UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIAS TECNOLÓGICAS
CURSO BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

APOLO MATOS PEDROSO
IGOR PADILHA DOS SANTOS

RELATÓRIO DO LABORATÓRIO DE CIRCUITOS

BOA VISTA RR
2024

APOLO MATOS PEDROSO
IGOR PADILHA DOS SANTOS

RELATÓRIO DO LABORATÓRIO DE CIRCUITOS
ATIVIDADE DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

Relatório referente à atividade do laboratório de circuitos da disciplina de arquitetura e organização de computadores, como requisito para obtenção da nota da avaliação parcial do semestre.
Professor: Dr. Herbert Oliveira Rocha

BOA VISTA RR
2024

1 INTRODUÇÃO

Relatório com objetivo de descrever os componentes da atividade “Laboratório de circuitos” bem como, explicar suas soluções, suas criações e suas funcionalidades. Para esse trabalho foi utilizado o programa Logisim para a construção e teste de todos os circuitos na qual, utilizando os conhecimentos de Circuitos Digitais, foram feitos, ao todo, 17 circuitos com com objetivos diferentes. Depois de tudo feito, os referidos circuitos foram postos em um repositório do GitHub, em forma de código fonte extraído pelo Visual Studio Code.

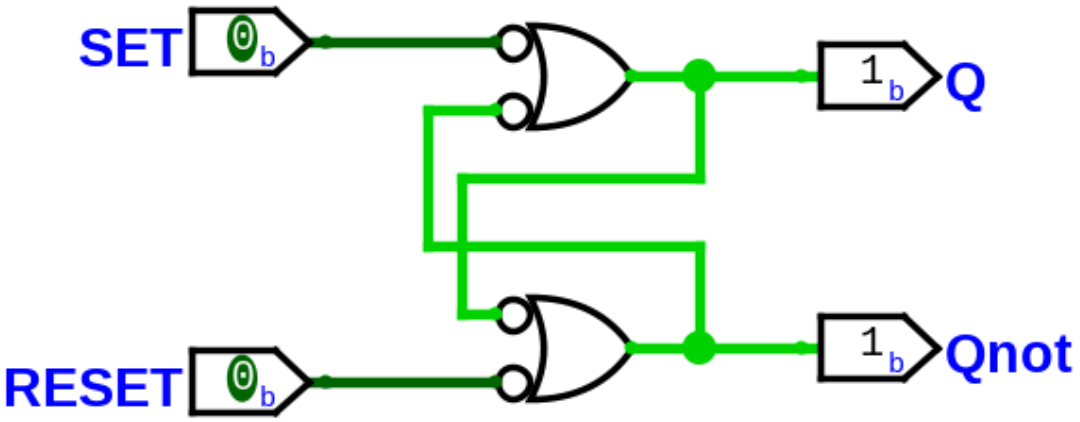
2 DESENVOLVIMENTO

2.1 FLIP-FLOP

O flip flop é um dispositivo utilizado para armazenar valores em células de memória, sendo uma das unidades mais básicas de armazenamento num computador. Existem vários designs de flip-flops, mas, de uma maneira geral, todos apresentam uma unidade Latch para fazer o registro de dados e portas lógicas para controlar os sinais de entrada.

2.1.1 Circuito Latch

Um latch é um circuito básico que também tem a função de armazenar dados, porém os circuitos do tipo latch respondem aos sinais de entrada de forma assíncrona. Existem várias formas de construir um latch, um NAND latch usa duas portas NAND interconectadas e dois sinais de entrada chamados de SET e RESET. Temos as saídas do circuito como Q e ~Q, que representam o valor armazenado e o seu inverso respectivamente.



Os sinais de entrada do NAND latch estão normalmente no estado alto, isso significa que para alterar o estado do circuito, o valor armazenado, são utilizados pulsos no valor baixo. Segue a tabela verdade de uma NAND latch:

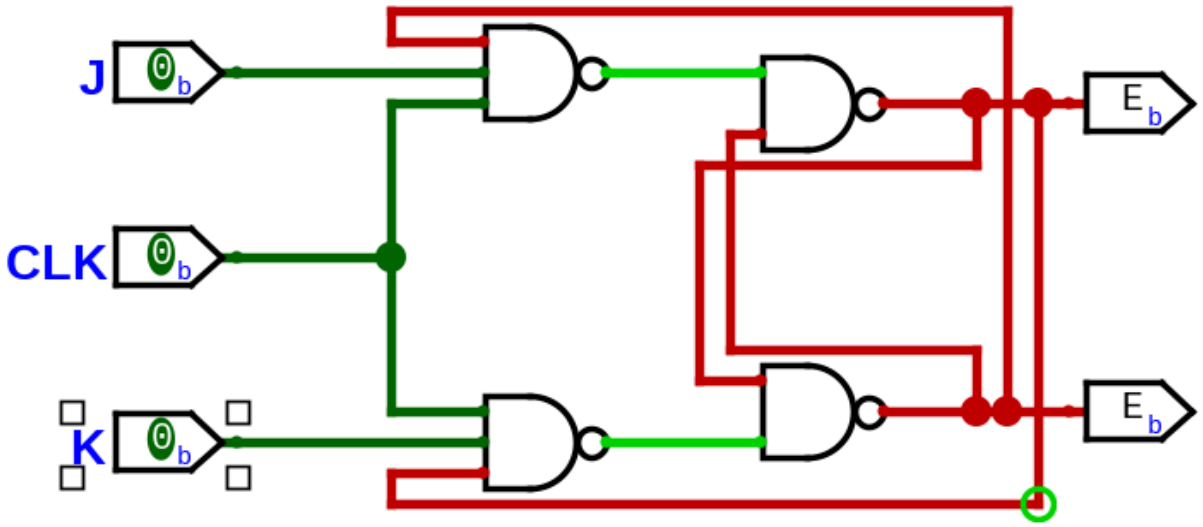
S (Set)	R (Reset)	Q (Atual)	Q' (Atual)	Q (Próximo)	Q' (Próximo)
0	0	0	1	1	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	1	Indefinido	Indefinido
1	1	1	0	Indefinido	Indefinido

Nota-se que no estado SET=0 e RESET=0 temos um comportamento indefinido, isso significa que o valor armazenado pelo latch não pode ser determinado, podendo ser 0 ou 1.

2.1.2 Flip-flop JK

O flip-flop JK adiciona algumas portas lógicas ao latch para adicionar controle e funcionalidades, além de tornar a resposta do circuito síncrona ao sinal da entrada

clock. Para entrada temos os sinais de controle J, K e clock e para a saída temos os sinais de dados do latch, Q e ~Q.



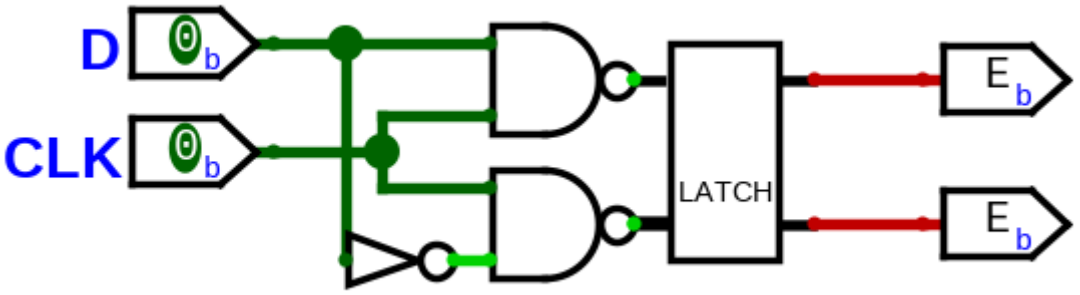
Como o flip-flop JK é um tipo de circuito síncrono, as suas operações só são realizadas quando o sinal de clock está em estado alto. O comportamento de um flip-flop JK pode ser observado completamente através da tabela:

J	K	Q (Estado)	Q' (Próximo)	Q' (Próximo)
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Um estado importante do flip-flop JK é o estado em que temos SET=RESET=1 ao sinal de clock alto. Nesse estado ocorre uma inversão no valor armazenado pelo latch, invertendo o valor de saída Q e ~Q.

2.1.3 Flip-flop D

O flip-flop D é um componente síncrono e apresenta o circuito de um latch internamente, assim como o flip-flop JK. O diferencial está nas entradas, enquanto o flip-flop JK os sinais J e K para conduzir o estado do latch, o flip-flop D utiliza apenas uma entrada D.



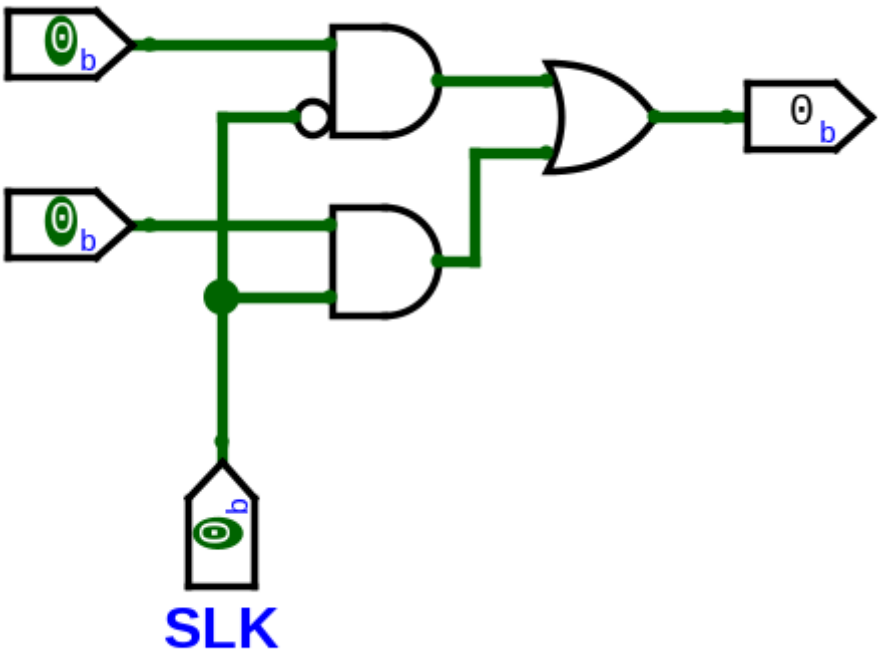
Como o flip-flop D também é um circuito síncrono, o seu funcionamento também é regido pelo sinal do clock. Podemos analisar os estados de um flip-flop D a partir da seguinte tabela:

(inserir tabela verdade do flip-flop D)

2.2 MULTIPLEXADOR

O circuito multiplexador é um circuito que recebe vários canais de entrada e redireciona para uma única saída. Esse componente é comumente utilizado para controlar o fluxo de canais de comunicação e saídas de vários circuitos para uma canal único.

