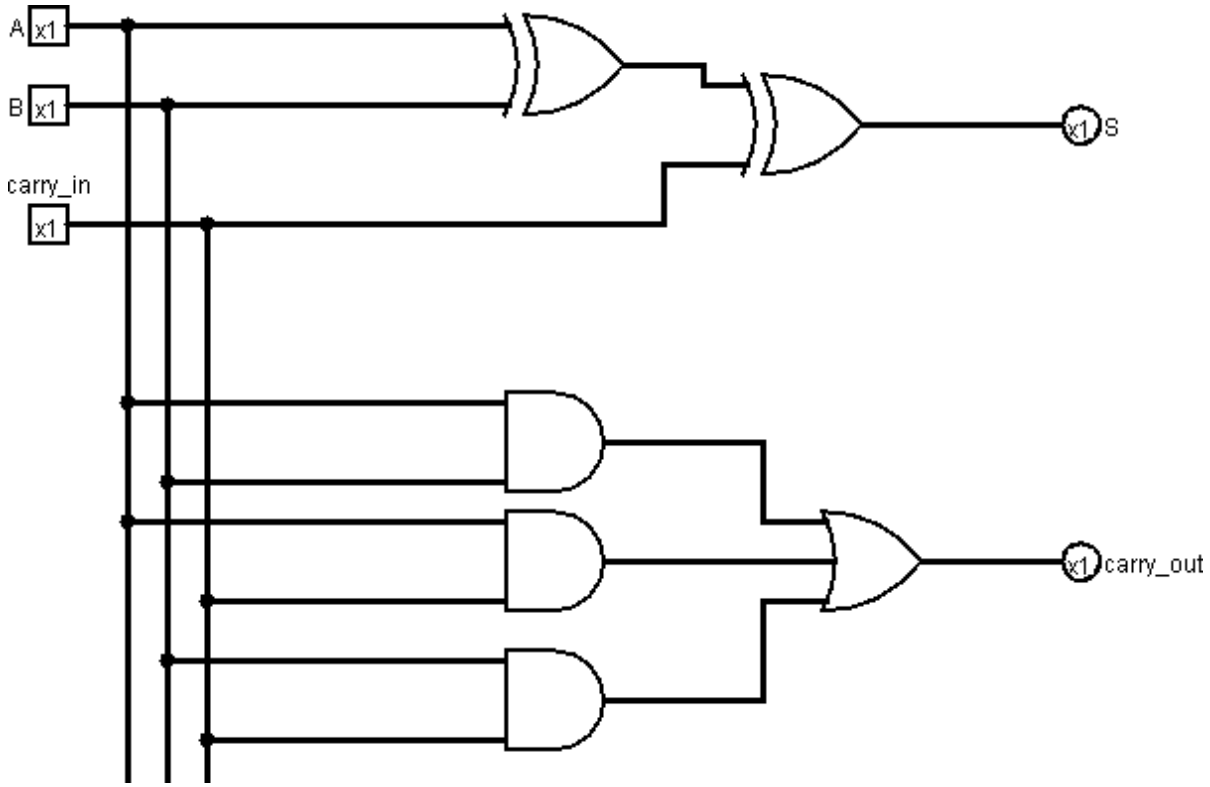




[COMPONENTE 04]. Somador de 8 bits que recebe um valor inteiro e soma com o valor 4.

Este circuito apresenta a implementação de um somador de 8 bits que recebe um valor inteiro de entrada e soma com o número fixo 4. Ademais, inclui a implementação do somador de 1 bit, que é o bloco básico utilizado para realizar a soma de dois bits com carry.

[CIRCUITO SOMADOR DE 1 BIT]



Esse circuito funciona soma dois bits de entrada A e B, considerando um carry de entrada cin (Carry-in) e gerando um carry de saída cout (Carry-out). A operação de um gera as seguintes expressões:

$S = A \oplus B \oplus C$

$Cout = (A \cdot B) + (A \cdot Cin) + (B \cdot Cin)$

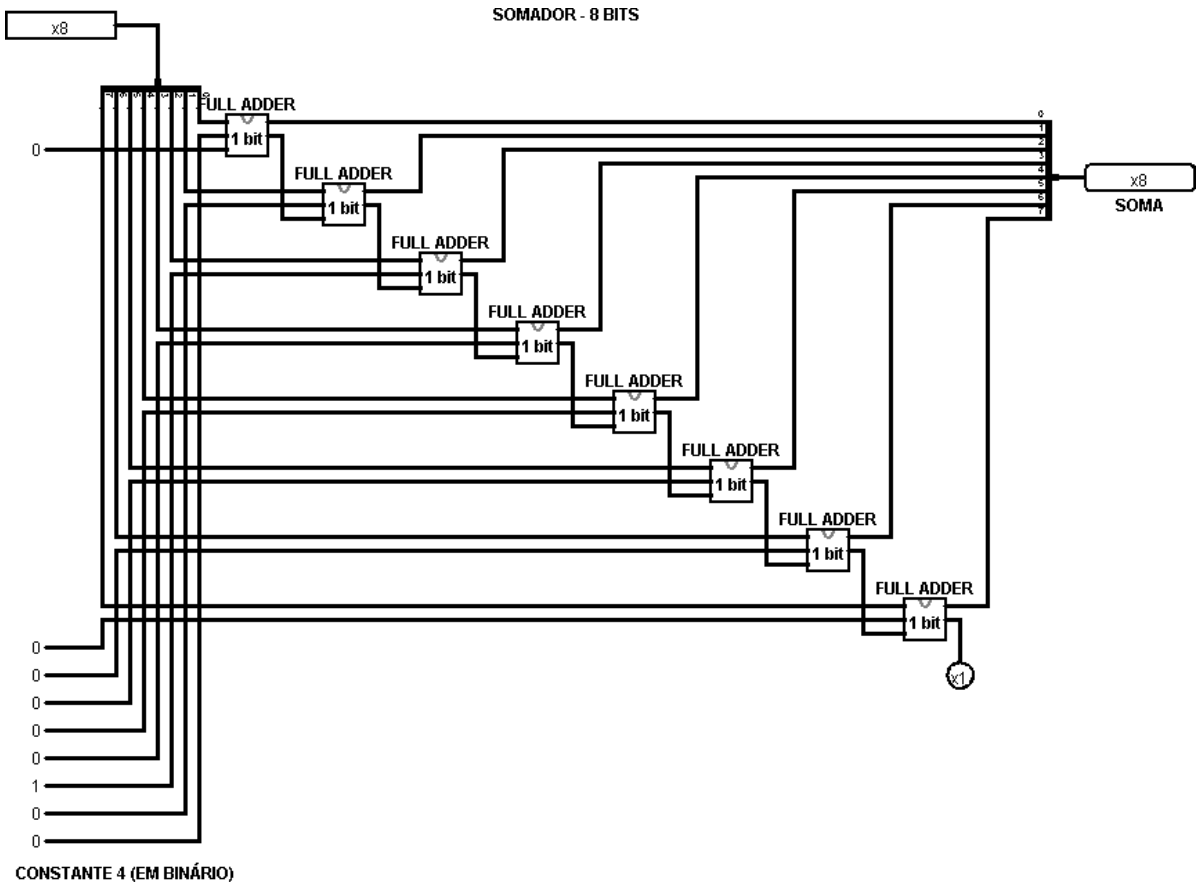
A entrada *carry-in* fornecerá um valor de um bit para ser adicionado ao montante (se for especificado), e a saída *carry-out* fornecerá um valor de um bit correspondente ao valor de *overflow* (transbordamento) que poderá ser enviado a outro componente.

[TABELA VERDADE]

A	B	cin	S	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

Para somar o valor de entrada A com 4, utilizaremos 8 Full Adders para somar cada par de bits de A e 4 (sendo 4 = 00000100).

[CIRCUITO COMPLETO]

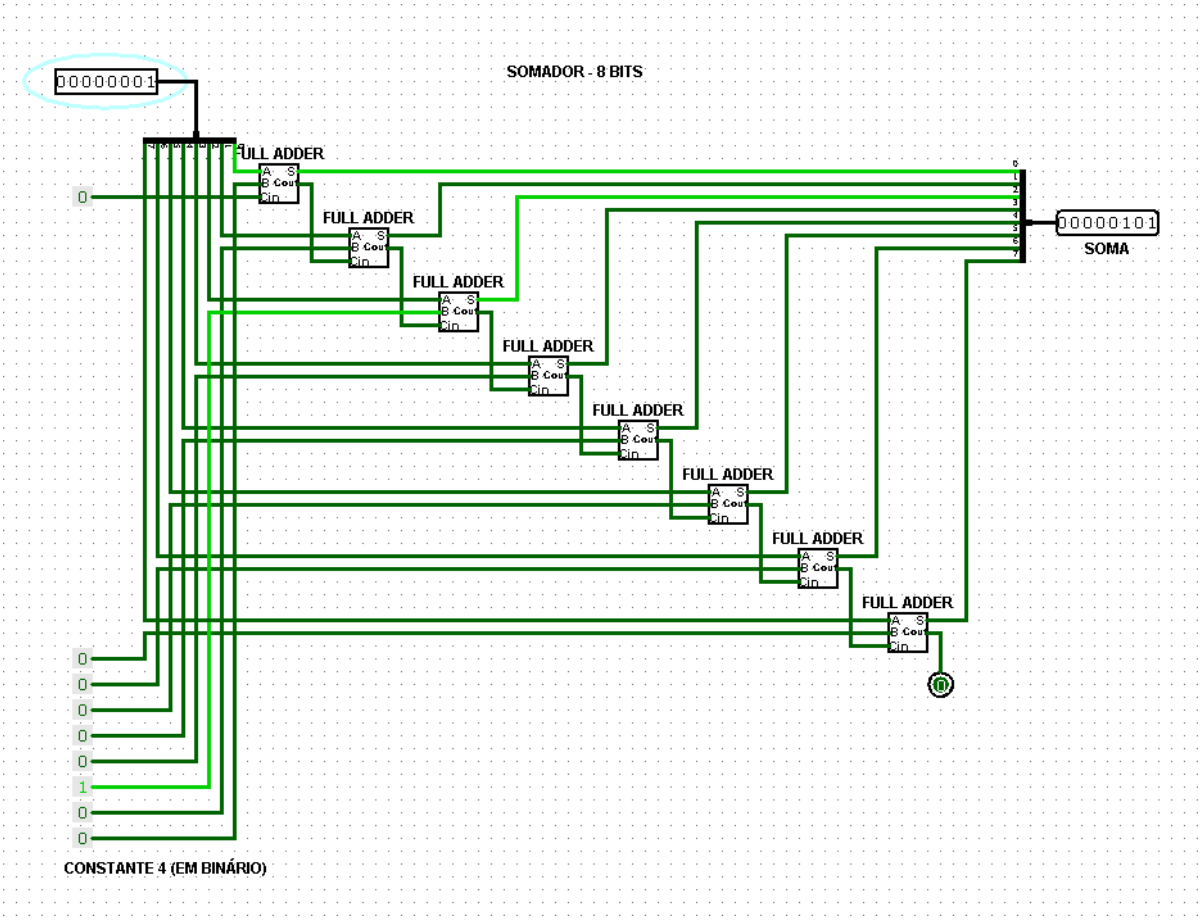


Somador de 1 bit: Cada Full Adder será responsável por somar Ai com 4i e o carry anterior.
Propagação do carry: O carry gerado por um somador de 1 bit será passado para o próximo somador de 1 bit.

Expressão Lógica do Somador de 8 Bits

A operação de soma para cada bit i será:
Si = Ai ⊕ 4i ⊕ Cin

[TESTES]

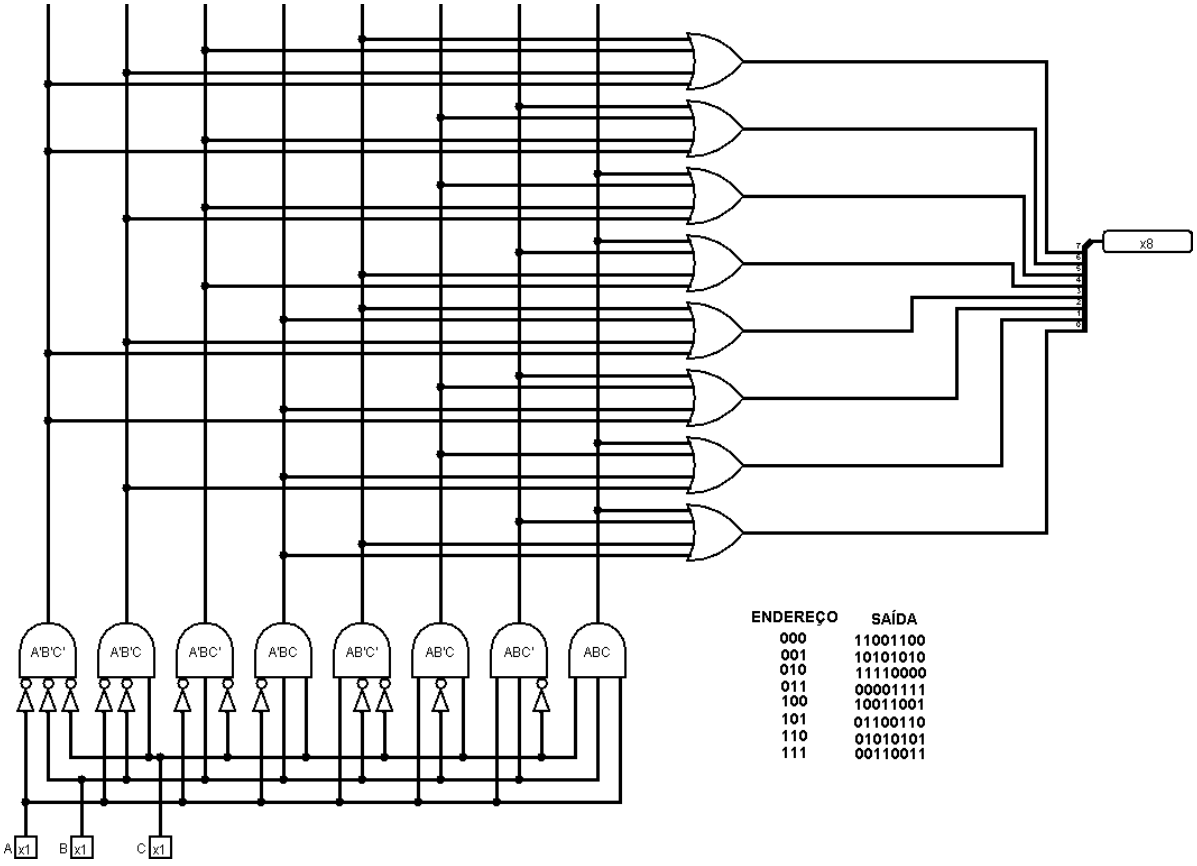


SOMANDO 1 (00000001) + 4 (00000100) = 5 (00000101)

[COMPONENTE 05]. Memória ROM de 8 bits.

