UNIVERSIDADE FEDERAL DE RORAIMA

CENTRO DE CIÊNCIA E TECNOLOGIA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

DCC301– ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES– 2024

PROF. DR. HEBERT OLIVEIRA ROCHA

LEONARDO VINÍCIUS LIMA CASTRO

ÁLEFE ALVES DA COSTA

LABORATÓRIO DE CIRCUITOS - CODIFICAÇÃO E SIMULAÇÕES

BOA VISTA, RR

2024

LEONARDO VINÍCIUS LIMA CASTRO

ÁLEFE ALVES DA COSTA

LABORATÓRIO DE CIRCUITOS - CODIFICAÇÃO E SIMULAÇÕES

Trabalho da disciplina de Arquitetura e Organização de Computadores do ano de 2024 apresentado à Universidade Federal de Roraima do curso de Bacharelado em ciência da computação.

Docente: Prof. Dr. Hebert O. Rocha

BOA VISTA, RR

2024

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1.INTRODUÇÃO

Este relatório técnico apresenta a construção de 17 componentes previamente selecionados, onde cada componente foi construído usando as portas lógicas, constantes, distribuidores e pinos de entrada e saída, toda essa implementação foi realizada no software Logisim versão 2.7.1, simulador de circuitos digitais amplamente utilizado em projetos acadêmicos e profissionais. O objetivo principal desse trabalho é a aplicação do conhecimento de circuito e documentação dos processos de descrição de cada componente, lógica aplicada e testes realizados para garantir sua funcionalidade.

2.COMPONENTES

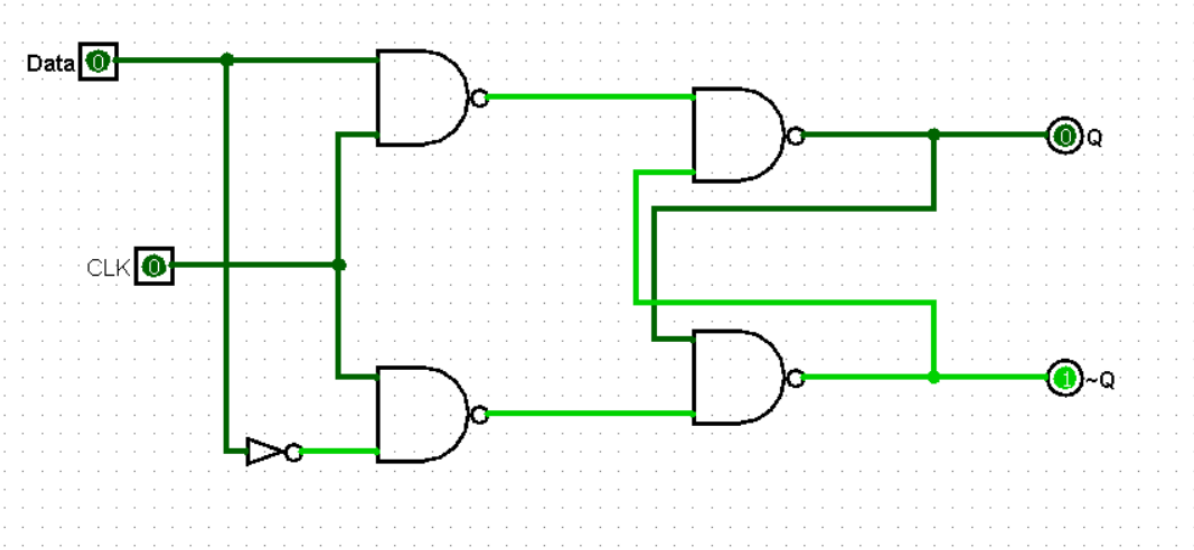
2.1. Registrador Flip-Flop do tipo D e do tipo JK

Flip-Flops são circuitos digitais usados para armazenar 1 bit de informação. Eles são componentes fundamentais na construção de memórias e dispositivos sequenciais, como contadores e registradores.

2.1.1 Registrador Flip-Flop do tipo D

O Flip-flop D é um circuito utilizado em geral para armazenar um único bit, como se fosse um registrador de 1 bit, ele é normalmente utilizado em circuitos sequenciais e dispositivos de armazenamento como contadores, registradores e memórias. Este circuito funciona da seguinte forma, ele depende do clock, quando ocorre a borda de subida, o valor que tiver na entrada D, vai para a saída Q.

Figura 1 - Flip-Flop D



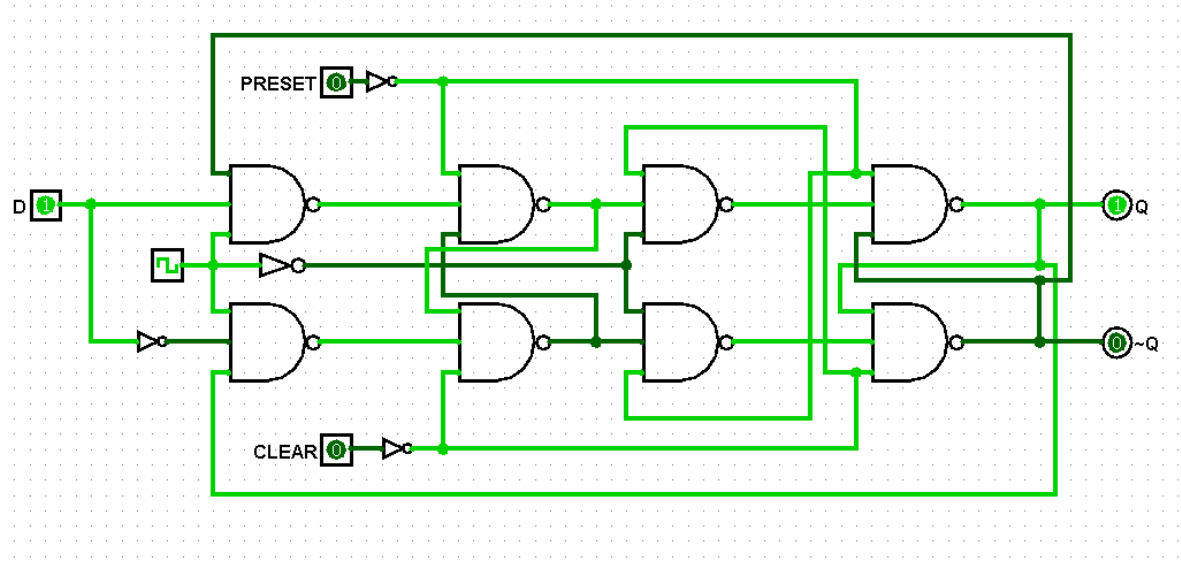
O circuito acima representa um Flip-Flop do tipo D, que é composto pelas portas lógicas NAND e NOT, possui duas entradas uma de clock(CLK) e uma de dado(D), e possui duas saídas Q e ~Q, onde Q representado o estado atual armazenado no Flip-Flop e ~Q o inverso.

Tabela 1 – Tabela verdade do Flip-Flop tipo D

D	CLK	Q	~Q
0	1	0	1
1	1	1	0

Nos testes feitos no Logisim o resultado foi essa tabela, onde há uma mudança de estado quando D recebe um dado e CLK recebe uma entrada alta, fazendo assim Q receber o estado atual do Flip-Flop, assim cumprindo o papel do circuito.

Figura 2 - Flip-Flop D com PRESET e CLEAR

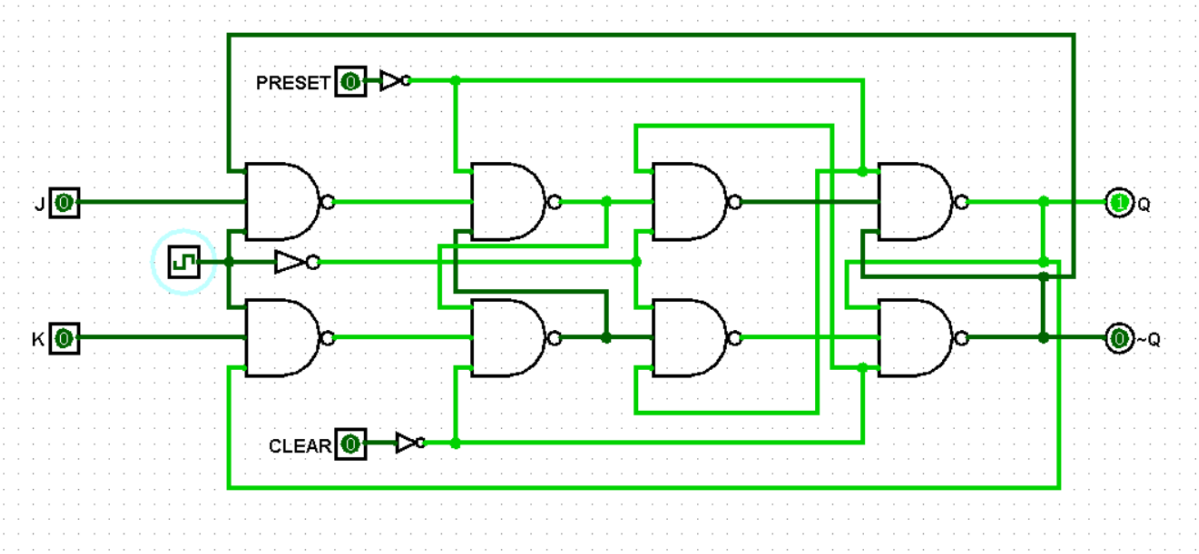


Os Flip-Flop D apresentado anteriormente não possui o PRESET e CLEAR onde na construção de alguns componentes acaba sendo útil, então há essas duas versões deste Flip-Flop. Onde o PRESET e CLEAR são entradas de controle usadas para definir diretamente o estado do flip-flop D. PRESET força a saída Q = 1, enquanto CLEAR força Q = 0, independentemente das entradas D e CLOCK. Eles são úteis para inicializar ou redefinir circuitos digitais rapidamente.

2.1.2 Registrador Flip-Flop do tipo JK

O Flip-Flop JK é uma extensão do Flip-Flop D que pode ser usado para diversas aplicações, como contadores e divisores de frequência, devido à sua versatilidade. Diferentemente do Flip-Flop D, o JK tem duas entradas, chamadas J e K, que permitem diferentes operações.

Figura 3 - Flip-Flop JK



O circuito acima representa um Flip-Flop do tipo JK, composto por portas lógicas NAND e NOT. Ele possui as entradas J, K, CLOCK (CLK), CLEAR e PRESET, e duas saídas, Q e ~Q. A saída Q representa o estado atual armazenado no Flip-Flop, enquanto ~Q é seu inverso. O comportamento do Flip-Flop JK é controlado pelas combinações das entradas J e K em relação ao sinal do CLOCK, além dos sinais CLEAR e PRESET, que permitem forçar a saída para 0 ou 1, respectivamente.

Tabela 2 - Tabela verdade Flip-Flop JK

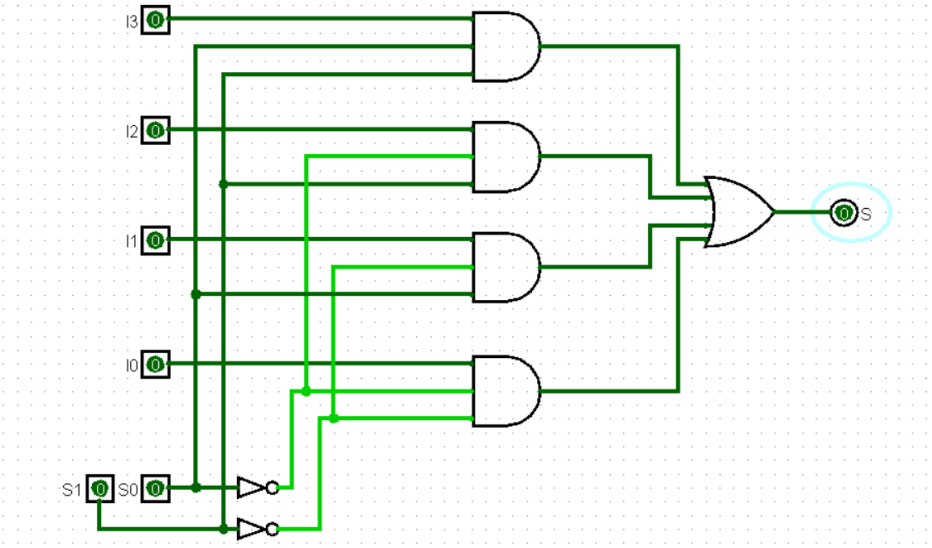
J	K	CLK	CLEAR	PRESET	Q	~Q	Descrição
x	x	x	0	1	1	0	Preset força Q = 1
x	x	x	1	0	0	1	Clear força Q = 0
0	0	1	0	1	1	0	Q = 1
0	0	1	1	0	0	1	Q = 0
1	0	1	0	0	1	0	Q = 1
0	1	1	0	0	0	1	Q = 0
1	1	1	0	0	1	0	Q = 1

Nos testes feitos no Logisim o resultado foi essa tabela, onde há uma mudança de estado quando JK recebe um dado no J, no K ou nos dois e CLK recebe uma entrada alta, fazendo assim Q = 1, quando o PRESET for ativado, pois ele força esse resultado, quando J =1 e K = 0, ou J = 1 e K = 1, o caso Q = 0, ocorre quando o CLEAR = 1, pois ele força esse resultado e quando J = 0 e K = 1. O CLEAR e PRESET independe do clock, já as entradas J e K são dependentes.

2.2. Multiplexador de 4 entradas

O multiplexador é um dispositivo digital que seleciona uma entre várias linhas de entrada e a direciona para uma única linha de saída. Ele é controlado por sinais chamados de linhas de seleção. No caso do multiplexador de 4 entradas, ele seleciona uma entre as 4 entradas e tem 2 linhas de seleção, a quantidade de linhas de seleção é dada pela fórmula $\mathrm{m} = \log_2(n)$, onde n é número de entradas.

Figura 4 - Multiplexador de 4 entradas



O circuito acima representa um Multiplexador de 4 entradas, que é composto pelas portas lógicas AND, OR e NOT, possui 6 entradas, duas de seleção S0 e S1, 4 entradas de dados para seleção sendo elas I0, I1, I2 e I3 e apenas uma saída S.

Tabela 3 - Tabela verdade do Multiplexador de 4 entradas

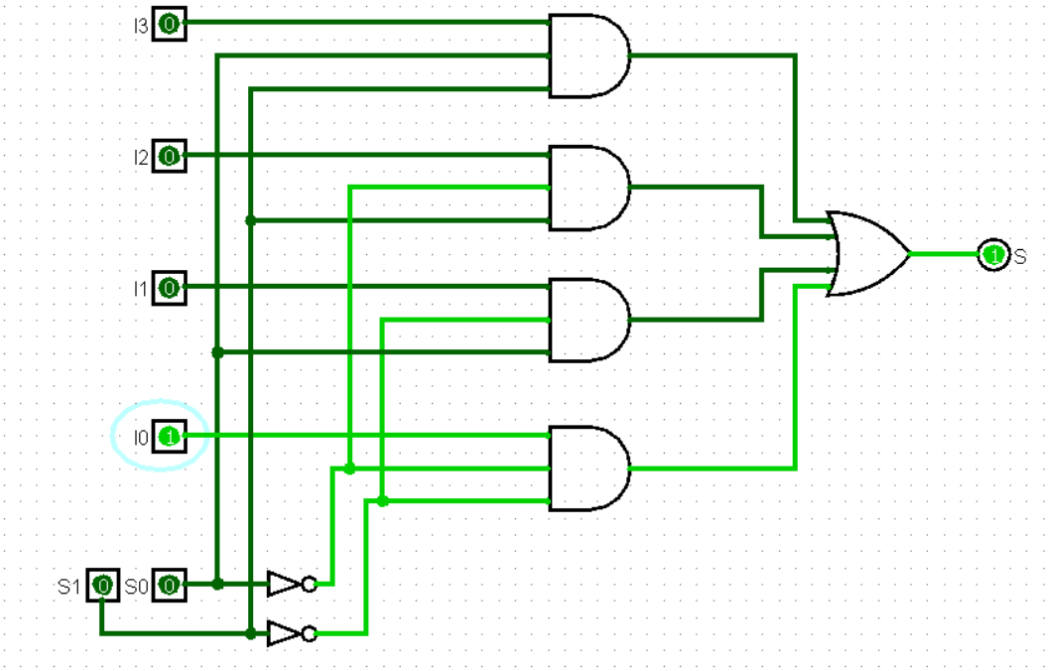
I0	I1	I2	I3	S0	S1	S
1	0	0	0	0	0	1
0	1	0	0	1	0	1
0	0	1	0	0	1	1
0	0	0	1	1	1	1

Nos testes feitos foi encontrado 4 casos como apresentado na tabela:

2.2.1. Caso 00

Neste caso, o caminho do AND que recebe a entrada I0 tem duas negações um para cada linha do seletor, fazendo assim ter todas entradas altas no AND e S = 1, quando I0 = 1.

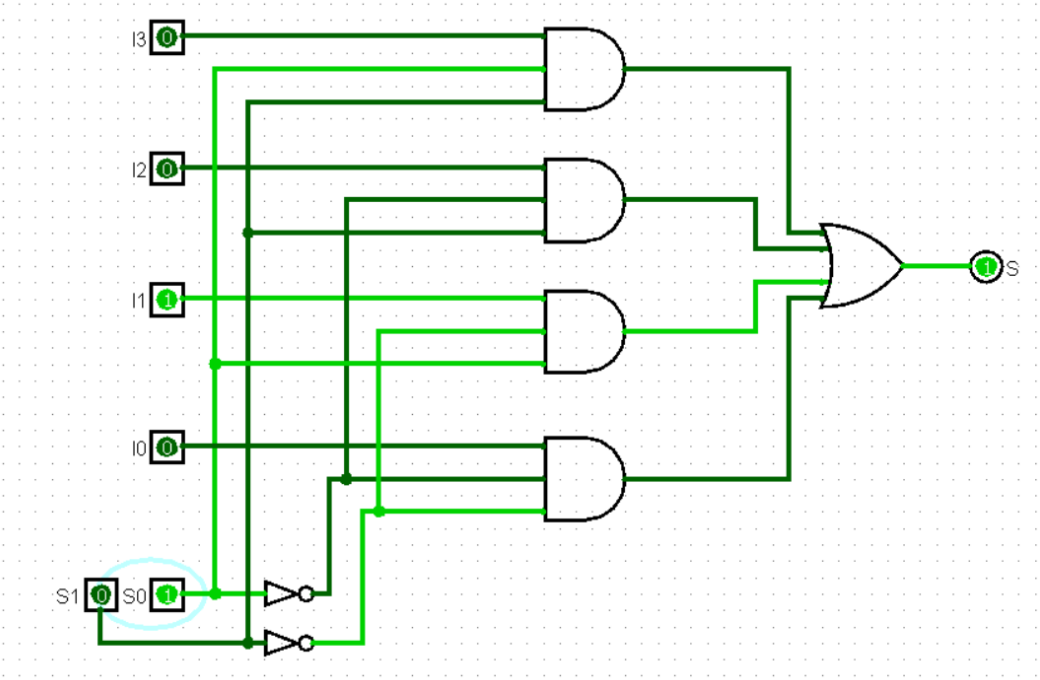
Figura 5 - Multiplexador de 4 entradas caso 00



2.2.2. Caso 01

Neste caso, o caminho do AND que recebe a entrada I1 tem uma negação na linha do seletor S1 e S0 = 1, fazendo assim ter todas entradas altas no AND e S = 1, quando I1 = 1.

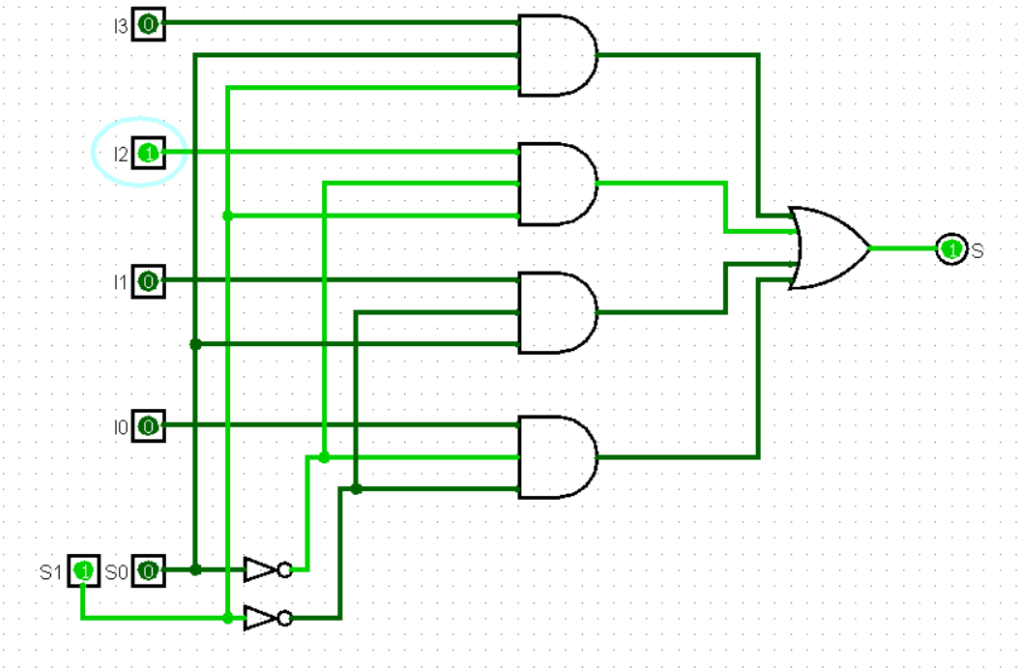
Figura 6 - Multiplexador de 4 entradas caso 01



2.2.3. Caso 10

Neste caso, o caminho do AND que recebe a entrada I2 tem uma negação na linha do seletor S0 e S1 = 1, fazendo assim ter todas entradas altas no AND e S = 1, quando I2 = 1.

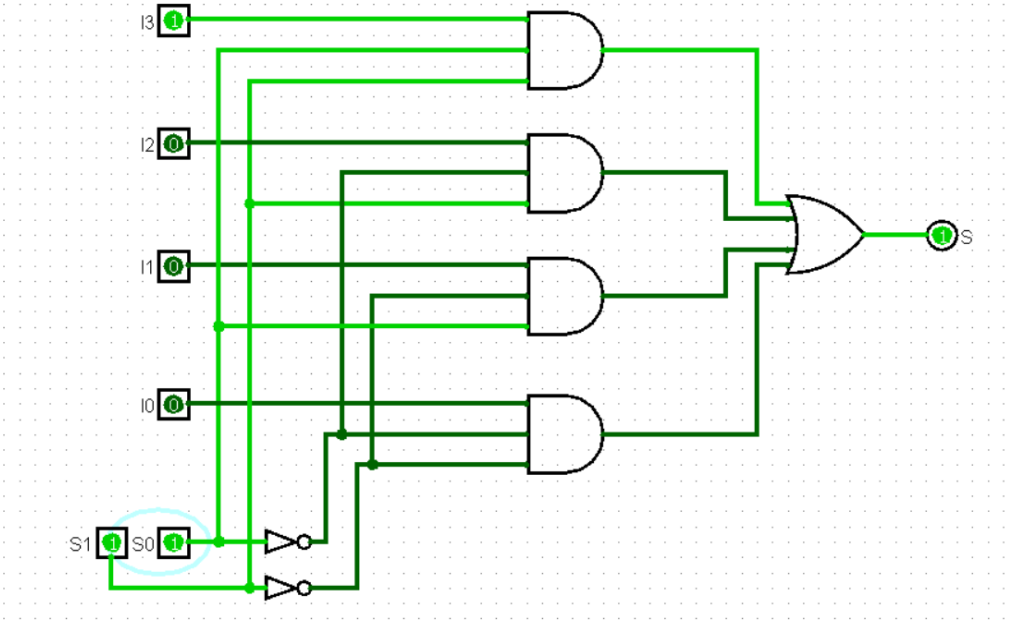
Figura 7 - Multiplexador de 4 entradas caso 10



2.2.4. Caso 11

Neste caso, o caminho do AND que recebe a entrada I3 tem as duas entradas de seleção altas, sendo S1 = 1 e S0 = 1, fazendo assim ter todas entradas altas no AND e S = 1, quando I3 = 1.

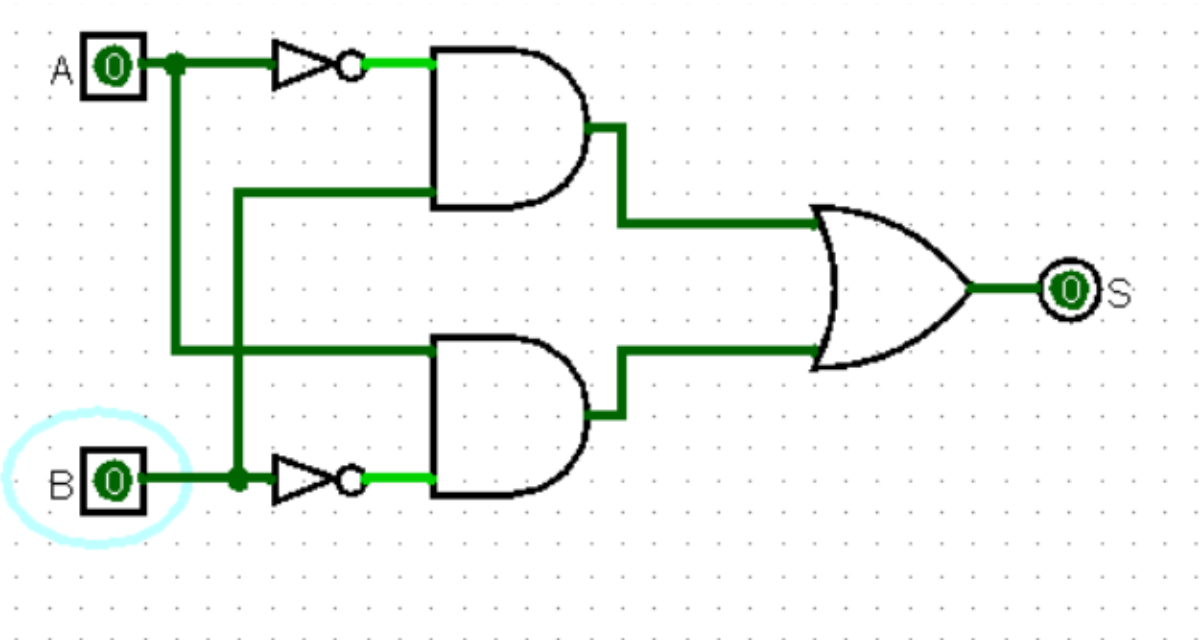
Figura 8 - Multiplexador de 4 entradas caso 11



2.3. Porta lógica XOR

A porta lógica XOR é uma porta digital que realiza a operação lógica de disjunção exclusiva. Ela tem a propriedade de retornar um valor 1 apenas quando suas entradas são diferentes, e 0 quando as entradas são iguais.

Figura 9 - Porta lógica XOR



O circuito acima representa a porta lógica XOR, que é composto pelas portas lógicas AND, OR e NOT, possui 2 entradas e apenas uma saída S.

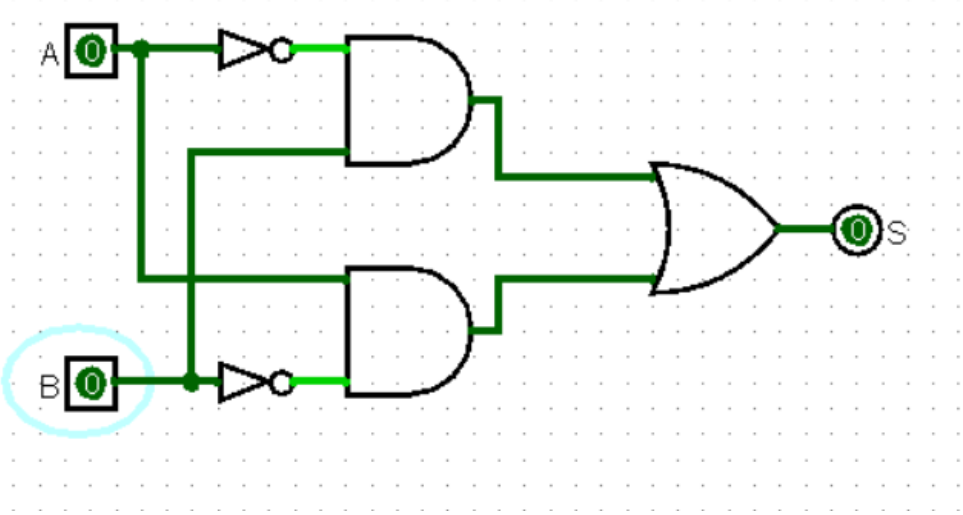
Tabela 4 - Tabela verdade do XOR

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

2.3.1. Caso 00

Neste caso as duas entradas são iguais a 0, logo não satisfaz a condição de as duas entradas serem diferentes para S ser igual a 1.