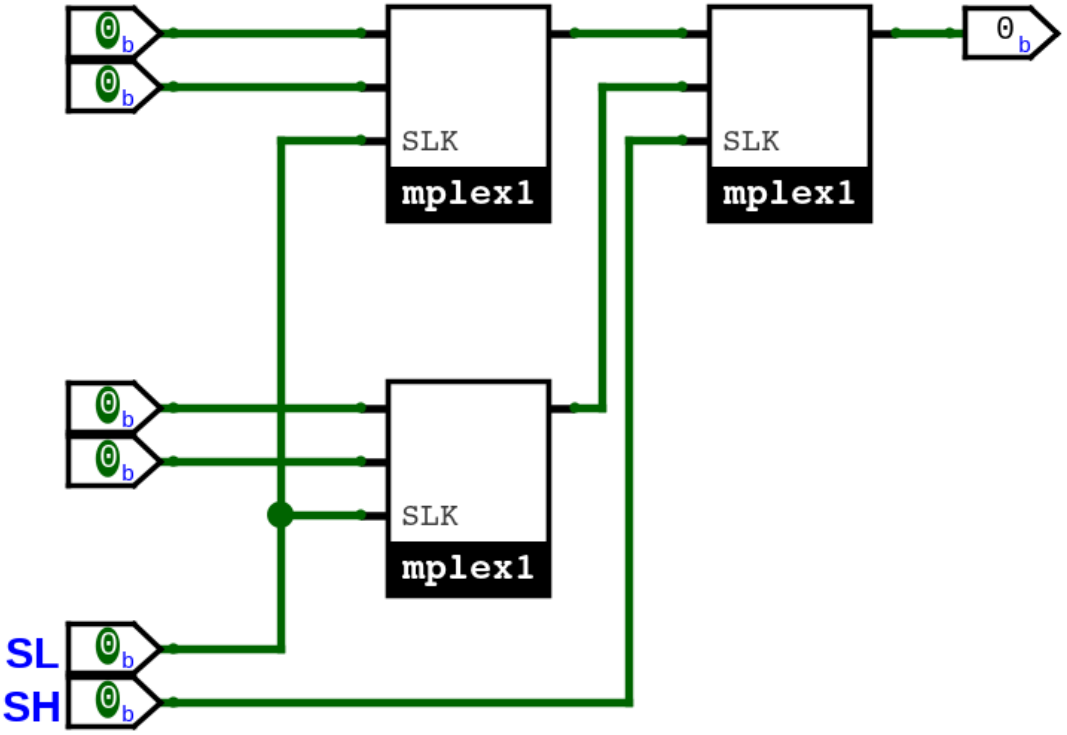
Para determinar qual canal de entrada deve ser redirecionado para a saída o multiplexador conta com uma porta seletora, que recebe o número de identificação do canal selecionado na forma binária.



O circuito do multiplexador acima tem canal de entrada de dados D e uma porta seletora S. Um multiplexador de duas entradas precisa uma porta seletora com 1 bit de largura para a sua porta seletora, já que 1 bit é suficiente para identificar todas as suas entradas. O funcionamento do multiplexador de duas entradas representado pelo circuito acima pode ser resumido da seguinte forma.

- Quando SLK=0, a primeira entrada é redirecionada
- Quando SLK=1, a segunda entrada é redirecionada

Para construir multiplexadores um maior número de entradas é possível fazer uma composição de multiplexadores menores. O circuito abaixo é um multiplexador de 4 entradas construído a partir de 3 multiplexadores de 2 entradas.



O multiplexador de 4 entradas necessita de uma porta seletora de 2 bits de largura, os bits de entrada estão representados por SL, o bit menos significativo, e SH o bit mais significativo.

O primeiro, SL, bit é utilizado para selecionar a entrada do primeiro grupo de multiplexadores. O segundo bit, SH, é utilizado para selecionar o resultado do grupo adequado para ser redirecionado a saída do circuito.

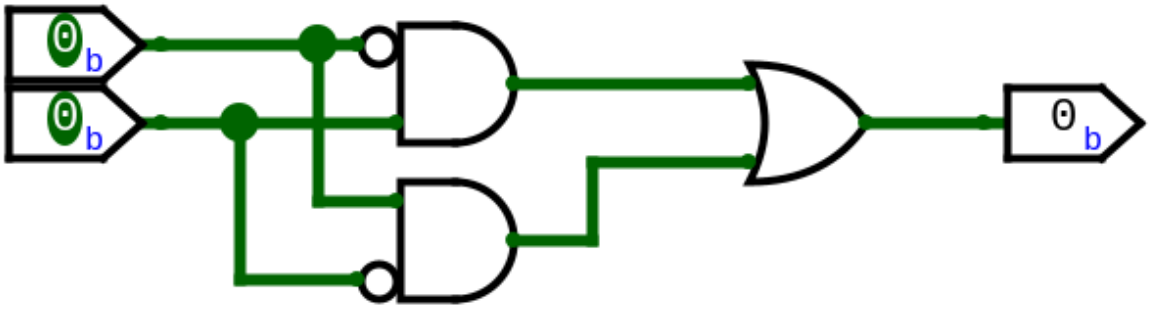
2.3 PORTA XOR

A porta XOR é um porta lógica básica utilizada para construção de circuitos mais complexos. A saída de uma porta XOR tem o nível alto exclusivamente quando apenas um dos seus valores de entrada está no nível alto, por isso essa porta também é chamada de ou exclusivo. O comportamento de uma porta XOR pode analisado a partir da seguinte tabela:

A	B	A ⊕ B
0	0	0
0	1	1

1	0	1
1	1	0

Analisando os casos em que a saída é 1 na tabela, podemos extrair a expressão $A \oplus B = (A \cdot \bar{B}) + (\bar{A} \cdot B)$. A partir dessa expressão é possível representar o comportamento da porta lógica utilizando as portas AND, OR e NOT.

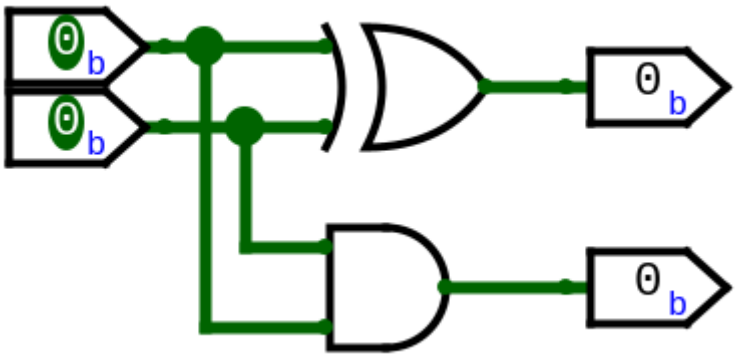


2.4 SOMADOR +4

Esse circuito recebe um número binário de 8 bits e acrescenta 4 ao seu valor. Para isso são utilizados somadores simples de 1 bit

2.4.1 Somador simples de 1 bit

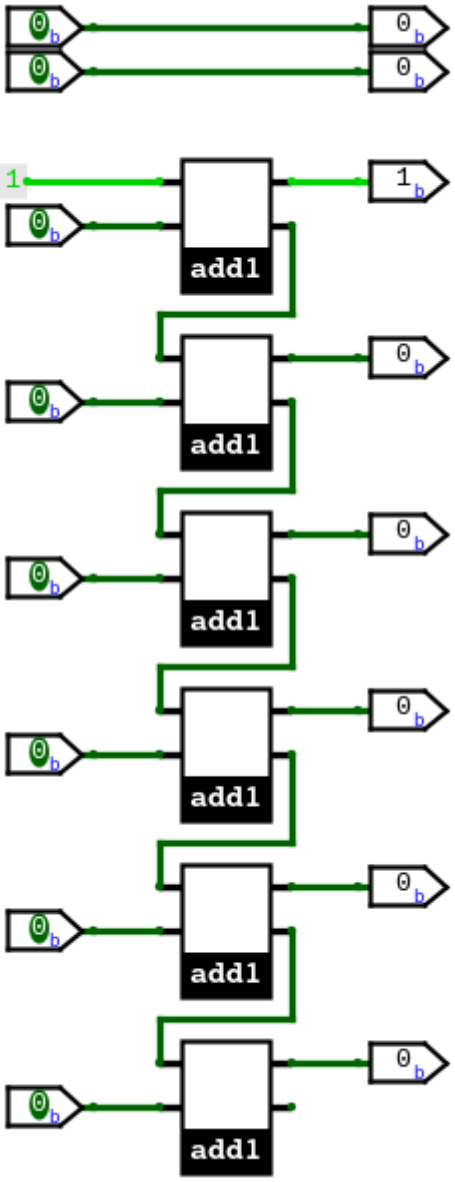
O somador é um circuito que realiza a operação de soma em dois operandos, no caso do somador de 1 bit cada operando tem 1 bit de largura. Esse tipo de somador é bem simples, possuindo apenas duas entradas e duas saídas. Cada entrada representa um operando e as saídas representam os bits do resultado da soma.



Para fazer o cálculo do primeiro bit de saída usa-se uma porta XOR e para o segundo bit usa-se uma porta AND.

2.4.2 Implementação

O circuito construído recebe oito entradas, cada uma sendo um bit do operando, e oito bits de saída para o resultado. Foram utilizados 5 somadores simples. de 1 bit, já que o circuito se aproveita do fato da representação do número 4 em binário ter os dois primeiros bits não menos significativos como 0. Dessa forma, a soma é realizada a partir do terceiro bit do operando, repetindo o valor dos dois primeiros bits de entrada na saída.



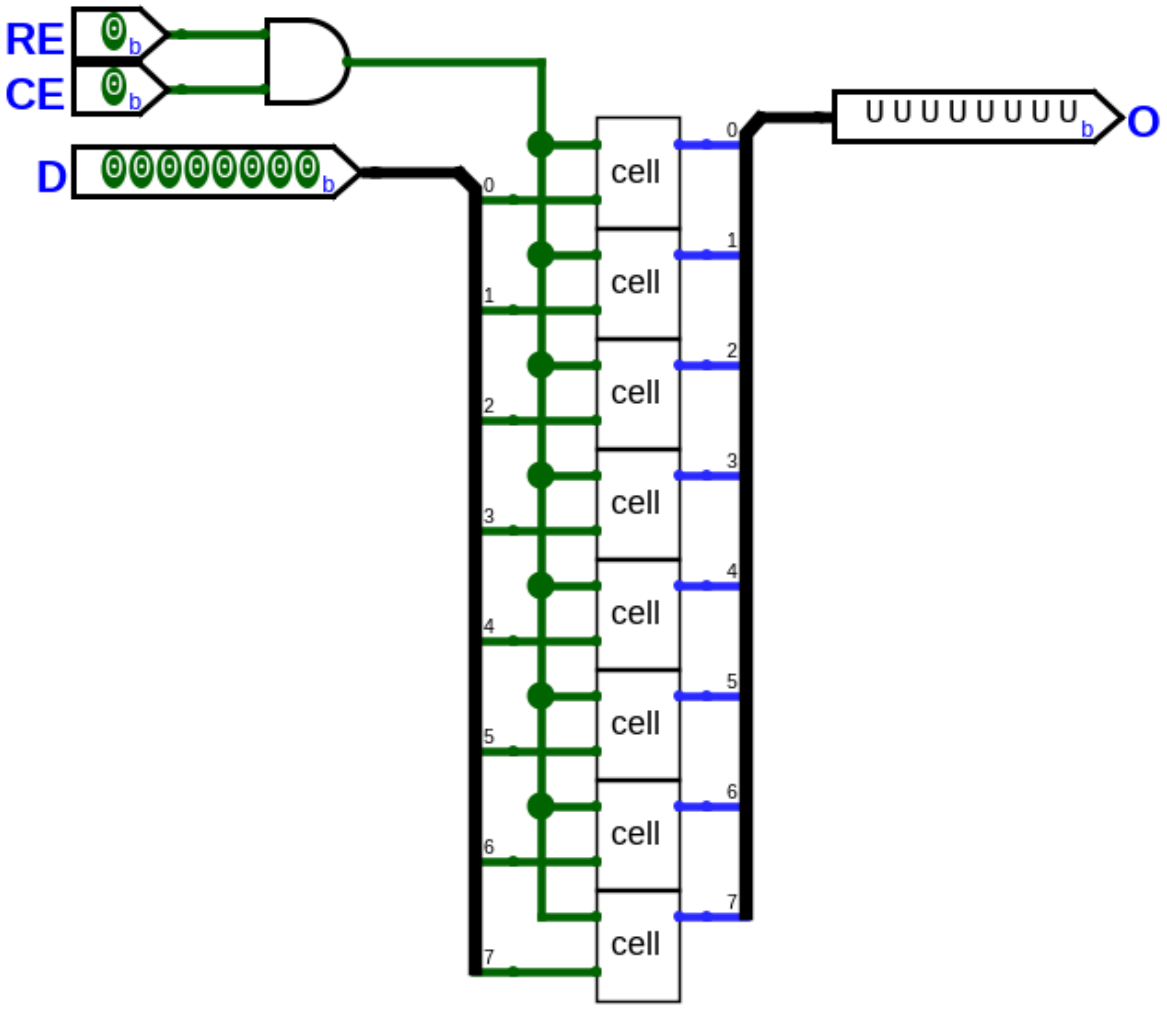
2.5 MEMÓRIA ROM

A memória ROM (Read-Only Memory) é um tipo de memória que armazena dados de forma permanente, ou seja, os dados gravados nela não podem ser modificados facilmente. Este tipo de memória é amplamente utilizado em sistemas embarcados, como computadores, microcontroladores e dispositivos eletrônicos, para armazenar programas ou dados que não precisam ser alterados com frequência, como o firmware de um dispositivo.