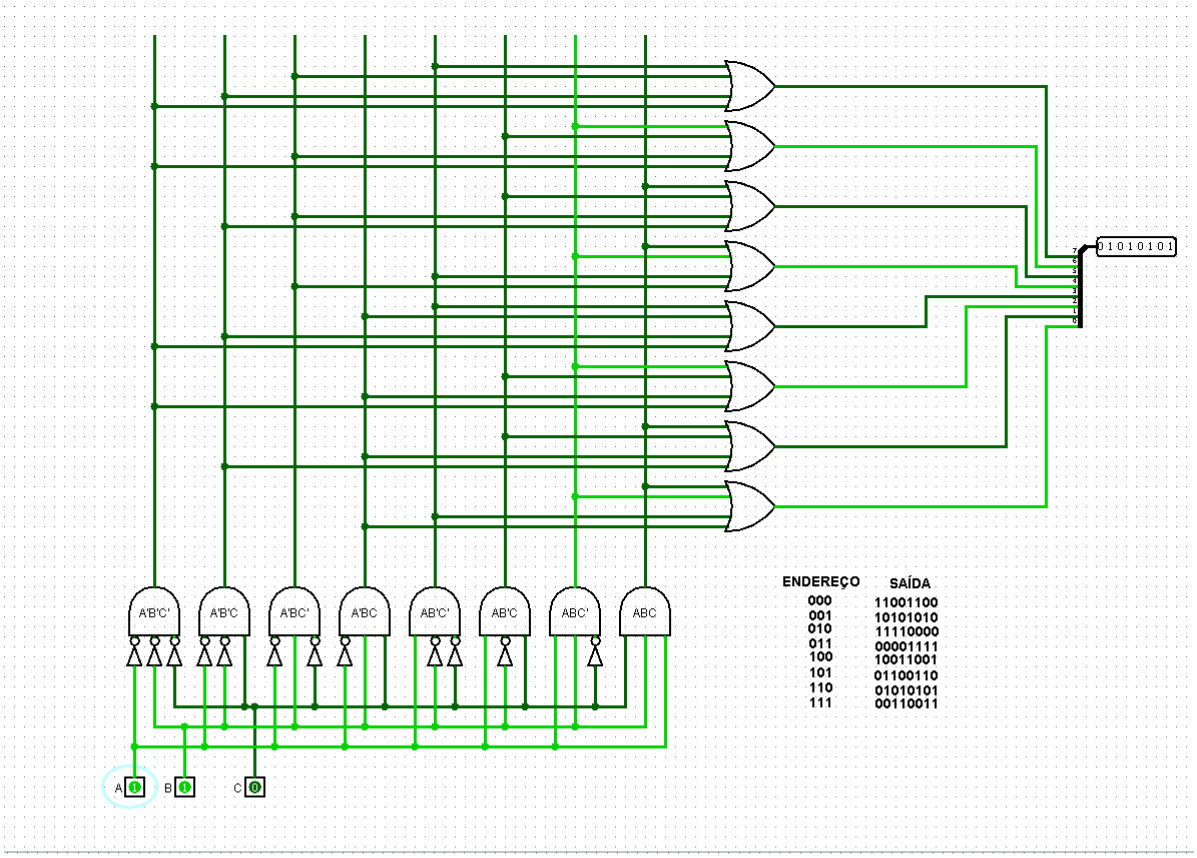
A Memória ROM (Read-Only Memory) é um tipo de memória não volátil utilizada para armazenar dados que não podem ser alterados durante a operação normal do sistema. Este relatório apresenta uma ROM de 8 bits, descrevendo sua estrutura, funcionamento e aplicações.
Uma ROM de 8 bits armazena palavras binárias de 8 bits em endereços distintos. O número de endereços depende do tamanho da ROM, que é especificado em termos de 2^n, onde n é o número de linhas de endereço.

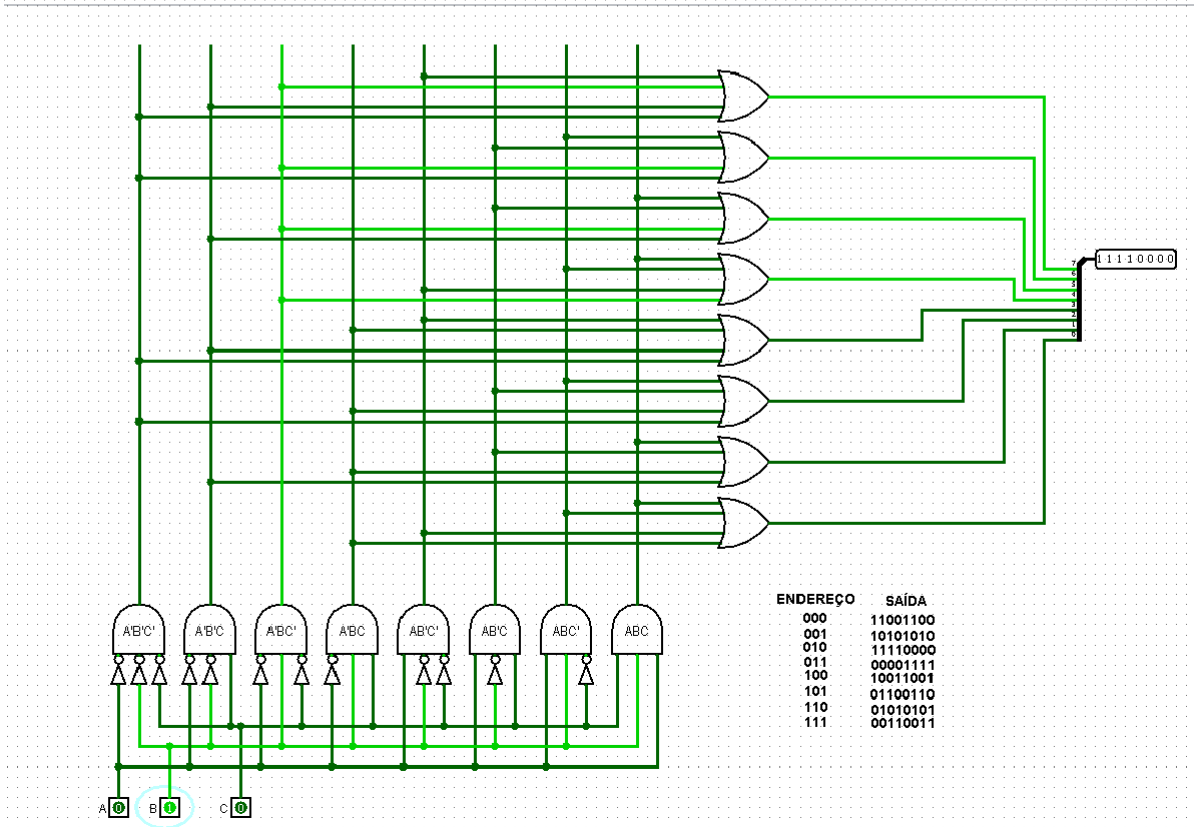
[CIRCUITO COMPLETO]



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

[TESTES]


A'B'C'
A'B'C
A'BC'
A'BC
AB'C'
AB'C
ABC'
ABC
A
B
C
01010101
ENDEREÇO SAÍDA
000 11001100
001 10101010
010 11110000
011 00001111
100 10011001
101 01100110
110 01010101
111 00110011

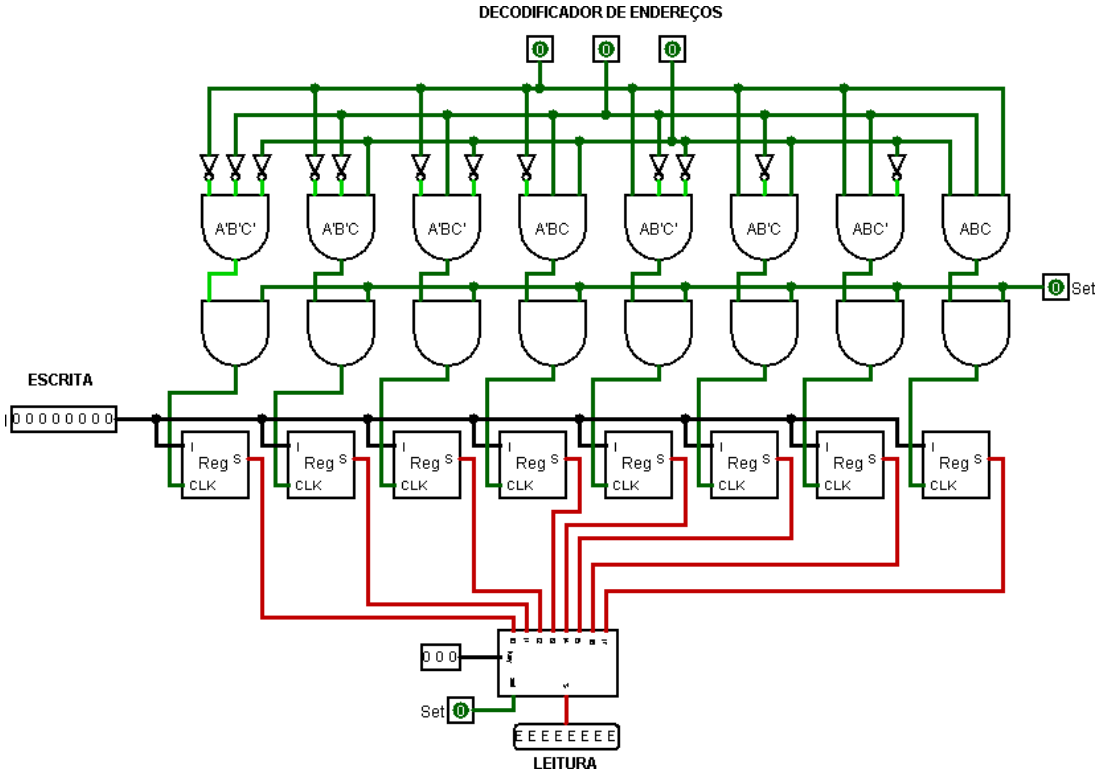

A'B'C'
A'B'C
A'BC'
A'BC
AB'C'
AB'C
ABC'
ABC
A
B
C
11110000
ENDEREÇO SAÍDA
000 11001100
001 10101010
010 11110000
011 00001111
100 10011001
101 01100110
110 01010101
111 00110011

1- Fornecimento de Endereço:
Um endereço binário é enviado às linhas de entrada da ROM.
2- Decodificação do Endereço:
O decodificador ativa a linha correspondente ao endereço fornecido.
3- Leitura dos Dados:
A linha ativa da grade de memória transmite os bits armazenados para as linhas de saída.
4- Saída:
Os 8 bits armazenados no endereço selecionado são disponibilizados na saída da ROM.

[COMPONENTE 6] - Memória RAM de 8 bits

Memória volátil usada para armazenar dados temporariamente em sistemas digitais. Ela permite a leitura e escrita de informações com acesso aleatório, ou seja, qualquer posição de memória pode ser acessada diretamente sem precisar percorrer outras posições.

[CIRCUITO COMPLETO]



Entradas:

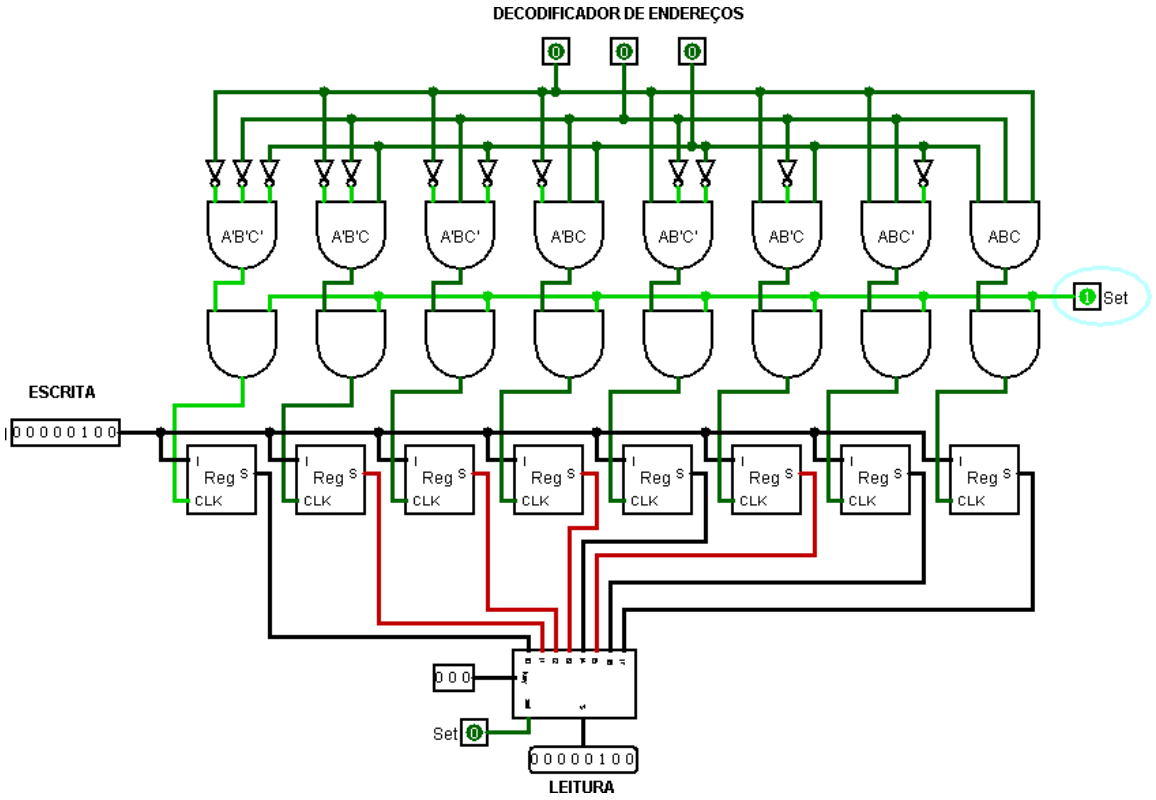
Endereços: Determinam a posição da memória a ser acessada.