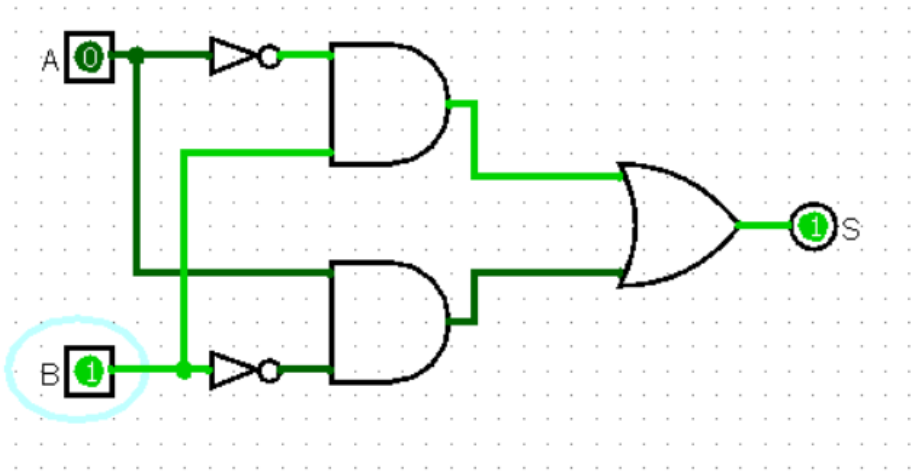
Figura 10 - Porta lógica XOR caso 00



2.3.2. Caso 01

Neste caso A = 0 e B = 1, sendo as duas entradas diferentes satisfaz a condição e S = 1.

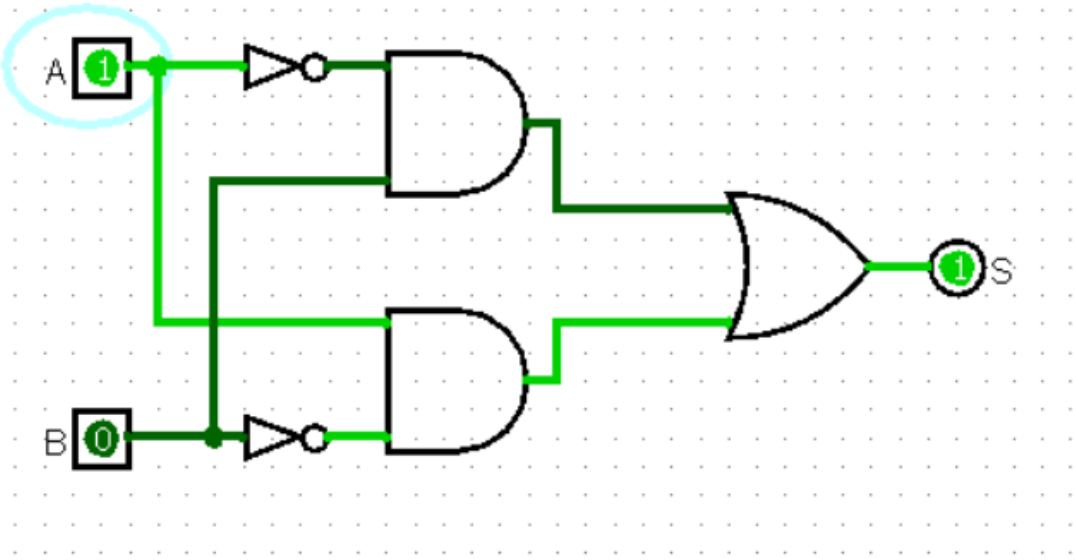
Figura 11 - Porta lógica XOR caso 01



2.3.3. Caso 10

Neste caso A = 0 e B = 1, sendo as duas entradas diferentes satisfaz a condição e S = 1.

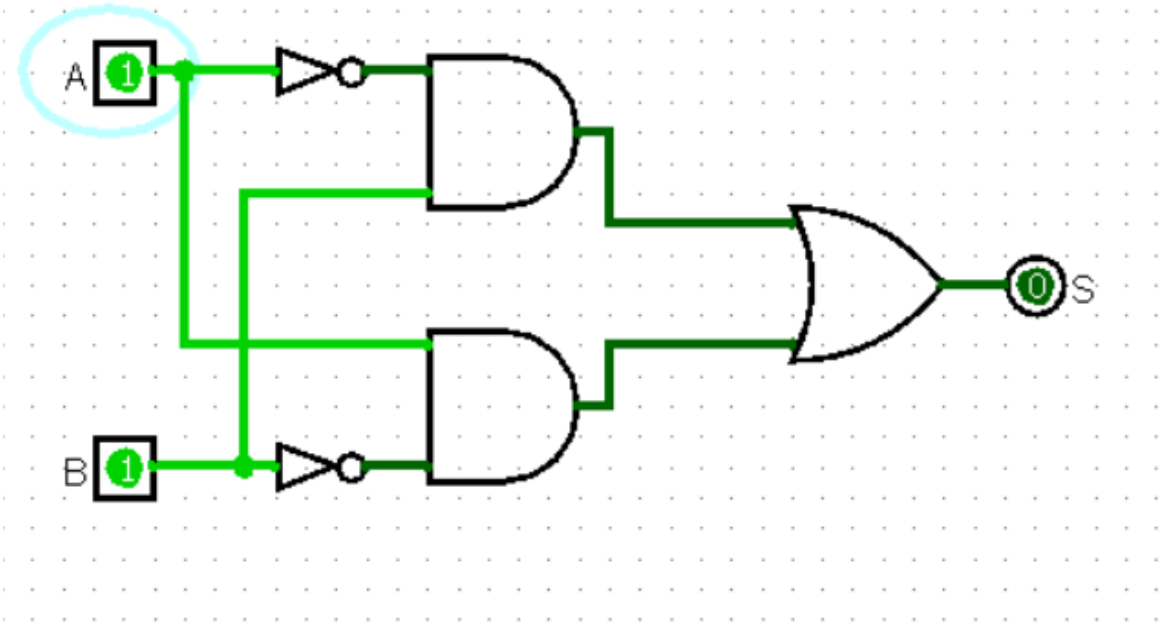
Figura 12 - Porta lógica XOR caso 10



2.3.4. Caso 11

Neste caso as duas entradas são iguais a 1, logo não satisfaz a condição de as duas entradas serem diferentes para S ser igual a 1.

Figura 13 - Porta lógica XOR caso 11



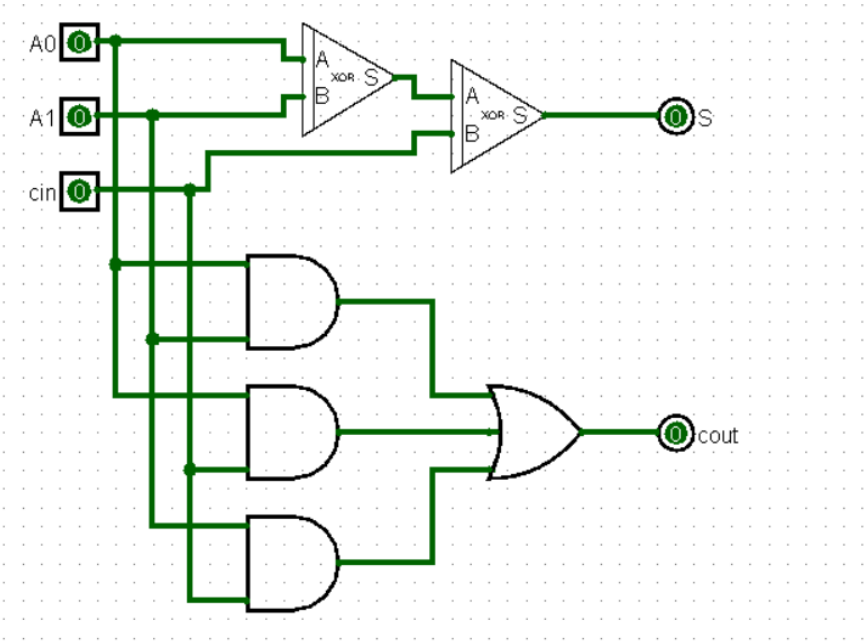
A XOR é essencial no projeto de Flip-Flops e dispositivos sequenciais. Por exemplo, ela é usada em Flip-Flops T (Toggle) e em circuitos como detectores de paridade.

2.4. Somador de 8 bits mais 4

2.4.1 Somador de 1 bit

Um somador de 1 bit é um circuito digital que realiza a soma de dois bits (A0 e A1) e, opcionalmente, considera um bit de transporte de entrada (Cin), gerando como resultado à soma(S). Podendo haver a saída (Cout) indicando se houve um "vai-um" na operação.

Figura 14 - Somador de 1 bit

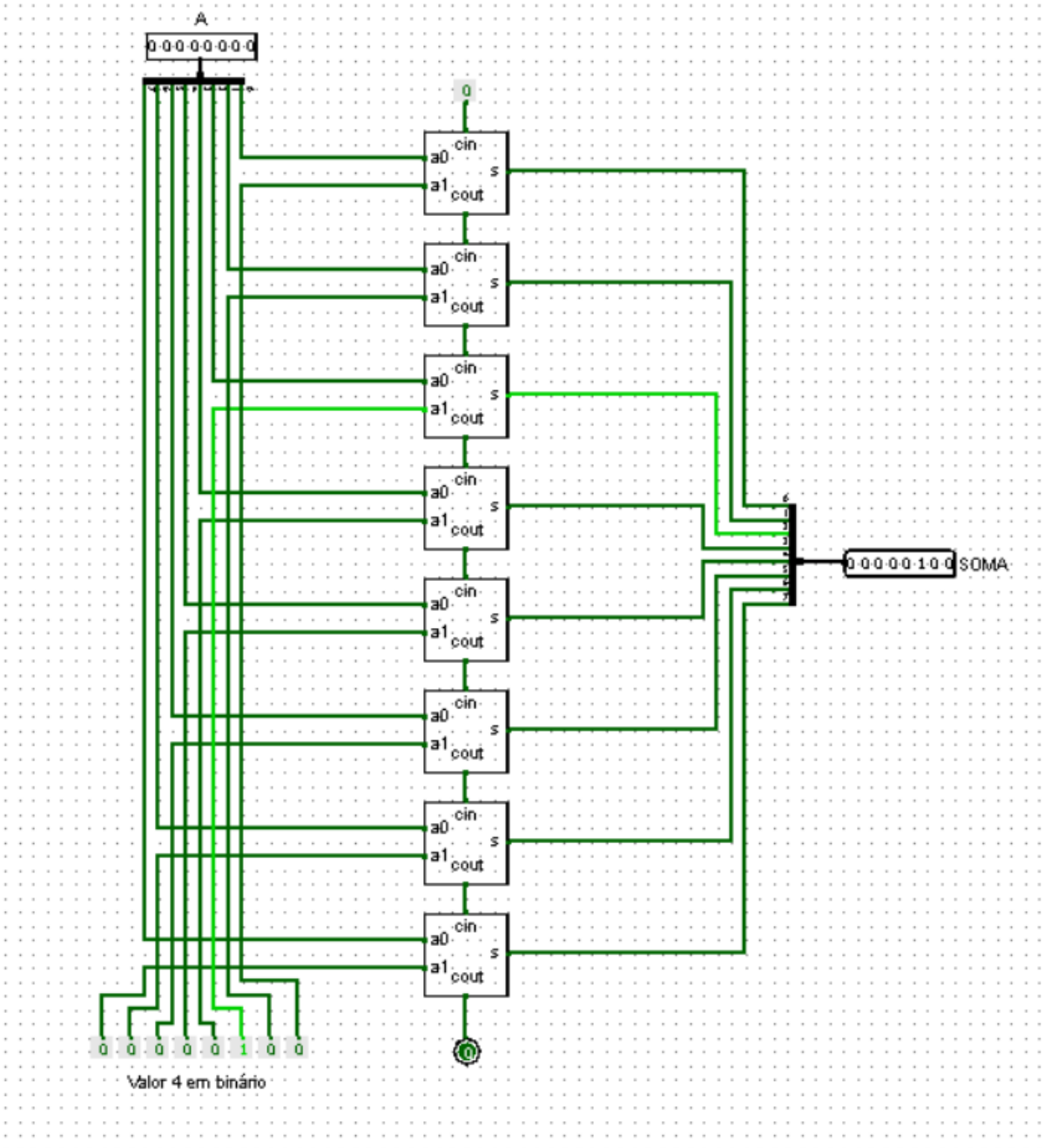


O circuito acima representa somador de 1 bit, que é composto pelas portas lógicas AND, OR e XOR, possui 3 entradas (A0, A1 e Cin) e duas saídas S e Cin. Onde quando A0 = 1 e A1 = 0 ou A0 = 0 e A1 = 1, S = 1 e Cout = 0, igualmente na soma de binário, e quando A0 = 1 E A1 = 1, S = 0 e Cout = 1, pois “estorou” a arquitetura de 1 bit, esse Cout pode ir para outro somador assim podendo haver a soma com mais bits, como 8 bits onde será apresentado no próximo circuito. Com isso o somador de 1 bits pode receber 1 bit pelo Cin, caso Cin = 1, A0 = 1, A1 = 1, as saídas de S = 1 e Cout = 1, assim fechando todos os casos de soma de 1 bit, que aprendemos em soma de binário.

2.4.2 Somador de 8 bits mais 4

Um somador de 8 bits é um circuito digital que realiza a soma de dois números binários de 8 bits (A e B) e, opcionalmente, considera um bit de transporte de entrada (Cin), gerando como resultado número binário de 8 bits representado pela saída SOMA e em caso de “estouro” de bits vai 1 para o Cout ocorrendo um overflow. Mas nesse caso o valor de B é constante sendo sempre 4, assim os 8 somadores de bit em paralelo recebem 0 menos o terceiro pois ele fica na posição que resulta no valor 4, como pode ser visto na imagem a abaixo.

Figura 15 - Somador de 8 bits mais 4

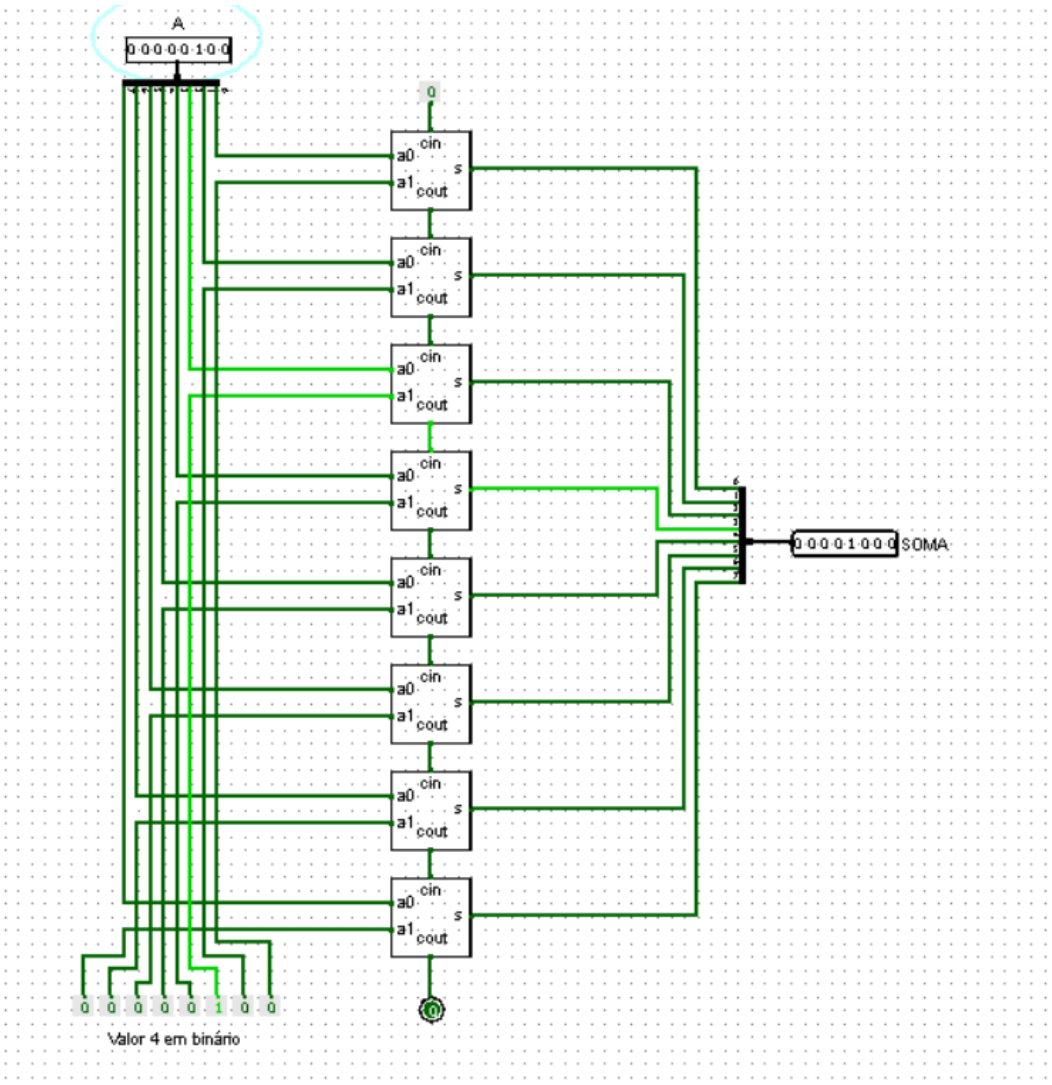


O circuito acima representa um somador de 8 bits mais 4, composto por 8 somadores de 1 bit, 2 distribuidores, 8 constantes sendo apenas uma igual a 1, 1 entrada de 8 bits(A) e uma saída de 8 bits (SOMA). O circuito funciona com cada somador de 1 bit recebendo os dados referente a sua posição na soma, onde ele tá preparado para receber um dado Cin e enviar um dado no Cout, além das duas entradas padrões, fazendo assim a clássica soma vai um e vem um, mas nesse caso, apenas a entrada A soma com 4 pois o valor de B tá predefinido em 4 em binário 00000100, logo abaixo vai ser exemplificado dois casos, um de soma comum e um de soma com resultado com overflow. A soma ocorre corretamente pois eles estão em paralelo, fazendo os dados serem distribuídos corretamente.

2.4.3 Caso de soma comum

A entrada A recebe o valor 00000100, 4 em binário, e a entrada B já está predefinida em 00000100, 4 em binário, resultando assim em 00001000, 8 em binário, como pode ser visto na imagem abaixo, demonstrando que a soma está correndo corretamente.

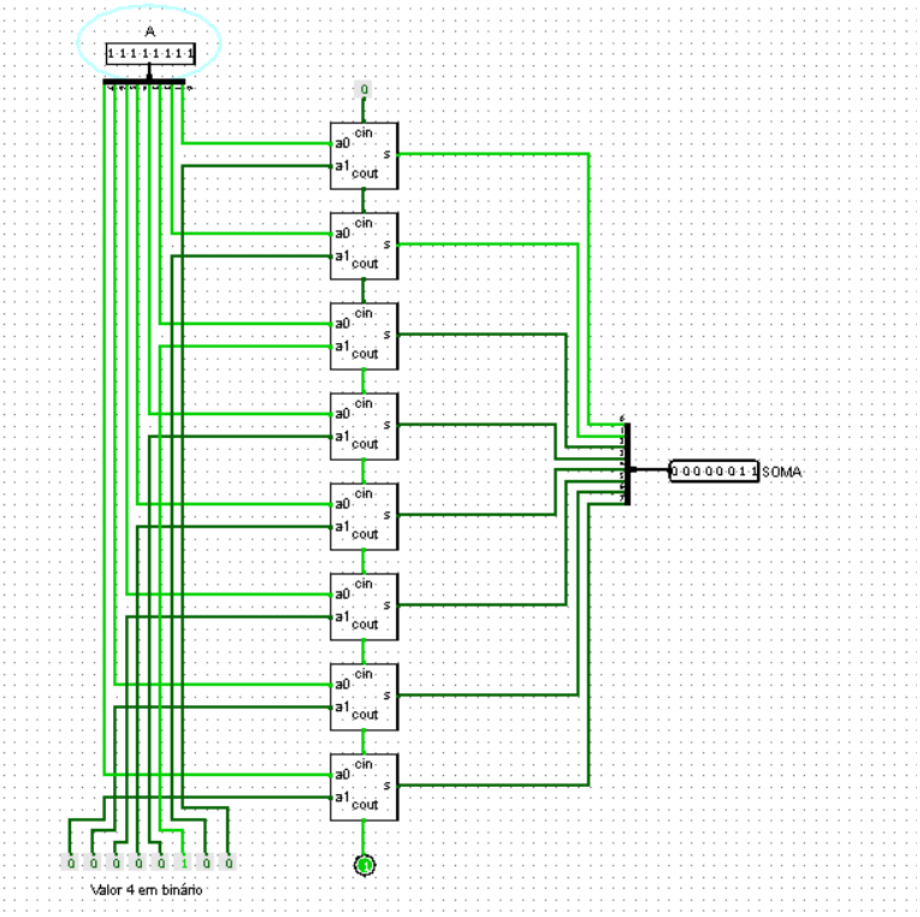
Figura 16 - Somador de 8 bits mais 4 caso comum



2.4.4 Caso de soma com overflow

A entrada A recebe o valor 11111111, 255 em binário, e a entrada B já está predefinida em 00000100, 4 em binário, resultando assim em 100000011, 259 em binário, como pode ser visto na imagem abaixo 259 em binário precisa de pelo menos 9 bits, então a saída em SOMA = 00000011 o overflow aparece em Cout = 1, comprovando que seguindo essa lógica de colocar os somadores de 1 em paralelo pode expandir o somador de 8 bits para um somador de mais de 8 bits.

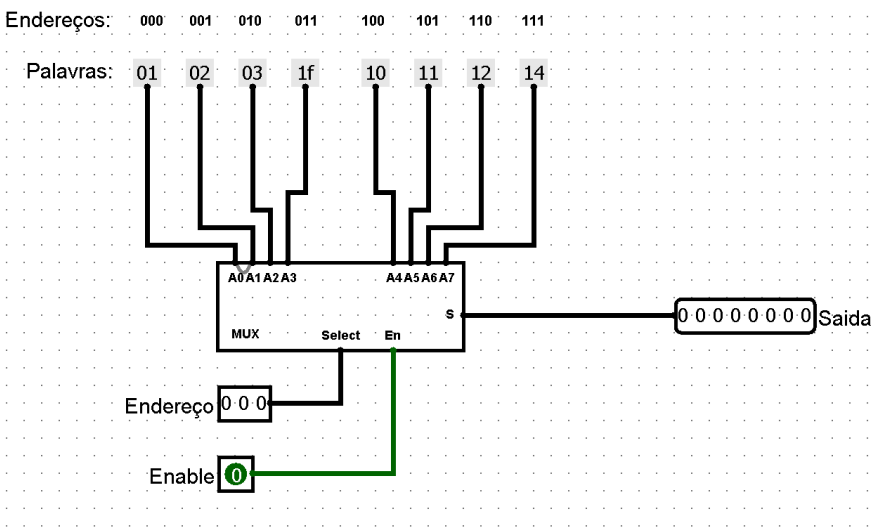
Figura 17 - Somador de 8 bits mais 4 caso estouro



2.5. Memória ROM de 8 bits

A memória apenas de leitura, mais conhecida como memória ROM, é uma memória normalmente projetada para fazer leitura de dados permanentes ou que raramente são alterados. Outras características desta que podem ser citadas são que esta é uma memória semicondutora e não-volátil.

Figura 18 - Memória ROM de 8 bits

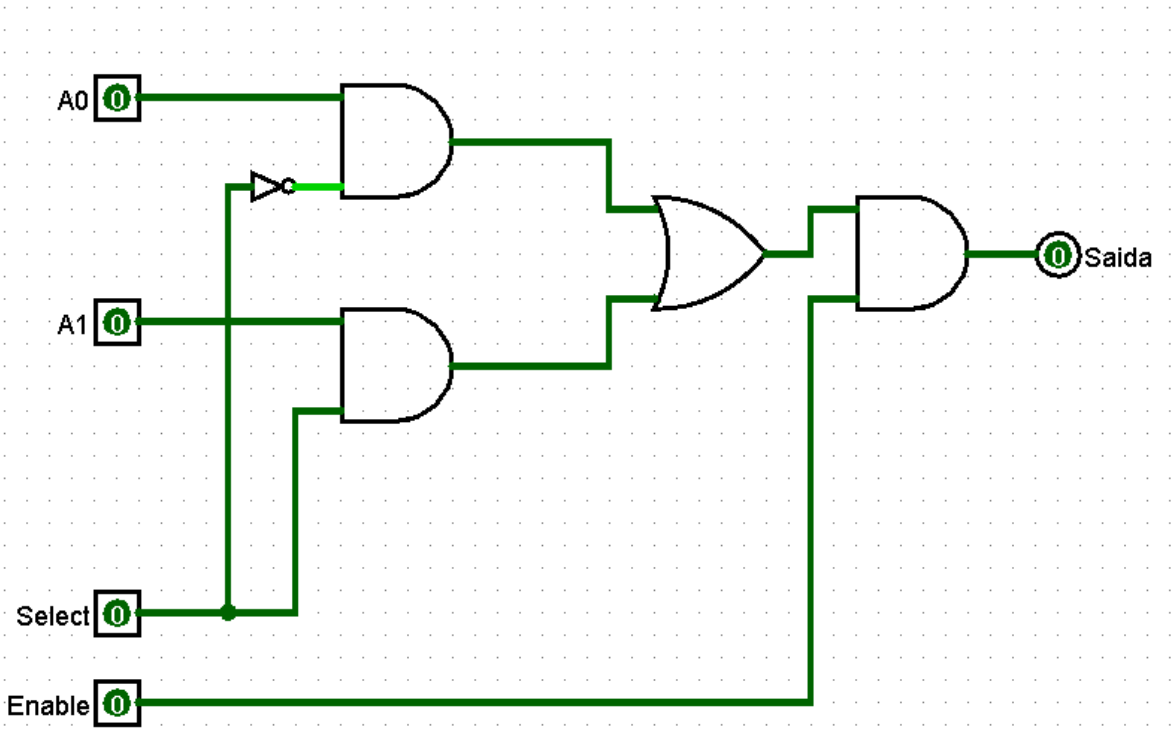


A imagem acima, mostra a implementação de uma memória ROM 8x8 no logisim, onde tem-se 8 constantes de 8 bits representando 8 palavras/dados de 8 bits que normalmente não são alterados. Estas palavras são acessadas por meio de um multiplexador 8x1 de 8 bits que seleciona um destes por vez por meio de um endereço que na imagem é representado por uma entrada de 3 bits. Existe também neste circuito uma entrada chamada Enable que caso seja 0 desabilita a leitura e caso seja 1 a leitura ocorre normalmente. A palavra lida é exibida na saída de 8 bits do multiplexador.

A seguir será mostrado como foi feita a implementação do multiplexador 8x1 de 8 bits utilizado na construção da memória ROM.