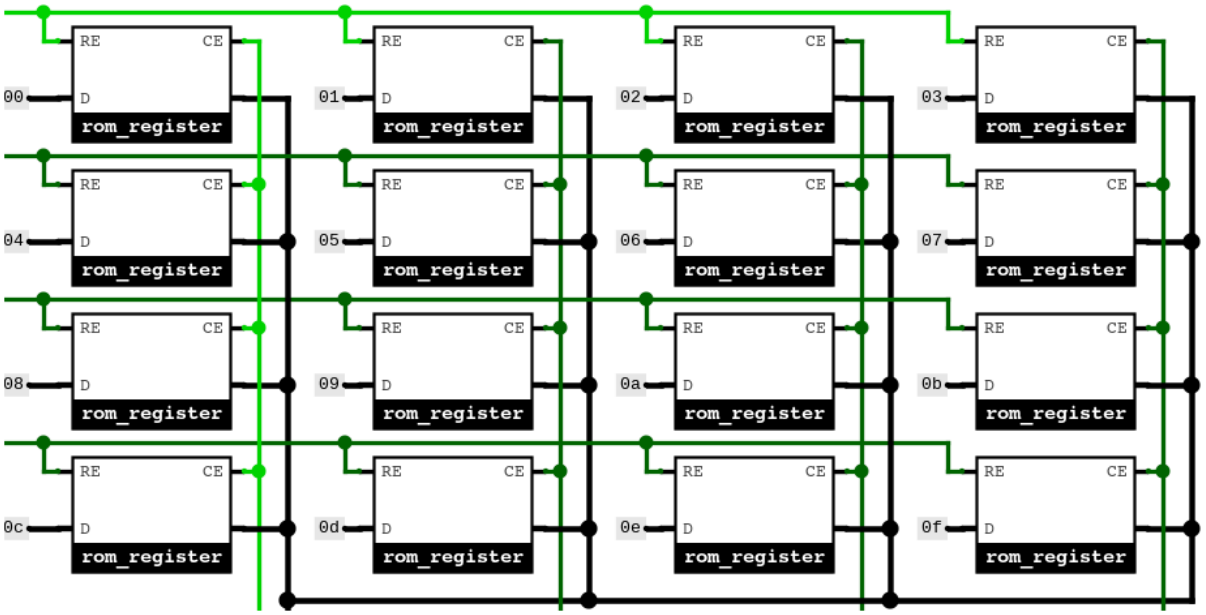
O funcionamento da memória ROM implementada é baseado em um arranjo de células de memória, onde cada célula é responsável por armazenar um valor de 8 bits. O endereço de memória é de 4 bits, permitindo acessar 16 células de memória. Cada registrador armazena uma palavra de 8 bits de dados.

A interface externa do circuito da memória expõe um canal de dados ADR, um sinal de controle CS e um canal de saída de 8 bits. O canal ADR é um canal de 4 bits de largura e é usado para selecionar a célula de memória que deve ser lida pela memória. Já o sinal CS (Chip Select) controla a leitura dos dados armazenados na memória, se o sinal CS apresenta o nível alto, a memória inicia o processo de leitura da célula de memória identificada no canal ADR.

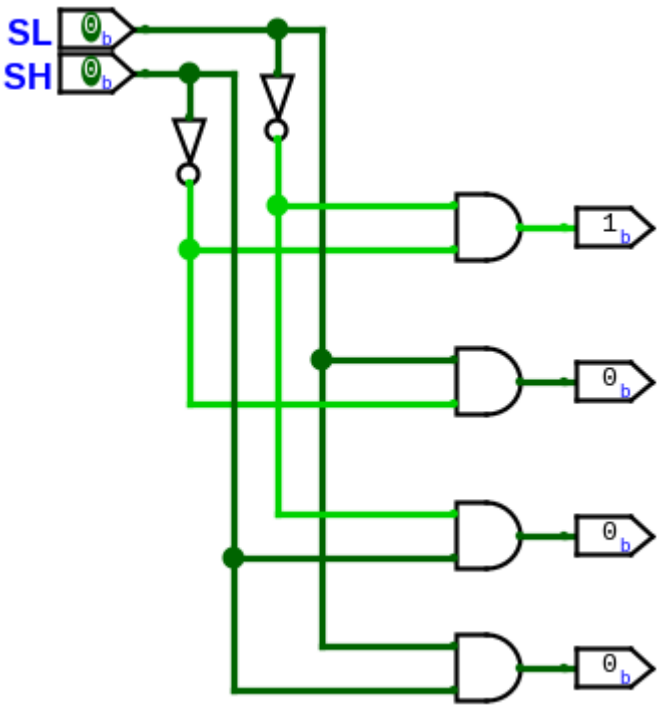
Cada célula de memória é composta por oito transistores que recebem um valor de entrada predefinido 8 bits e recebe dois sinais de leitura RE (Row enable) e CE (Column enable), como cada célula de memória armazena 8 bits, cada célula tem uma linha de saída de 8 bits de largura. A célula entra no modo de leitura exclusivamente quando RE e CE estão em nível alto. Na leitura, a célula de memória indica o bit 0 conduzindo a linha de saída para o nível lógico baixo e indica o bit 1 mantendo a linha em estado de alta impedância. Quando uma célula não está sendo lida, todas as suas linhas de saída são mantidas em estado de alta impedância.



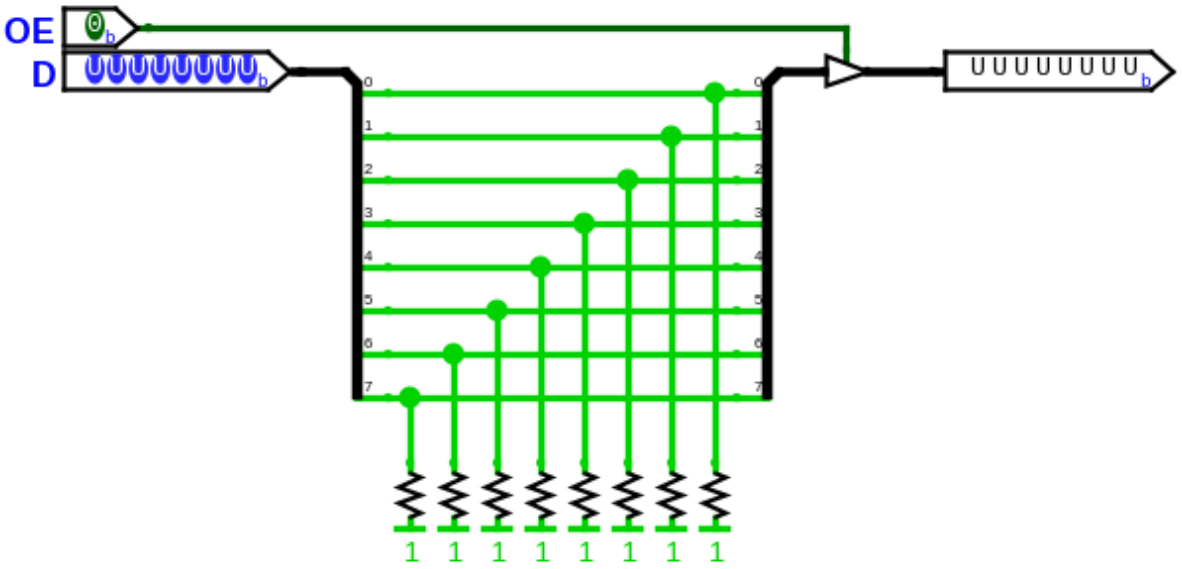
As células de memória estão dispostas em forma de grid, cada linha compartilha um mesmo sinal RE e cada coluna compartilha um sinal CE, todas células as saídas das células estão conectadas a mesma linha de saída. Quando o processo de leitura começa, o endereço em ADR é dividido em duas partes de 2 bits de largura, a parte menos significativa é utilizada e a parte mais significativa são utilizadas para decodificar a linha e a coluna da célula de memória a ser lida, respectivamente.



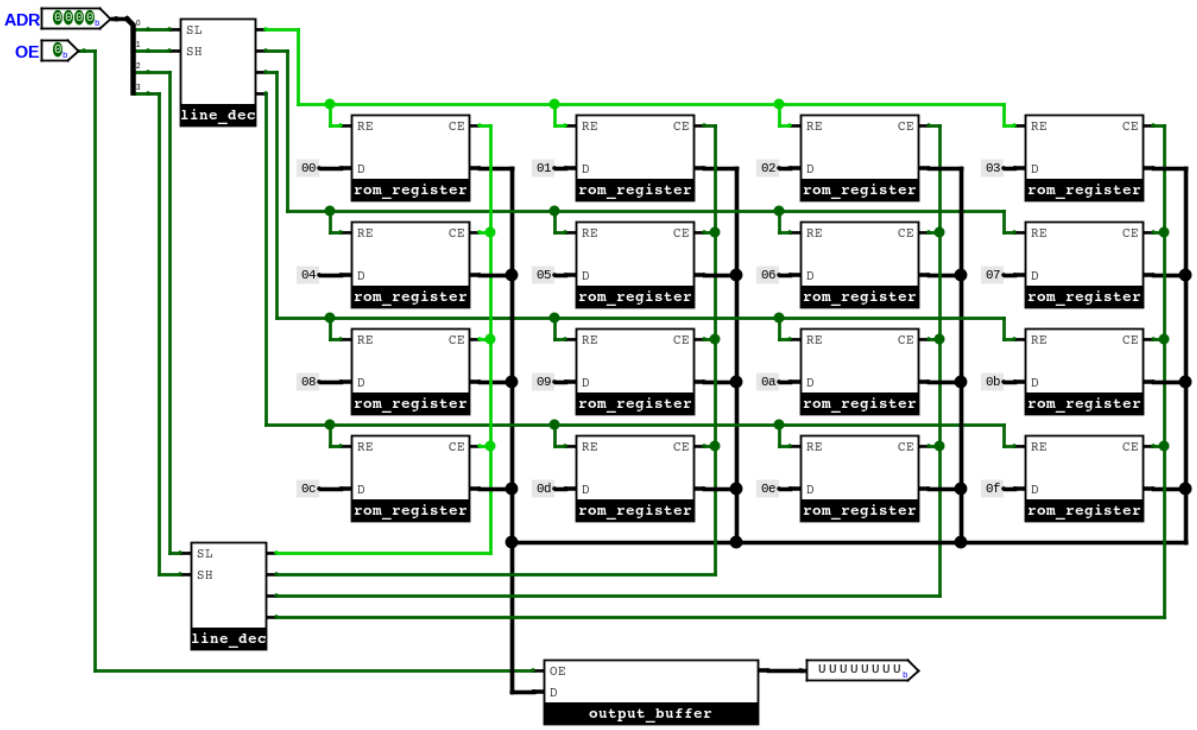
Os decodificadores são circuitos simples que utilizam uma combinação de portas AND, NOT e OR para determinar qual linha está sendo selecionada. O circuito do decodificador recebe duas linhas de entrada, para a indicar a seleção, e tem 4 linhas de saída. As linhas são mantidas num estado normalmente baixo. Quando uma linha é selecionada, o estado da linha de saída é conduzido para o nível alto.