Dados: Dados de entrada para escrita.

Sinais de controle: Incluem Write Enable (WE) para ativar a escrita e Chip Select (CS) para ativar a memória.

Saídas:
Os dados lidos de uma posição de memória.

[TESTES]



[COMPONENTE 07]. Banco de Registradores de 8 bits.

Um banco de registradores é um conjunto de registradores organizados de forma que suas operações (leitura e escrita) possam ser realizadas com rapidez e eficiência. Este relatório descreve a implementação e funcionalidade de um banco de registradores de 8 bits, um componente essencial em sistemas digitais como unidades de processamento.

Registradores: Um banco de registradores de 8 bits consiste em múltiplos registradores, cada um com capacidade de armazenar 8 bits de dados.

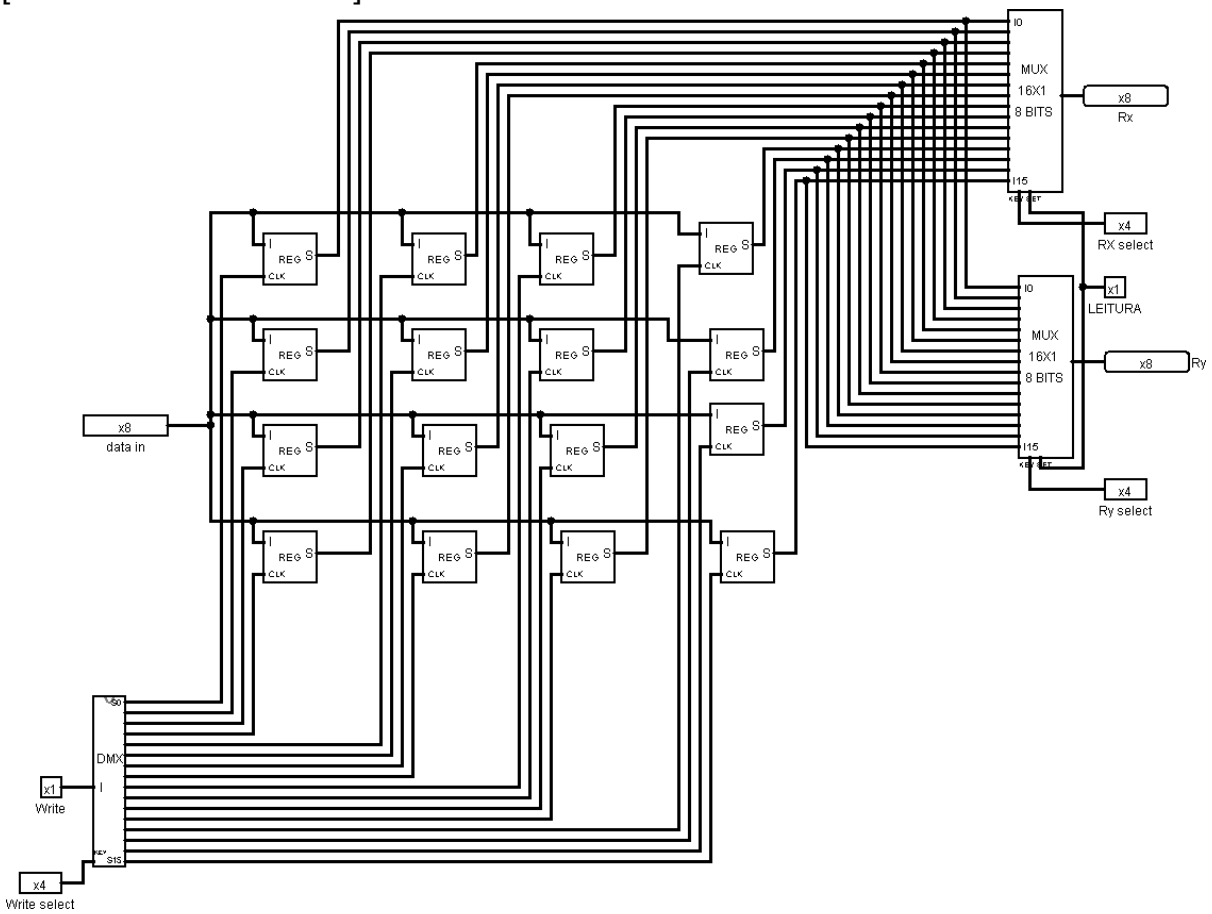
Linhas de Controle:

Endereço de Seleção: Escolhe qual registrador será acessado.
Sinal de Escrita (Write Enable): Habilita a escrita de novos dados no registrador selecionado.

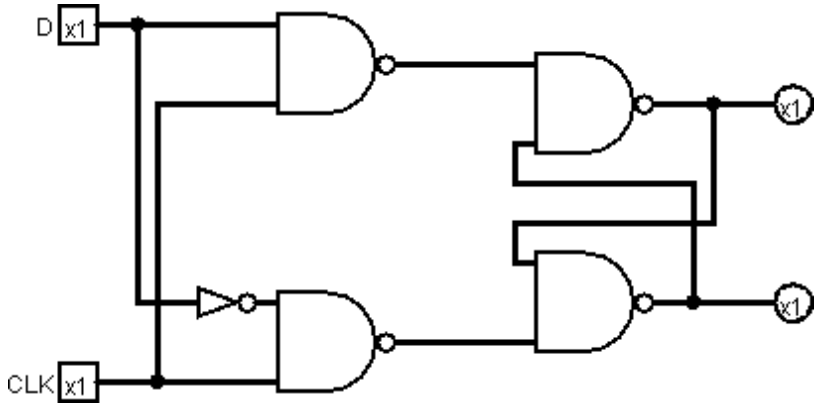
Sinal de Leitura (Read Enable): Habilita a leitura do registrador selecionado.
Linhas de Dados:
Entrada de Dados: Linhas para escrita nos registradores.
Saída de Dados: Linhas para leitura do conteúdo armazenado.
Configuração Típica:
Número de Registradores: 2^n, onde n é o número de bits das linhas de endereço.
Tamanho da Palavra: 8 bits por registrador.

Funcionamento do Banco de Registradores
Escrita:
Fornece o endereço do registrador desejado por meio das linhas de seleção.
Ativa o sinal de escrita (Write Enable).
Insere os 8 bits de dados nas linhas de entrada.
O registrador no endereço selecionado armazena os dados.
Leitura:
Fornece o endereço do registrador desejado.
Ativa o sinal de leitura (Read Enable).
O conteúdo do registrador selecionado é disponibilizado nas linhas de saída.

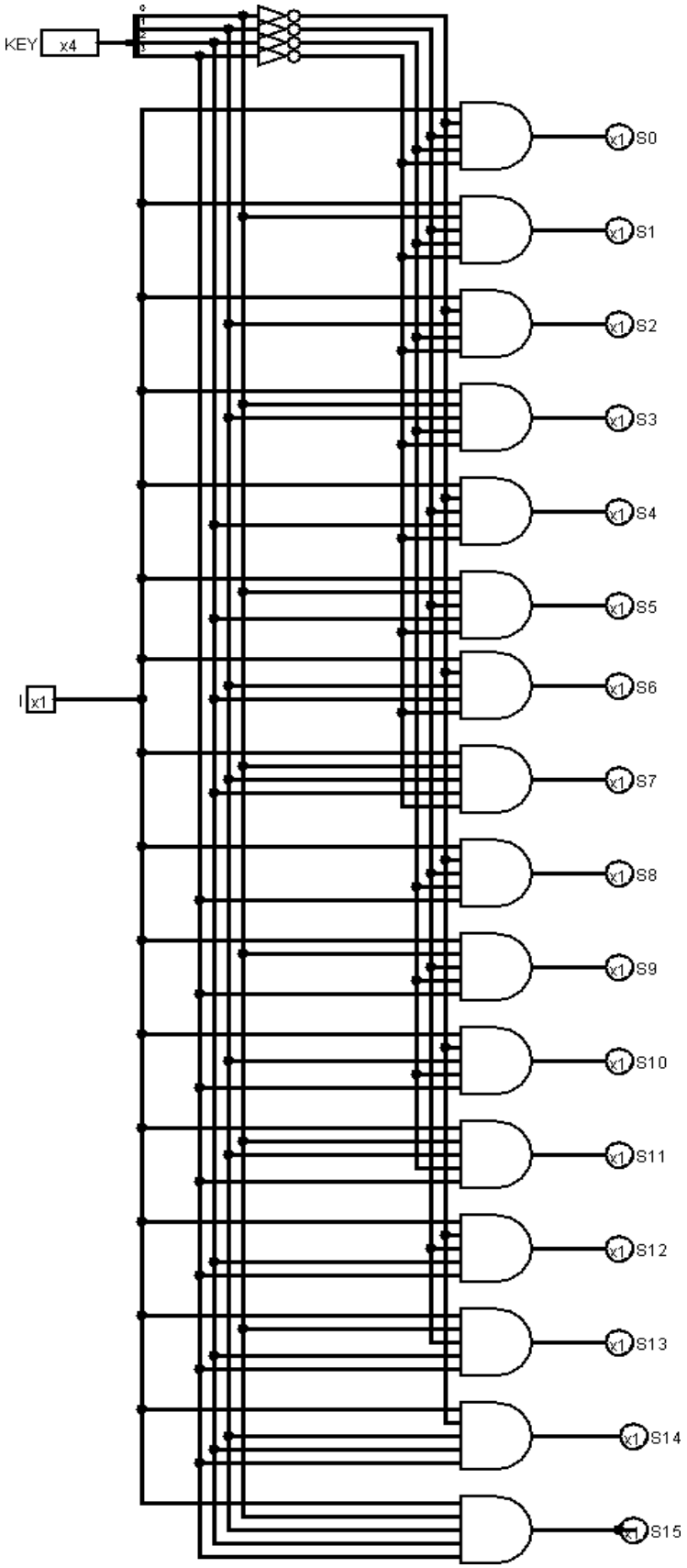
[CIRCUITO COMPLETO]



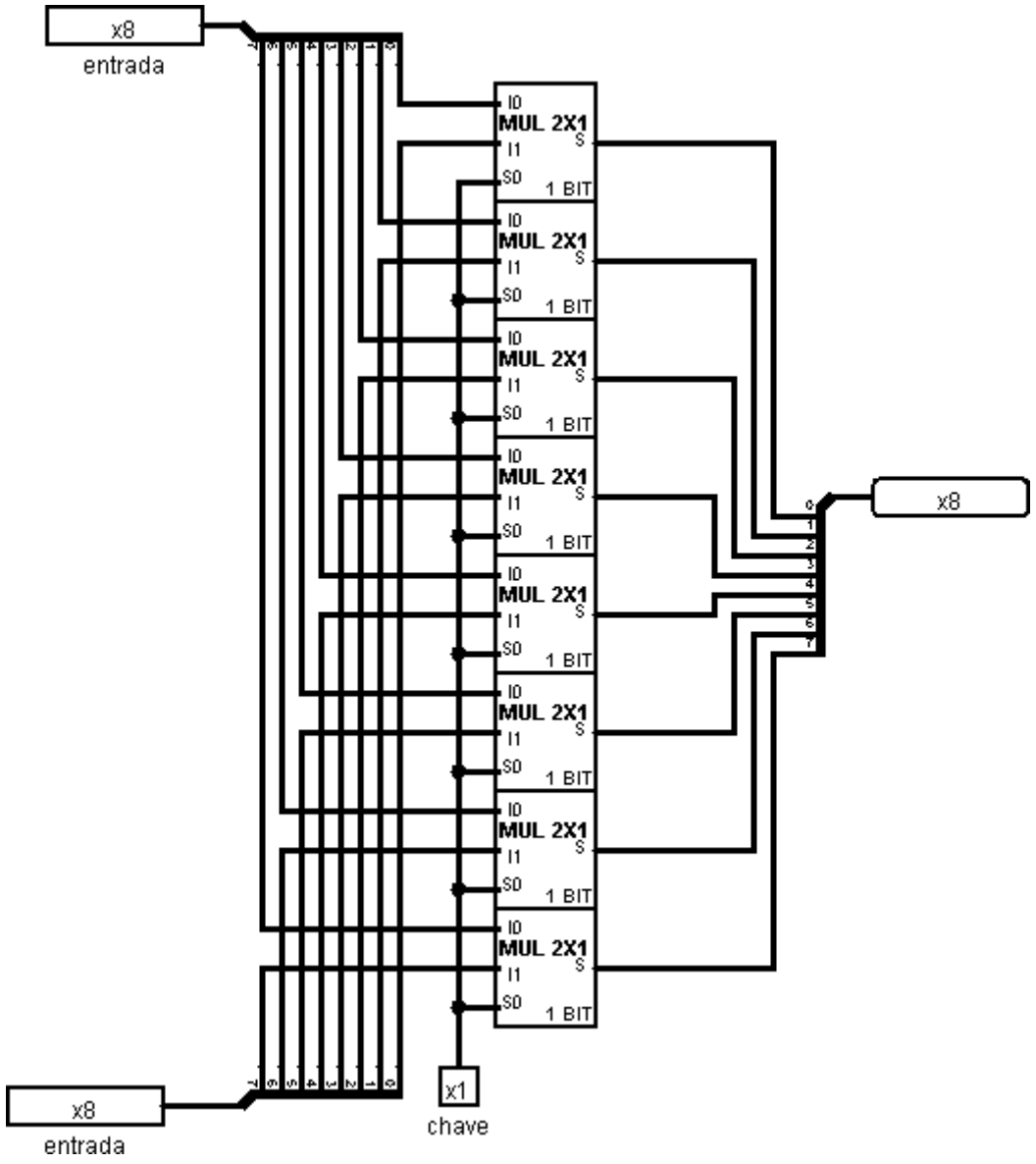
[FLIP-FLOP TIPO D]