2.5.1. Multiplexador 2x1 de 1 bit

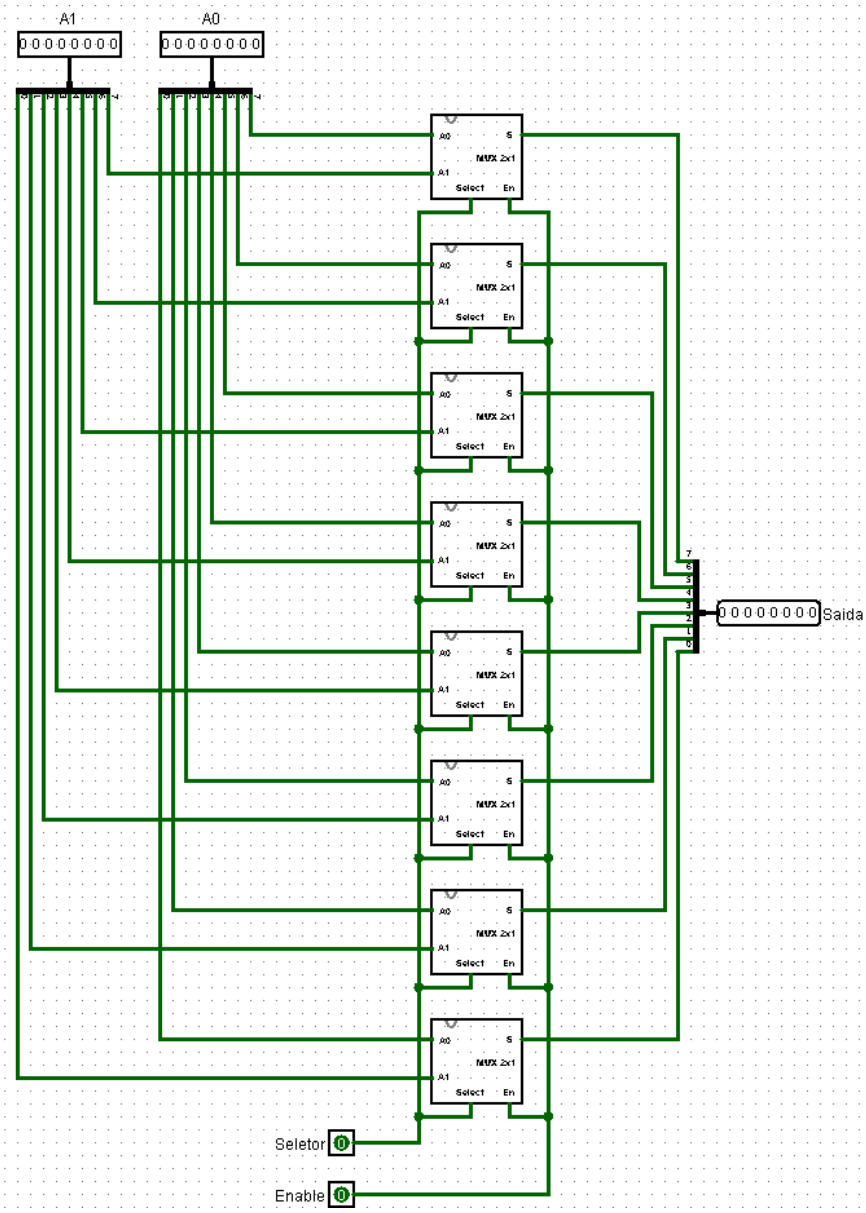
Figura 19 - Multiplexador 2x1 de 1 bit



A imagem acima mostra a construção de um multiplexador 2x1 básico. Esse circuito tem 2 entradas de dado, A0 e A1, que são selecionadas pela entrada chamada de Select. Caso Select seja 0, o dado que deve ser transferido à saída é A0, caso 1, o dado A1 que irá à saída. A entrada Enable serve para habilitar ou não essa transferência de dado a saída, caso Enable seja igual a 0, o dado não chega à saída, caso seja 1, a operação do multiplexador ocorre normalmente.

2.5.2. Multiplexador 2x1 de 8 bits

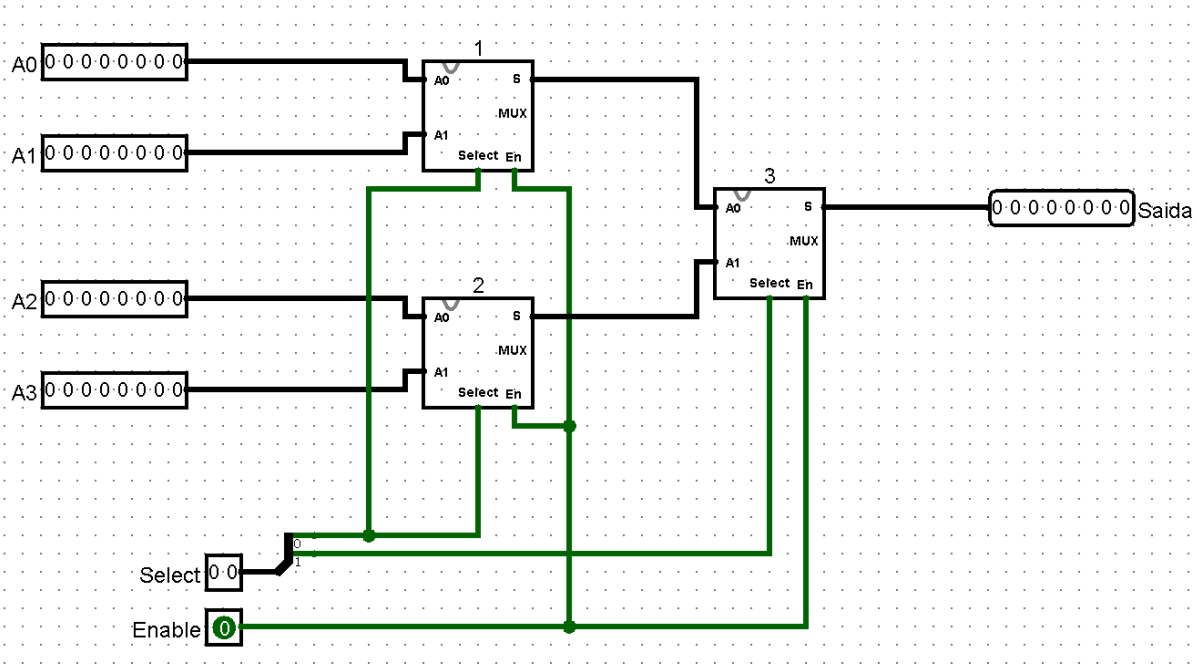
Figura 20 - Multiplexador 2x1 de 8 bits



Utilizando 8 multiplexadores 2x1 de 1 bit, iguais aos do item anterior, é possível construir um multiplexador 2x1 que trabalhe com 2 entradas de 8 bits. Neste novo multiplexador, cada uma das 8 unidades de multiplexadores 2x1 de 1 bit ficará responsável por selecionar 1 bit de cada entrada. É importante notar que este circuito funciona da mesma forma que o anterior, no qual a entrada Select seleciona um dos dados e a entrada Enable possibilita ou não a passagem do dado selecionado para a saída. Por isso, ambas devem ser entradas comuns aos 8 multiplexadores para então possibilitar a seleção correta dos dados. A diferença entre este circuito e o anterior está nas entradas de dados A0 e A1 que são de 8 bits, nos distribuidores utilizados para distribuir os bits dessas entradas entre os multiplexadores e na saída de 8 bits.

2.5.3. Multiplexador 4x1 de 8 bits

Figura 21 - Multiplexador 4x1 de 8 bits



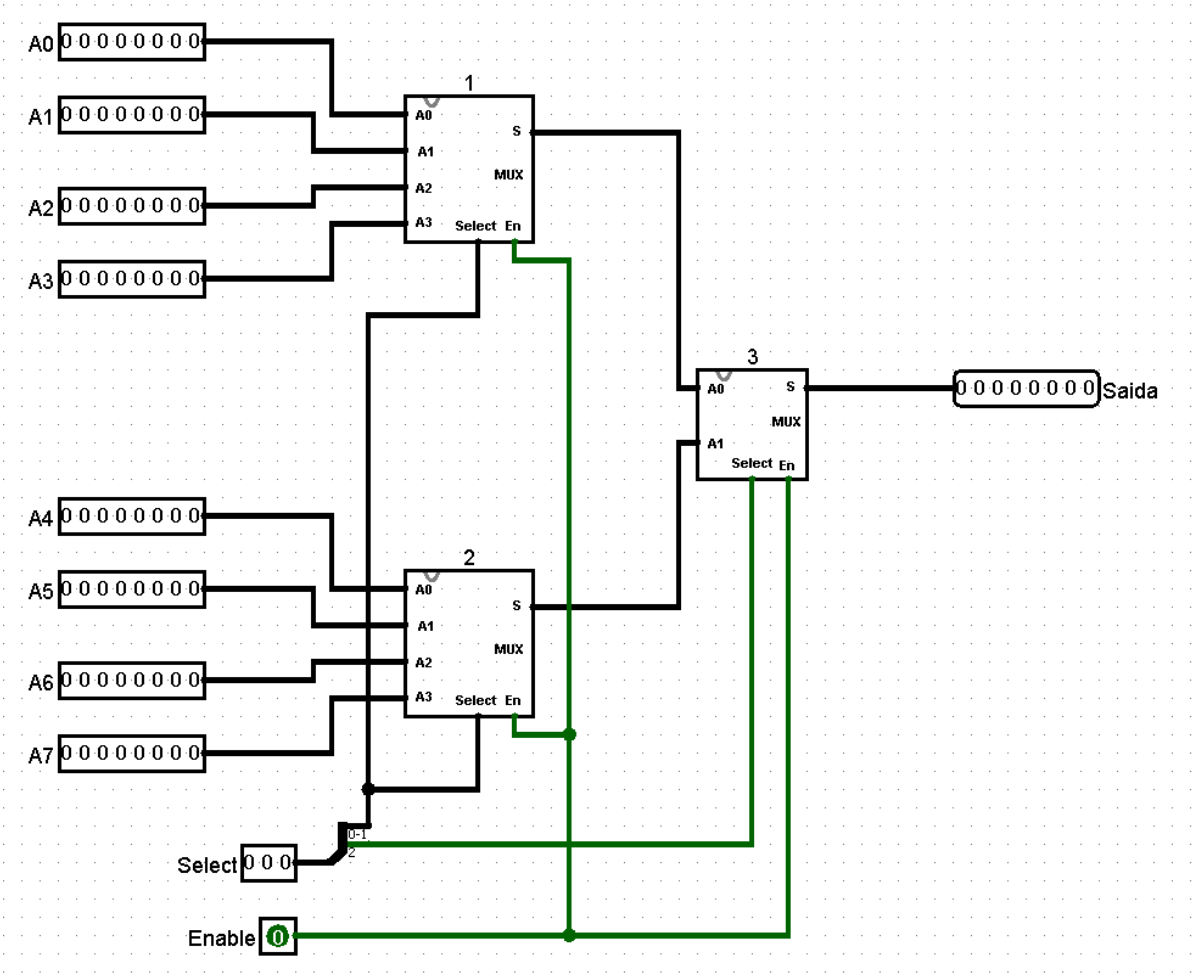
A imagem acima mostra o último item necessário para construir um multiplexador 8x1 de 8 bits, este item é o multiplexador 4x1 de 8 bits. Este novo circuito é constituído de 3 multiplexadores 2x1 de 8 bits, 4 entradas de 8 bits correspondentes aos dados, 1 entrada de 2 bits que será o Select, 1 entrada de 1 bit que será o Enable e 1 saída de 8 bits para mostrar o dado selecionado.

A lógica por trás deste circuito, é fazer a seleção de 2 entradas por vez. O primeiro multiplexador seleciona uma das entradas A0 ou A1 e transfere o dado selecionado para sua saída que está conectada a entrada A0 do terceiro multiplexador. O segundo multiplexador lida com as entradas A2 e A3, selecionando uma destas e enviando para a entrada A1 do terceiro multiplexador. O último multiplexador por sua vez fica responsável por selecionar uma das saídas dos multiplexadores 1 e 2 e assim mostrar o resultado. Diferente dos 2 últimos multiplexadores mostrados, este tem uma entrada de seleção de 2 bits, em que o primeiro bit fica responsável por fazer a seleção dos multiplexadores 1 e 2, e o último bit faz a seleção no multiplexador 3.

Para explicar de forma mais direta, quando a entrada Select é igual a 00, o dado que deve aparecer na saída é o mesmo que corresponde à entrada A0, quando o Select é igual a 01, o dado que aparecerá na saída será o correspondente à A1, quando Select igual a 10, o dado de A2 deve aparecer na saída e quando Select igual a 11, o dado A3 aparecerá. Isto ocorrerá, apenas quando Enable for igual a 1, caso Enable seja 0, a saída será nula.

2.5.4. Multiplexador 8x1 de 8 bits

Figura 22 - Multiplexador 8x1 de 8 bits



Por fim, temos na imagem acima a implementação do multiplexador 8x1 de 8 bits que é feito usando 2 multiplexadores 4x1 de 8 bits e 1 multiplexador 2x1 de 8 bits, ambos já apresentados nos itens 2.5.4 e 2.5.3 respectivamente. Este circuito consiste também de 8 entradas de 8 bits que são os dados simbolizados de A0 a A7, uma entrada de seleção de 3 bits chamada de Select, uma entrada de 1 bit correspondente ao Enable e uma saída de 8 bits.

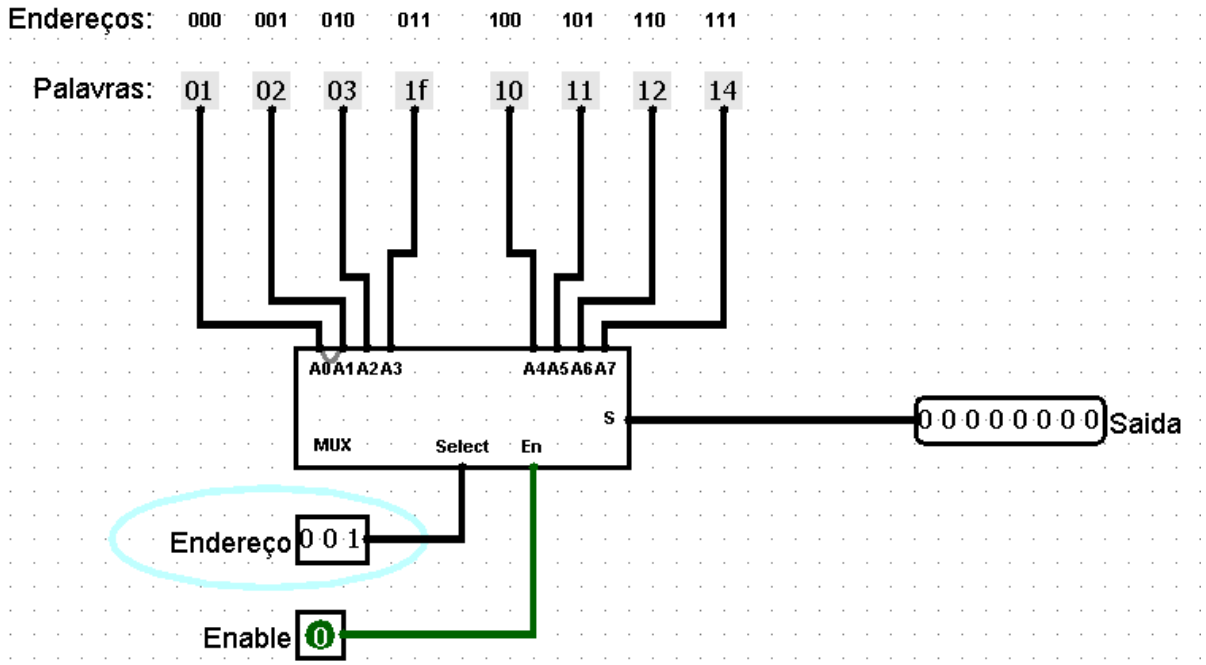
O funcionamento deste circuito é dado, assim como no multiplexador 4x1 de 8 bits, dividindo se o trabalho de seleção das 8 entradas, onde o primeiro multiplexador 4x1 de 8 bits se encarrega das entradas A0 a A3 e o segundo multiplexador se encarrega das entradas A4 a A7 de modo que a seleção nestes 2 multiplexadores é feita a partir dos 2 primeiros bits de seleção da entrada Select. O resultado da seleção do multiplexador 8x1 de 8 bits é feito pelo multiplexador 2x1 de 8 bits, neste caso o terceiro multiplexador que seleciona umas das saídas dos multiplexadores anteriores conforme o último bit de seleção da entrada Select. Todo este processor explicado acima, só é valido quando a entrada Enable é igual a 1, caso Enable seja igual a 0, o resultado da saída do multiplexador sempre será 00000000.

Sabendo do funcionamento do multiplexador de 8x1 de 8 bits usando para a construção da memória ROM, abaixo são apresentados alguns casos de teste para a memória ROM construída.

2.5.5. Caso Enable Desativado

Neste caso, com o Enable igual a 0, independente da palavra selecionada pela entrada Select que neste caso é a palavra de endereço 001, a saída apresentará um valor final de 00000000, pois quando Enable é igual a 0 a leitura da memória ROM é desativada.

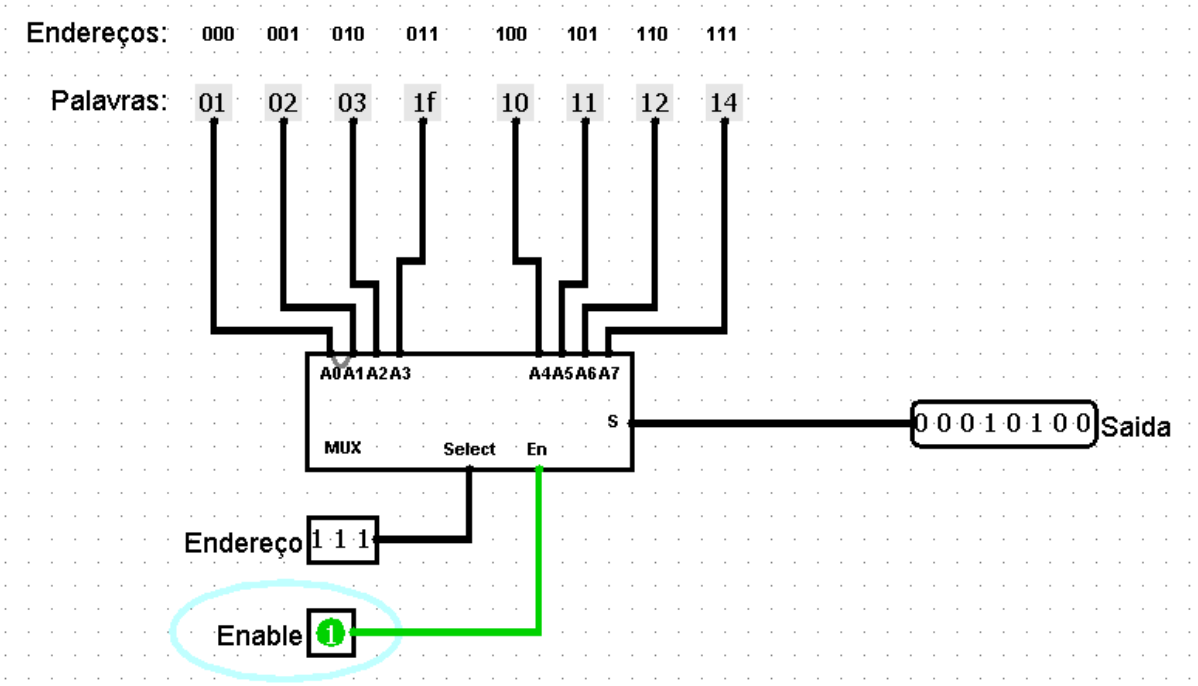
Figura 23 - Memória ROM Caso Enable Desativado



2.5.6. Caso Enable Ativado

Ao contrário do caso anterior, quando Enable é igual 1, a leitura da memória ROM é ativada, fazendo com que a saída receba a informação da palavra selecionada normalmente. Neste caso a palavra selecionada foi a de endereço 111.

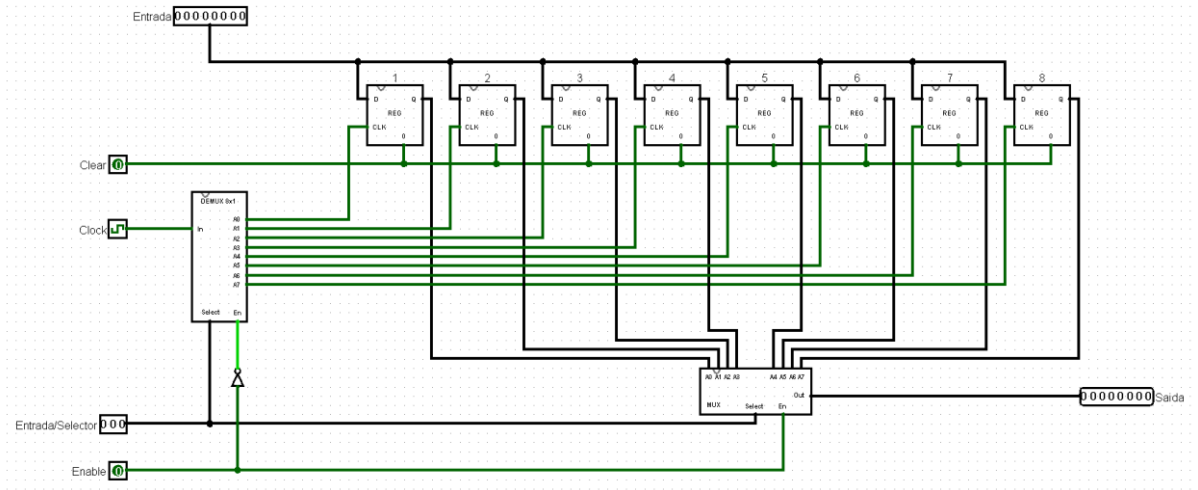
Figura 24 - Memória ROM Caso Enable Ativado



2.6. Memória RAM de 8 bits

Memória RAM é um termo empregado para se referir às memórias cujo tempo de acesso em qualquer endereço da memória é igual. Existem diversos tipos de memórias que podem receber esta alcunha, um exemplo é a própria memória ROM apresentada no componente anterior. Quando se trata de RAM em memórias semicondutoras, normalmente se refere às memórias do tipo RWM (Read-Write-Memory) que são aquelas em que é possível realizar as operações de leitura e escrita de dados.

Figura 25 - Memória RAM de 8 bits



Acima é mostrada a implementação de uma memória RAM de 8x8, na qual se tem 8 palavras de 8 bits representadas por 8 registradores paralelos, construídos utilizando flip-flops do tipo D, nos quais os dados são gravados ou lidos.

Nesta memória RAM, a escrita é feita nos registradores que recebem a mesma entrada, mas que somente um registrador pode armazenar uma informação por vez, devido ao demultiplexador 1x8 que limita o pulso de clock nesses registradores, conforme o endereço passado na entrada Selector.

A operação de leitura nesta memória RAM é feita de forma análoga a de leitura da memória ROM, onde se tem um multiplexador 8x1 de 8 bits que recebe as saídas dos registradores e conforme a passagem do endereço na entrada Selector é apresentado uma saída correspondente a informação do registrador selecionado.

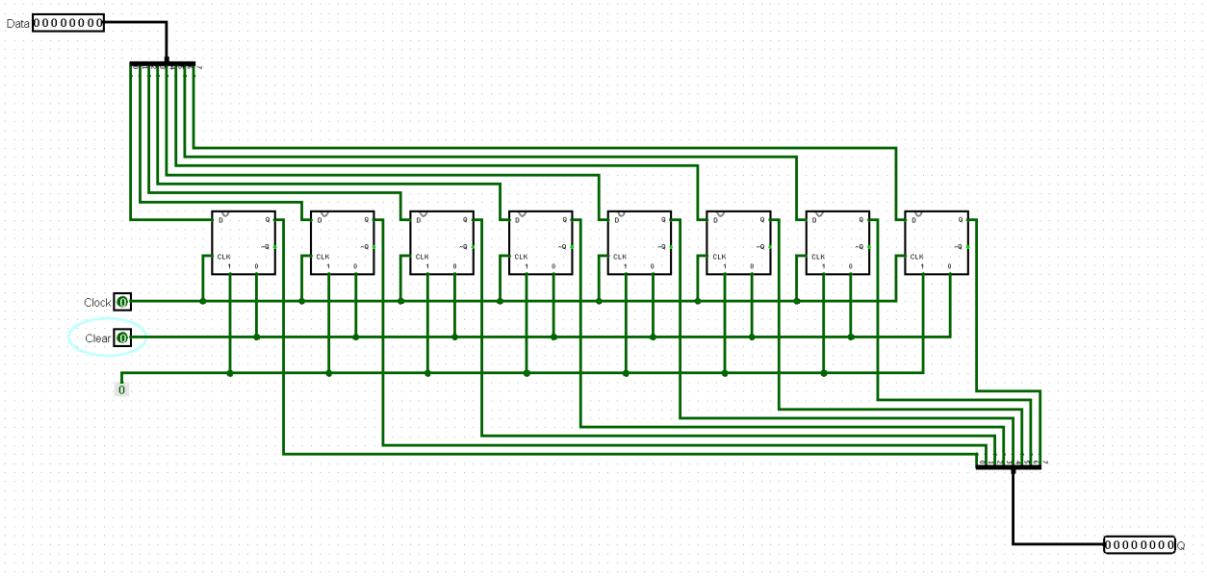
Vale ressaltar que nesta implementação não é possível realizar a leitura e escrita de dados ao mesmo tempo, pois a entrada Enable quando igual a 1 habilita a leitura no multiplexador, mas desabilita a escrita no demultiplexador e quando igual a 0 o inverso acontece.

A seguir será mostrado como foi feita a implementação dos registradores e do demultiplexador utilizados na construção desta memória RAM.

2.6.1. Registrador Paralelo de 8 bits

Utilizando 8 flip-flops do tipo D apresentados na Figura 2 deste relatório, é possível criar um registrador paralelo capaz de armazenar um dado de 8 bits iguais aos usados na criação da memória RAM 8x8.

Figura 26 - Registrador Paralelo de 8 bits

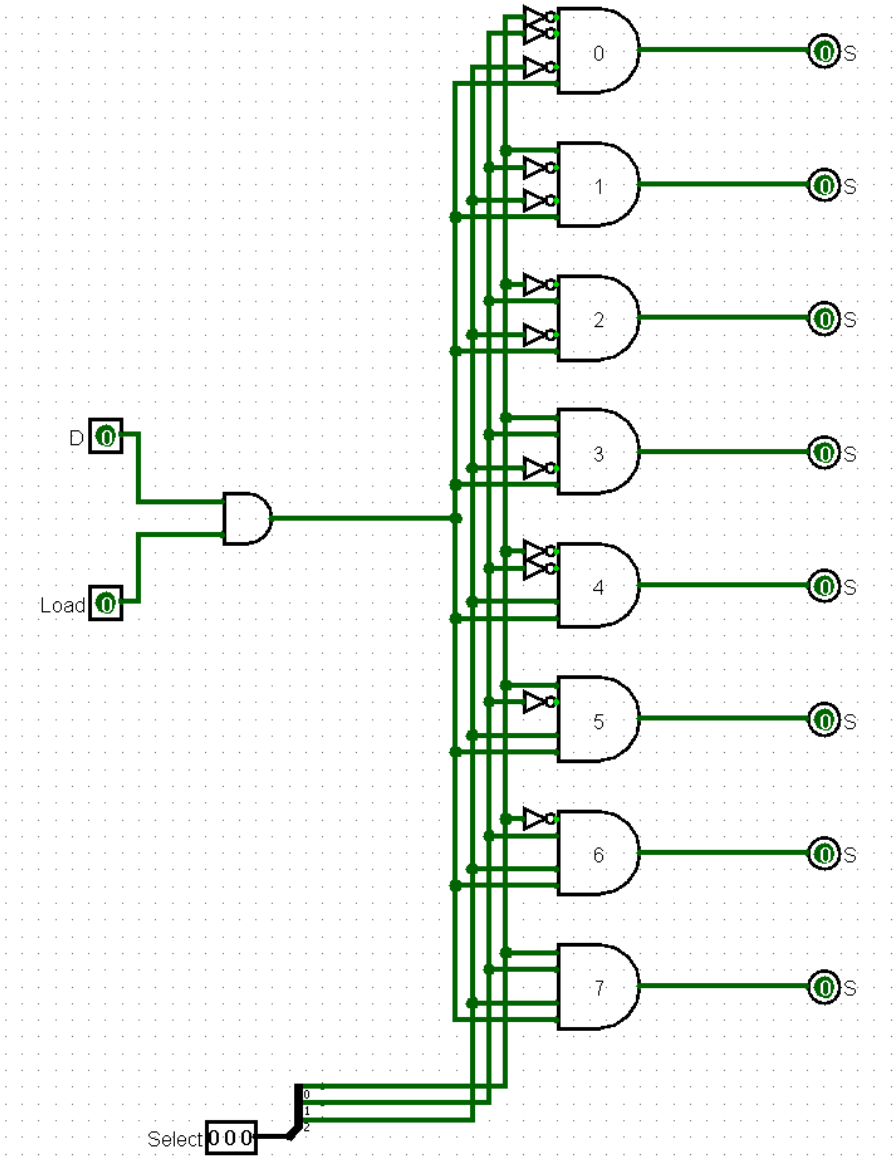


A imagem acima mostra a implementação de um registrador de 8 bits. Nele se pode observar a entrada de dados de 8 bits chamada de Data, a saída de 8 bits Q, a entrada de Clock que permitirá a gravação dos dados da entrada, a entrada Clear que reseta todos os flip-flops e a entrada constante 0 que não é utilizada na implementação deste registrador.