O resultado da leitura das células da memória é processado pelo circuito do Output Buffer, que recebe as linhas de saída. A função desse circuito é manter normalizar o estado das linhas que estejam em estado de alta impedância para o nível alto, isso é feito através de um resistor pull-up conectado a cada linha de entrada. Além disso, para evitar que valores inválidos sejam transmitidos para a saída, o circuito do Output Buffer também recebe um sinal de controle OE, que só permite a leitura das linhas em caso esteja no nível alto.



O processo de leitura da memória ROM envolve ativar a linha de controle CS e enviar os dados de endereço da célula para a entrada ADR. O endereço da célula será decodificado pelos decodificadores de linha e coluna que enviarão os sinais de ativação para a linha e coluna da célula selecionada. A célula selecionada escreverá os dados na linha de saída compartilhada e os dados serão transformados e redirecionados pelo Output Buffer para a linha de saída da memória ROM.



2.6 MEMÓRIA RAM

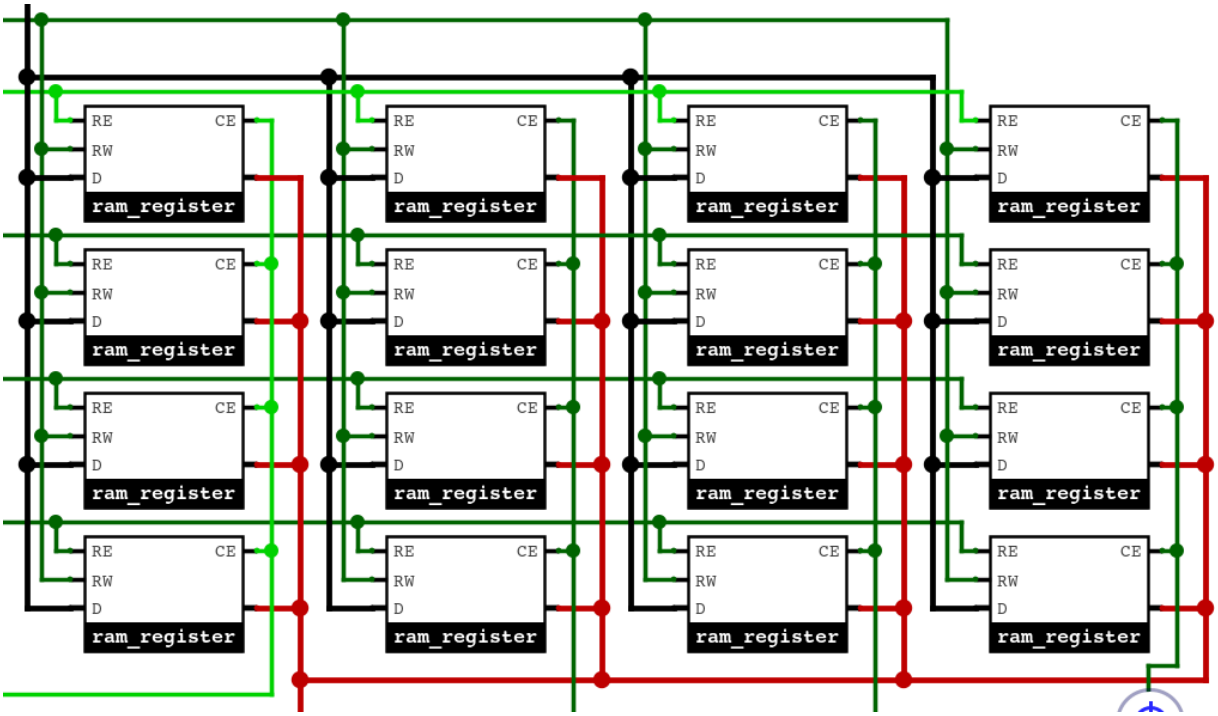
A memória RAM (Random Access Memory) é um tipo de memória temporária, conhecida como volátil, usada para armazenar dados de curto prazo durante a execução de programas e operações do sistema. Diferente da memória ROM, que armazena dados permanentemente, a RAM permite a leitura e a gravação de dados, mantendo-os enquanto o dispositivo está ligado. Quando o sistema é desligado, os dados na RAM são perdidos.

O funcionamento da memória RAM depende de um arranjo de células de memória, onde cada célula armazena um valor de 8 bits. A memória possui um endereço de 4 bits, permitindo o acesso a 16 células de memória. Cada registrador contém uma palavra de 8 bits de dados.

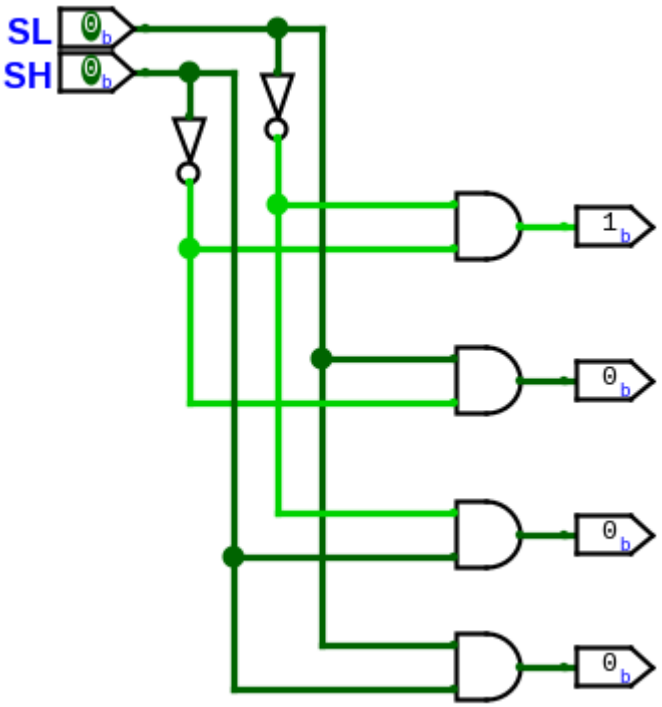
A interface externa da memória expõe dois canais de dados, ADR e Di, dois sinais de controle, CS e RW, e um canal de saída Do de 8 bits. O canal ADR, com 4 bits de largura, seleciona a célula de memória a ser lida. O canal Di é usado para escrever dados na memória. O sinal CS (Chip Select) controla a leitura e escrita de dados; quando CS está em nível alto, a memória inicia a leitura ou escrita da célula de memória identificada no canal ADR.

Cada célula de memória, composta por um registrador de 8 bits, recebe uma linha de dados D e três sinais de controle: RE (Row enable), CE (Column enable) e RW (Read/Write). Cada célula armazena 8 bits usando um registrador interno e possui uma linha de saída de 8 bits de largura. A célula só opera em leitura ou escrita quando RE e CE estão em nível alto. A operação de leitura é indicada pelo sinal RW em nível baixo, direcionando o valor armazenado no registrador para a saída. Na operação de escrita, o sinal RW está em nível alto e a célula grava os dados da linha D no registrador. Quando não está em uso, a linha de saída da célula fica em estado de alta impedância.

As células de memória estão organizadas em uma grade, onde cada linha compartilha o sinal RE e cada coluna, o sinal CE. As saídas das células são conectadas a uma linha de saída comum, e as entradas de dados estão na linha Di, com os sinais RW também compartilhados. Durante a leitura, o endereço em ADR é dividido em duas partes de 2 bits: a parte menos significativa decodifica a coluna e a parte mais significativa decodifica a linha da célula a ser lida.

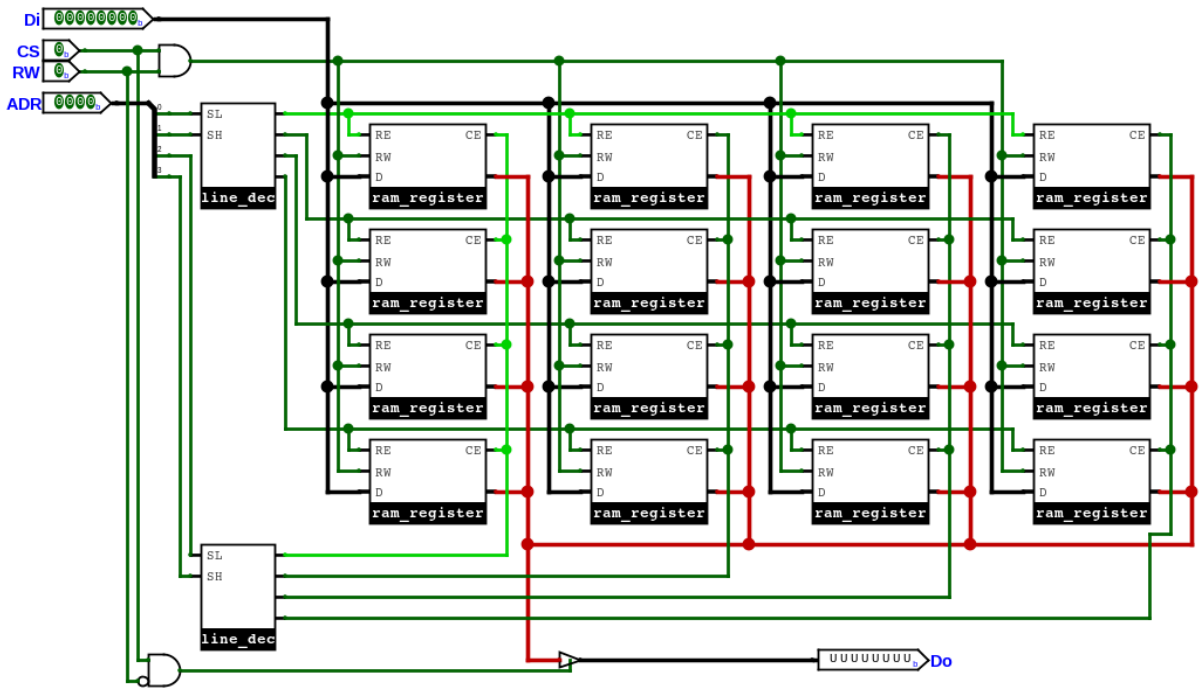


Os decodificadores utilizam portas lógicas AND, NOT e OR para determinar a linha selecionada. Recebendo duas linhas de entrada para a seleção, o decodificador possui 4 linhas de saída, normalmente em nível baixo. Quando uma linha é selecionada, o estado da saída muda para nível alto.



O processo de leitura da memória RAM envolve ativar o sinal de controle CS, baixar o nível da entrada RW e enviar o endereço da célula para ADR. O endereço é decodificado pelos decodificadores de linha e coluna, enviando sinais de ativação para a célula correspondente. A célula escreve os dados na linha de saída compartilhada, controlada por um buffer acionado pelo sinal RE, para evitar transmissão de valores inválidos.

Durante a escrita, os sinais CS e RW estão em nível alto, e os dados de endereço e a serem gravados são enviados para ADR e Di. A célula de memória é ativada pela decodificação de linha e coluna do endereço ADR. O sinal RW é direcionado para a linha RW compartilhada, permitindo que a célula selecionada leia e grave os dados na linha de entrada compartilhada.