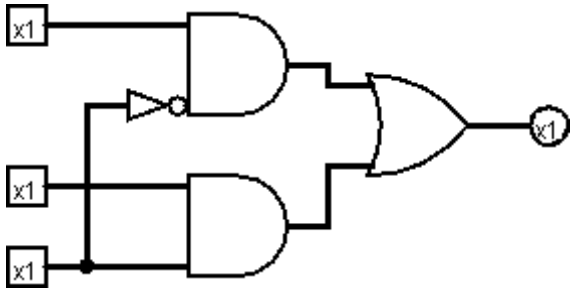
[DEMUX 1X16]


KEY
x4
I
x1
x1 S0
x1 S1
x1 S2
x1 S3
x1 S4
x1 S5
x1 S6
x1 S7
x1 S8
x1 S9
x1 S10
x1 S11
x1 S12
x1 S13
x1 S14
x1 S15

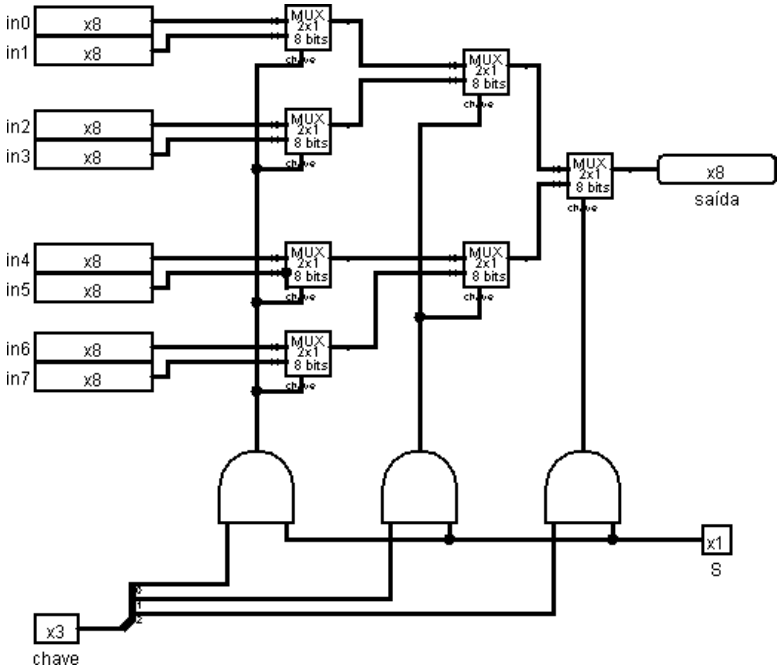
[MUX 2X1 8 BITS]



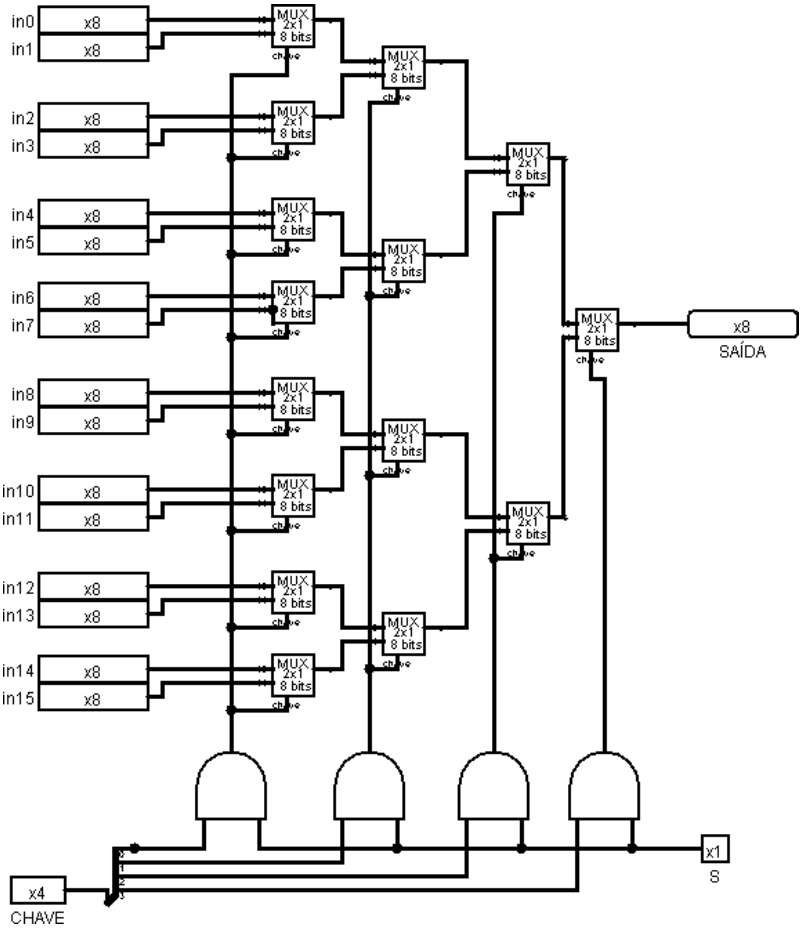
[MUX 2X1 1 BIT]



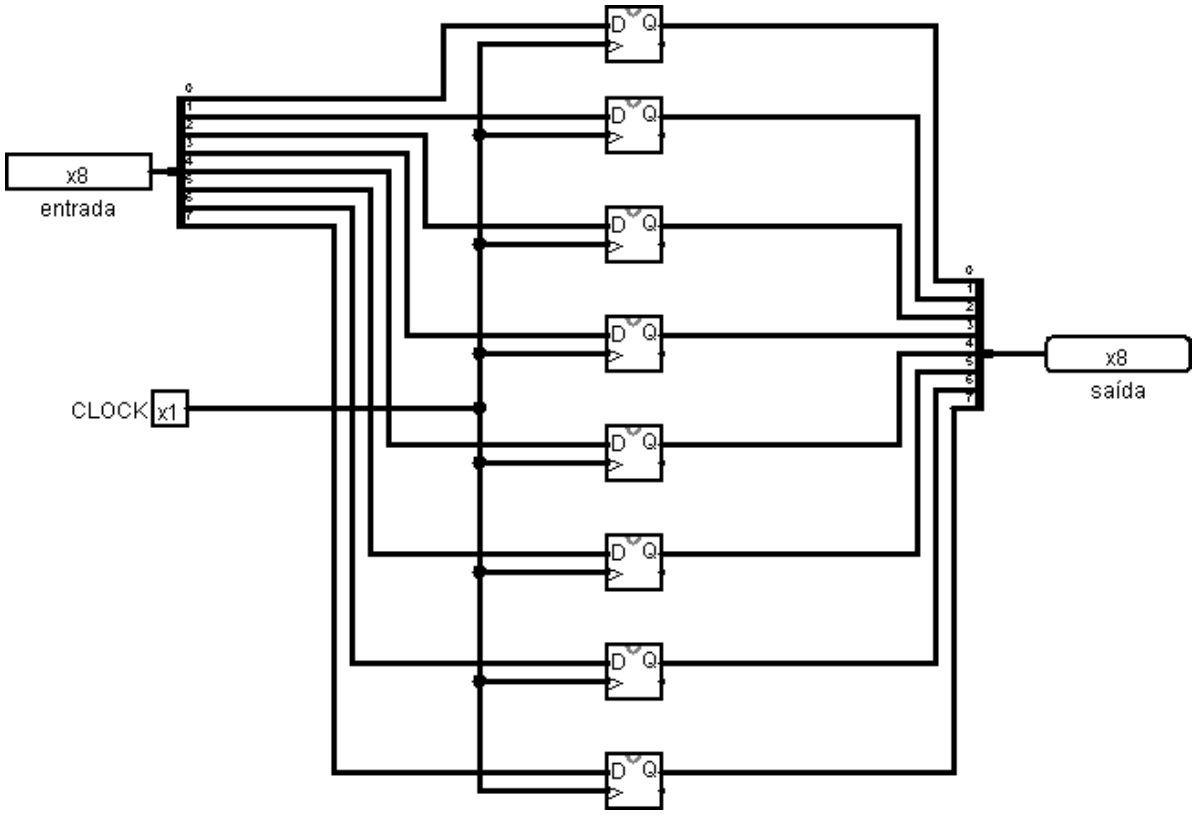
[MUX 8X1 8 BITS]



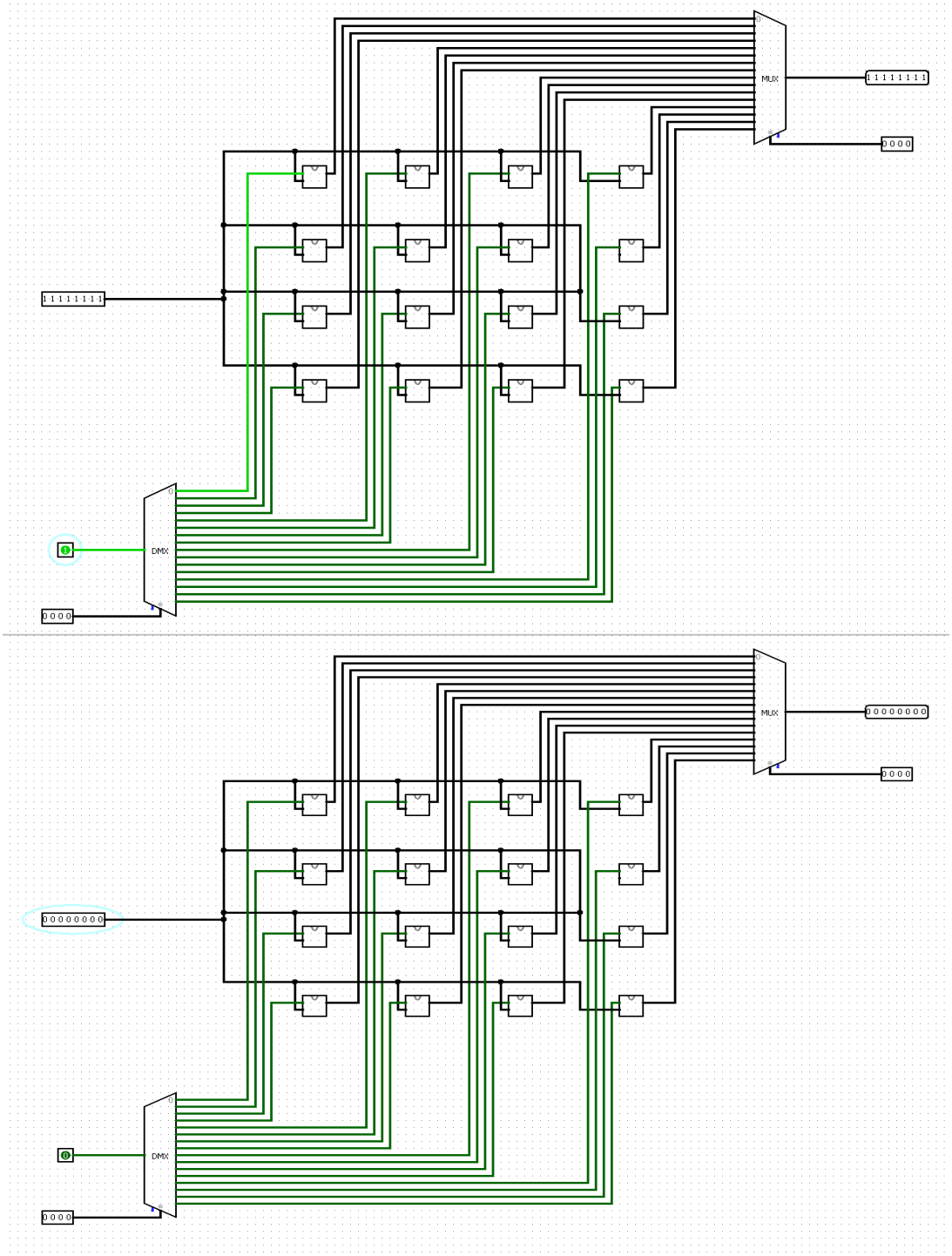
[MUX 16X1 8 BITS]



[REGISTRADOR]