O funcionamento deste registrador se dá de forma paralela, ou seja, os dados da entrada são gravados todos ao mesmo tempo nos flip-flops, bastando um pulso de clock. Deve se também utilizar 2 distribuidores que permitiram a entrada e a saída dos dados de forma paralela. Esse registrador apresentado, possibilita a gravação e leitura de dados na memória RAM, sem este componente é impossível guarda os dados desta.

2.6.2. Demultiplexador 1x8 de 1 bit

Figura 27 - Demultiplexador 1x8



Um demultiplexador é um circuito que realiza uma operação inversa a do multiplexador, ou seja, recebe uma entrada de dado e distribui esse dado as suas saídas conforme uma chave de seleção.

O demultiplexador apresentado na imagem possui uma entrada de 1 bit chamada de D que é o dado, uma entrada de 1 bit chamada de Load que habilita ou não a operação do demultiplexador, uma entrada de seleção de 3 bits chamada Select que seleciona qual das 8 saídas devem apresentar o dado da entrada D e as 8 saídas que conforme a seleção, uma delas apresenta o dado de entrada.

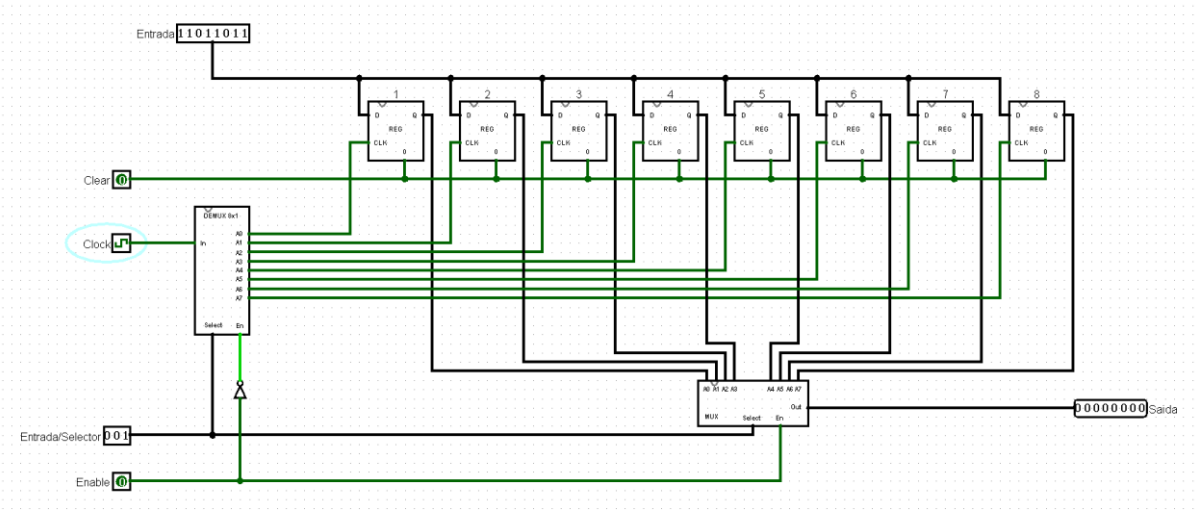
No caso da memória RAM, o dado de entrada do demultiplexador é o sinal de clock, que dependendo da seleção habilitam ou não um registrador salvar uma informação. Ou seja, caso Select seja 000, apenas o primeiro registrador gravará a entrada, caso Select seja 001, o segundo registrador grava o dado da entrada, caso 010, o terceiro grava e assim por diante.

A seguir é mostrado alguns casos de testes feitos com a memória RAM.

2.6.3. Caso Escrita

Para escrita de dados na memória RAM, a entrada Enable deve ser igual a 0 habilitando assim a escrita. Com Enable igual a 0 basta escolher o endereço do registrador que deve gravar a informação da entrada.

Figura 28 - Memória RAM Caso Escrita

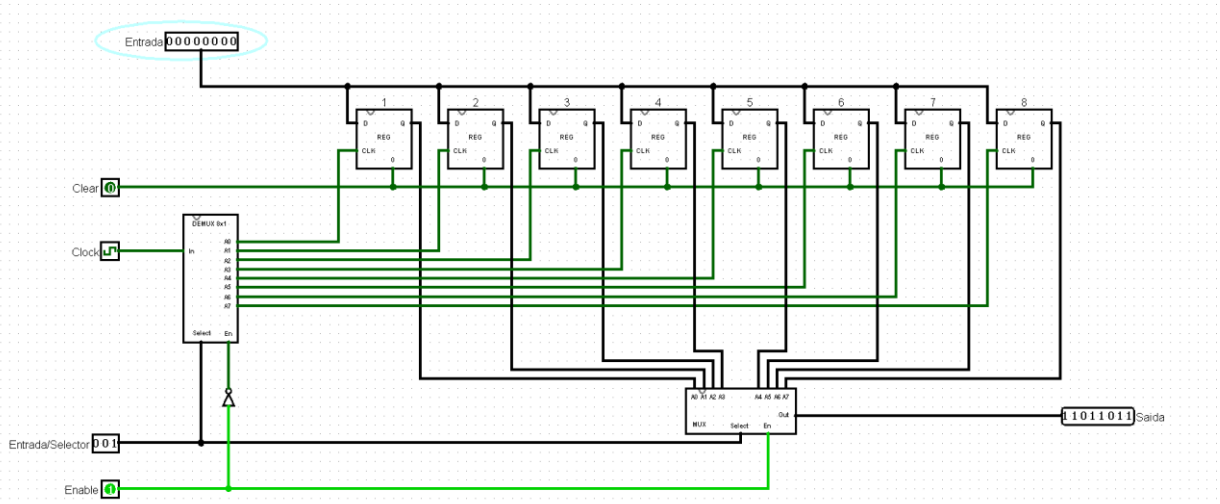


A imagem acima mostra a entrada de dado que contém a informação 11011011, que deve ser gravada no segundo registrador que tem endereço 001, para sacramentar a operação, basta um pulso de clock.

2.6.4. Caso Leitura

Para realizar a leitura de dados em algum endereço da memória RAM, a entrada Enable deve ser igual a 1.

Figura 29 - Memória RAM Caso Leitura

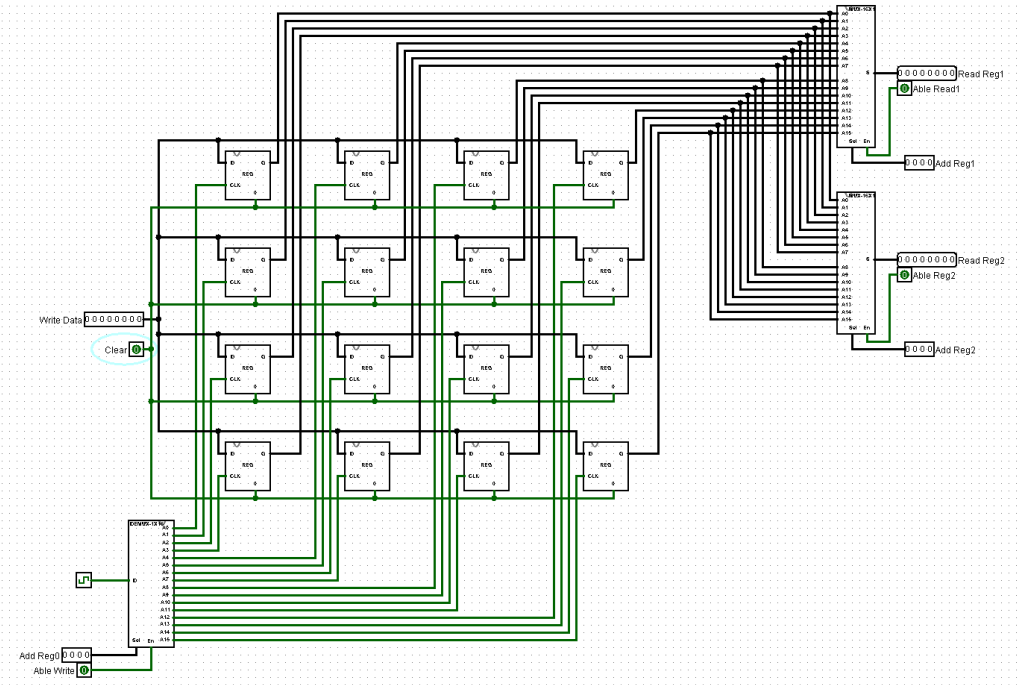


A imagem acima mostra a leitura no endereço 001, em que foi feita a gravação da informação 11011011, pode se observar que o mesmo dado anteriormente escrito, aparece na saída, validando assim tanto a leitura e a escrita deste circuito.

2.7. Banco de Registradores de 8 bits

O banco de registradores é um componente digital essencial no funcionamento de um processador, responsável por armazenar temporariamente as informações que estão sendo processadas em um dado momento. Ele consiste em um conjunto organizado de registradores, permitindo a execução de operações de leitura e de escrita sendo otimizado para acesso extremamente rápido às informações diretamente utilizadas pelo processador.

Figura 30 – Banco de Registradores de 8 bits



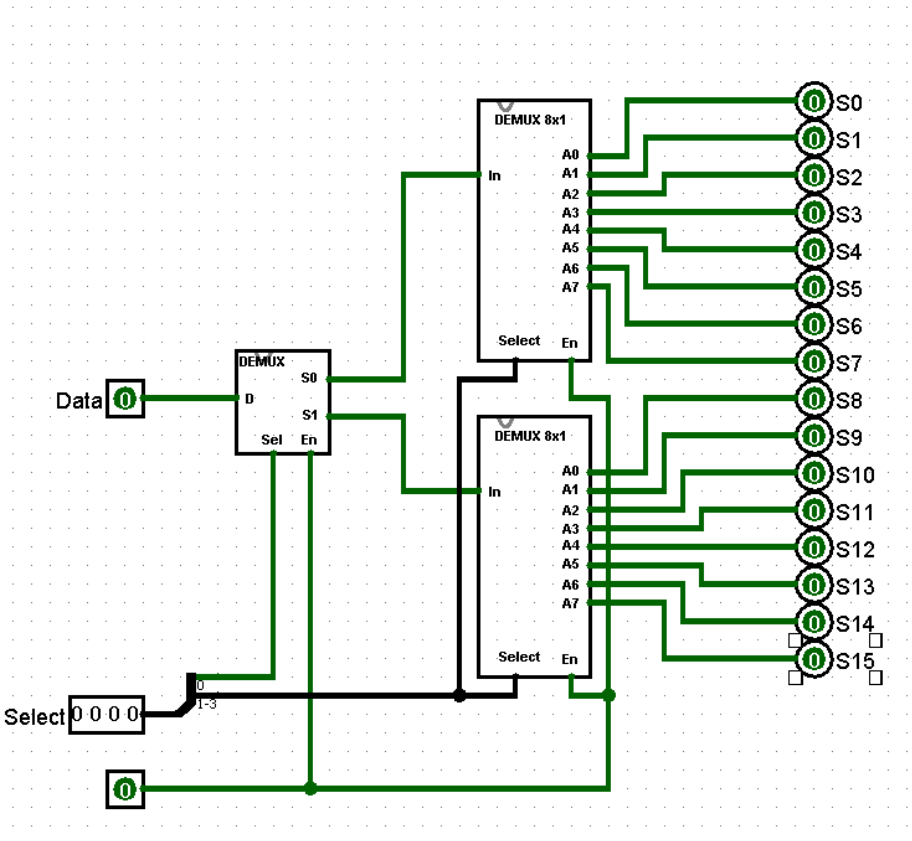
A imagem apresentada ilustra um banco de registradores implementado no Logisim. Este banco é composto por 16 registradores de 8 bits cada e possui uma arquitetura que inclui um demultiplexador 16x1, utilizado para habilitar o pulso de clock exclusivamente para o registrador que irá realizar a gravação de dados. Além disso, ele conta com dois multiplexadores responsáveis pela leitura dos dados armazenados nos registradores.

Com relação as entradas deste banco, ele possui um total de 8 entradas, sendo uma entrada de 8 bits destinada ao dado a ser gravado, três seletores de endereço de 4 bits cada (Add Reg), utilizados para selecionar qual registrador será utilizado para leitura ou escrita, três sinais de habilitação de 1 bit (Able), que ativam ou desativam as operações disponíveis no banco e um sinal de limpeza (Clear) de 1 bit, utilizado para apagar o conteúdo armazenado nos registradores.

Abaixo é mostrada a implementação do demultiplexador de 1x16 e do multiplexador 16x1 de 8 bits. Deve se levar em conta que foi utilizado na implementação destes dispositivos outros componentes já descritos anteriormente que apenas serão citados.

2.7.1. Demultiplexador 1x16 de 1 bit

Figura 31 - Demultiplexador 1x16 de 1 bit



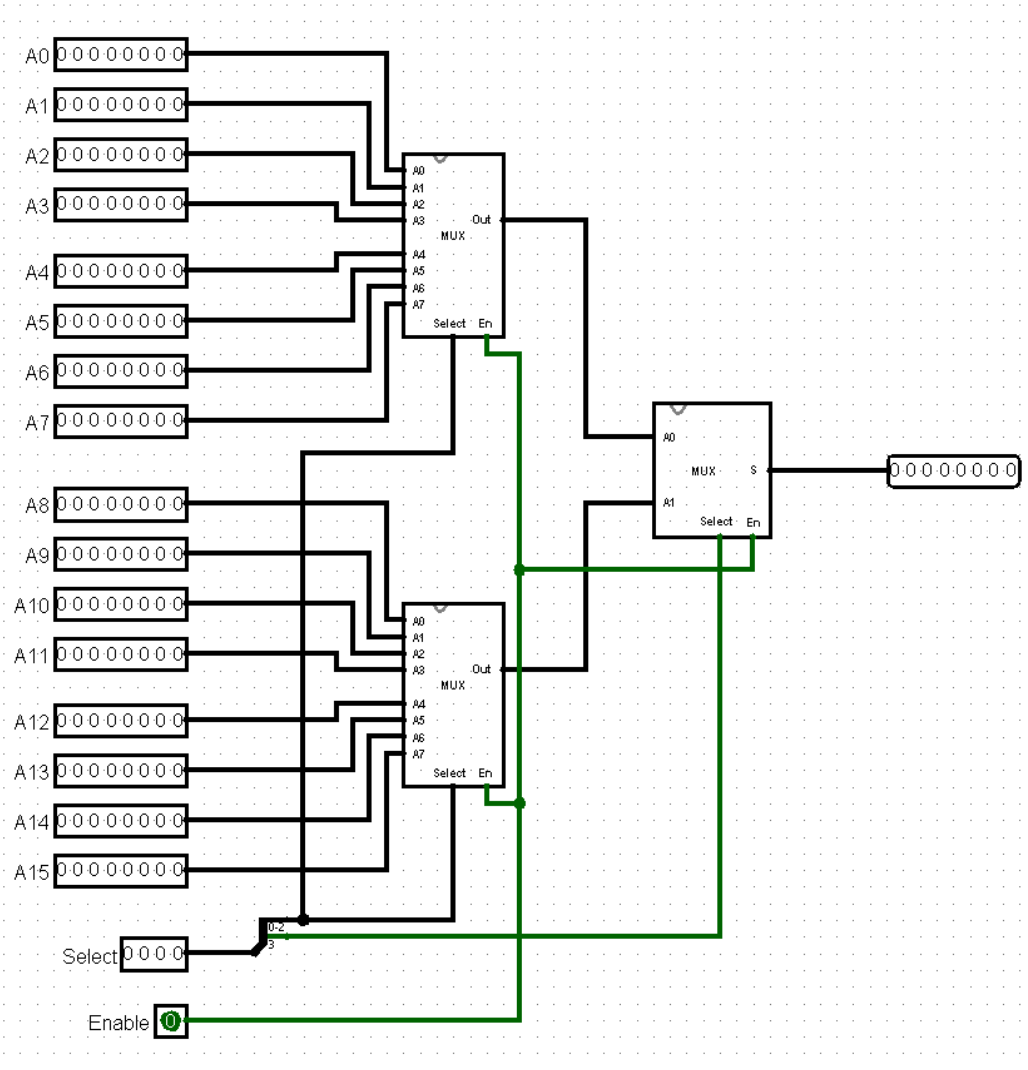
O demultiplexador apresentado na imagem é implementado utilizando dois demultiplexadores 1x8 apresentados no item 2.6.2 e um demultiplexador 2x1, configurados de forma hierárquica.

Ele possui três entradas principais, sendo uma entrada de 1 bit denominada Data, que é o dado a ser direcionado para a saída correspondente; uma entrada de 4 bits que funciona como seletor de endereço, responsável por determinar qual saída será ativada; uma entrada de 1 bit chamada Enable, utilizada para habilitar ou desabilitar o funcionamento do demultiplexador.

Caso o seletor seja igual a 0000, o dado de entrada é direcionado para a saída S0, caso seja 0001 o dado é direcionado para a saída S1, caso 0010 o dado sai em S3 e assim por diante.

2.7.2. Multiplexador 16x1 de 8 bits

Figura 32 – Multiplexador 16x1 de 8 bits



O multiplexador 16x1 de 8 bits é implementado utilizando uma combinação de dois multiplexadores 8x1 de 8 bits e um multiplexador 2x1 de 8 bits, previamente descritos nos itens 2.5.4 e 2.5.2 respectivamente. Essa configuração permite ampliar o número de entradas para 16, mantendo a lógica de seleção eficiente.

Este circuito possui um total de 18 entradas, sendo 16 destas para os dados de 8 bits (A0 a A15), uma entrada de 4 bits corresponde ao seletor de endereço (Select) e uma entrada de 1 bit correspondente ao Enable.

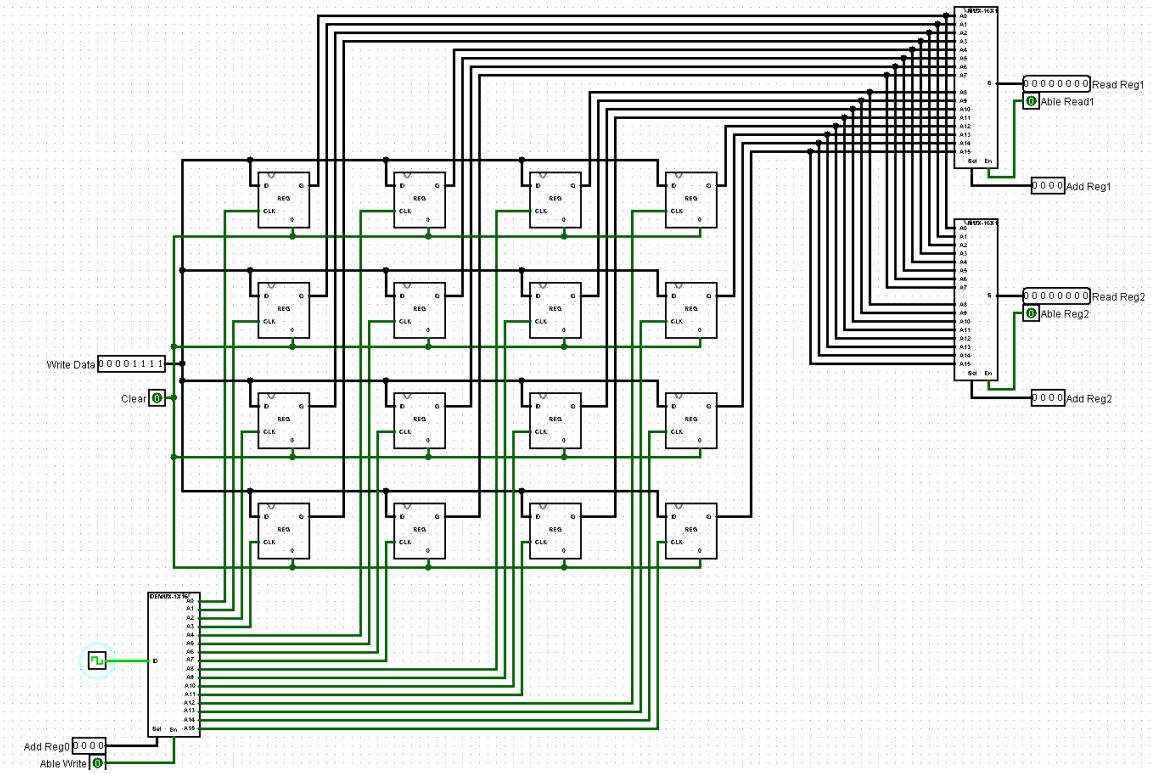
O funcionamento do multiplexador ocorre da seguinte maneira, quando o sinal enable está ativado, o multiplexador analisa o valor do seletor de endereço. Caso o seletor seja 0000, o dado da entrada A0 será direcionado para a saída, para o seletor 0001, o dado da entrada A1 será exibido e se o seletor for 0010, o dado da entrada A2 aparecerá na saída. Esse padrão se repete para todas as combinações possíveis do seletor de endereço, até o valor 1111, que corresponderá ao dado da entrada A15.

Sabendo do seu funcionamento e de seus componentes, a seguir são apresentados alguns casos de teste aplicados ao banco de registradores.

2.7.3. Caso Operações Desabilitadas

Ao tentar gravar ou ler dados com as entradas de habilitação de operação (Able's) desligadas, é possível notar que nenhum dado é gravado ou lido mesmo aplicando um pulso de clock. Esse comportamento está de acordo com o esperado, pois as entradas de habilitação atuam como controladores de suas respectivas operações.

Figura 33 - Banco de Registradores Caso OP Desabilitadas

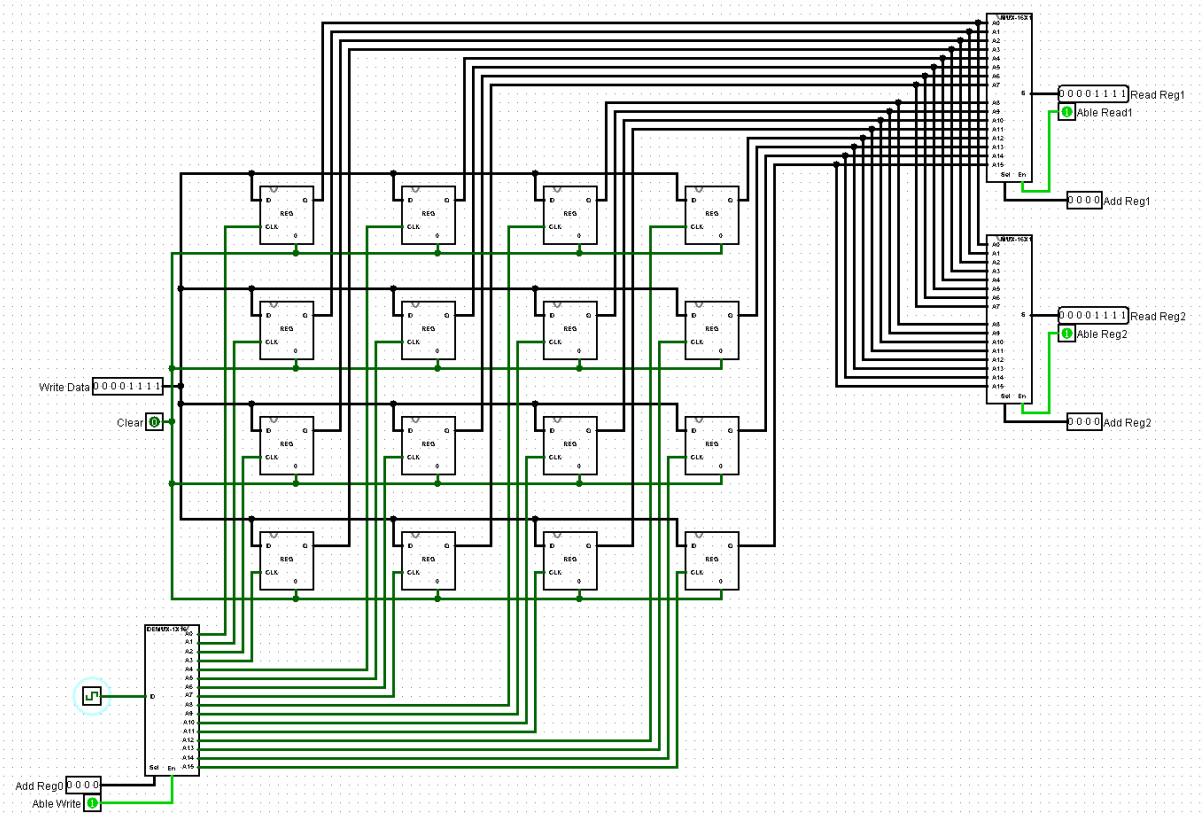


A imagem acima mostra uma tentativa de escrita e leitura no registrador de endereço 0000 que não foi permitida pelas entradas Able Write, Able Read1 e Able Read2 estarem desativadas.

2.7.4. Caso Operações Habilitadas

Uma vez que as entradas de habilitação de operação estão ativadas é esperado que as operações de escrita e leitura no banco de registradores ocorram normalmente, pois neste estado o circuito permite que um dado seja gravado ou lido em um registrador.

Figura 34 - Banco de Registradores Caso OP Habilitadas



Na imagem acima é possível observar que com as entradas de habilitação de operação ativas, a escrita e leitura de dados no registrador de endereço 0000 ocorre normalmente.

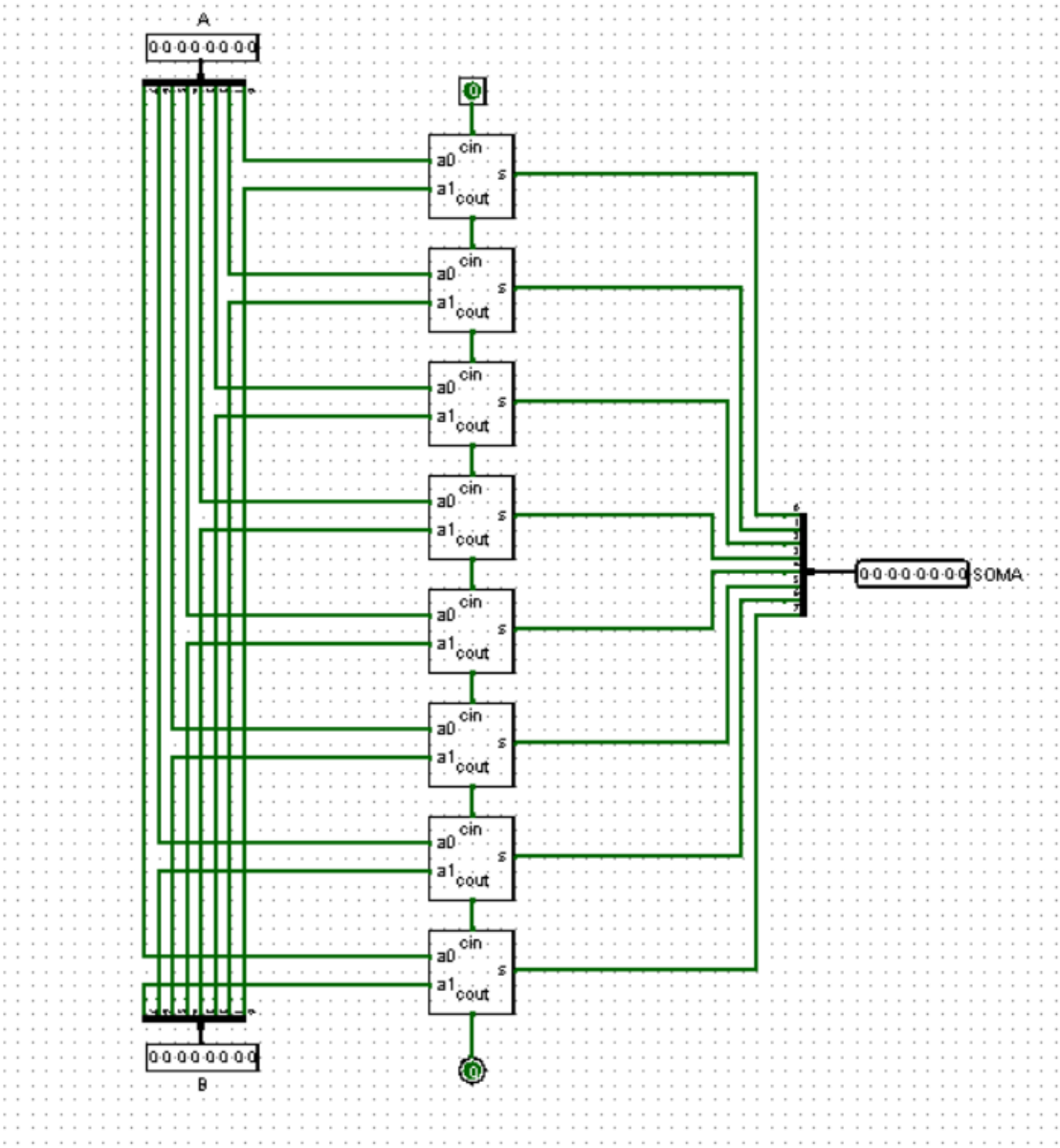
É importante relembrar que neste circuito as entradas de habilitação de operação são independentes, ou seja, é possível realizar somente a escrita ou também somente a leitura de dados, sem que uma entrada Able influencia outra operação.