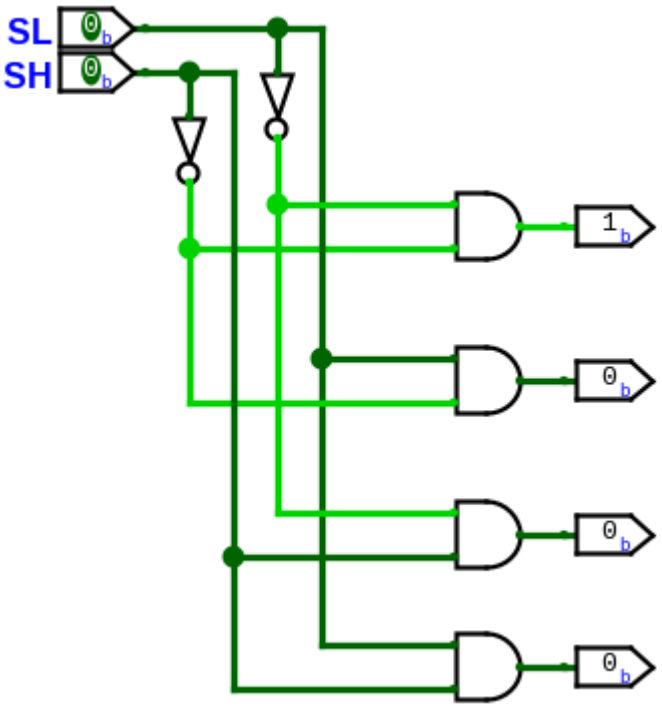


2.7 BANCO DE REGISTRADORES

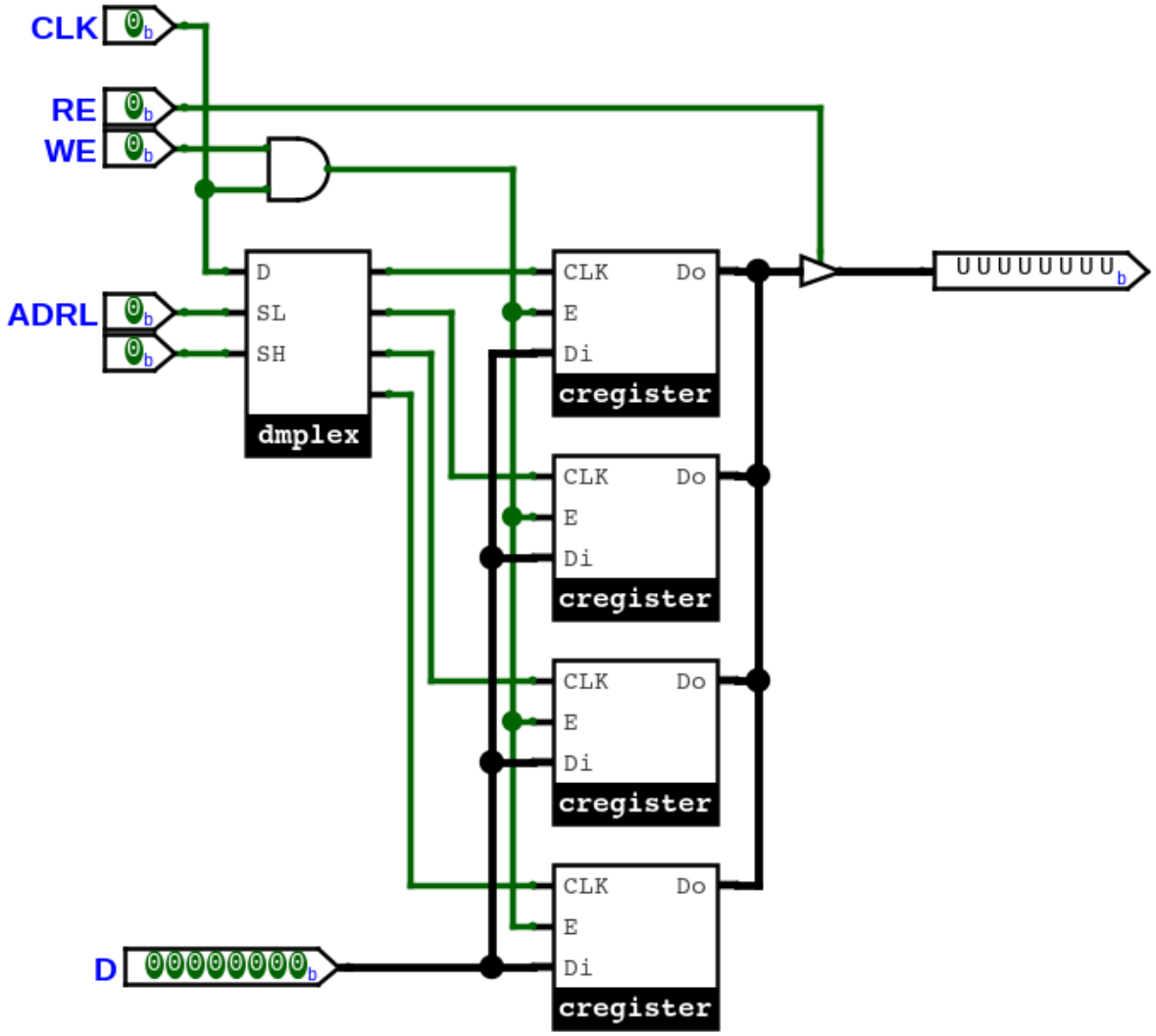
Banco de registradores é um circuito que armazena dados, permitindo leitura e escrita de valores em registradores distintos. Nesta implementação, utiliza-se um endereço de 2 bits para selecionar um registrador, dados de 8 bits para armazenamento e duas flags de controle: RE (Read Enable) e WE (Write Enable).

O banco é composto por 4 registradores capazes de armazenar valores de 8 bits cada. Cada registrador possui uma linha de dados D e dois sinais de controle: CLK (Clock) e E (Enable). O registrador só é ativado para operações de leitura ou escrita quando o CLK está em nível alto. Durante a leitura, o sinal E fica em nível alto, permitindo que o registrador grave os dados na linha D. Para leitura dos dados presentes no registrador, um sinal de ativação é enviado pelo CLK; quando ativado, o registrador transmite os dados armazenados para a linha de saída.

O decodificador é um circuito simples que combina portas lógicas AND, NOT e OR para identificar a linha selecionada. Recebendo duas linhas de entrada para seleção, o decodificador possui 4 linhas de saída, normalmente em nível baixo. Quando uma linha é selecionada, o estado da saída muda para nível alto.



Os registradores estão alinhados, compartilhando o sinal E. As saídas dos registradores conectam-se à mesma linha de saída, e as entradas de dados estão na linha D, com sinais RW também compartilhados. Durante a leitura ou escrita, o endereço em ADR é decodificado, ativando o registrador correspondente. Para a escrita, o sinal WE é enviado a todos os registradores, mas apenas o registrador ativado realiza a operação, lendo os dados na linha D. Na leitura, o registrador ativo transmite os dados para a linha de saída compartilhada, controlada pelo sinal RE. Quando RE não está ativo, a linha de saída fica em estado de alta impedância.



2.8 SOMADOR DE 8 BITS

A soma binária é uma operação fundamental em sistemas digitais, onde os números são representados em base dois. Nesse sistema, cada dígito pode ser 0 ou

1, e a soma de dois números binários segue regras específicas semelhantes à adição decimal, mas ajustadas para a base dois.

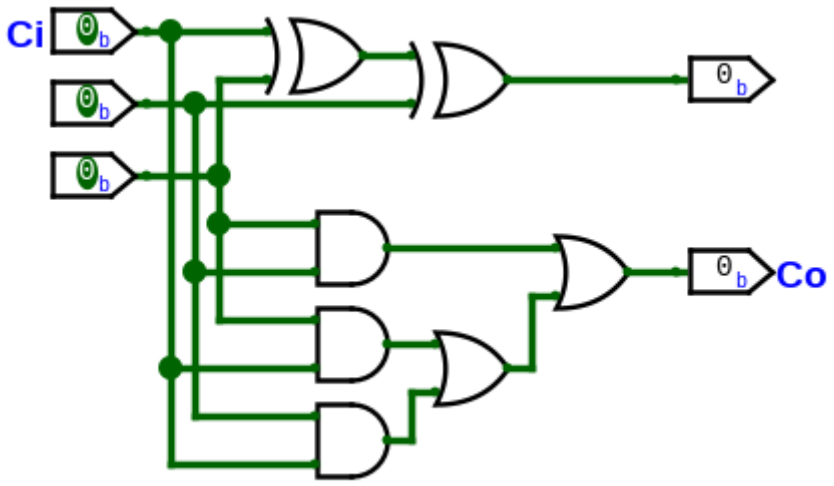
Para somar dois bits, consideramos as entradas A e B, além de um bit de transporte (carry-in) da operação anterior. O resultado da soma é dado pela combinação desses bits, e pode gerar um bit de saída (soma) e um bit de transporte para a próxima operação (carry-out).

A	B	Cin	S	Cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

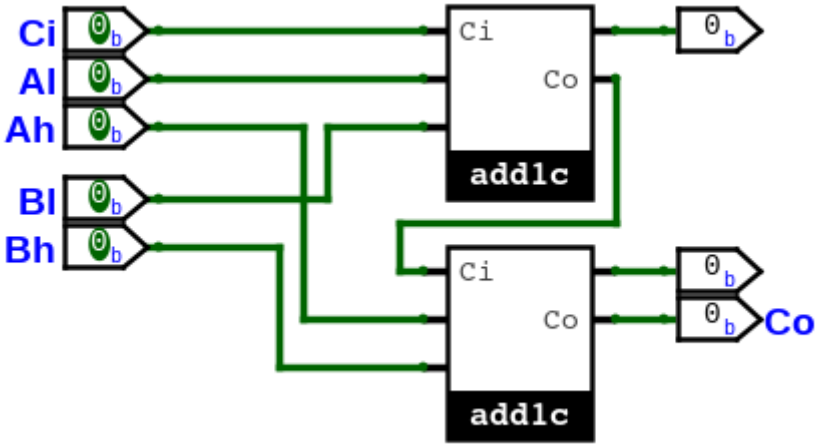
A partir dessa tabela pode extrair as expressões:

- $S = A \oplus B \oplus Cin$
- $Cout = (A \cdot B)+(A \cdot Cin)+(Cin \cdot B))$.

Dessa forma o circuito de um somador de 1 bit completo é:

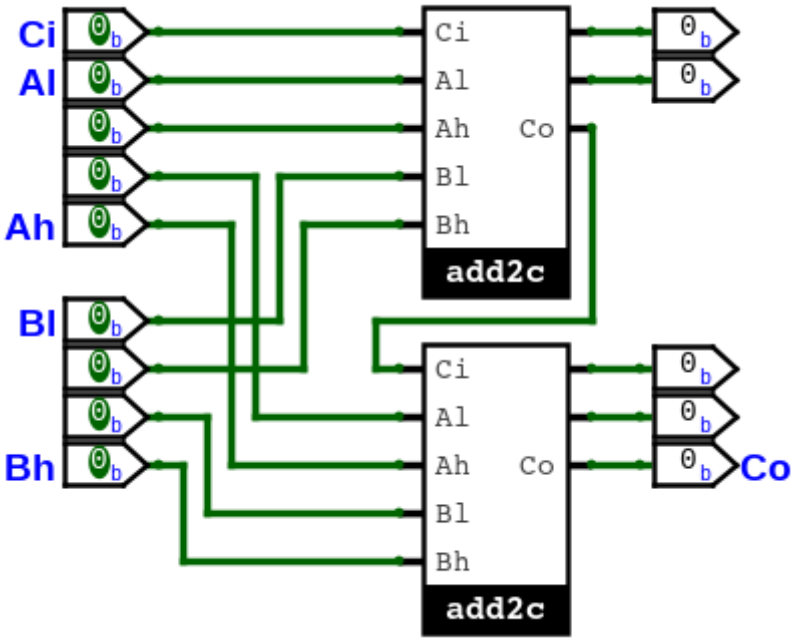


Um somador de 2 bits pode ser construído a partir de dois somadores de 1 bit. Neste arranjo, o carry-out do primeiro somador é conectado ao carry-in do segundo somador, permitindo que o bit de transporte seja levado em conta na próxima soma.