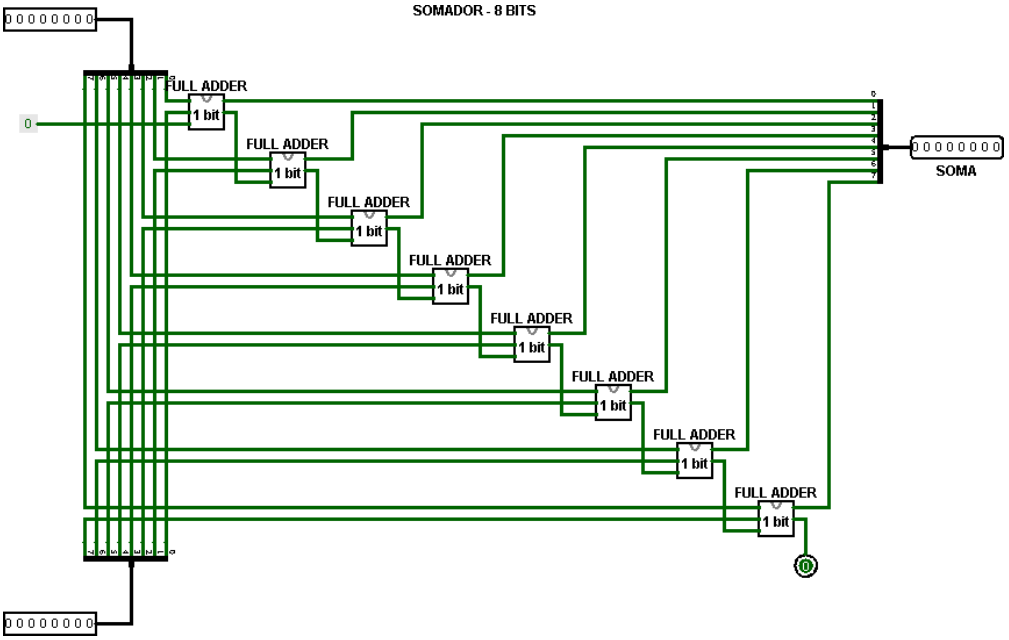


[TESTES]


MUX
11111111
0000
11111111
DMX
0000
MUX
00000000
0000
00000000
DMX
0000

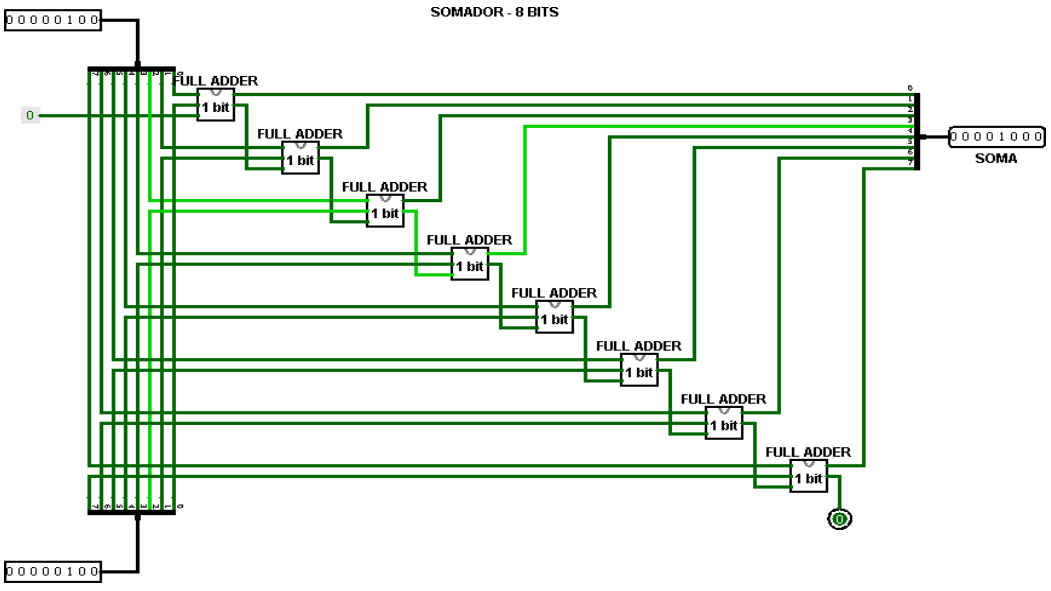
[COMPONENTE 8] - Somador de 8 bits
Circuito digital que realiza a operação de adição de dois números binários de 8 bits cada. Ele é composto por oito somadores completos (full adders) conectados em série, de forma que o carry (vai-um) de cada bit seja propagado para o próximo bit mais significativo.

[CIRCUITO COMPLETO]



Possui operandos de 8 bits, oito full adders encadeados e uma saída soma de 8 bits e um carry-out (indicando overflow ou "vai-um" para um possível próximo somador).

[TESTES]
00000100 + 00000100 (4 + 4)



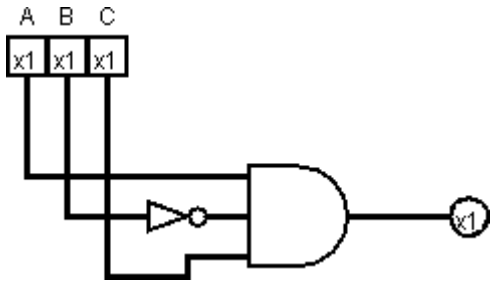
[COMPONENTE 09]. Construa um detector de sequência binária para identificar a sequência "101" em um fluxo de entrada.

O detector verifica bit a bit a entrada e sinaliza quando encontra a sequência "101". Ele utiliza flip-flops para armazenar o estado atual do reconhecimento e portas lógicas para realizar a transição de estados e a identificação

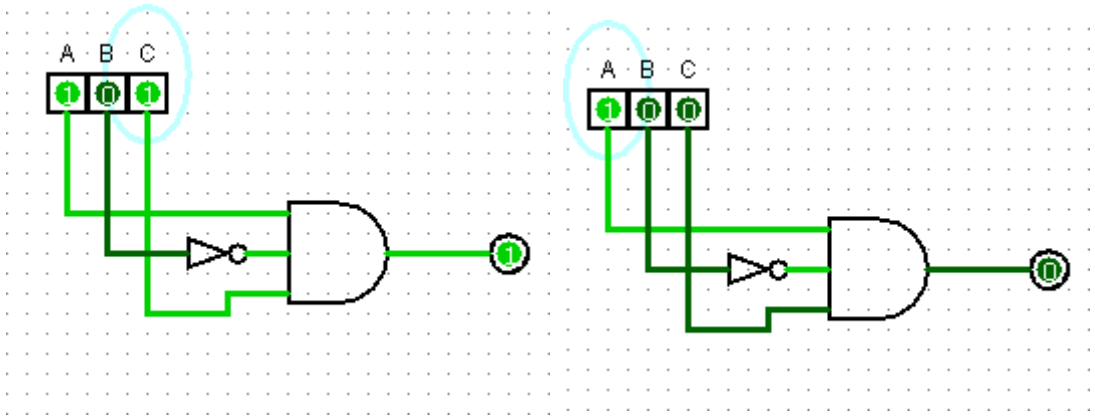
[TABELA VERDADE]

A	B	C	x
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

[CIRCUITO COMPLETO]



[TESTES]

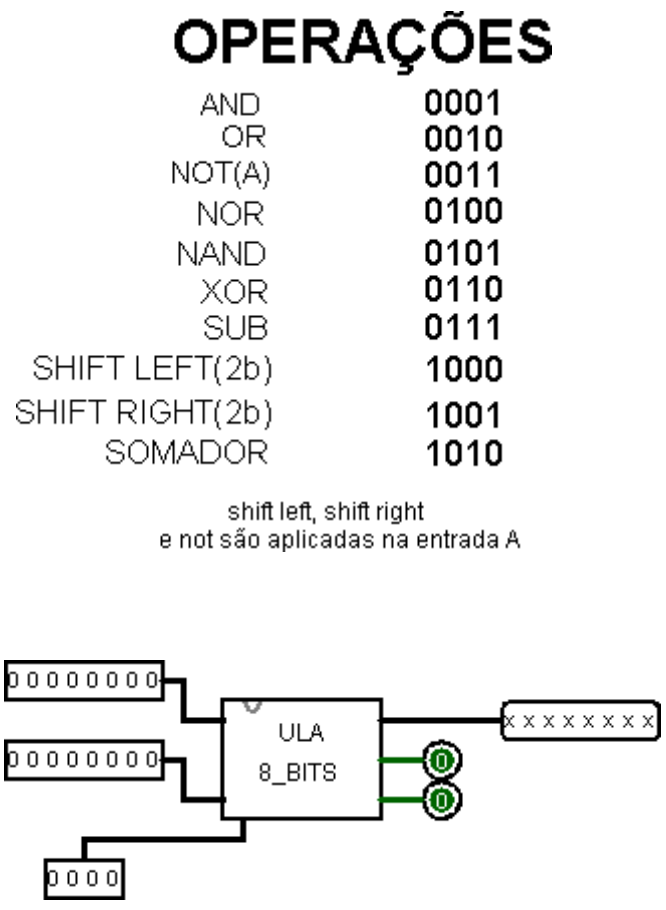


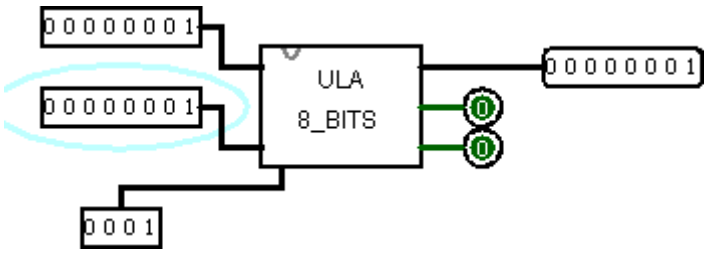
[COMPONENTE 10]. ULA de 8 bits com as seguintes operações: AND, OR, NOT, NOR, NAND, XOR, SHIFT de 2 bits à esquerda, SHIFT de bits à direita, soma e subtração.

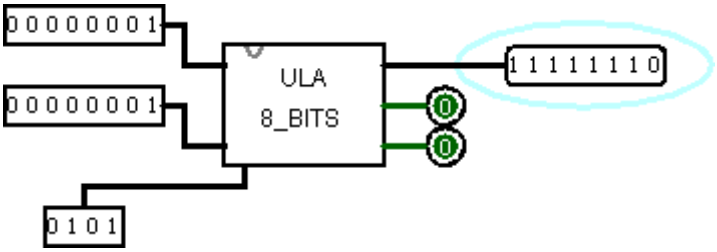
A Unidade Lógica e Aritmética (ULA) é um componente essencial em sistemas digitais, responsável por executar operações lógicas e aritméticas em dados binários. Este relatório descreve a implementação de uma ULA de 8 bits, capaz de realizar as operações: AND, OR, NOT, NOR, NAND, XOR, SHIFT à esquerda, SHIFT à direita, soma e subtração.

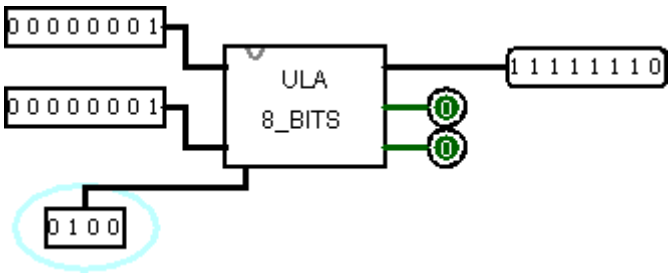
A ULA recebe como entradas dois operandos de 8 bits (A e B), um sinal de controle de 4 bits que define a operação desejada e um sinal de transporte de entrada (Carry In) usado em operações aritméticas. Como saída, a ULA retorna um resultado de 8 bits (R) e um transporte de saída (Carry Out) nas operações de soma e subtração.

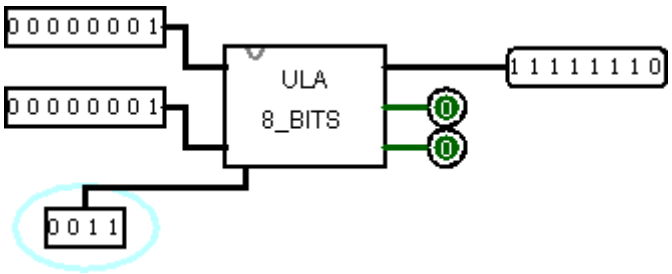
[TESTES DOS CIRCUITOS]

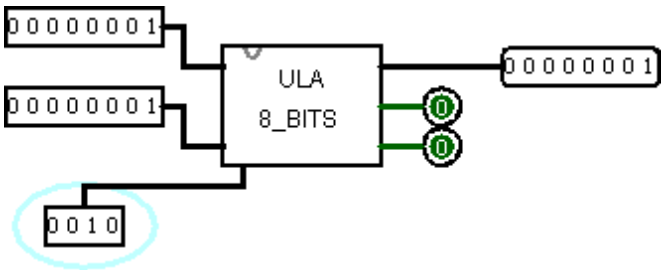


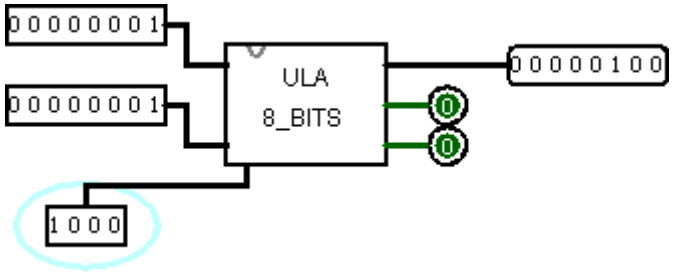

0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
0 0 0 1
ULA
8_BITS
0
0
0 0 0 0 0 0 0 1

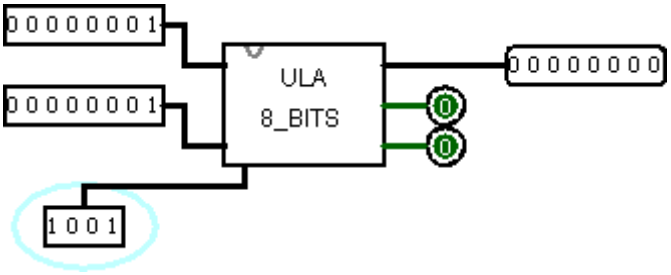

0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
0 1 0 1
ULA
8_BITS
0
0
1 1 1 1 1 1 1 0