2.8. Somador de 8 bits

Um somador de 8 bits é um circuito digital que realiza a soma de dois números binários de 8 bits (A e B) e, opcionalmente, considera um bit de transporte de entrada

(Cin), gerando como resultado número binário de 8 bits representado pela saída SOMA e em caso de “estouro” de bits vai 1 para o Cout ocorrendo um overflow. A soma ocorre por conta dos 8 somadores de bit em paralelo e as suas entradas recebem os bits correspondentes a suas posições, como pode ser visto na imagem a abaixo.

Figura 35 - Somador de 8 bits



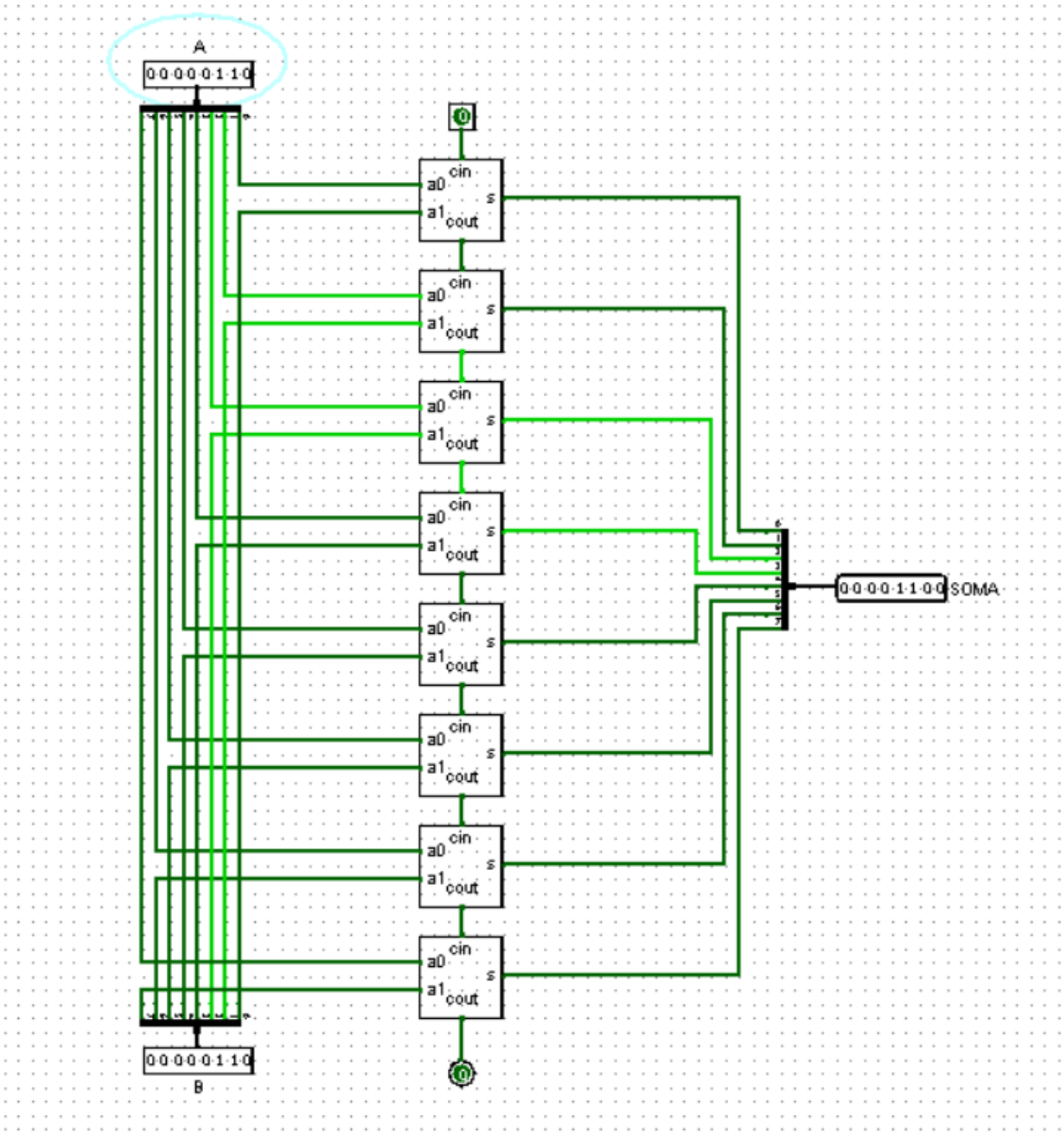
O circuito acima representa um somador de 8 bits, composto por 8 somadores de 1 bit, 3 distribuidores, duas entradas de 8 bits (A e B) e uma saída de 8 bits (SOMA). O circuito funciona com cada somador de 1 bit recebendo os dados referente a sua posição na soma, onde ele tá preparado para receber um dado Cin e enviar um dado no Cout, além das duas entradas padrões, fazendo assim a clássica soma vai um e vem um, logo abaixo vai ser exemplificado dois casos, um de soma comum e

um de soma com resultado com overflow. A soma ocorre corretamente pois eles estão em paralelo, fazendo os dados serem distribuídos corretamente.

2.8.1 Caso de soma comum

No caso de uma soma comum, a entrada A recebe um valor de 8 bits que nesse caso é 00000110, em decimal é 6, e a entrada B também recebe 00000110, como pode se observar cada bit tem sua posição e com o uso do distribuidor esses valores vão pra somadores de 1 bit de suas respectivas posições, assim ocasionando no processo de soma correto, usado a saída Cout de somador e a entrada Cout de outro para conectar os somadores, assim realizando os transportes de bits correto para realização da soma. Abaixo tem o exemplo do valor 00000110(2) + 00000110(2) = 00001100(2), sendo em decimal 6 + 6 = 12, essa soma ocorreu espeitando o limite de bits, que são 8 bits.

Figura 36 - Somador de 8 bits caso comum

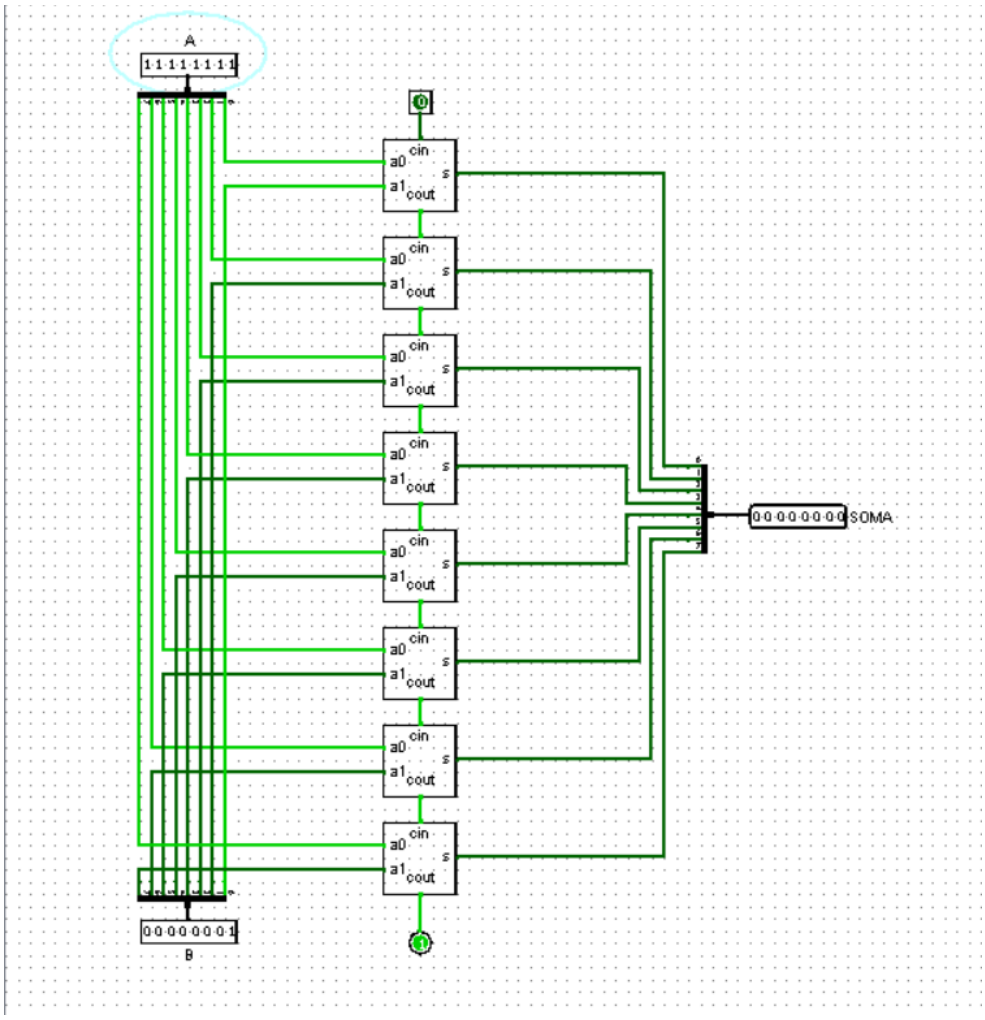


2.8.2 Caso de soma com overflow

No caso de uma soma com overflow, a entrada A recebe um valor de 8 bits, que neste caso é 11111111, representando o valor decimal 255, e a entrada B recebe 00000001, que em decimal é 1. Cada bit tem sua posição, e utilizando o distribuidor, esses valores são enviados para somadores de 1 bit em suas respectivas posições, realizando o processo de soma corretamente. A saída de cada somador gerará um bit de soma, e o valor do bit de transporte, denominado Cout, é passado como entrada para o somador seguinte, garantindo que os transportes de bits sejam feitos corretamente.

No exemplo, temos a soma de 11111111(2) + 00000001 (2) = 100000000 (2). Onde 11111111 é 255 em decimal, 00000001 é 1 em decimal e 100000000 é 256 em decimal. No entanto, como estamos trabalhando com um limite de 8 bits, ocorre o *overflow*, pois o valor 256 não pode ser representado com apenas 8 bits. Nesse caso, a saída final será truncada para 8 bits, gerando um resultado incorreto de 00000000 em binário, ou 0 em decimal.

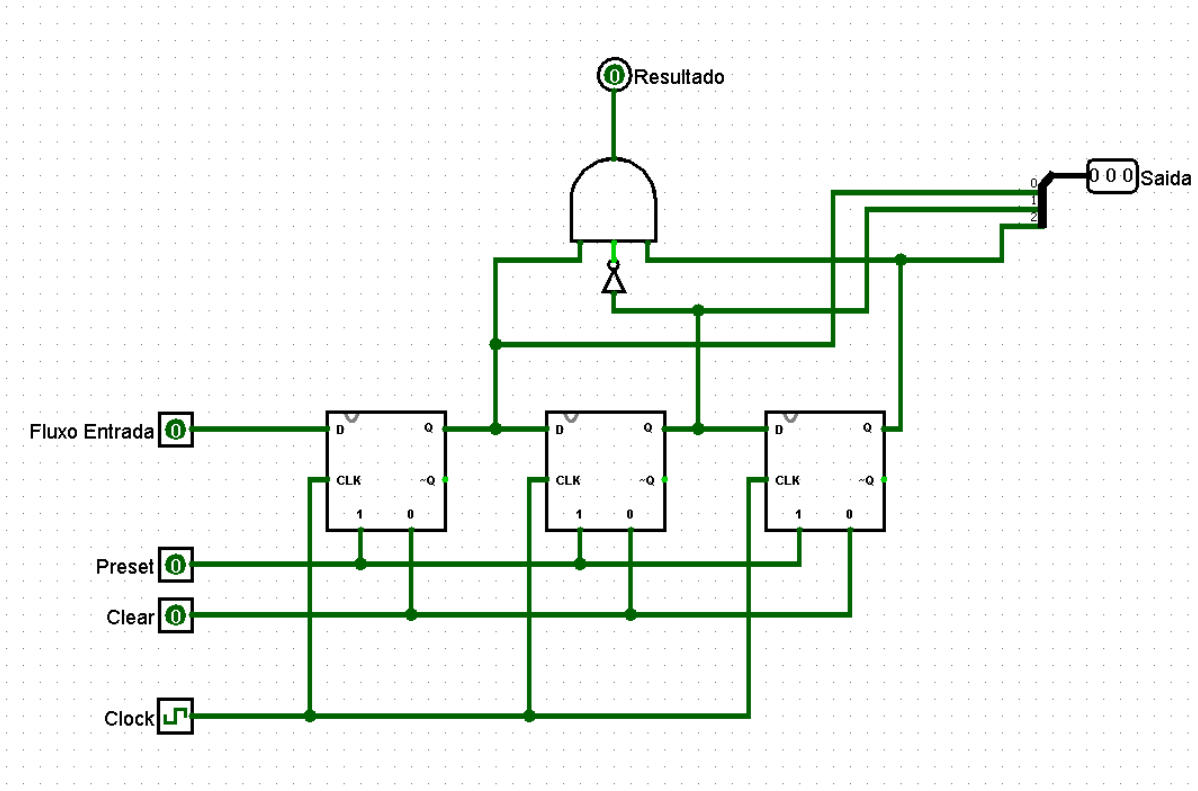
Figura 37 - Somador de 8 bits caso estouro



2.9. Detector de Sequência Binária 101

Um detector de sequência binária é um circuito capaz de detectar uma sequência de sinais de entrada em um conjunto de dados binários. Ou seja, é um circuito capaz de determinar se um conjunto de bits recebidos correspondem a uma sequência pré-estabelecida. Esse tipo de circuito é amplamente usando em sistemas de comunicação, processamento de dados e processamento de sinais digitais.

Figura 38 – Detector de Sequência Binária 101



A imagem acima ilustra a implementação de um detector de sequência binária 101. Este circuito, possui um total de 5 entradas sendo uma correspondente a entrada que deve ser analisada, as entradas padrões Clear e Preset dos flip-flops D e o clock responsável por atualizar os valores de entrada nos flip-flops D. Existem também neste circuito 2 saídas, uma saída de 1 bit chamada de Resulta que caso seja 1, indica que a sequência foi encontrada e uma saída de 3 bits que mostra a sequência atualmente em análise.

Este circuito possui 3 flip-flops do tipo D já apresentados no item 2.1.1, mais precisamente na Figura 2. Esses flip-flops então ligados de forma serial, com suas saídas feitas de forma paralela, na qual são elas que devem ser analisadas a fim de verificar se a sequência 101 foi atingida.

A seguir são apresentados alguns casos de testes para comprovar a eficiência do circuito.

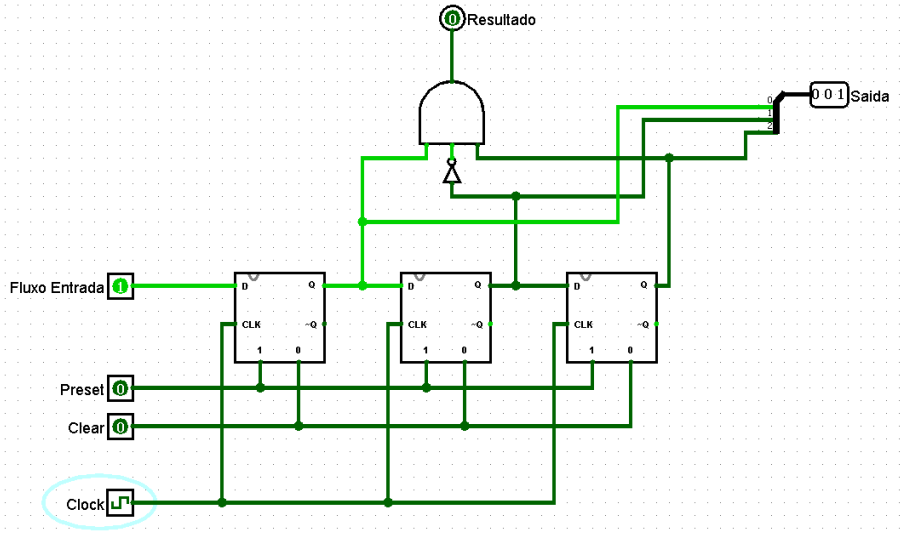
2.9.1. Caso Inicial

Todos os flip-flops armazenam o valor 0 e o clock não apresentou borda de descida. Neste momento o circuito está em estado de repouso assim como na Figura 38.

2.9.2. Caso Primeiro Pulso

Após a primeira borda de descida o valor da entrada passa a ser armazenado pelo primeiro flip-flop, enquanto os demais ainda armazenam 0.

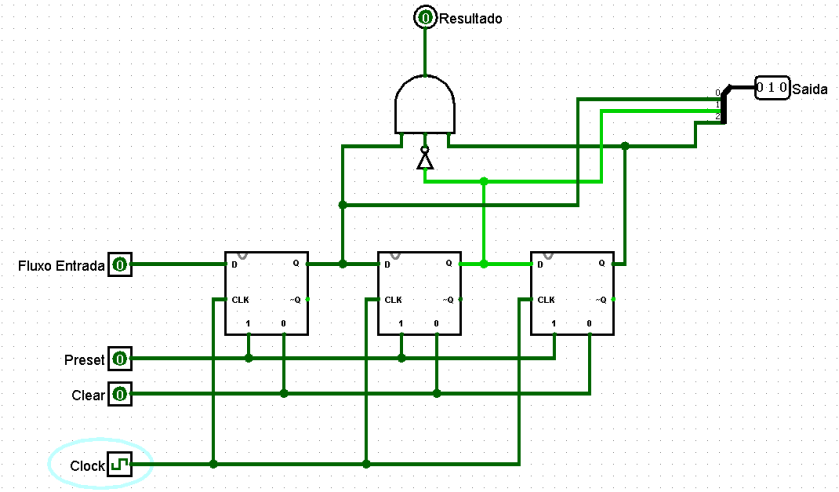
Figura 39 – Detector de Sequência Caso Primeiro Pulso



2.9.3. Caso Segundo Pulso

No segundo pulso de clock, a informação armazenada no primeiro flip-flop, é passada para o segundo. Neste mesmo instante um novo dado é atualizado no primeiro flip-flop conforme o valor da entrada.

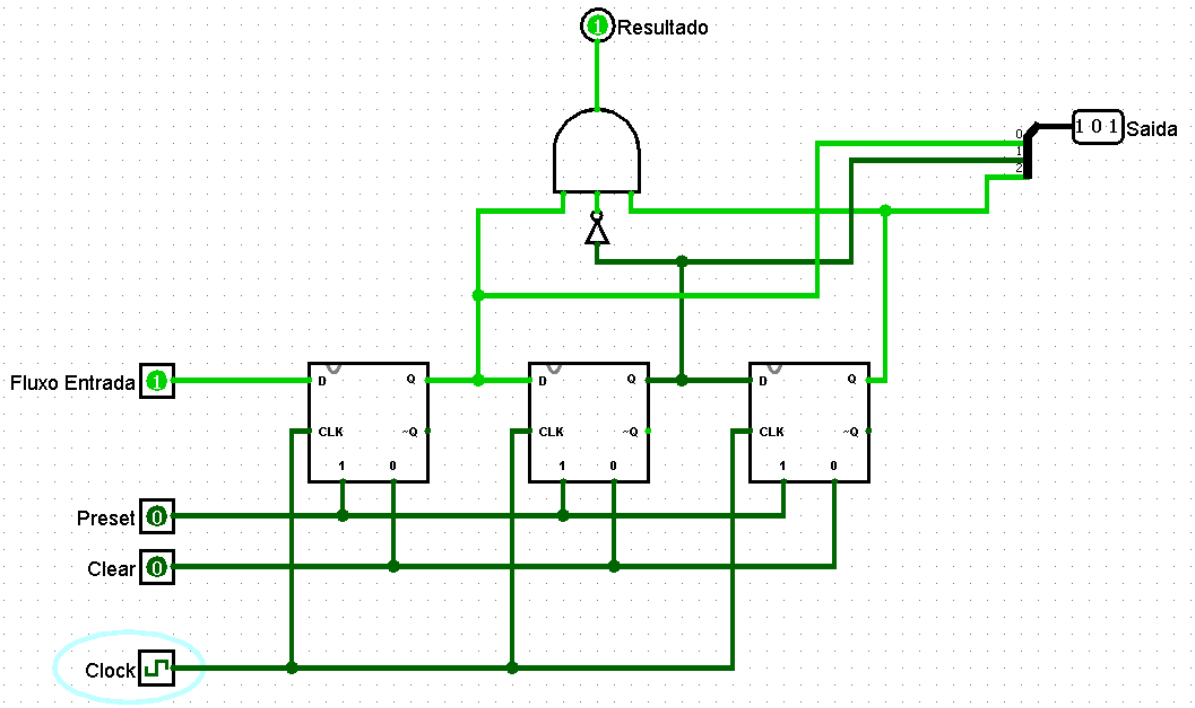
Figura 40 – Detector de Sequência Caso Segundo Pulso



2.9.4. Caso Terceiro Pulso

No terceiro pulso de clock, a informação do segundo flip-flop é passada para o terceiro. O segundo flip-flop é atualizado com a informação anterior do primeiro flip-flop e o primeiro recebe o valor atual da entrada.

Figura 41 - Detector de Sequência Caso Terceiro Pulso

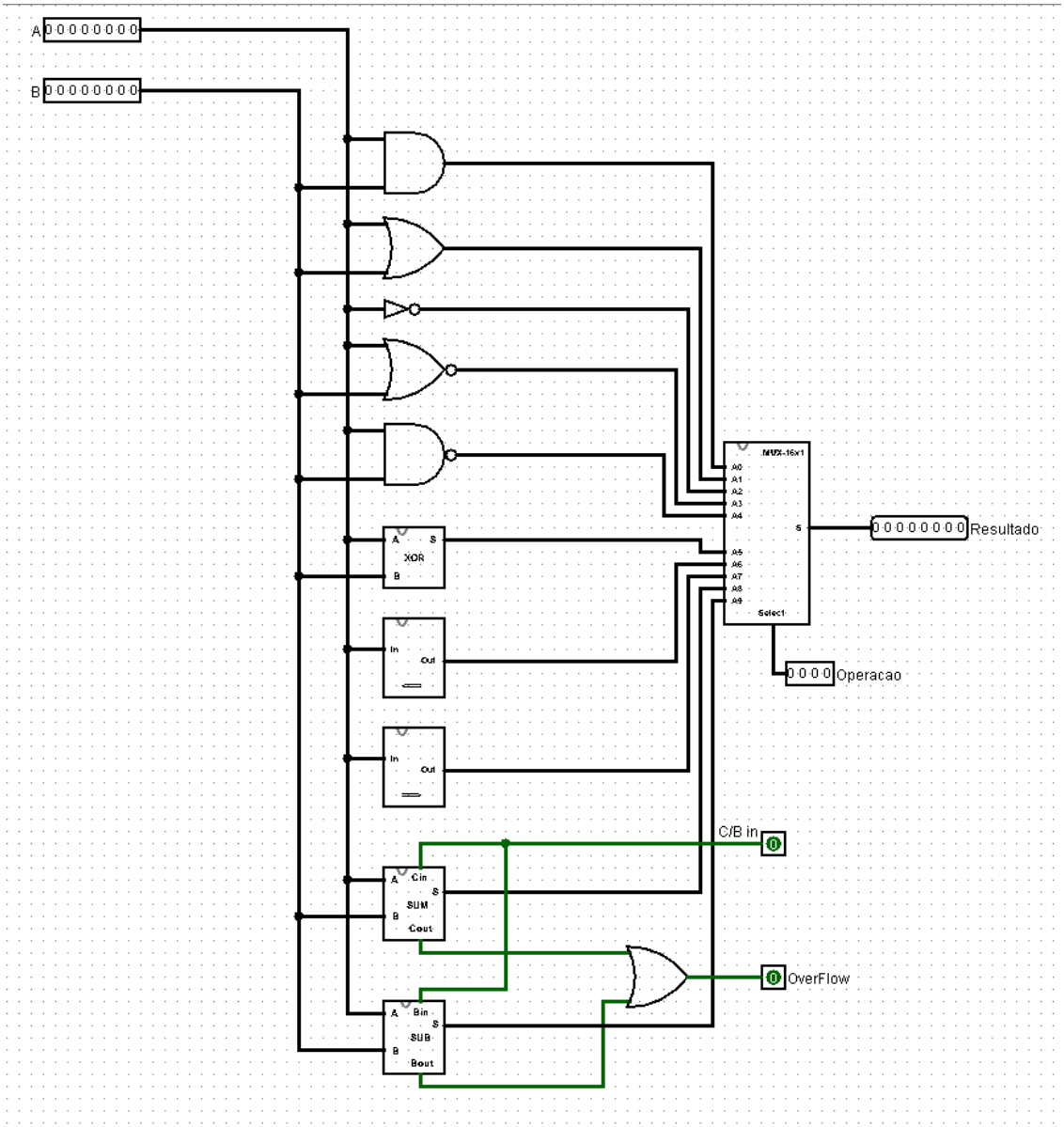


Após 3 pulsos de clock, já é possível obter um resultado sobre os últimos 3 valores da entrada. Deve se notar que antes de 3 pulsos de clock é impossível obter um resultado válido para o detector de sequência, visto que os registradores armazenam zero em seus estados iniciais. Dessa forma o circuito necessita de no mínimo 3 ciclos de clock para avaliar adequadamente a sequência de entrada.

2.10. Unidade Lógica Aritmética de 8 Bits

Uma Unidade Lógica e Aritmética (ULA) é um componente fundamental em sistemas digitais, responsável por realizar operações lógicas e aritméticas sobre números representados em circuitos lógicos. Geralmente, uma ULA recebe dois operandos como entradas e possui uma entrada de controle adicional, que especifica qual operação deve ser executada. Essas operações podem incluir somas, subtrações, multiplicações, divisões, além de operações lógicas como AND, OR, XOR, entre outras. A ULA desempenha um papel crucial no processamento de dados, sendo utilizada em processadores, controladores e outros dispositivos eletrônicos.

Figura 42 - Unidade Lógica Aritmética



A imagem logo acima exibe a implementação de uma ULA no Logisim, configurada para realiza um total de 10 operações que podem ser acessadas a partir dos seguintes códigos binários mostrados na tabela abaixo.

Tabela 5 - Códigos de Operações da ULA

Código	Operação
0000	AND bit a bit
0001	OR bit a bit
0010	NOT bit a bit
0011	NOR bit a bit
0100	NAND bit a bit

0101	XOR bit a bit
0110	SHIFT à Esquerda
0111	SHIFT à Direita
1000	Soma
1001	Subtração

Essa ULA foi desenvolvida utilizando componentes padrões do Logisim, incluindo portas lógicas básicas, como a porta XOR previamente descrita no item 2.3 deste relatório, além de deslocadores de bits à direita e à esquerda, um somador de 8 bits mencionado no item 2.8 e um subtrator de 8 bits. O resultado de cada operação é direcionado para um multiplexador 16x1 de 8 bits, também já detalhado anteriormente no item 2.7.2. Esse multiplexador é responsável por selecionar, por meio de um código binário de 4 bits (conforme especificado na tabela 5), qual resultado será exibido na saída.

Essa ULA ainda possui 2 de entrada de 8 bits (A e B) sobre os quais as operações são realizadas, 1 entrada de 4 bits para a seleção das operações e uma saída 8 bits que fornece o resultado.

Nos próximos tópicos, serão detalhadas as implementações dos subcircuitos responsáveis pelos deslocadores de bits e pelo subtrator presentes na ULA.