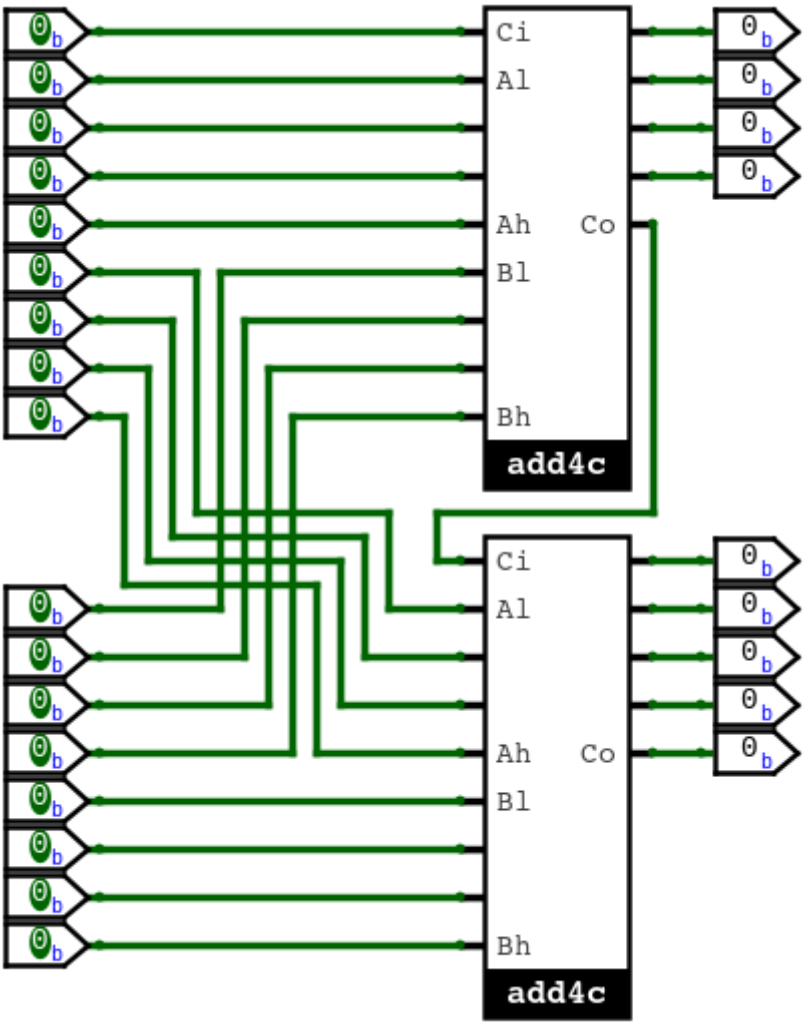


Para criar somadores de maior largura, como um somador de 4 bits ou de 8 bits, utilizamos o mesmo princípio de encadeamento através dos bits de transporte.

- Para montar um somador de 4 bits, utilizamos quatro somadores de 1 bit. Os carries são propagados de cada somador para o seguinte, conectando o carry-out de um ao carry-in do próximo. Este método permite somar vetores de 4 bits, com a soma final levando em conta todos os bits de transporte intermediários.



- Um somador de 8 bits é composto por dois somadores de 4 bits. O carry-out do primeiro somador de 4 bits é conectado ao carry-in do segundo somador de 4 bits, assegurando a propagação correta do carry entre os dois segmentos do circuito.



2.9 SEQUÊNCIA 101

O detector de sequência 101 é um circuito digital utilizado para identificar a presença da sequência binária "101" em um fluxo contínuo de bits. Este tipo de circuito é essencial em sistemas de comunicação e processamento de sinais, onde a detecção de padrões específicos é necessária.

A máquina de estados do detector de sequência 101 possui quatro estados principais. Abaixo está a descrição de cada estado e suas transições baseadas na entrada de bits:

Estado	Entrada	Próximo Estado	Saída
0	0	0	0

0	1	1	0
1	0	2	0
1	1	1	0
2	0	0	0
2	1	3	1
3	0	2	0
3	1	1	0

Cada estado pode ser descrito da seguinte maneira:

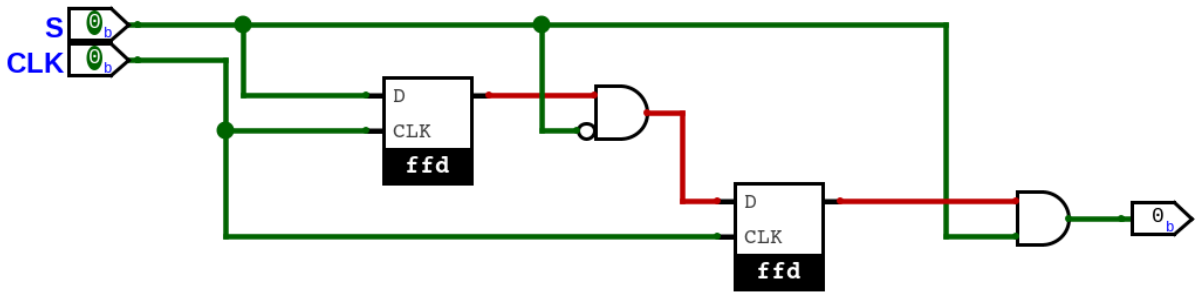
- Estado 0: Este é o estado inicial, onde o circuito aguarda o início da sequência. Quando a entrada é '1', ele transita para o Estado 1.
- Estado 1: Neste estado, o circuito detectou o primeiro '1' e agora aguarda '0' para continuar a sequência. Se a entrada for '0', ele transita para o Estado 2. Se for '1', ele permanece no Estado 1.
- Estado 2: Neste estado, o circuito detectou '10' e agora aguarda '1' para completar a sequência. Se a entrada for '1', ele transita para o Estado 3 e a saída é 1, indicando a sequência detectada. Se for '0', ele retorna ao Estado 0.
- Estado 3: Este é o estado final, indicando que a sequência "101" foi detectada. Se a entrada for '0', ele transita para o Estado 2 para continuar a detecção. Se for '1', ele transita para o Estado 1.

Com base na análise da tabela de transição de estados, podemos derivar expressões lógicas que indicam quando o circuito está em um determinado estado. Chamamos os estados 1, 2 e 3 como E1, E2 e E3, respectivamente, e a entrada como S.

As expressões lógicas são:

- $E1 = S$
- $E2 = \overline{S} \cdot E1$
- $E3 = S \cdot E2$

O circuito de detecção utiliza flip-flops para armazenar os estados da sequência. Como o estado final E3 só pode ser alcançado após passar por dois outros estados, é necessário utilizar dois flip-flops para armazenar os estados E1 e E2. Isso garante que o circuito possa transitar corretamente entre os estados e detectar a sequência "101".

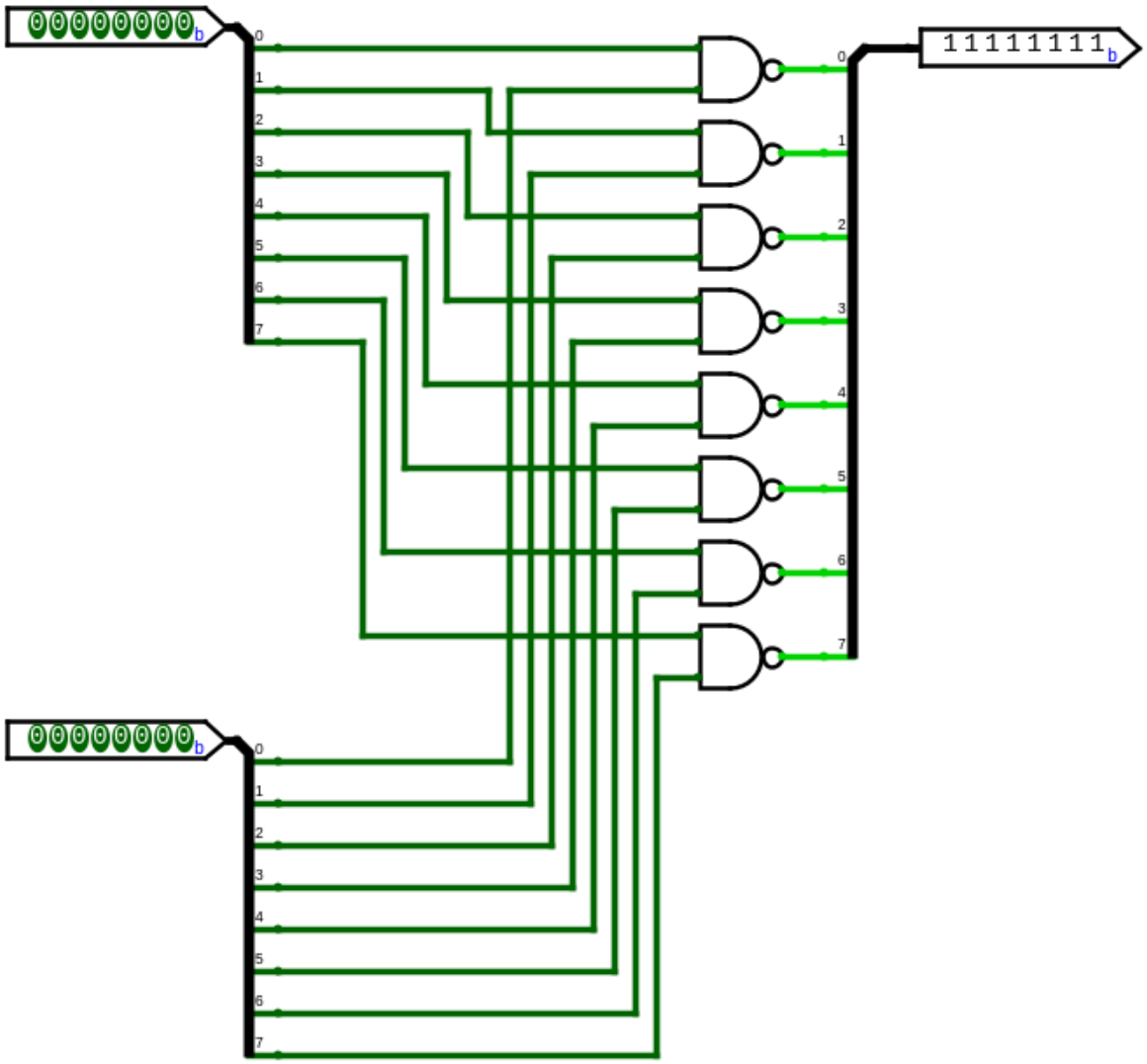


2.10 UNIDADE LÓGICA ARITMÉTICA

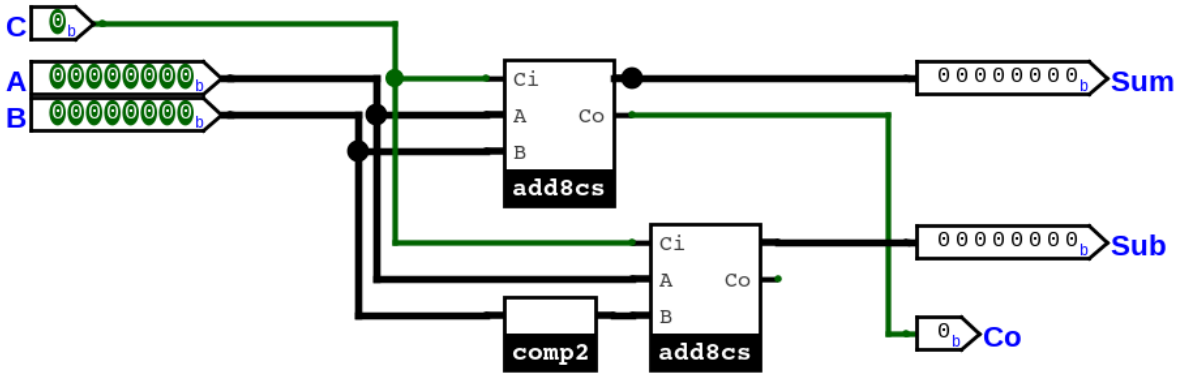
Uma Unidade Lógica e Aritmética (ULA) é um componente fundamental em processadores e sistemas de computação, responsável por realizar operações lógicas e aritméticas. O circuito de ULA implementado suporta as seguintes operações: AND, OR, NOT, NOR, NAND, XOR, SHIFT de 2 bits à esquerda, SHIFT de bits à direita, soma e subtração.