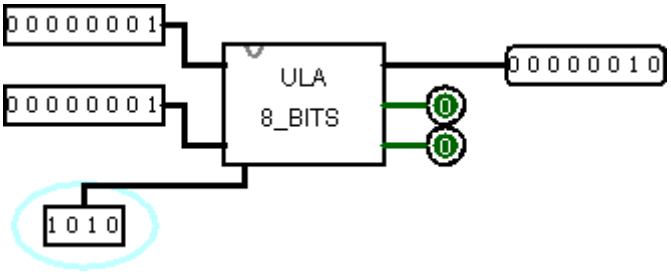

0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
0 1 0 0
ULA
8_BITS
0
0
1 1 1 1 1 1 1 0

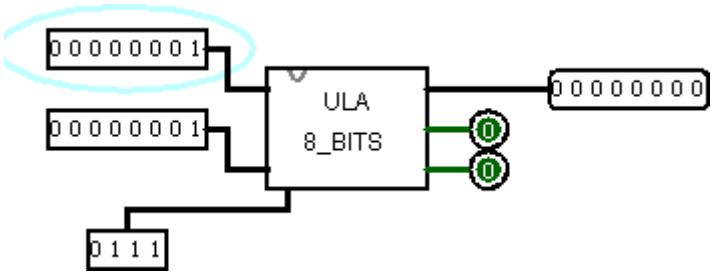

0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
0 0 1 1
ULA
8_BITS
0
0
1 1 1 1 1 1 1 0

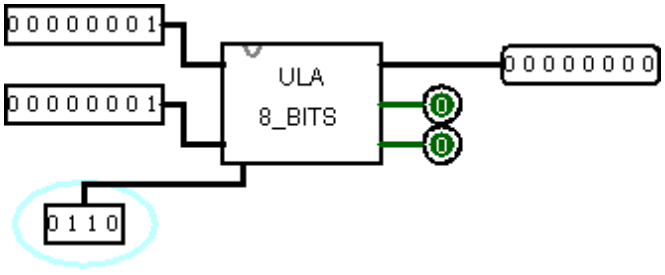

0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
0 0 1 0
ULA
8_BITS
0
0
0 0 0 0 0 0 0 1


0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
ULA
8_BITS
0 0 0 0 0 1 0 0
0
0
1 0 0 0


0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
ULA
8_BITS
0 0 0 0 0 0 0 0
0
0
1 0 0 1


0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
ULA
8_BITS
0 0 0 0 0 0 1 0
0
0
1 0 1 0

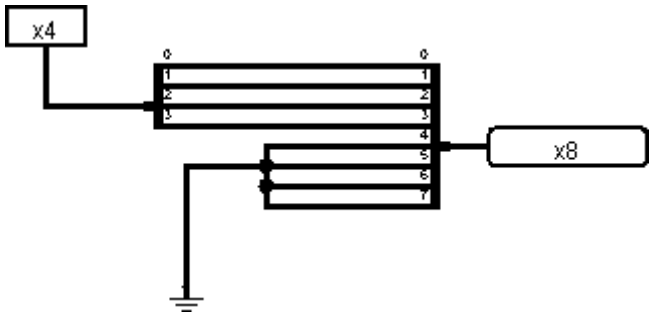

0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
ULA
8_BITS
0 0 0 0 0 0 0 0
0
0
0 1 1 1


0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
ULA
8_BITS
0 0 0 0 0 0 0 0
0
0
0 1 1 0

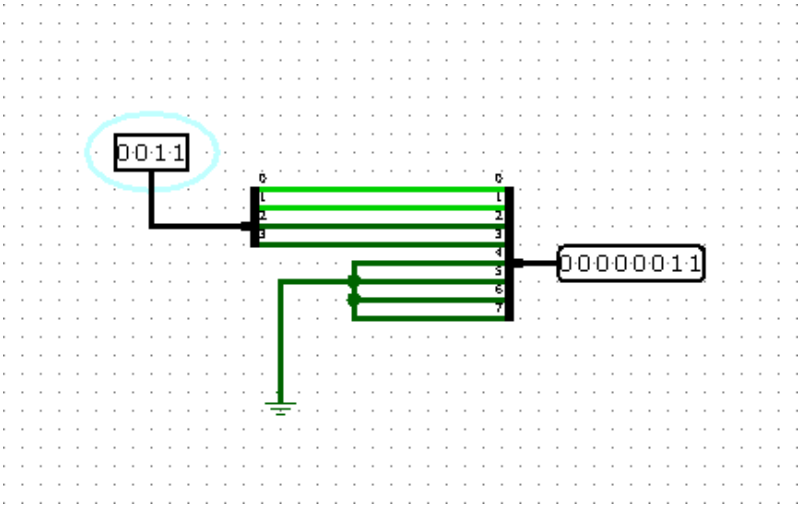
[COMPONENTE 11]. Extensor de sinal de 4 bits para 8 bits.

Um extensor de sinal é um componente digital utilizado para ajustar o tamanho de uma palavra binária ao contexto de um sistema maior, preservando seu significado numérico. Este relatório descreve a construção de um extensor de sinal que converte números de 4 bits para 8 bits.

[CIRCUITO COMPLETO]



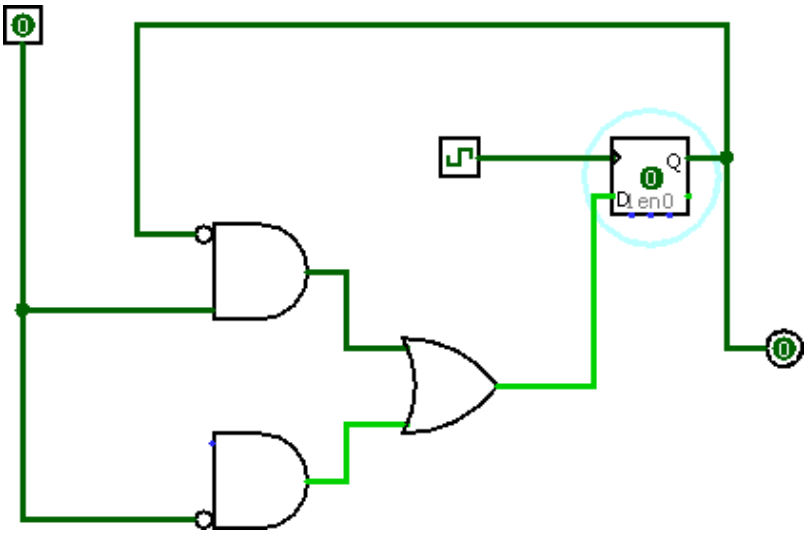
[TESTES]



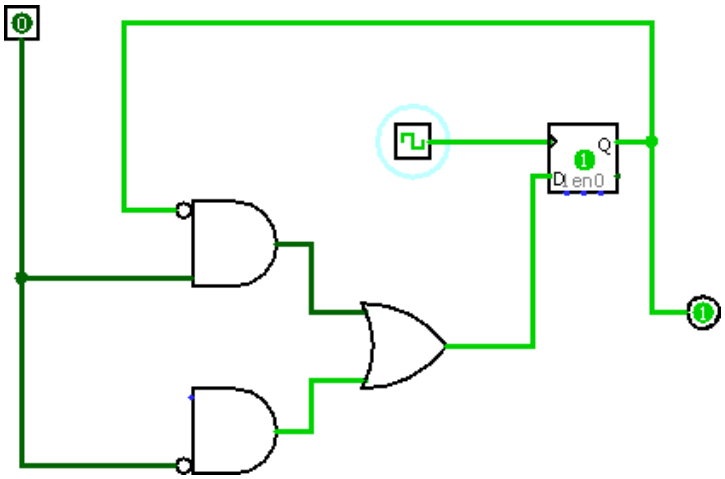
[COMPONENTE 12]. Implemente uma máquina de estados utilizando portas lógicas. É um modelo computacional usado para representar sistemas que mudam de comportamento com base em entradas e condições internas. Ela é amplamente utilizada em engenharia de software, automação e design de circuitos digitais.

[TESTES]

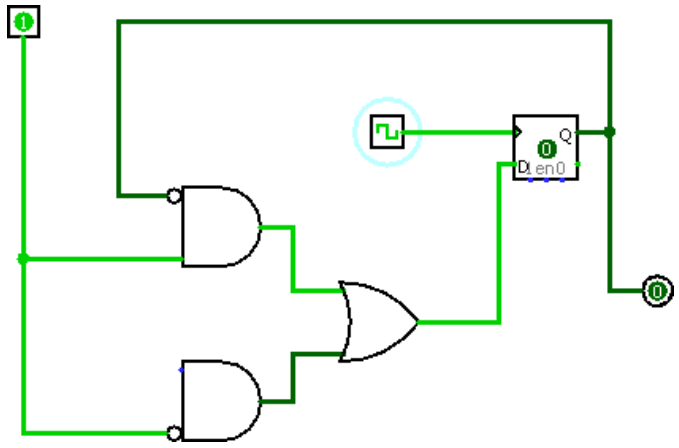
ENTRADA = 0, Flip Flop D = 0



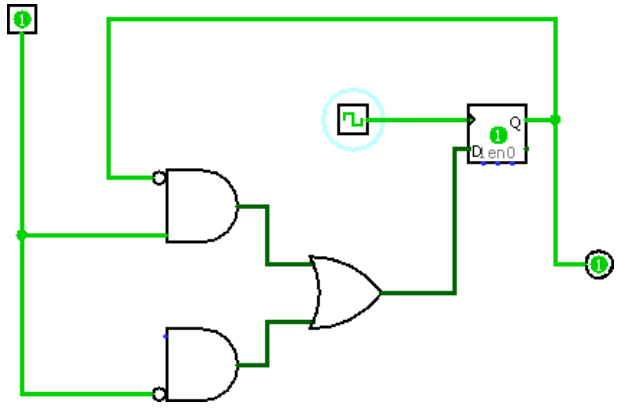
ENTRADA = 0, Flip Flop D = 1



ENTRADA = 1, Flip Flop D = 0



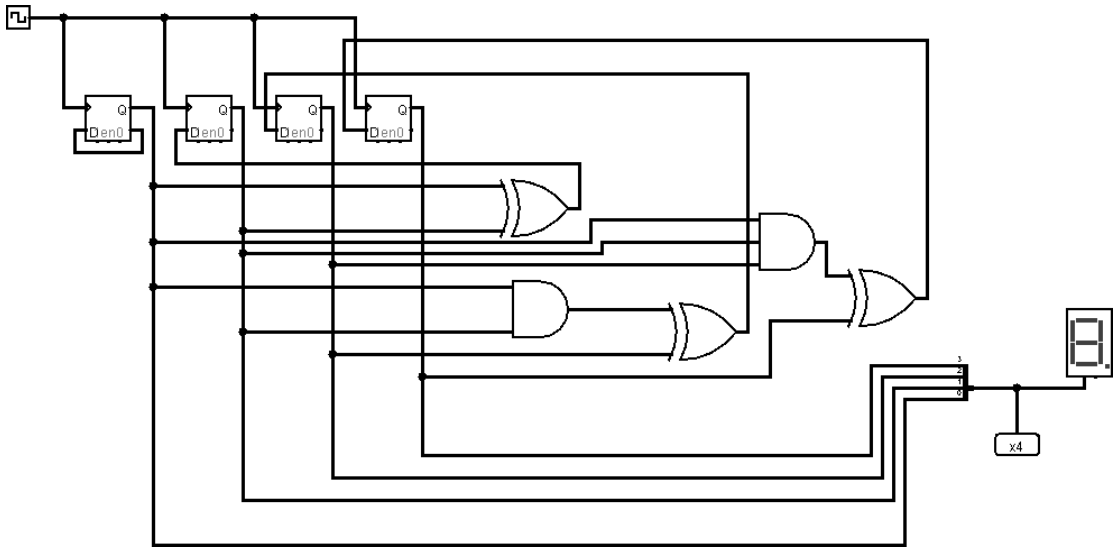
ENTRADA = 1, Flip Flop D = 1



[COMPONENTE 13]. Contador Síncrono.

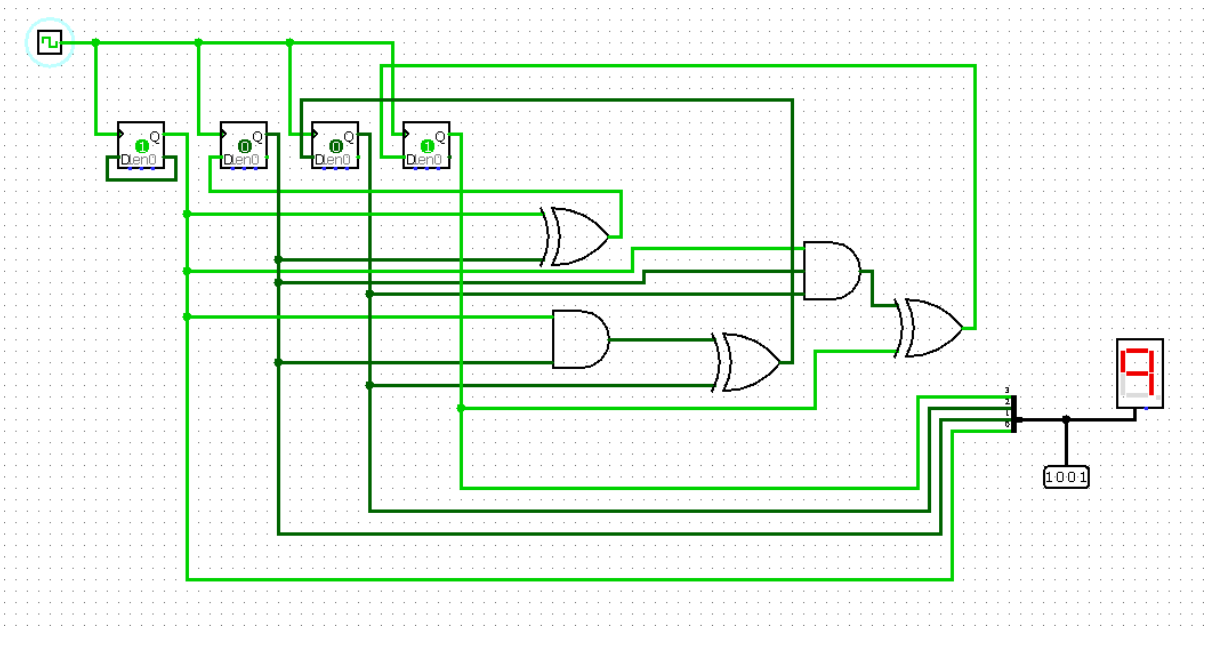
Um contador síncrono é um circuito sequencial utilizado para contar pulsos de clock de maneira ordenada, onde todas as mudanças de estado ocorrem simultaneamente em resposta ao clock. Este relatório descreve a implementação de um contador síncrono de 4 bits utilizando flip-flops do tipo D.