2.10.1. Deslocadores

Figura 43 - Deslocador de 1 Bit à Esquerda

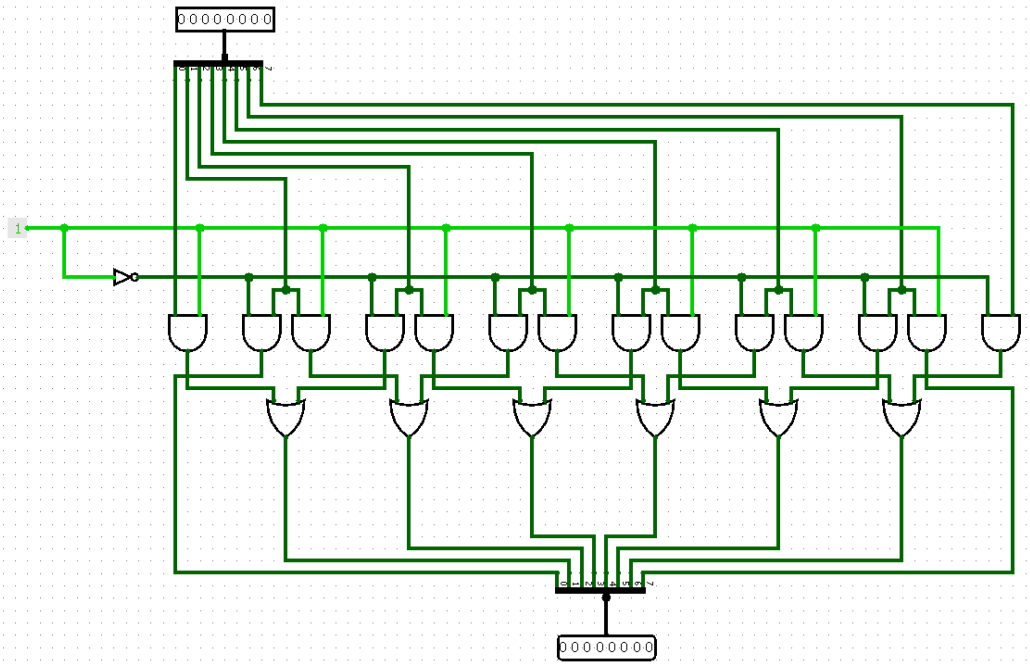
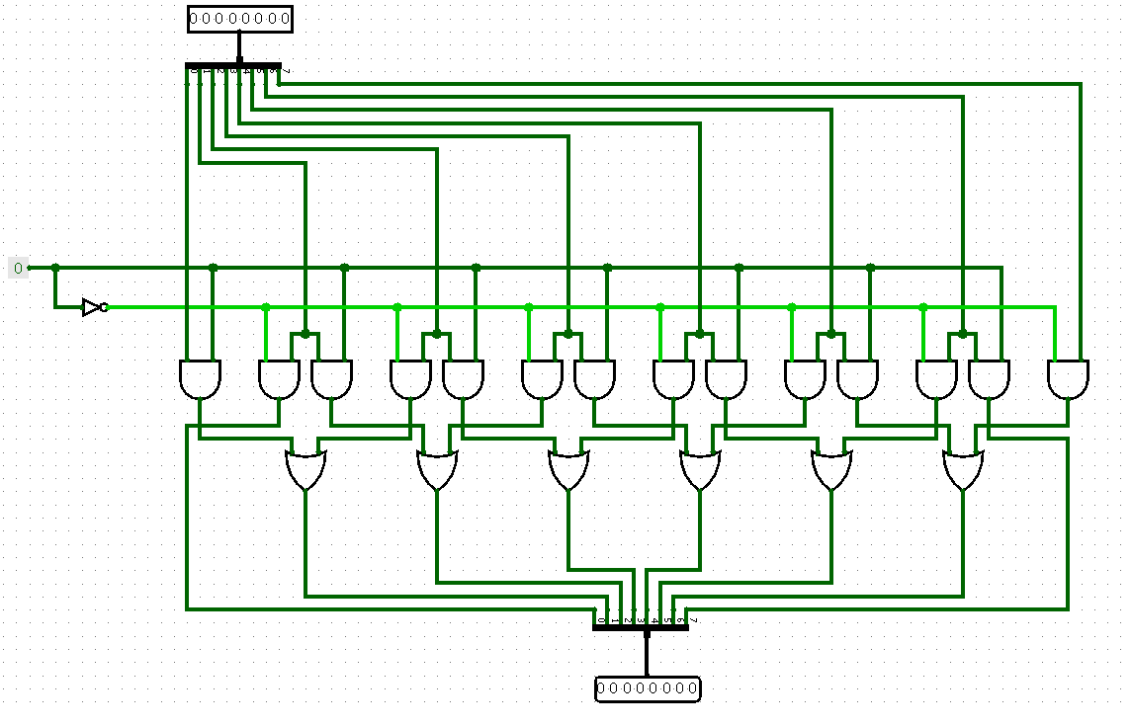


Figura 44 - Deslocador de 1 Bit à Direita



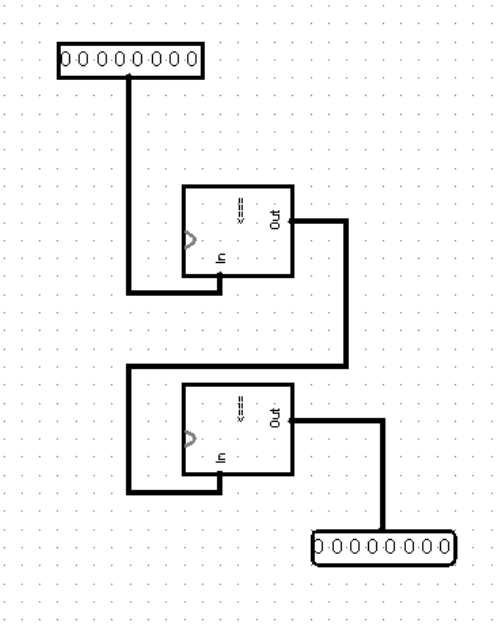
Nas últimas duas figuras, é possível observar a implementação dos deslocadores de bit para a esquerda e para a direita com deslocamento de 1 bit. Esses circuitos são estruturalmente semelhantes, diferenciando-se apenas pelo comportamento controlado por uma constante de controle.

Quando a constante de controle é igual a 1, o circuito opera como um deslocador para a esquerda, movendo todos os bits para posições de maior peso e inserindo 0 na posição menos significativa (LSB).

Quando a constante de controle é igual a 0, o circuito atua como um deslocador para a direita, movendo os bits para posições de menor peso e preenchendo a posição mais significativa (MSB) com 0.

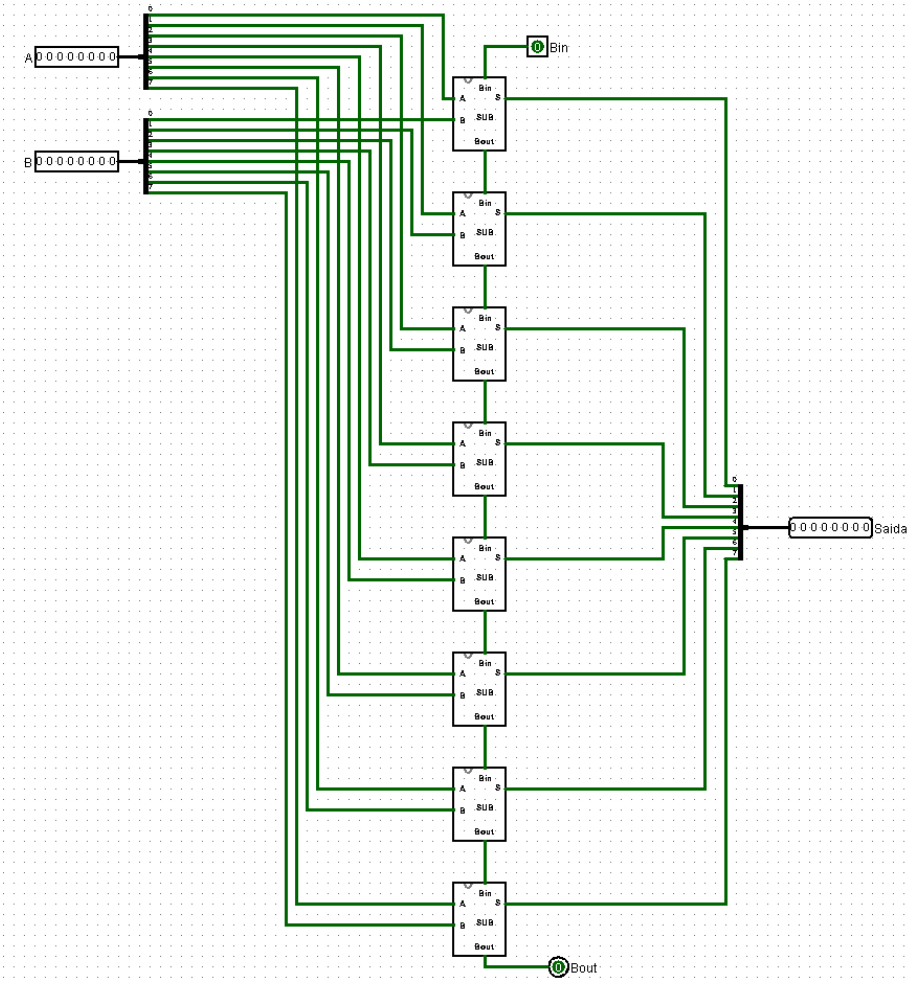
Portanto, para implementar um circuito deslocador de 2 bits, é suficiente conectar dois componentes deslocadores de 1 bit que operem na mesma direção, seja para a esquerda ou para a direita, como ilustrado na imagem abaixo.

Figura 45 - Deslocador de 2 Bit à Esquerda



2.10.2. Subtrator de 8 bits

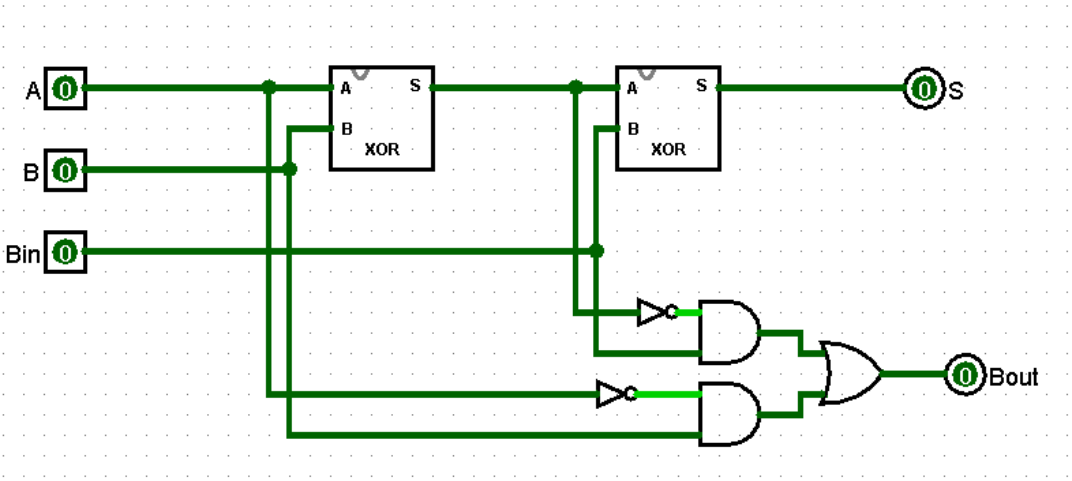
Figura 46 – Subtrator de 8 Bits



Um circuito subtrator de 8 bits é construído a partir da combinação de 8 subtratores de 1 bit conectados em cascata. Cada subtrator de 1 bit realiza a operação de subtração bit a bit, levando em consideração o borrow (empréstimo) do bit menos significativo.

Para implementar o subtrator de 1 bit, utiliza-se a base de um somador de 1 bit, modificando a expressão de saída de Carry-Out. A nova expressão que define o comportamento do subtrator é ~A*B+~(A xor B) * Cin, onde Carry In será chamado de Borrow In no subtrator

Figura 47 – Subtrator de 1 Bit

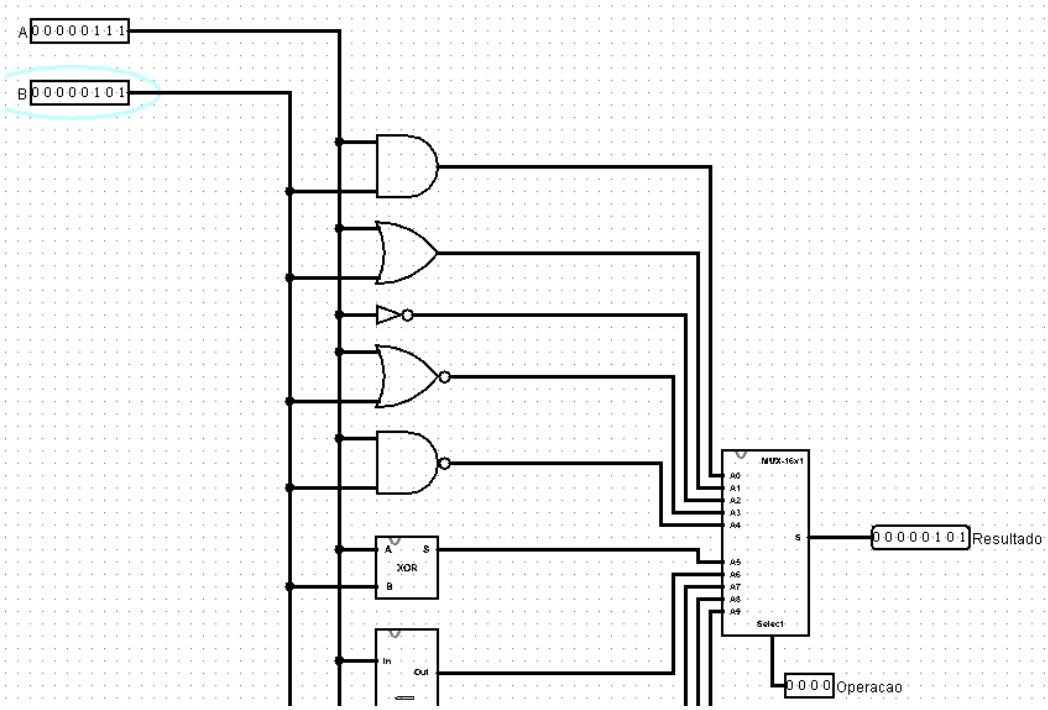


A seguir são implementados alguns casos de teste sobre a ULA.

2.10.3. Caso 0000

No caso em que o seletor de operações da ULA é configurado com o código 0000, o resultado exibido na saída corresponde à operação AND bit a bit realizada entre os dois operandos de entrada (A e B). Essa operação avalia cada par de bits dos operandos, aplicando a lógica AND, em que a saída é 1 somente quando ambos os bits de entrada são 1.

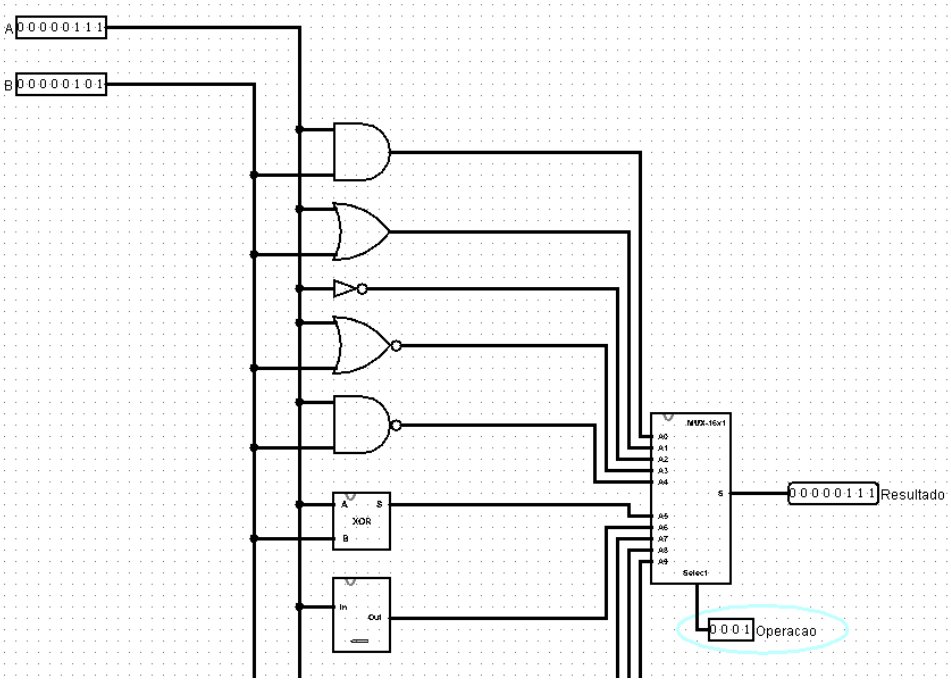
Figura 48 – ULA Caso 0000



2.10.4. Caso 0001

Quando o seletor de operações da ULA é configurado com o código 0001, a operação OR bit a bit é realizado entre as entradas e o resultado é mostrado na saída. Nesta operação, cada par de bits dos operandos é avaliado utilizando a lógica OR, em que a saída é 1 se pelo menos um dos bits de entrada for 1.

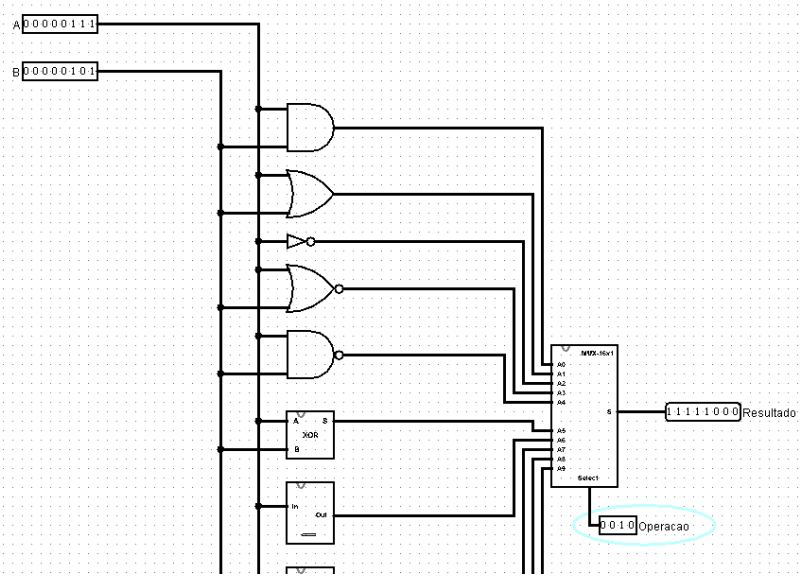
Figura 49 – ULA Caso 0001



2.10.5. Caso 0010

Quando o seletor de operações da ULA está configurado com o código 0010, a operação realizada é a NOT bit a bit aplicada ao operando A. Na operação NOT, cada bit do operando de entrada A é invertido.

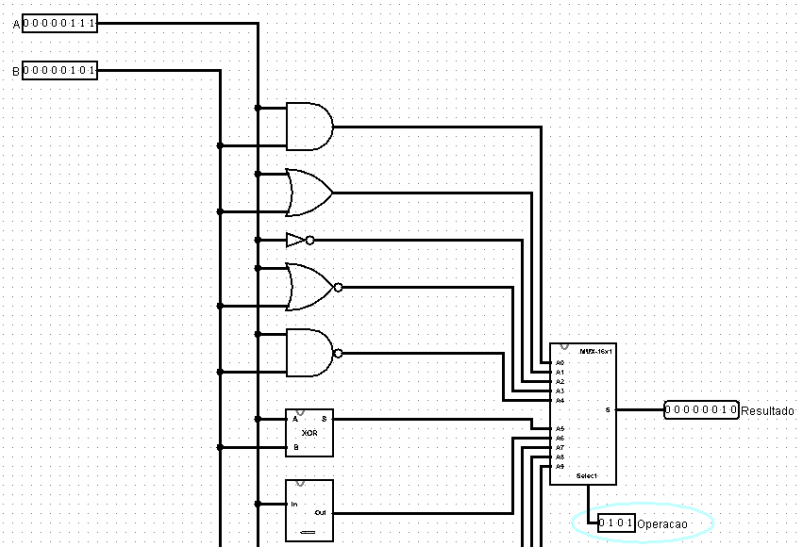
Figura 50 – ULA Caso 0010



2.10.6. Caso 0101

Quando o seletor de operações da ULA está configurado com o código 0101, a operação realizada é a XOR bit a bit entre os dois operandos de entrada (A e B). Na operação XOR (OU exclusivo), a saída para cada par de bits dos operandos A e B é 1 se os bits forem diferentes, e 0 se forem iguais.

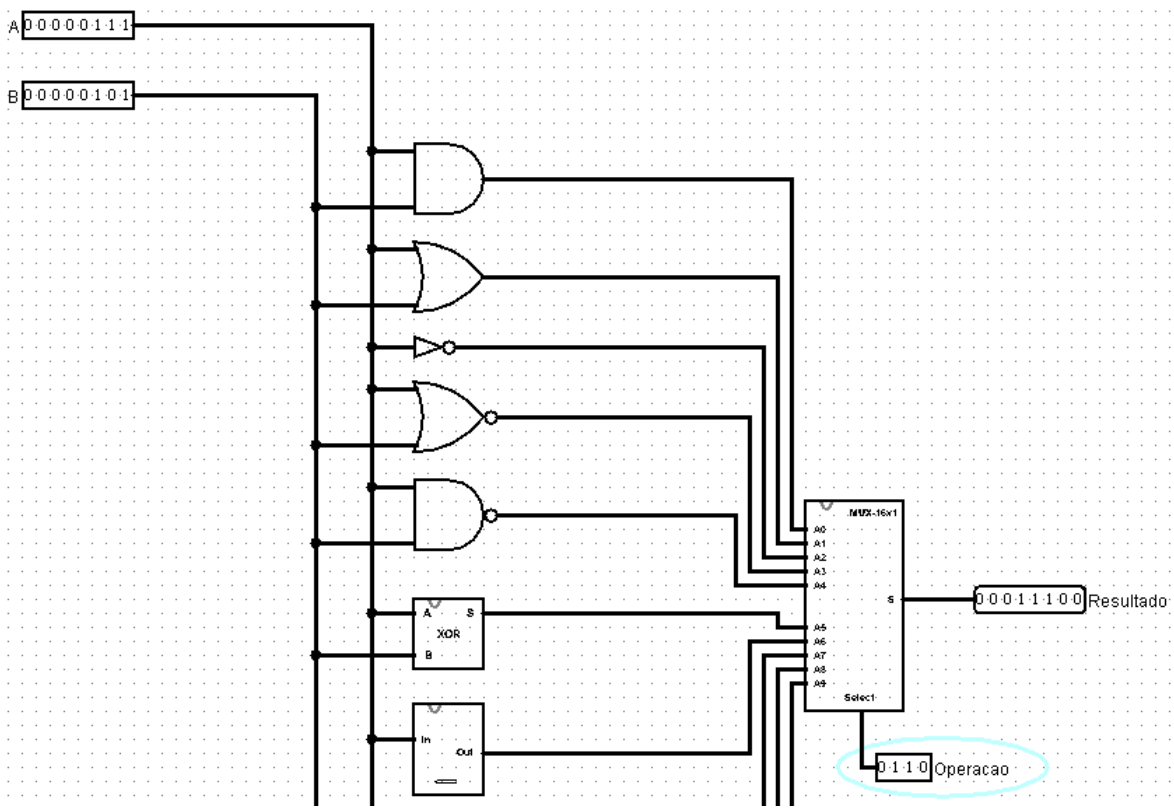
Figura 51 – ULA Caso 0101



2.10.7. Caso 0110

Quando o seletor de operações da ULA está configurado com o código 0110, a operação realizada é o deslocamento de 2 bits à esquerda. Neste caso, os bits do operando A são deslocados duas posições para a esquerda, e, como resultado, os dois bits menos significativos (à direita) são preenchidos com zeros. Essa operação é equivalente a multiplicar o número binário por 4.

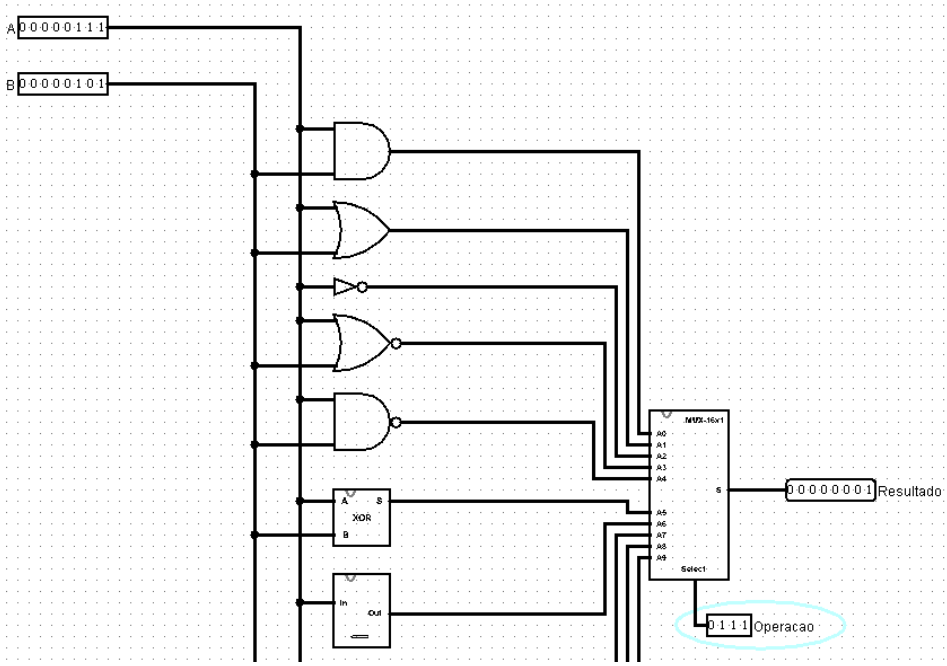
Figura 52 – ULA Caso 0110



2.10.8. Caso 0111

Quando o seletor de operações da ULA está configurado com o código 0111, a operação realizada é o deslocamento de 2 bits à direita. Nesse caso, os bits do operando A são deslocados duas posições para a direita, e os dois bits mais significativos (à esquerda) são preenchidos com zeros. Esse tipo de operação é equivalente a dividir o número binário por 4 (deslocamento aritmético para a direita).

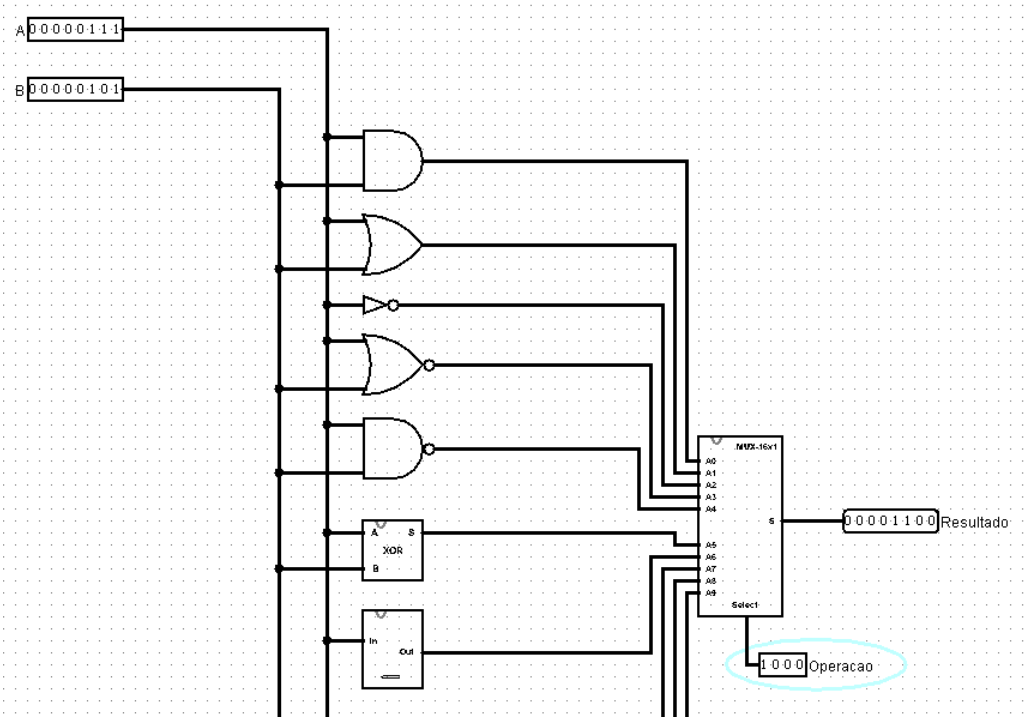
Figura 53 – ULA Caso 0111



2.10.9. Caso 1000

Quando o seletor de operações da ULA está configurado com o código 1000, a operação realizada é a soma dos dois operandos de entrada (A e B). Essa operação é realizada bit a bit utilizando um somador binário, onde os bits correspondentes de A e B são somados, levando em consideração o carry (vai-um) gerado em cada etapa.