A ULA recebe dois operandos, A e B, cada um com 8 bits de largura, e uma flag Cin para o carry-in da soma. Além disso, a ULA recebe um sinal de controle de 4 bits de largura, chamado Ctrl, que determina qual operação será realizada. Com 4 bits de controle, podemos codificar até 16 operações diferentes, mas nesta implementação apenas as 10 primeiras serão utilizadas.

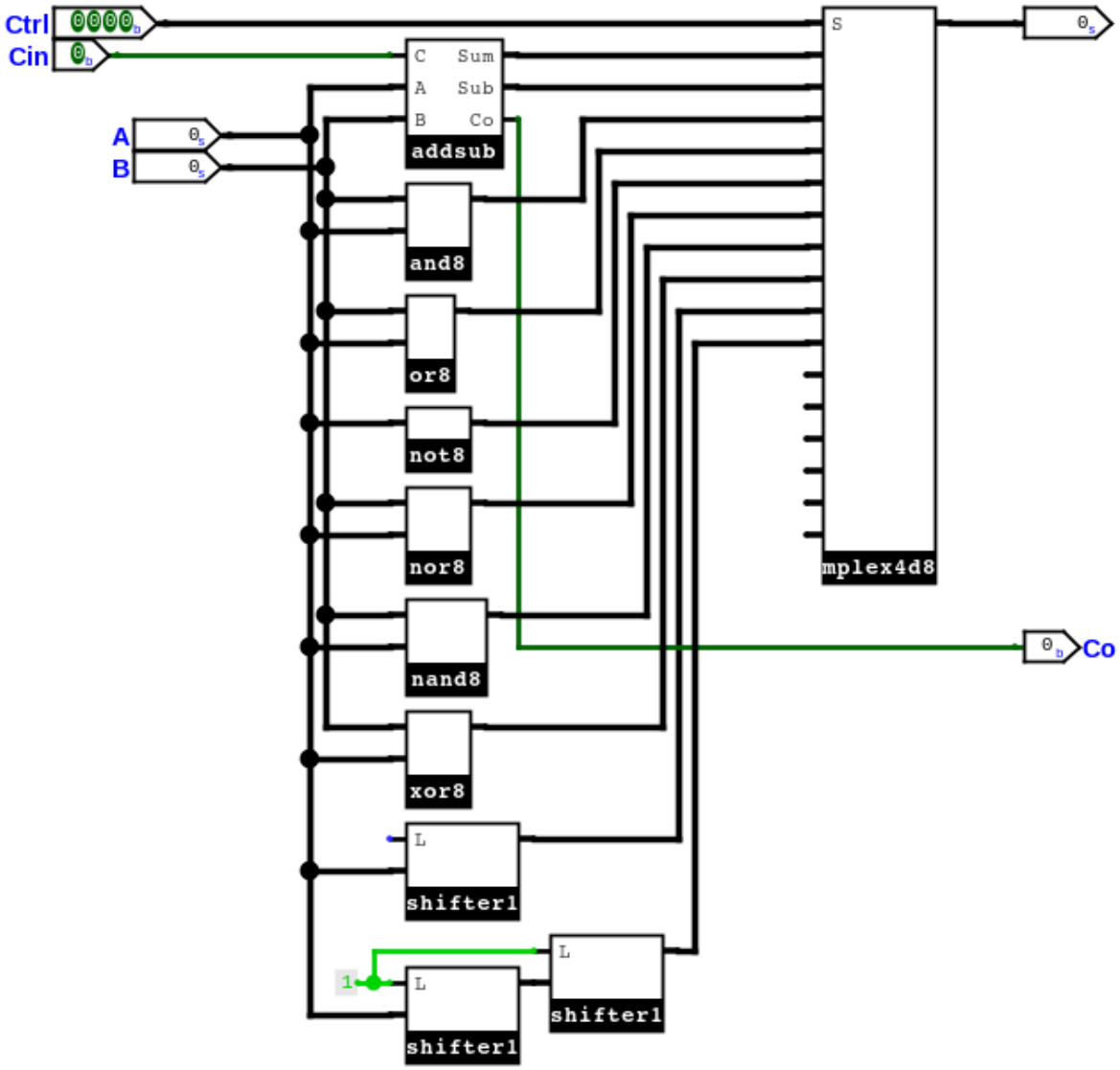
As operações AND, OR, NOT, NOR, NAND e XOR aplicam as respectivas operações lógicas entre os bits de A e B. O enésimo bit de saída é o resultado da operação lógica entre os bits A[n] e B[n]. Isso pode ser observado pela implementação da operação NAND abaixo.



Para a soma e a subtração foi criado um circuito chamado dedicado que faz as operações de soma e subtração simultaneamente em dois operandos A e B. Na parte de subtração, o circuito converte o B para a notação de complemento de 2 e faz a operação de soma entre o A e o complemento de B.



Para selecionar qual operação deve ser mostrada como resultado, utilizamos um multiplexador de 16 entradas. Embora todas as 16 entradas estejam presentes, apenas as 10 primeiras são utilizadas nesta implementação. O sinal de controle Ctrl é usado para selecionar qual das operações será a saída da ULA.



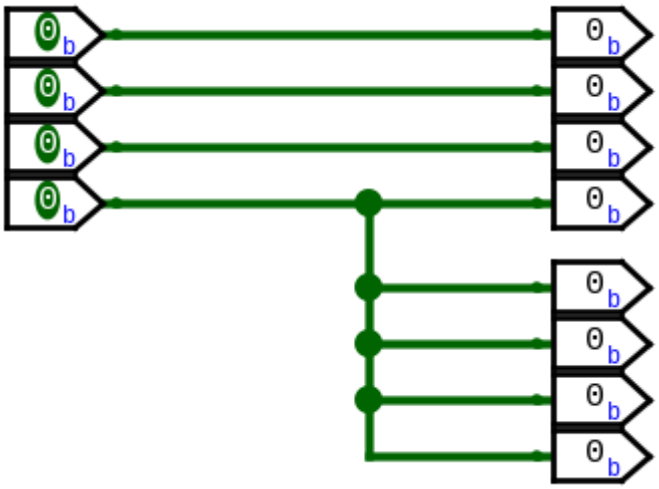
2.11 EXTENSOR DE SINAL

Um extensor de sinal é um circuito usado para aumentar a largura de um dado binário, mantendo seu valor e sinal corretos. Neste caso, vamos falar sobre um extensor de sinal que transforma um valor de 4 bits em um valor de 8 bits, usando a técnica de complemento de 2.

O complemento de 2 é uma forma de representação de números binários que permite a fácil manipulação e operação de números negativos. Nesta representação, o bit mais significativo (o bit de maior ordem) também atua como o bit de sinal. Se o bit de maior ordem é '0', o número é positivo; se for '1', o número é negativo. O extensor de sinal de 4 bits para 8 bits replica o valor do bit mais significativo (bit de

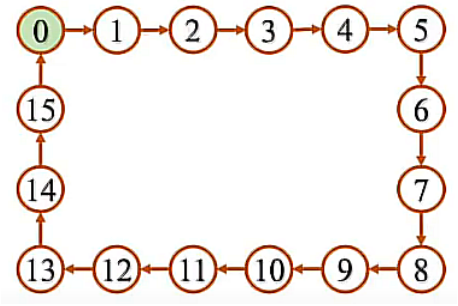
sinal) para os bits estendidos, garantindo que o número resultante preserve o mesmo valor e sinal.

Um circuito extensor de sinal consiste basicamente em fios que conectam o bit de sinal mais significativo do valor de 4 bits às novas posições de bit nas extensões de 8 bits. O circuito pode ser implementado da seguinte maneira:



2.12 CONTADOR SÍNCRONO

Contadores Síncronos é um circuito digital formado por flip-flops em paralelos, tal que todas as entradas clocks estejam conectados na mesma fonte de clock. Em um contador síncrono, todos os estados possíveis devem produzir um próximo estado específico, determinado pela sequência que o contador vai seguir. Portanto, na tabela verdade do contador, é necessário colocar esse próximo estado do lado do estado atual. Para sabermos o próximo estado do contador deve-se ser conhecido a ordem da contagem, seja ela crescente ou decrescente, para ser possível fazer o diagrama de estados, na qual mostra a sequência de números que o contador irá mostrar. Para esse circuito será adotado uma sequência de 0 a 15 com 4 bits de entrada e 4 bits de saída, quando o contador chegar no número 15 o seu proximo estado será o número 0, fazendo assim repetir toda sequência novamente, como mostra o diagrama abaixo.



Após ser definido, deve ser criado o circuito de controle, que é o circuito que ficará responsável pela sequência de números e de enviar o próximo estado aos flip flops. Nesse caso, na tabela verdade os bits de saída são um número a mais que os bits de entrada como mostrado abaixo.

Q3	Q2	Q1	Q0	D3	D2	D1	D0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

Com a tabela verdade definida, tira-se as expressões de cada saída e depois construir o circuito de controle.

D0: ~Q0

D1: ~Q1 Q0 + Q1 ~Q0

D2: ~Q2 Q1 Q0 + Q2 ~Q1 + Q2 ~Q0

D3: ~Q3 Q2 Q1 Q0 + Q3 ~Q2 + Q3 ~Q1 + Q3 ~Q0

Para esse projeto, será usado os flips-flops do tipo D, cada saída do circuito de controle (D0, D1, D2 e D3) será ligada com cada entrada D de 4 flip-flops. O clock de todos os flip-flops serão ligados por um pino de entrada de dado, esse pino definirá o momento de mudança de cada estado. Cada saída Q dos flip-flops terá um pino de saída, que mostrará o número atual do contador. Por fim todas as entradas CLEAR do flip-flops estarão ligadas para limpar a o estado atual e recomeçar a contagem.

2.13 DETECTOR DE PARIDADE ÍMPAR

O detector de paridade ímpar é um circuito digital utilizado para verificar se um número binário de 8 bits possui um número ímpar de bits 1. A verificação de paridade é amplamente utilizada em sistemas de comunicação e armazenamento para detectar erros de transmissão ou de dados.

O detector de paridade ímpar utiliza portas lógicas XOR para calcular a paridade. A operação XOR entre os bits do número resulta em 1 se o número de bits 1 for ímpar e 0 se o número de bits 1 for par.