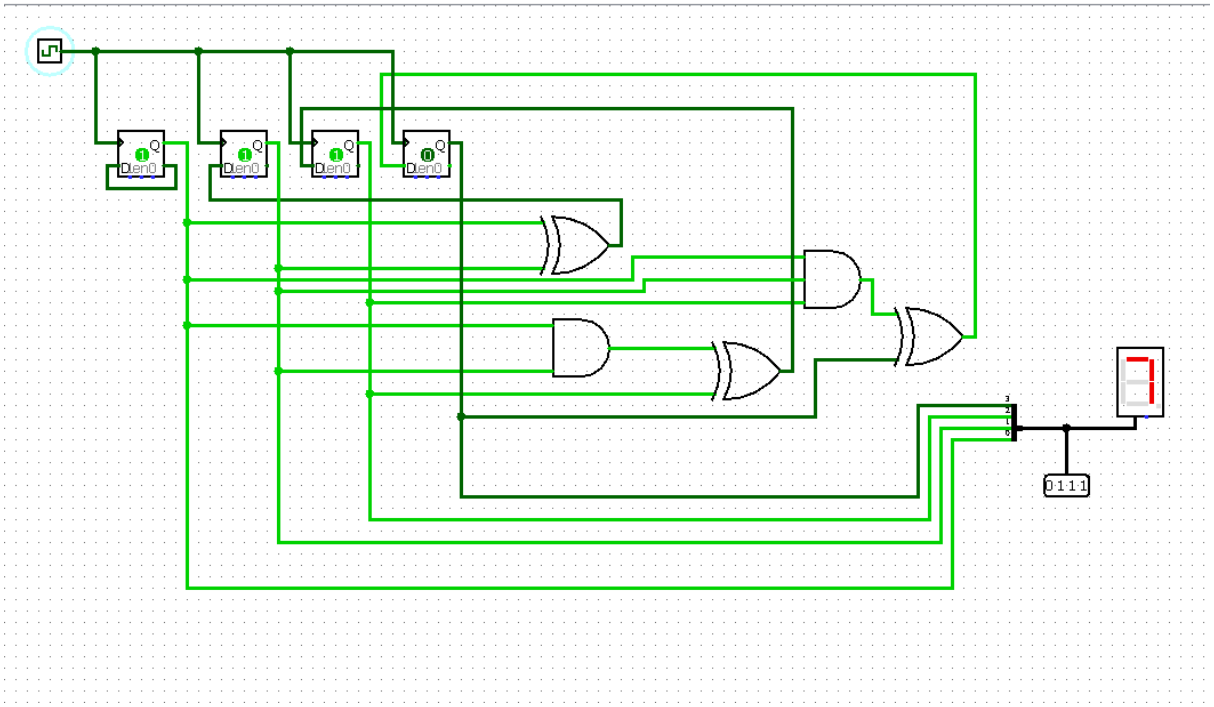
[CIRCUITO COMPLETO]



Utilizei o visor hexadecimal apenas por curiosidade, mas o circuito independe dele, pois coloquei o pino de saída.

[TESTES]

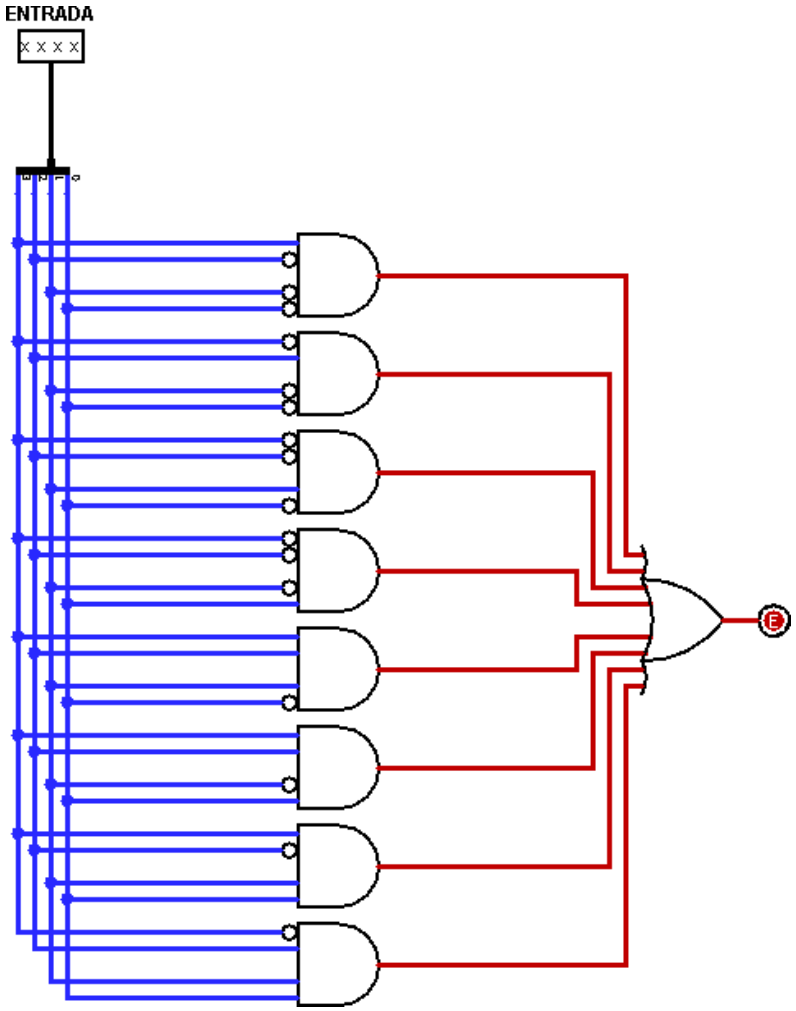




[COMPONENTE 14] Detector de Paridade ímpar

Circuito digital que verifica se o número de bits 1 em uma sequência binária é ímpar. Ele é amplamente utilizado em sistemas digitais e de comunicação para detecção de erros.

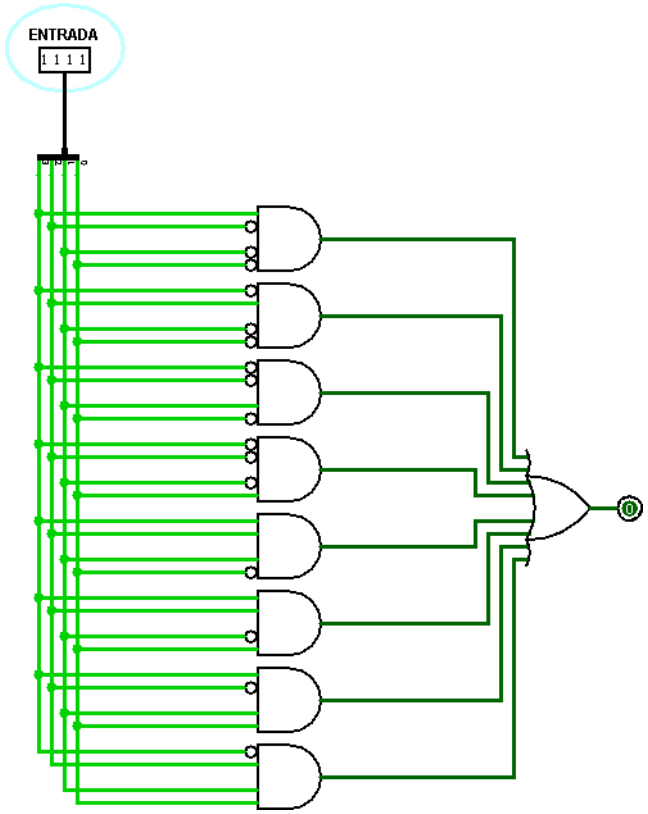
[CIRCUITO COMPLETO]



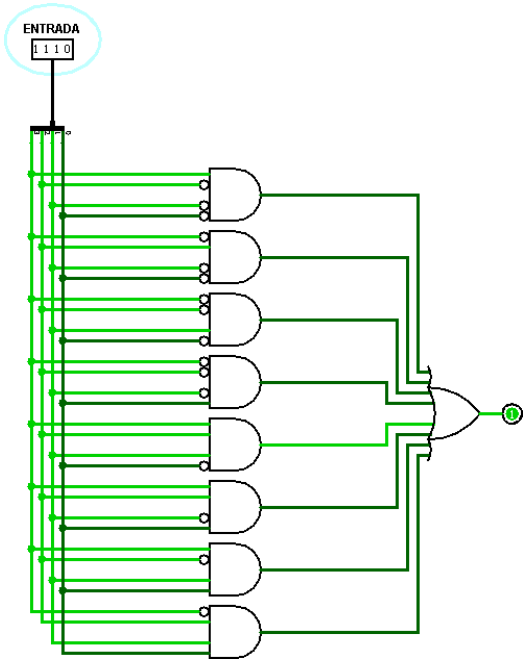
Possui uma entrada de 4 bits e uma saída que assume o valor 1 se o número de bits 1 for impar e 0 se o número de bits 1 for par.

[TESTES]

Número de bits 1 ímpar

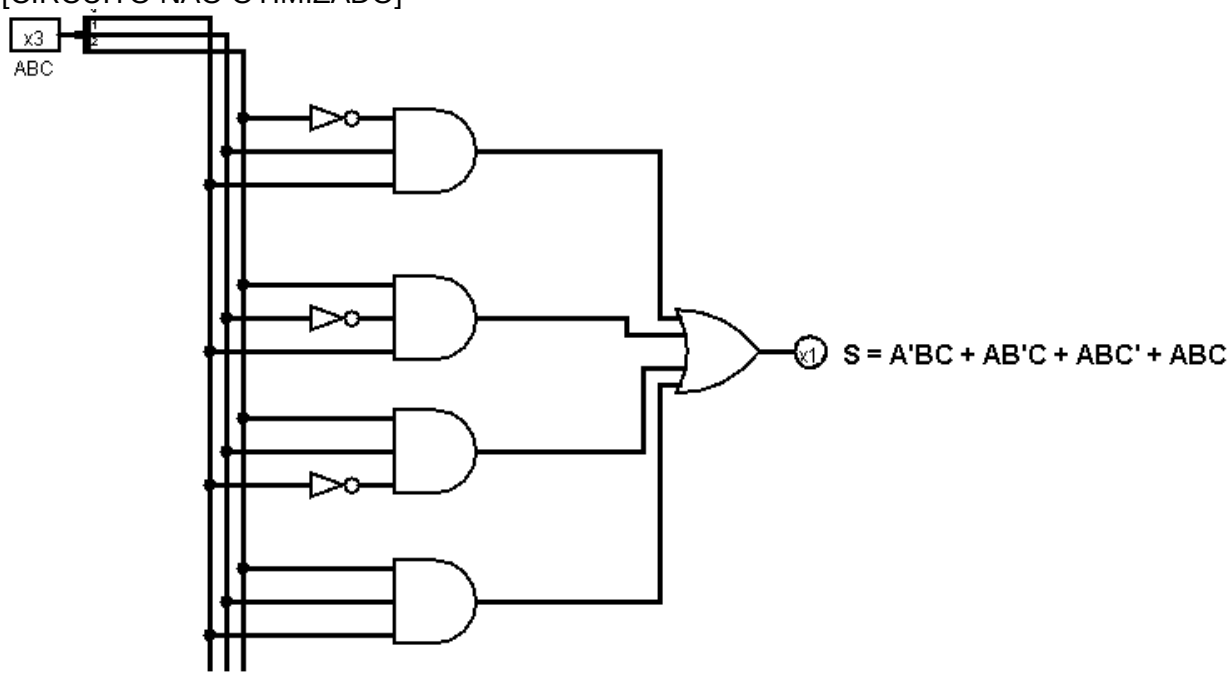


Número de bits 1 par

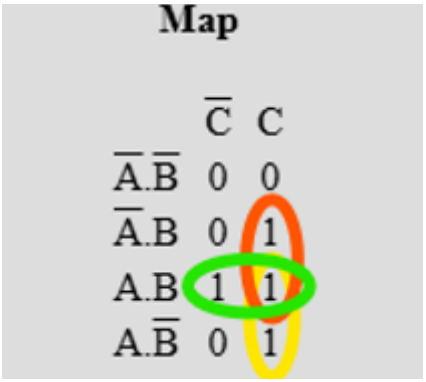


[COMPONENTE 15]. Resolva um problema de otimização lógica utilizando mapas de Karnaugh e implemente o circuito otimizado.