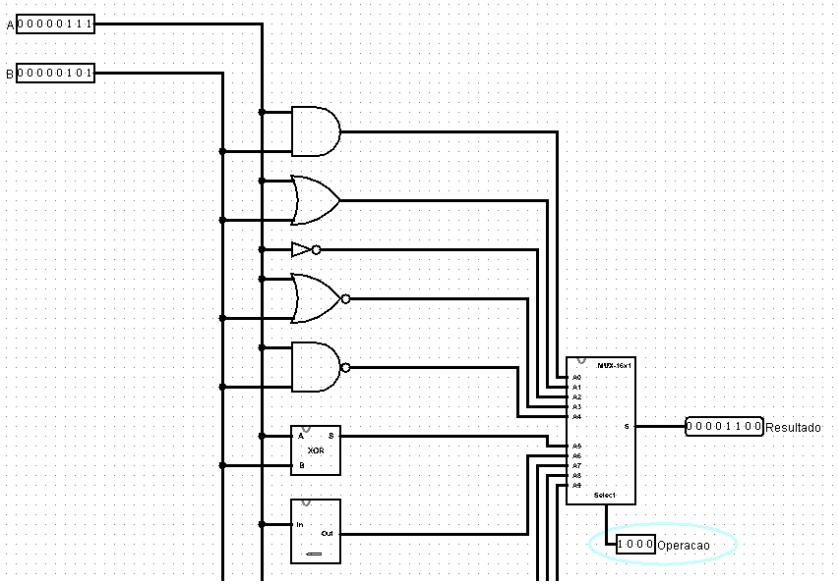
Figura 54 – ULA Caso 1000



2.10.10. Caso 1001

Quando o seletor de operações da ULA está configurado com o código 1001 a operação de subtração entre os operandos A e B é realizada diretamente com base na lógica de subtração binária.

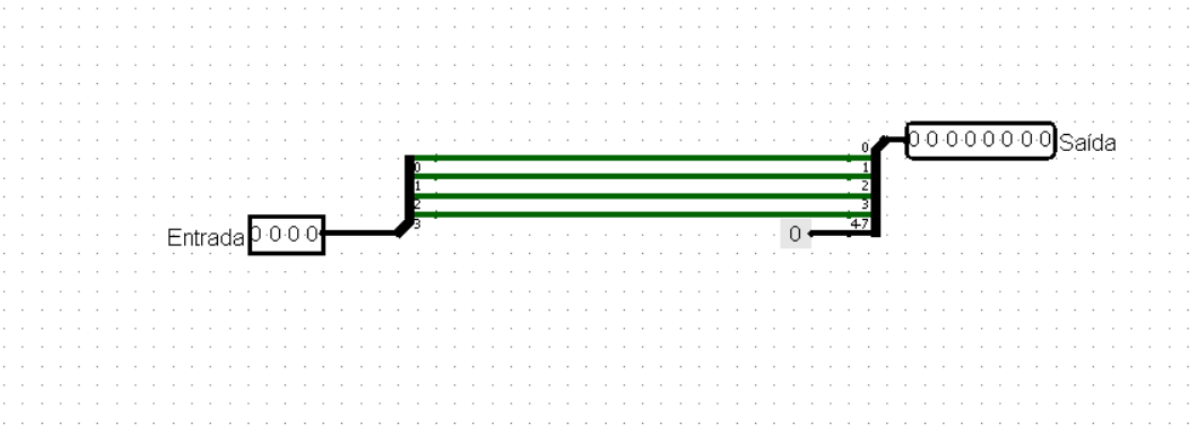
Figura 55 – ULA Caso 1001



2.11. Extensor de sinal de 4 bits para 8 bits

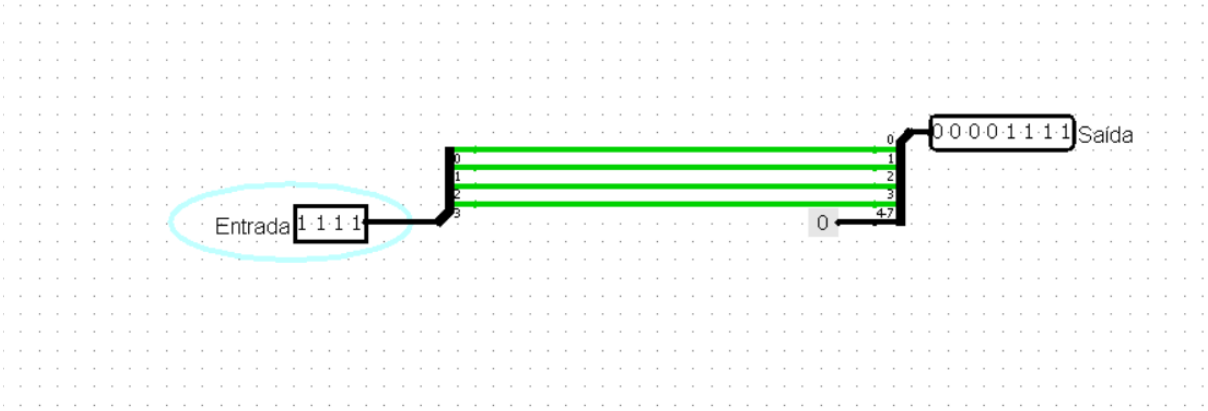
Um extensor de sinal é um circuito digital utilizado para ampliar o número de bits de um valor binário, mantendo a precisão do número em termos de sinal (positivo ou negativo). O extensor de sinal de 4 bits para 8 bits é um circuito que transforma um número de 4 bits em um número de 8 bits, replicando o bit mais significativo (o bit de sinal) para os 4 bits mais significativos da saída, garantindo que o número continue com o mesmo valor.

Figura 56 - Extensor de sinal de 4 bits para 8 bits



O circuito acima representa um extensor de sinal de 4 bits para 8 bits, composto por uma entrada de 4 bits, 2 distribuidores e uma saída de 8 bits, onde bits menos significativos na posição 0-3 no distribuidor de entrada recebem a entrada de 4 bits e consequentemente vão para mesma posição no distribuidor da saída, onde os outros 4 bits recebem o valor 0 da contante, esse distribuidor trabalha apenas com valores positivos.

Figura 57 - Extensor de sinal de 4 bits para 8 bits entrada 1111(2)



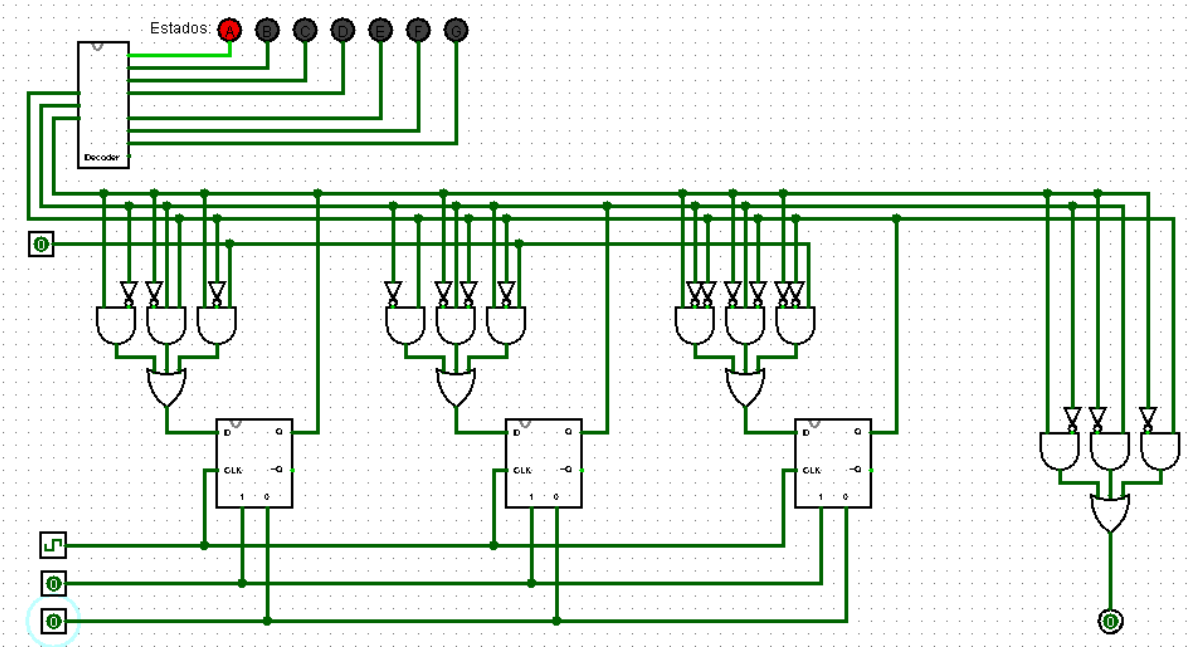
Na imagem acima vemos uma entrada de 4 bits, essa entrada sendo 1111(2), 15 em decimal, e saída sendo 00001111(2), que também é 15 em decimal, pelo uso do extensor agora seria possível essa entrada fazer uma soma, no somador de 8 bits criado no laboratório de circuitos, logo é percebido que o extensor é útil para deixar na mesma arquitetura do circuito que deseja utilizar.

2.12. Máquina de Estados

O termo máquina de estados refere-se a um circuito digital que passa por uma sequência de estados predeterminados, controlados por um sinal de clock e outros sinais de entrada. Em outras palavras, uma máquina de estados é projetada para mudar seu comportamento ou estado com base em condições específicas, sendo frequentemente utilizada em sistemas de controle, processamento de sinais e circuitos lógicos sequenciais.

Sabendo disso, a seguir é mostrado um circuito que descreve o seguinte problema. Dada uma entrada X e uma saída Y, quando X passar de 0 para 1, Y deves ser igual a 1 por 5 pulsos de clock e então retorne a 0, mesmo que X ainda seja igual 1.

Figura 58 – Máquina de Estados



Para um melhor entendimento do problema e para visualizar claramente a máquina de estados que deveria ser implementada, foi feito um diagrama de transição de estados descrito na tabela abaixo.

Tabela 6 - Diagrama de Transição de Estados

Estado Atual	Entrada X	Estado Próximo	Saída Y
A	0	A	0
A	1	B	0
B	0 ou1	C	1
C	0 ou 1	D	1
D	0 ou 1	E	1
E	0 ou 1	F	1
F	0 ou 1	G	1
G	0	A	0
G	1	G	0

Com base no diagrama apresentado, é possível observar que a máquina de estados deve ser composta por 7 estados. O estado A é o estado inicial, onde X é igual a 0. Os estados B a F, a saída Y permanece igual a 1, após X passar de 0 para 1. O estado G é o estado de espera, onde a máquina deve permanecer caso X ainda seja igual a 1, até que ocorra uma transição para o estado A, caso X se torne 0 novamente.

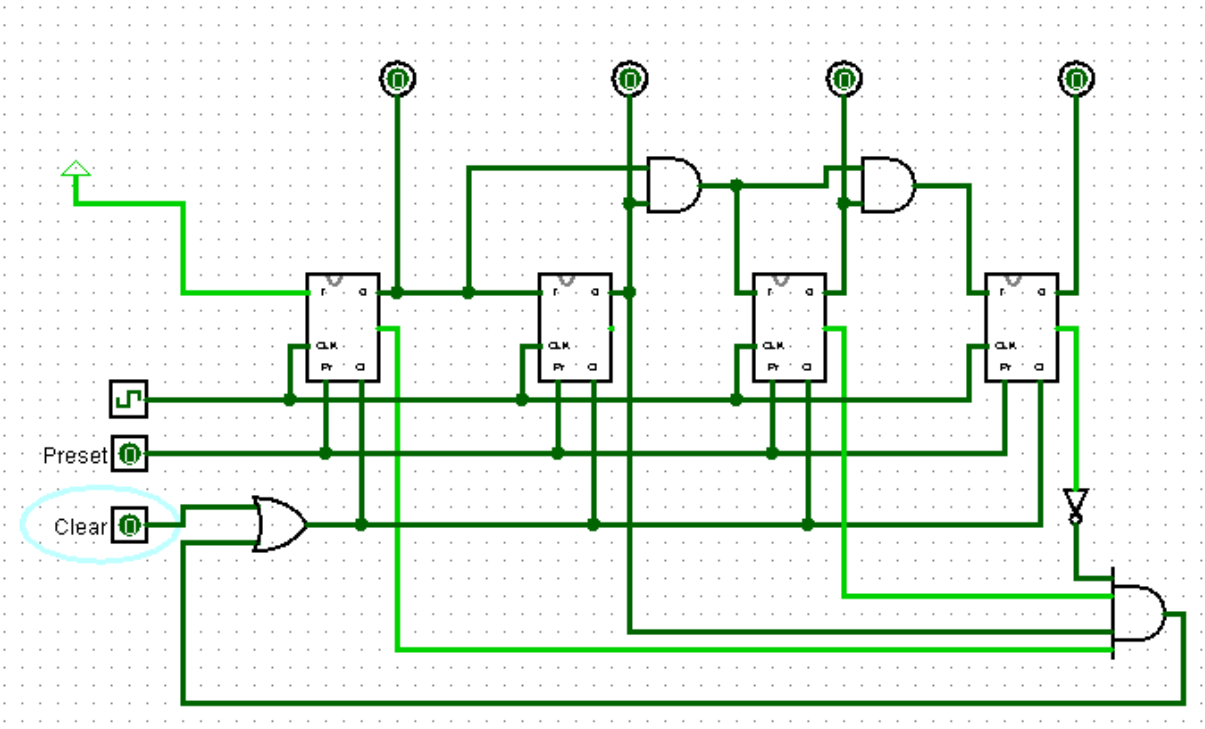
Com essas informações já é possível descrever a máquina de estados implementada, que possui 3 flip-flops do tipo D com entradas clear e preset apresentados no item 2.1.1 deste relatório, uma entrada X e uma saída Y. Com base em algumas expressões que são obtidas utilizando uma tabela de transição de

estados e uso mapas de Karnaugh foi possível chegar a expressões que descrevem a passagem de estados da máquina e sua saída.

2.13. Contador Síncrono

Um contador síncrono é um circuito digital composto por flip-flops conectados em paralelo, de forma que todas as entradas de clock estejam ligadas à mesma fonte. Esses contadores são circuitos sequenciais capazes de realizar contagens binárias de maneira controlada, pois contam com um circuito combinacional externo. Esse circuito utiliza como entradas as saídas Q e Q′ de cada flip-flop, e suas saídas são conectadas às entradas dos flip-flops utilizados, permitindo a contagem síncrona.

Figura 59 - Contador Síncrono



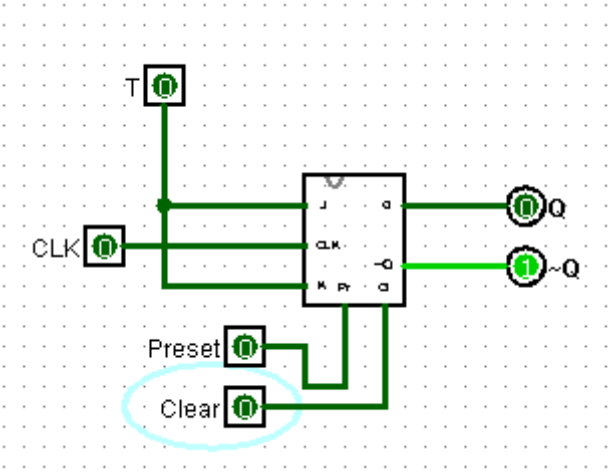
A imagem acima apresenta a implementação de um contador síncrono década, construído a partir de 4 flip-flops tipo T, os quais compartilham uma entrada de clock em comum. Dessa forma, esse tipo de contador elimina o problema de acúmulo de atraso de propagação entre os flip-flops existente em contadores assíncronos, já que todos respondem ao mesmo pulso de clock. As mudanças de estado de cada flip-flop ocorrem de forma coordenada, sendo diferenciadas apenas pelo momento em que cada flip-flop deve alterar seu estado.

2.13.1. Flip-Flop tipo T

O flip-flop tipo T é uma versão simplificada do flip-flop JK apresentado no item 2.1.2. Ele pode ser construído a partir de um flip-flop JK, conectando a entrada T às

entradas J e K do flip-flop JK, de forma que ambas recebam o mesmo valor como mostra a imagem abaixo.

Figura 60 - Flip-Flop T



Neste flip-flop quando a entrada T é 0, o estado do flip-flop não muda, ou seja, ele mantém o valor armazenado e quando igual a 1, o flip-flop alterna entre os estados 0 e 1 a cada pulso de clock, ou seja, ele troca de estado.

2.13.2. Casos de Testes

A seguir, são apresentadas as imagens que ilustram o funcionamento do circuito contador implementado. Essas imagens demonstram como o contador síncrono década, utilizando os 4 flip-flops tipo T, realiza a contagem binária de forma coordenada, com todos os flip-flops respondendo ao mesmo pulso de clock, sem o acúmulo de atraso de propagação entre eles.

Figura 61 - Contador Síncrono 0001

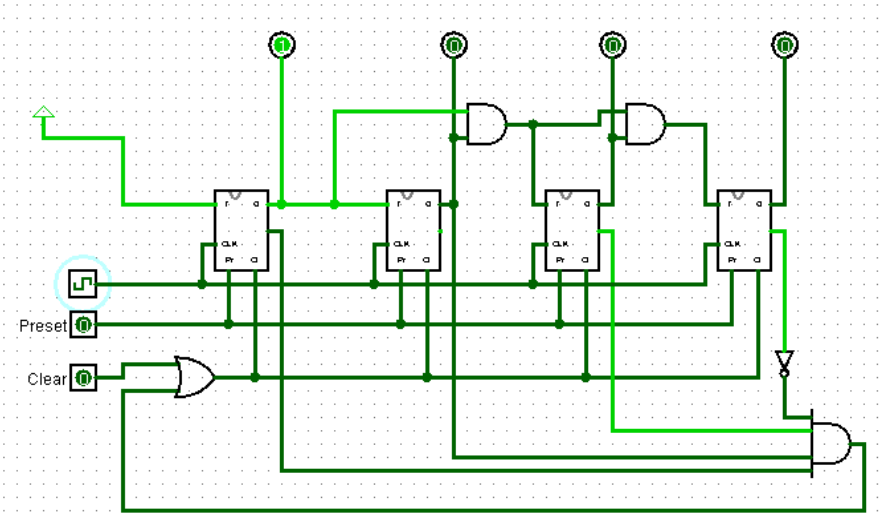


Figura 62 - Contador Síncrono 0010

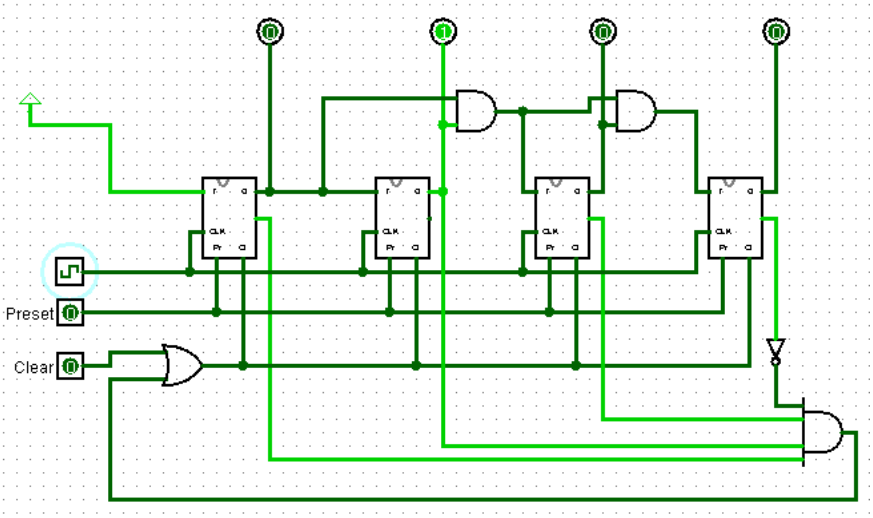


Figura 63 - Contador Síncrono 0011

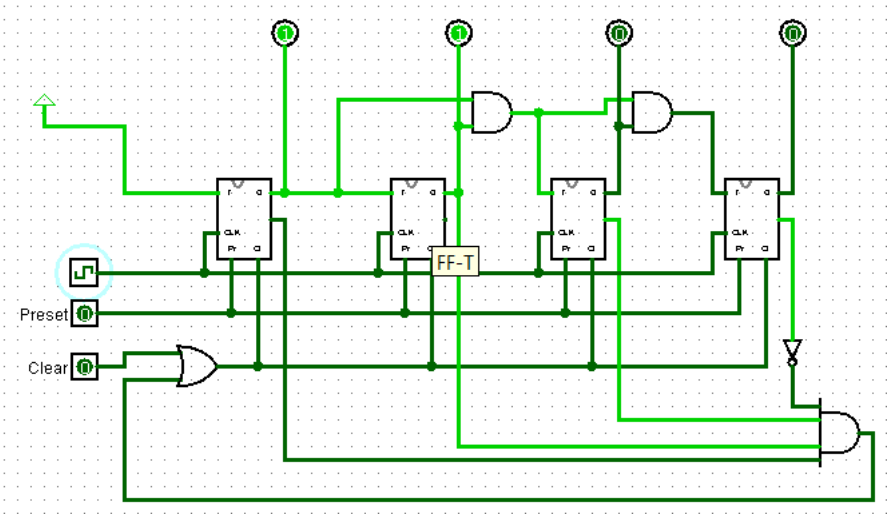


Figura 64 - Contador Síncrono 0100

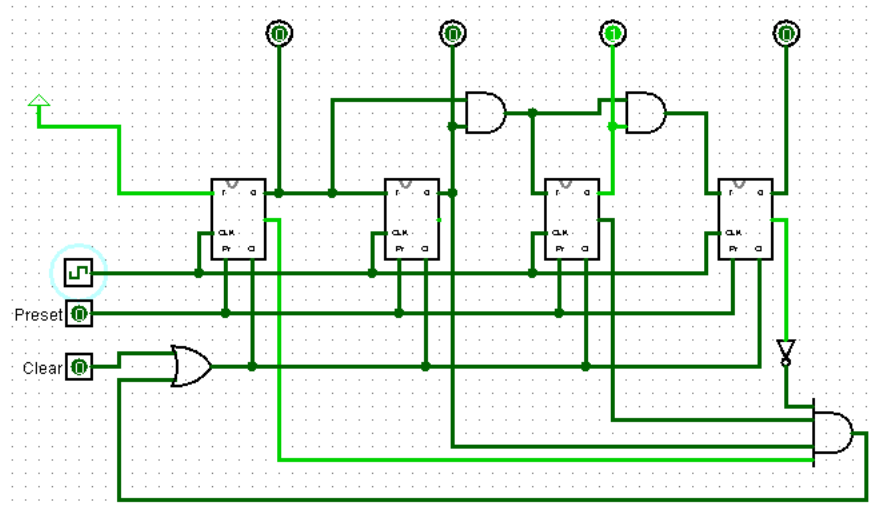


Figura 65 - Contador Síncrono 0101

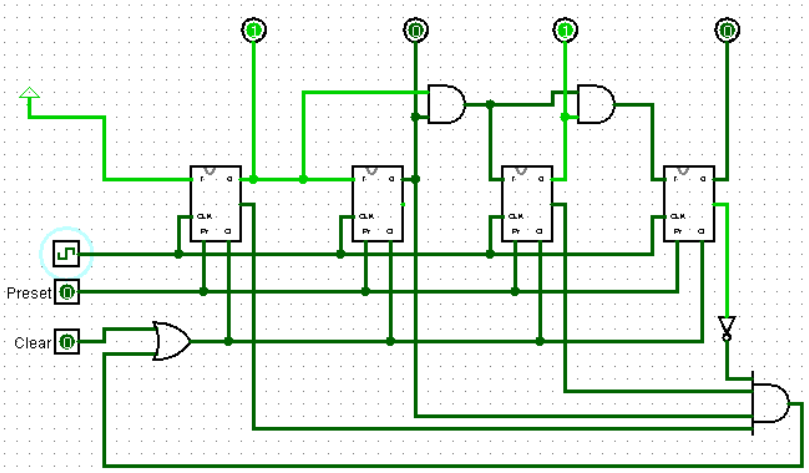


Figura 66 - Contador Síncrono 0110

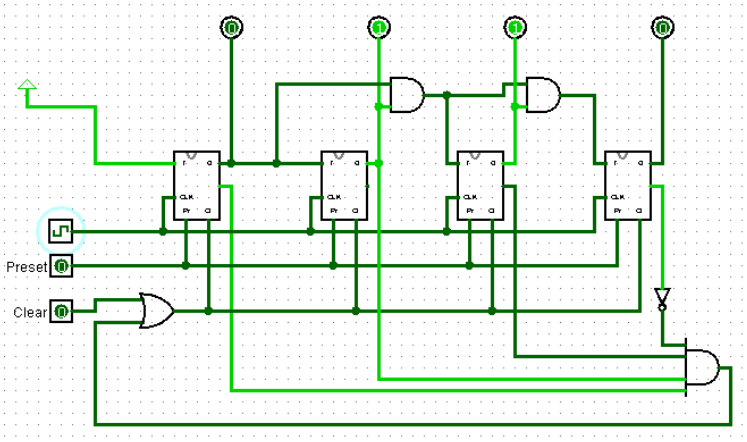


Figura 67 - Contador Síncrono 0111

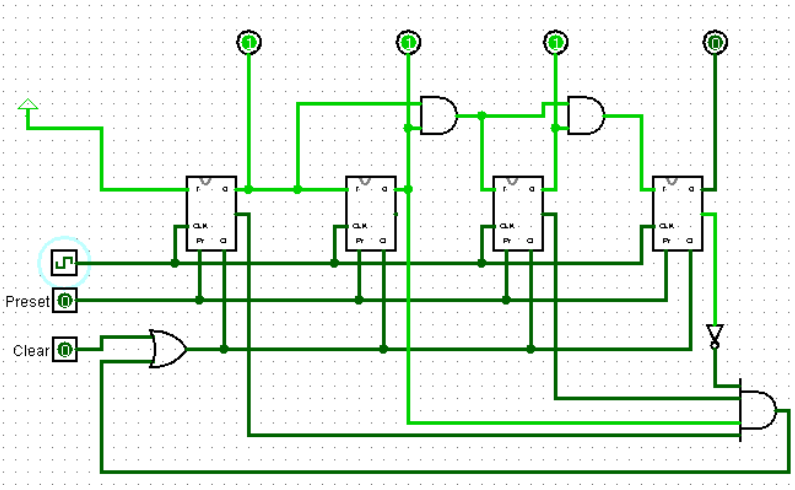


Figura 68 - Contador Síncrono 1000

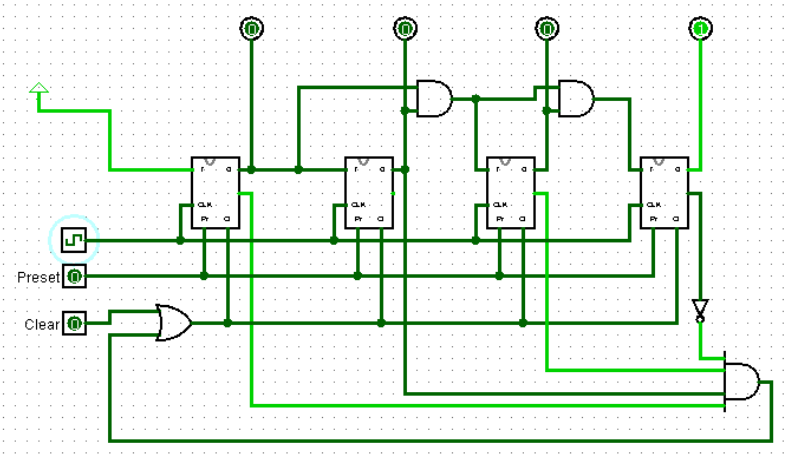
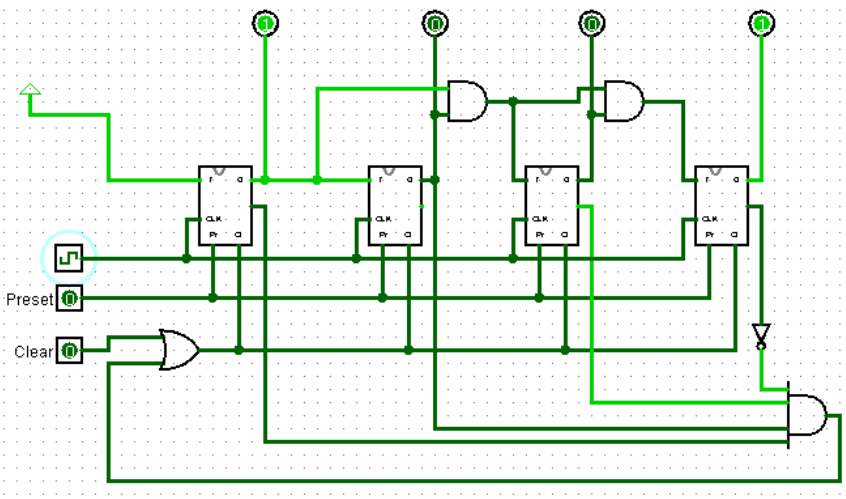


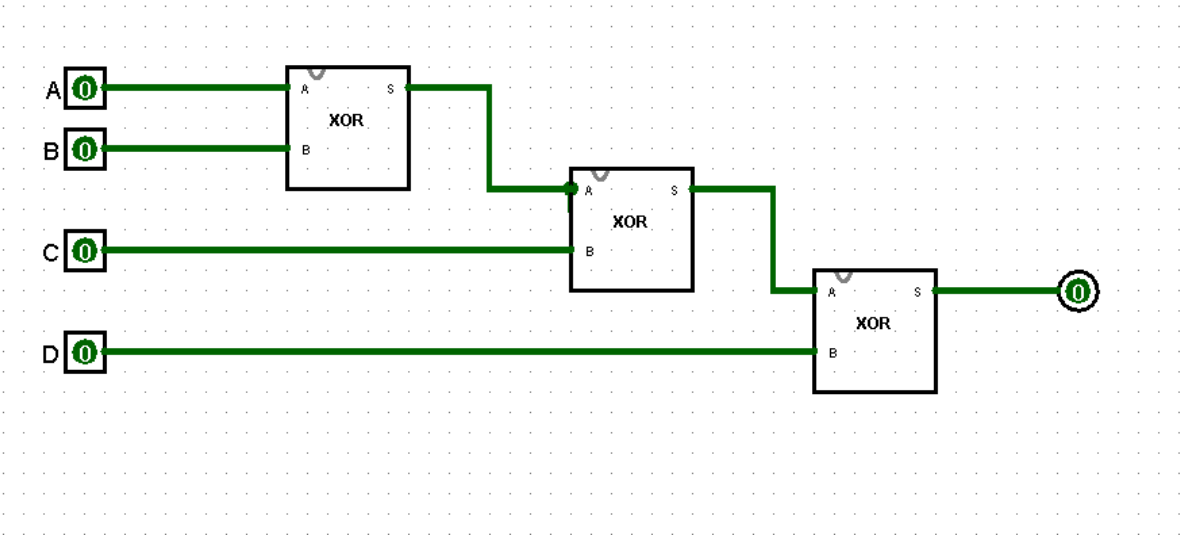
Figura 69 - Contador Síncrono 1001



2.14. Detector de Paridade Ímpar

Um circuito detector de paridade ímpar é um circuito digital utilizado para verificar se a quantidade de bits com valor 1 em um conjunto de dados binários é ímpar. Ele funciona aplicando a operação lógica XOR entre os bits de entrada, pois a operação XOR resulta em 1 quando a quantidade de entradas iguais a 1 é ímpar.