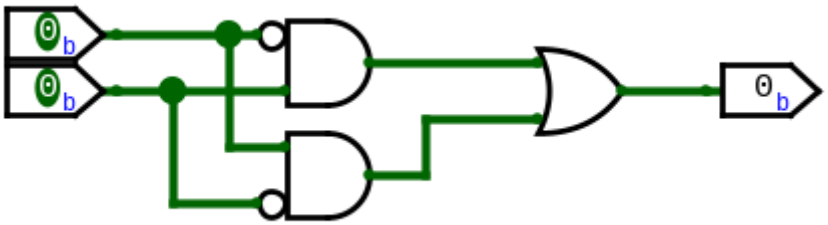
Com a análise da tabela verdade da operação XOR podemos determinar que a saída da operação XOR é 1 somente quando o número de bits 1 nas entradas A e B é ímpar

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1

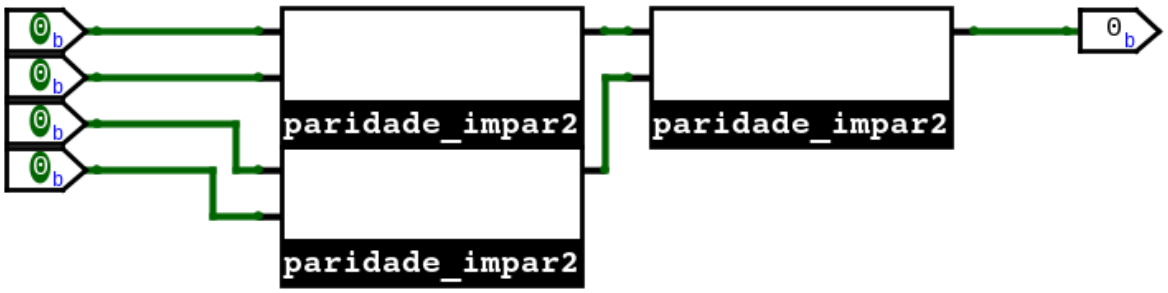
1	1	0

Podemos conectar detectores de paridade menores para criar um detector de paridade de maior tamanho porque a operação XOR é associativa e comutativa. Isso significa que a ordem em que aplicamos as operações XOR não altera o resultado. Portanto, podemos dividir o problema em partes menores, calcular a paridade de cada parte e depois combinar os resultados usando mais portas XOR.

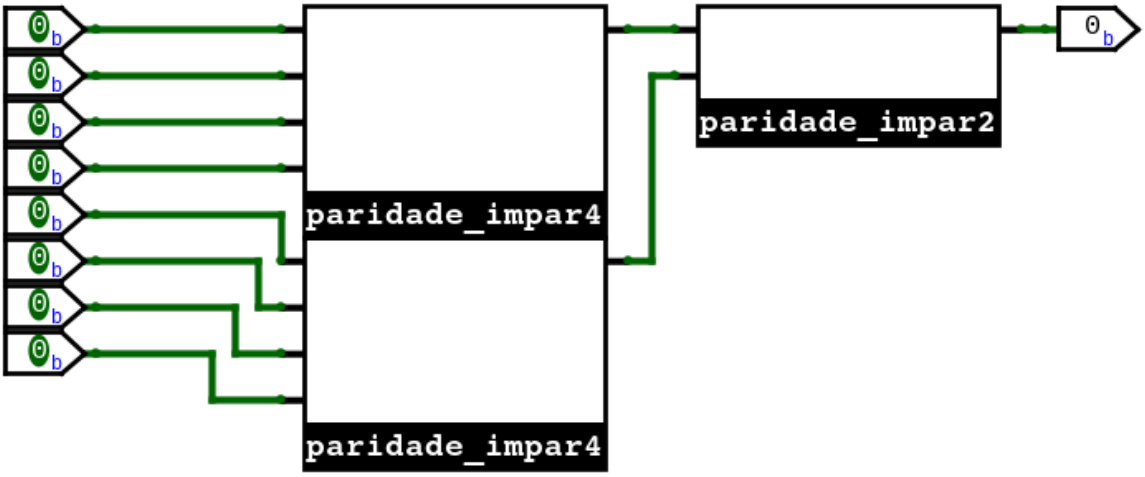
Para detectar a paridade ímpar de 2 bits, usamos uma única porta XOR. A saída será 1 se os dois bits forem diferentes e 0 se ambos forem iguais.



Para detectar a paridade de 4 bits, podemos agrupar os bits em dois pares de 2 bits. Em seguida, usamos detectores de 2 bits em cascata. A saída de cada par de 2 bits é combinada novamente por um detector de 2 bits para formar o detector de 4 bits.



Para detectar a paridade em 8 bits, combinamos dois detectores de paridade de 4 bits. As saídas dos dois detectores de 4 bits são passadas por um detector de 2 bits. A saída final indica se a quantidade de bits 1 no número de 8 bits é ímpar.



2.14 DECODIFICADOR DE 7 SEGMENTOS

Um decodificador é um CI (circuito interno) que recebe um código binário e o decodifca, através dos pinos de entrada, para um código hexadecimal. No caso do display de 7 segmentos, pode-se apenas decodificar números de 4 bits na entrada e a saída até o número 9 (1001 em binário). Para começar a construção do circuito deve-se saber a quantidade de saídas, para isso, se observa o data sheet do display, que é o mapa de pinos de entrada do dispositivo com o seus nomes.

O data sheet é mostrado que o display tem 7 entradas nomeadas com as letras de A até G, a partir daí se constrói a tabela verdade e tiramos a expressão de cada saída do circuito.

A1	A2	A3	A4	A	B	C	D	E	F	G	Display
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9
1	0	1	0	0	0	0	0	0	0	0	
1	0	1	1	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	
1	1	0	1	0	0	0	0	0	0	0	

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

Como o display só enumera até o número 9, a partir desse número, as saídas devem estar no estado lógico baixo.

Utilizado o mapa de Karnaugh, tira-se a expressão das saídas

A: ~A1 ~A2 ~A4 + ~A1 A3 + ~A1 A2 A4 + A1 ~A2 ~A3
B:~A1 ~A2 + ~A1 ~A3 ~A4 + ~A2 ~A3 + ~A1 A3 A4
C:~A2 ~A3 + ~A1 A4 + ~A1 A2
D:~A1 ~A2 ~A4 + ~A1 ~A2 A3 + ~A1 A3 ~A4 + ~A1 A2 ~A3 A4 + A1 ~A2 ~A3
E:~A2 ~A3 ~A4 + ~A1 A3 ~A4
F: ~A1 ~A3 ~A4 + ~A1 A2 ~A3 + ~A1 A2 ~A4 + A1 ~A2 ~A3
G: ~A1 ~A2 A3 + ~A1 A2 ~A3 + ~A1 A2 ~A4 + A1 ~A2 ~A3

Com a tabela verdade e as expressões, é o momento de montar o circuito no logisim e fazer as lições com a entrada do display.

2.15 DETECTOR DE NÚMEROS PRIMOS

Um circuito detector de números primos é projetado para verificar se um número binário de 4 bits é primo ou não, utilizando uma tabela verdade. Este circuito realiza verificações lógicas simples para determinar se o número é divisível apenas por 1 e por ele mesmo. No caso de números de 4 bits, os valores vão de 0 a 15, então os números primos entre detectados são 2, 3, 5, 7, 11 e 13.

A tabela verdade é usada para determinar se um número binário de 4 bits é primo. O circuito verifica a divisibilidade com base nas condições que definem os números primos. A tabela identifica uma saída 1 para números primos e 0 para não primos.

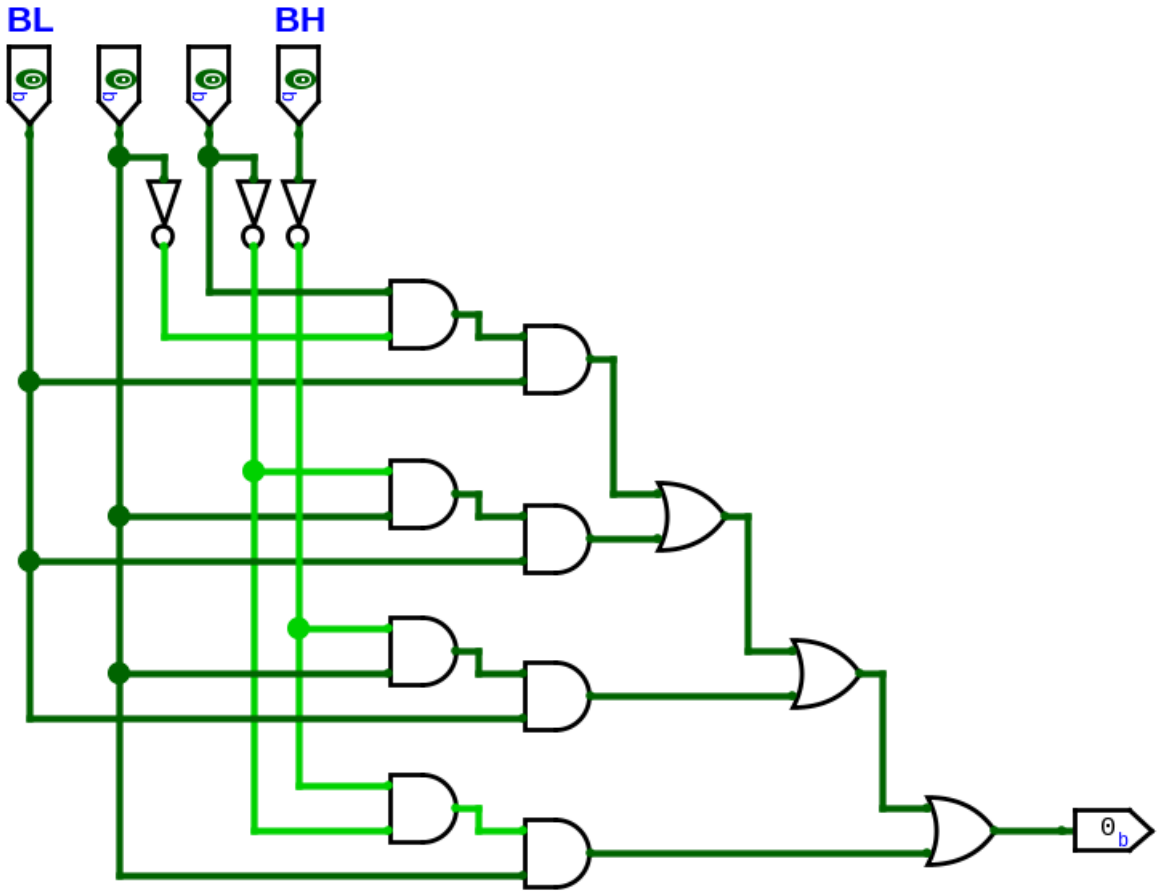
Entrada (4 bits)	Número Decimal	Primo (Saída)
0000	0	0
0001	1	0
0010	2	1
0011	3	1
0100	4	0
0101	5	1

0110	6	0
0111	7	1
1000	8	0
1001	9	0
1010	10	0
1011	11	1
1100	12	0
1101	13	1
1110	14	0
1111	15	0

A partir da tabela verdade, podemos derivar uma expressão lógica que identifica os números primos. Utilizamos cada bit de entrada do número como uma variável e aplicamos a técnica do Mapa de Karnaugh. Assim obtemos a seguinte expressão lógica:

$$\text{Saída} = (B \cdot \bar{C} \cdot D) + (\bar{B} \cdot C \cdot D) + (\bar{A} \cdot C \cdot D) + (\bar{A} \cdot \bar{B} \cdot C)$$

Dessa forma, a implementação do circuito digital baseado utiliza apenas portas lógicas simples, AND, NOT e OR. O circuito recebe quatro entradas, os bits do número a ser testado, marcadas com BL sendo o menos significativo e BH sendo o mais significativo. A saída consiste num bit, sendo 1 quando o número de entrada for primo e 0 quando o número não for primo.



3 CONCLUSÃO

Graças ao apoio da literatura e do programa Logisim, foi possível chegar aos resultados esperados no estudo, construção e descrição dos circuitos de maneira direta porém detalhada, bem como os seus códigos-fonte postado no repositório. O trabalho possibilita a aprendizagem de certos componentes básicos e fundamentais de circuitos digitais.

4 REFERÊNCIAS

UNIVERSIDADE FEDERAL DE RORAIMA. **Normas para Apresentação dos Trabalhos Técnico Científicos da UFRR**. 3ª Edição. Boa Vista-RR: UFRR, 2017. 103p. Disponível em: <https://drive.google.com/file/d/1iIIQLun0kmHC_pD5G_6PfPm8AGbk0H0U/view >. Acesso em: 10/12/2024.

WIDMER, Neal S. MOSS, Gregory L. TOCCI, Ronald J. **Digital Systems**: Principal and Applications. 12ª Edição. Estados Unidos. Pearson, 2018. 1025 p.