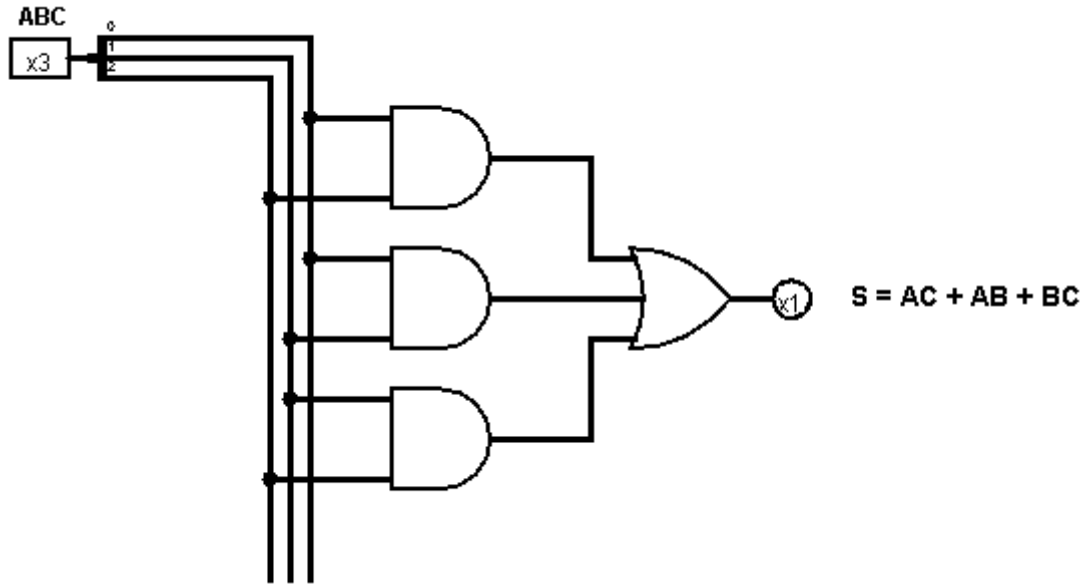
[CIRCUITO NÃO OTIMIZADO]



[MAPA DE KARNAUGH]



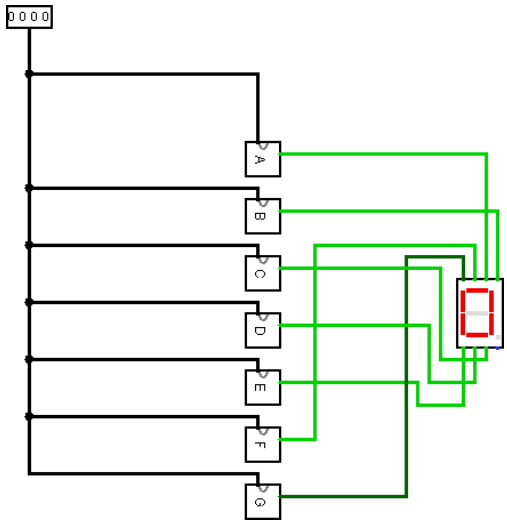
[CIRCUITO OTIMIZADO]



[COMPONENTE 16]. Decodificador de 7 Segmentos: Projete um circuito que converta um número binário de 4 bits para os sinais necessários para acionar um display de 7 segmentos (formato hexadecimal).

Circuito digital que converte uma entrada binária ou BCD (Binary-Coded Decimal) em sinais adequados para acionar os segmentos de um display de 7 segmentos. Esse tipo de display é amplamente utilizado para exibir números em dispositivos como relógios digitais, calculadoras e sistemas embarcados.

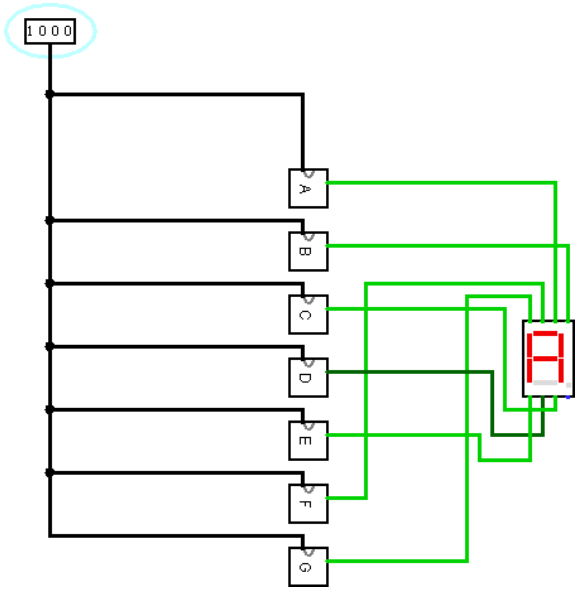
[CIRCUITO COMPLETO]



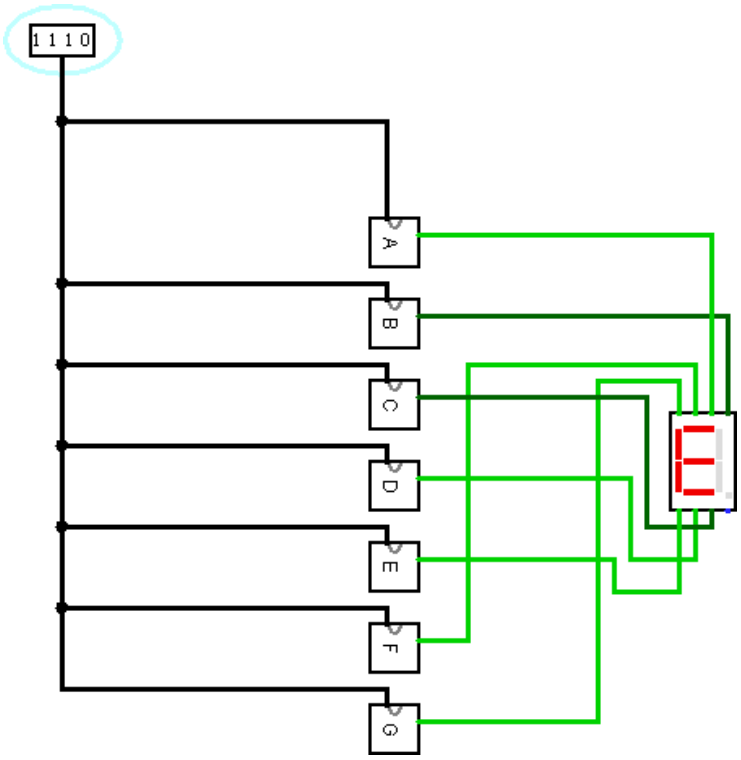
Entrada de 4 bits que representam os números de 0 a 9 e 7 linhas de saída, cada uma controlando um segmento específico do display (a, b, c, d, e, f, g).

[TESTES]

ENTRADA: 1000



ENTRADA: 1110



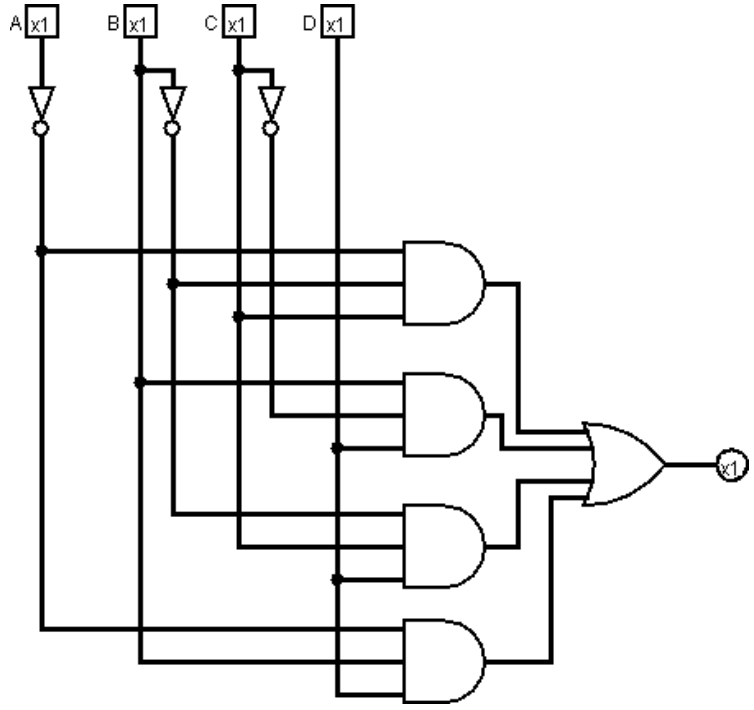
[COMPONENTE 17]. Detector de Número Primo: Crie um circuito que detecte se uma entrada binária de 4 bits representa um número primo. Utilize portas lógicas e mapas de Karnaugh para simplificar o circuito.

Basicamente o circuito emite sinal positivo para entrada quando o circuito detecta um número primo.

[MAPA DE KARNAUGH]

	$\overline{C}.\overline{D}$	$\overline{C}.D$	$C.D$	$C.\overline{D}$
$\overline{A}.\overline{B}$	0	0	1	1
$\overline{A}.B$	0	1	1	0
$A.B$	0	1	0	0
$A.\overline{B}$	0	0	1	0

[CIRCUITO COMPLETO]

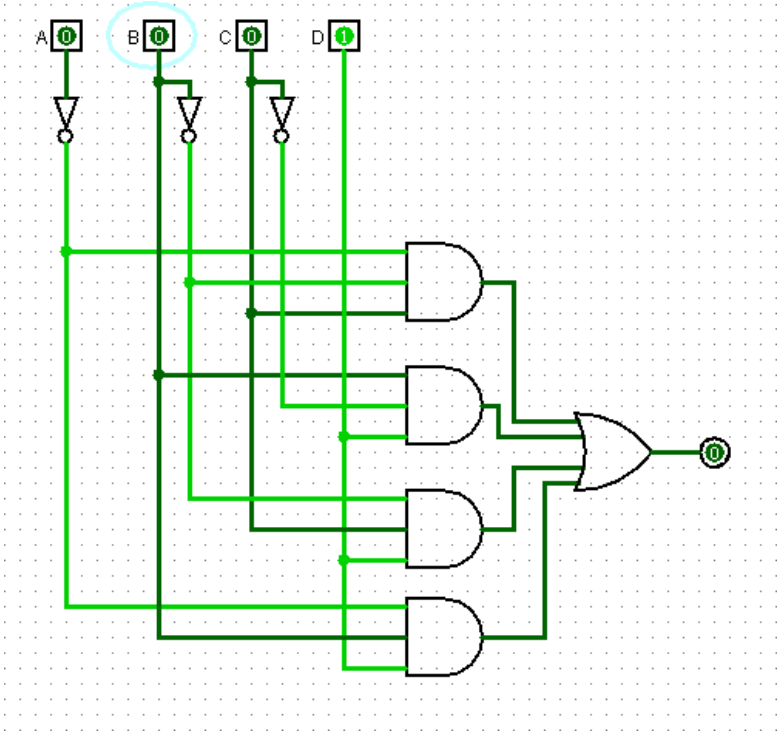


[TABELA VERDADE]

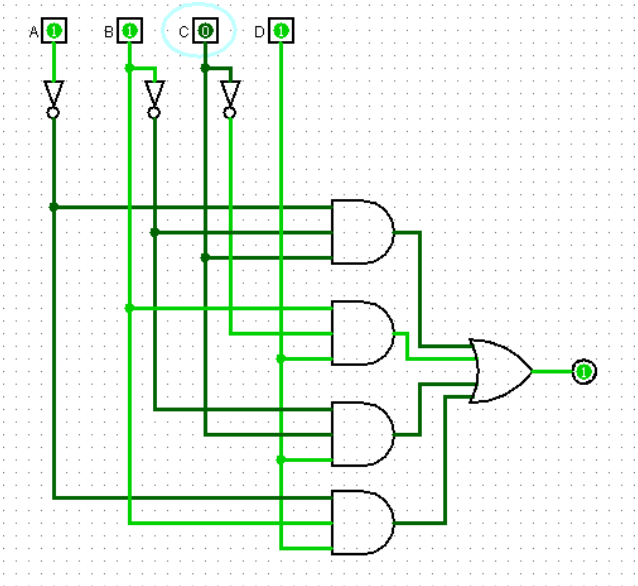
A	B	C	D	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

[TESTES]

0001 = 1 (não é primo)



Entrada 1101 = 13 (é primo)



CONCLUSÃO

A conclusão deste trabalho envolveu a criação e análise de diversos circuitos digitais essenciais para sistemas lógicos, utilizando apenas pinos de entrada e portas lógicas. Ao longo do desenvolvimento, foram abordados componentes como registradores, multiplexadores, somadores, decodificadores e detectores.

Através da utilização de portas AND, OR, NOT e suas combinações, foi possível criar circuitos eficiente, com foco na redução do número de portas necessárias para a implementação dos sistemas. Como a exibição de números em displays de 7 segmentos e a verificação de números primos em uma entrada binária. A utilização dos Mapas de Karnaugh permitiu simplificar as expressões booleanas, reduzindo o número de portas lógicas necessárias e melhorando a eficiência dos circuitos.

Em suma, este trabalho demonstrou a importância do uso de técnicas de simplificação e otimização no design de circuitos digitais, permitindo a criação de sistemas mais eficientes, com menor consumo de recursos e maior confiabilidade.

REFERENCIAS

http://www.cburch.com/logisim/docs/2.7/pt/html/libs/mem/flipflops.html

https://www.saber360.com.br/multiplexadores-e-dmux

https://embarcados.com.br/xor/

https://tecnoblog.net/responde/o-que-e-memoria-rom/

https://www.techtudo.com.br/noticias/2022/11/o-que-e-memoria-ram-descubra-qual-e-a-sua-funcao.ghtml

https://www.newtoncbraga.com.br/como-funciona/1214-art0159.html

https://www.saber360.com.br/somadores

https://pt.wikipedia.org/wiki/Unidade_l%C3%B3gica_e_aritm%C3%A9tica

https://www.youtube.com/watch?v=u_vr3CXRFHY

https://embarcados.com.br/cd4511-decodificador-display-7-segmentos/

https://www.youtube.com/watch?v=LwGUPlE_2Fk