Figura 70 – Detector de Paridade Ímpar

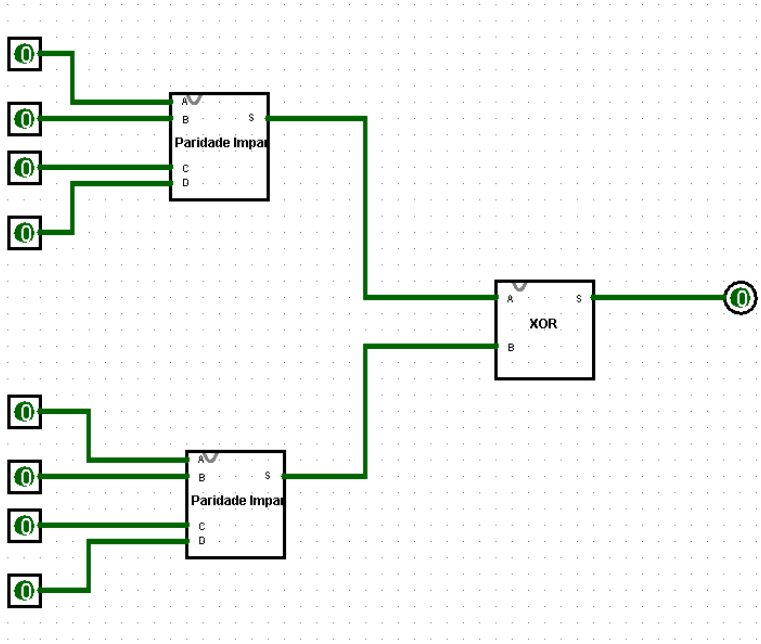


A imagem acima apresenta a implementação de um detector de paridade ímpar de 4 entradas, utilizando portas lógicas XOR, conforme descrito anteriormente neste relatório. O circuito possui uma saída de 1 bit, que indica o resultado da detecção: 1 caso seja identificada uma paridade ímpar e 0 caso contrário.

A lógica do circuito baseia-se na verificação de duas entradas por vez, utilizando a porta XOR, que retorna um valor verdadeiro (1) apenas quando as entradas são diferentes. Dessa forma, o circuito acumula a detecção de paridade para todas as entradas.

Para expandir esse circuito para um maior número de entradas, pode-se utilizar duas implementações do detector de 4 entradas, combinando suas saídas com uma porta XOR adicional. Esse método mantém a lógica e garante a detecção de paridade ímpar para conjuntos maiores de entradas, como observado na imagem a seguir.

Figura 71 – Detector de Paridade Ímpar de 8 Entradas



2.14.1. Casos de Testes

A seguir, são apresentadas as imagens que comprovam a eficiência do circuito detector de paridade ímpar implementado.

Figura 72 – Detector de Paridade Ímpar Teste 1

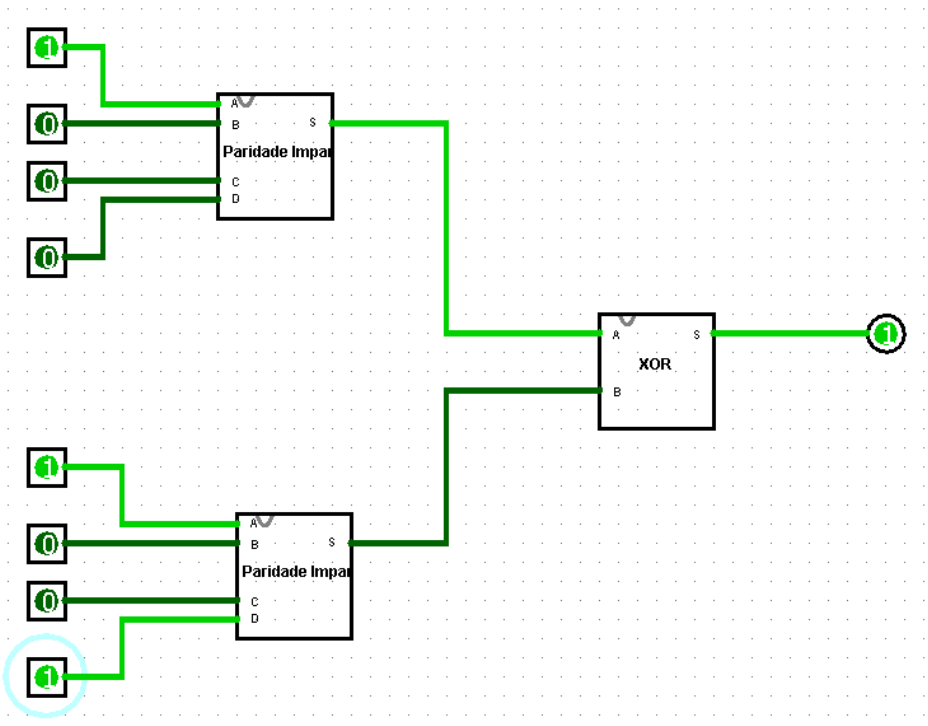
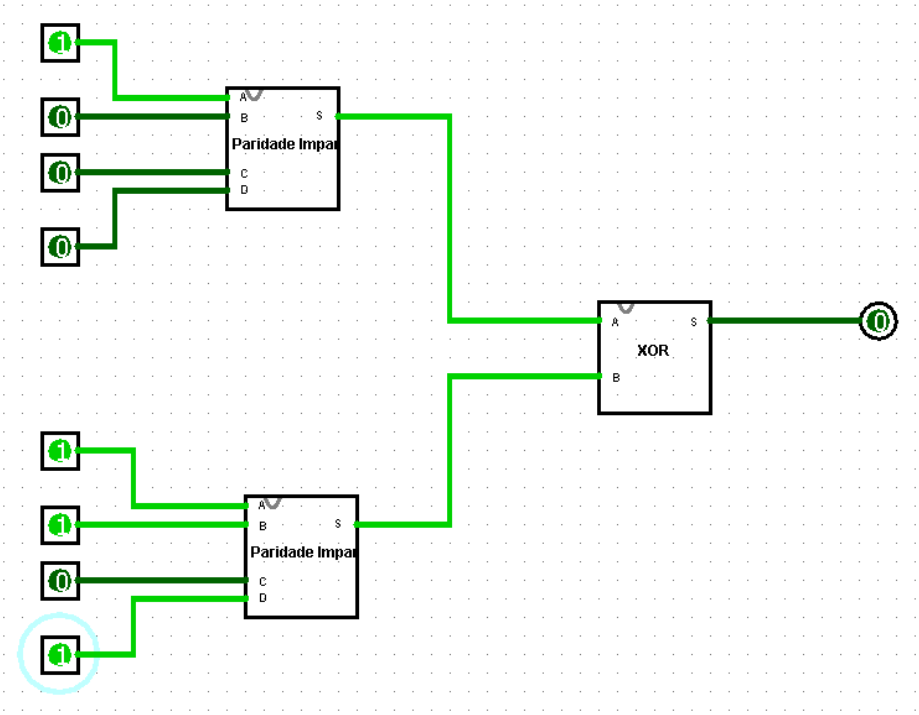


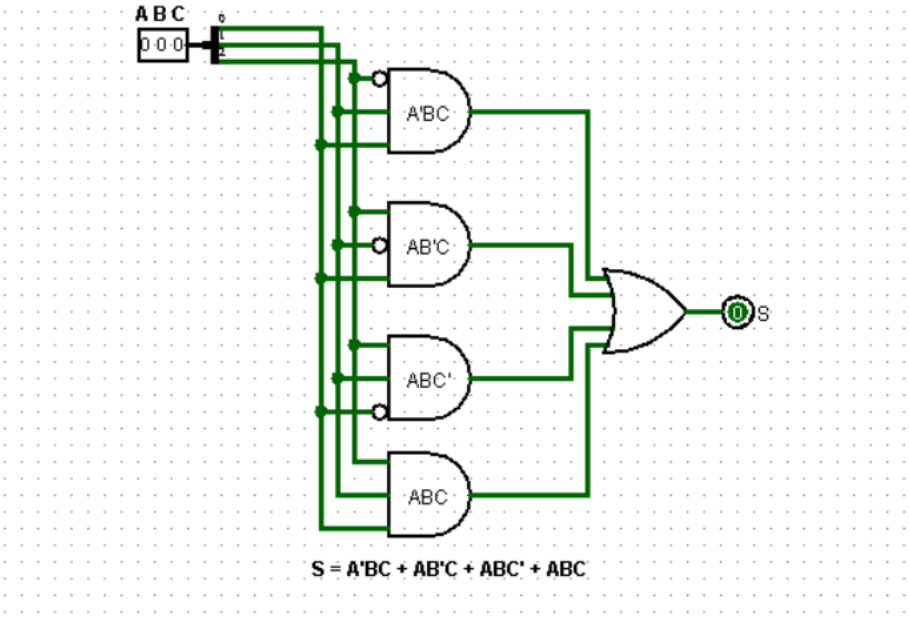
Figura 73 – Detector de Paridade Ímpar Teste 2



2.15. Otimização de circuito usando mapa de Karnaugh

O mapa de Karnaugh é uma ferramenta gráfica utilizada para simplificar expressões booleanas e, consequentemente, otimizar circuitos digitais. Com isso a construção de um circuito se torna mais fácil e com menos elementos.

Figura 74 - Circuito não otimizado

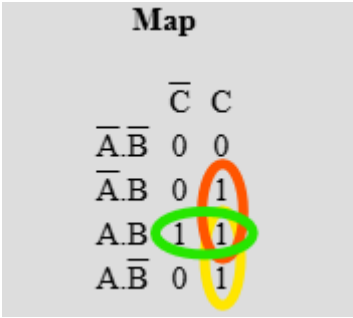


O circuito acima é composto por uma entrada de 3 bits, onde cada bit tem sua respectiva entrada no AND, onde tem quatro AND, três NOT, um OR e uma saída S, sua saída S = 1, quando 2 dos 3 bits são 1. Como podemos ver a posição 0 na entrada de 3 bits equivale a C, a posição 1 equivale a b e a posição 2 equivale a A.

Tabela 7 - Tabela verdade do circuito não otimizado

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

Figura 75 - Mapa de Karnaugh



Com a otimização usando o mapa de Karnaugh a tabela verdade e a expressão booleana foi bem simplificada, fazendo a construção do circuito ser mais simples e com menos componentes.

Figura 76 - Circuito otimizado com mapa de Karnaugh

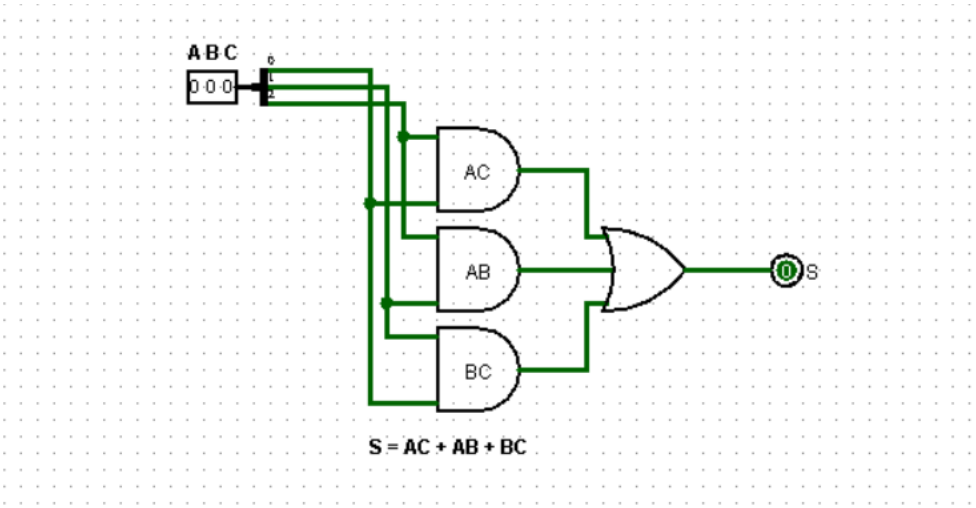


Tabela 8 - Tabela verdade do circuito otimizado com mapa de Karnaugh

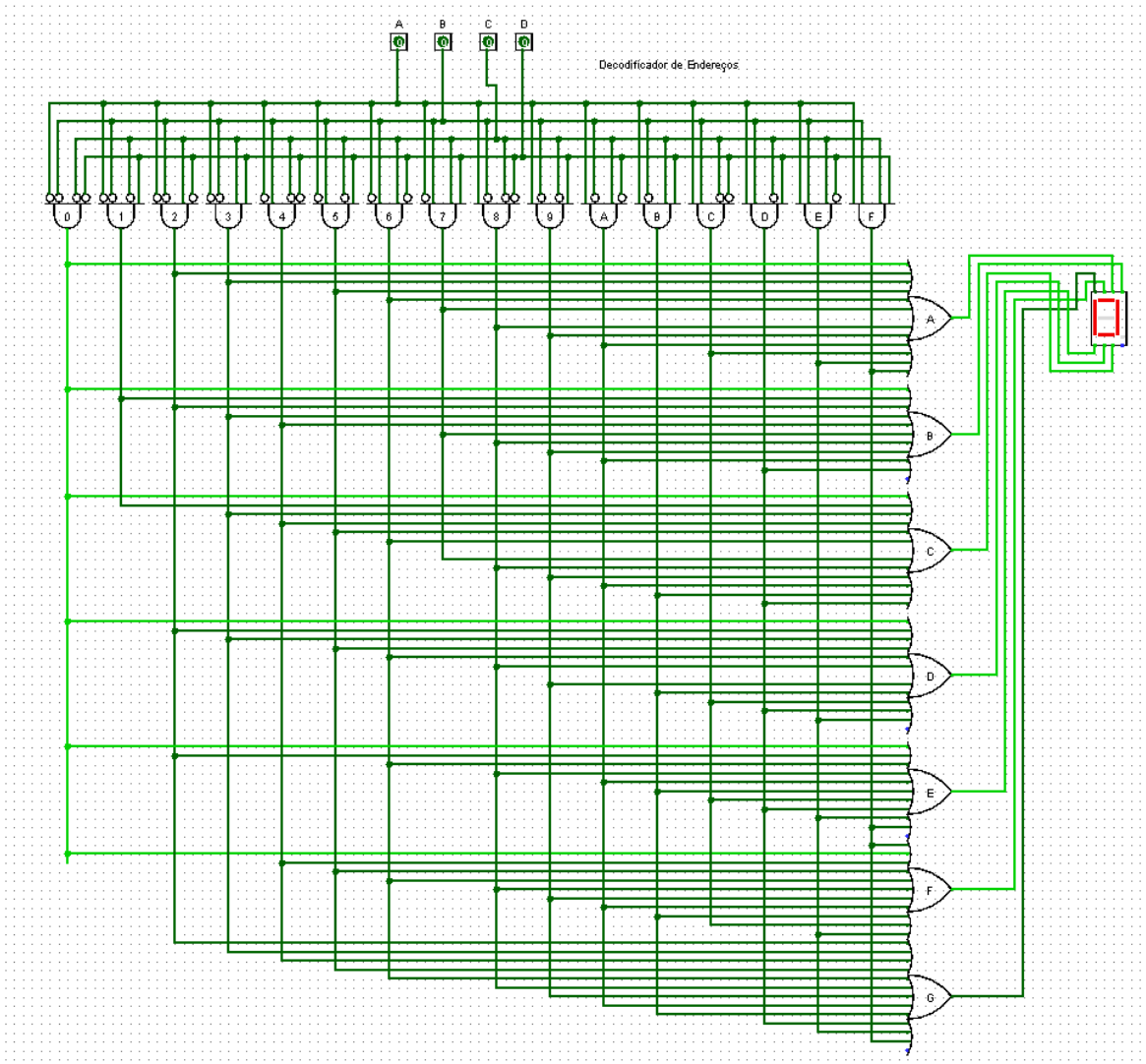
A	B	C	S
1	0	1	1
1	1	0	1
0	1	1	1

2.16. Decodificador de 7 Segmentos

O decodificador de 7 segmentos de 4 entradas é um dispositivo eletrônico utilizado para converter um número binário em sinais que acionam um display de 7

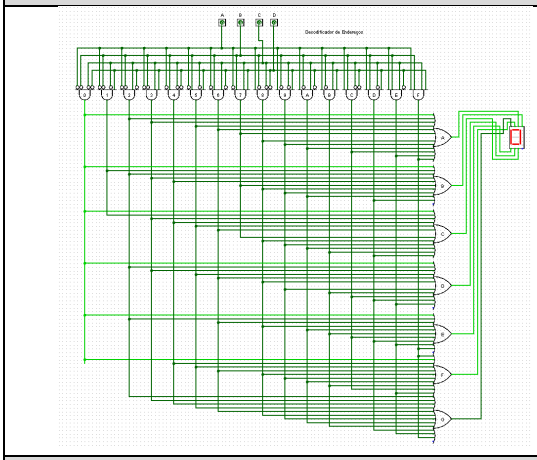
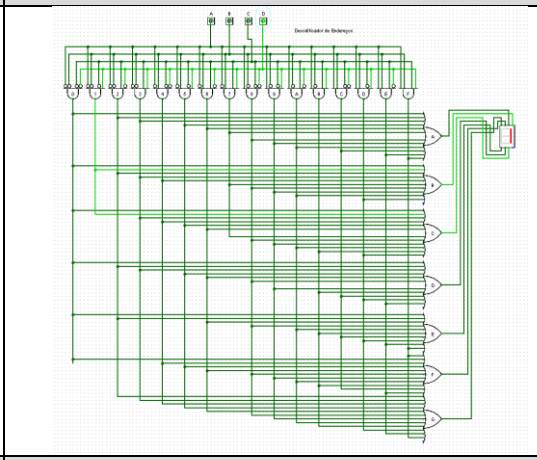
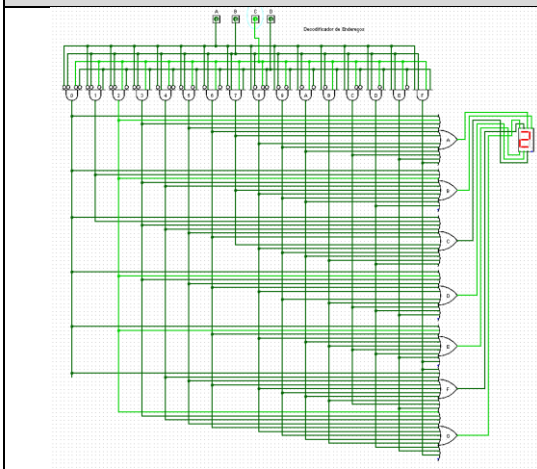
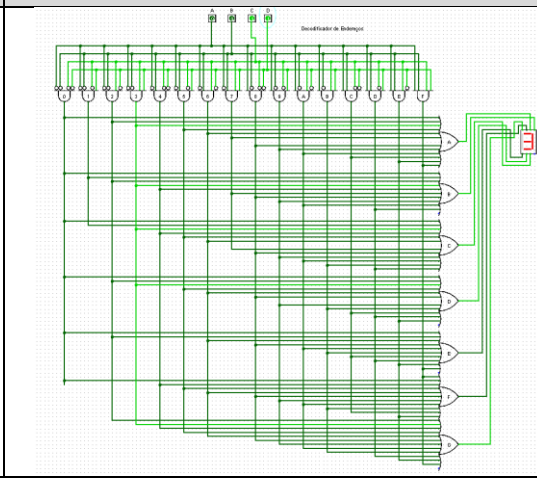
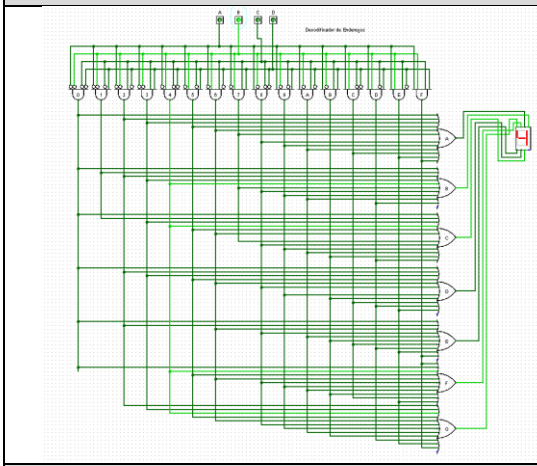
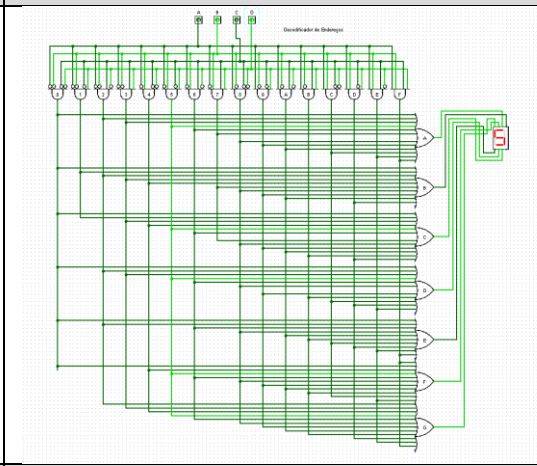
segmentos, representando visualmente números em formato hexadecimal (0-9 e A-F). O display é composto por sete LEDs dispostos de forma a formar os números de 0 a 9 e as letras A à F, com cada segmento do display sendo controlado por um sinal individual, tendo suas posições representadas por letras (a, b, c, d, e, f e g).

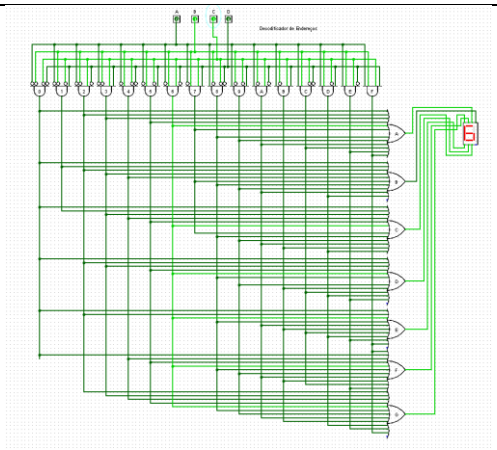
Figura 77 - Decodificador de 7 Segmentos



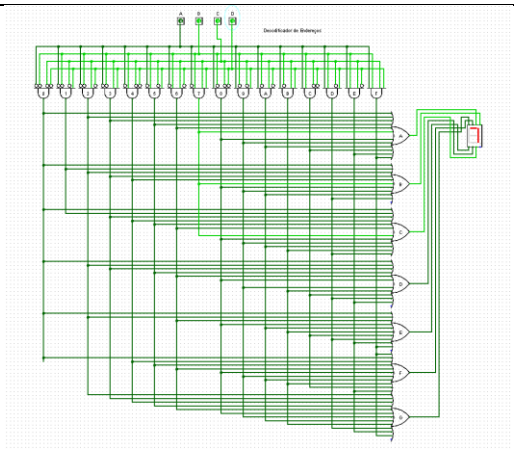
O circuito acima é composto por um display de 7 segmentos, 16 AND, onde cada um equivale a um endereço do decodificador, 7 OR, um para cada segmento do display, 4 entradas para representar os valores de 0 a 15 em hexadecimal, cada AND teve sua entrada negada quando necessário para ter uma saída alta quando for colocado nas entradas seu respectivo endereço, na parte dos OR que representam cada segmento foi conectado apenas naquele que havia necessidade de o segmento ser acesso. Para melhor visualização e montagem cada AND, foi colocado o rótulo de seu respectivo endereço em hexadecimal que vai de 0 à F, assim como cada OR recebeu a letra correspondente a seu segmento, as letras dos segmentos indo de A à F.

Tabela 9 - Funcionamento do Decodificador de 7 Segmentos

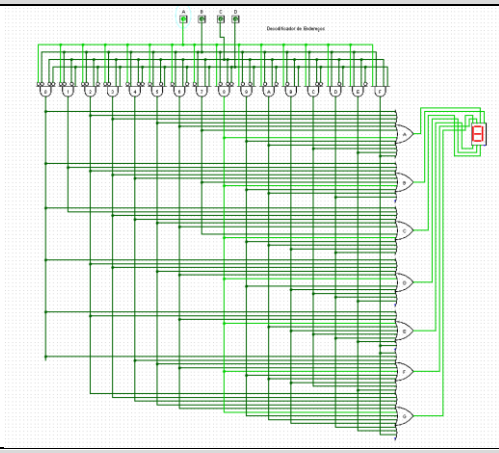
Caso 0	Caso 1
	
Caso 2	**Caso 3**
	
Caso 4	**Caso 5**
	
Caso 6	**Caso 7**



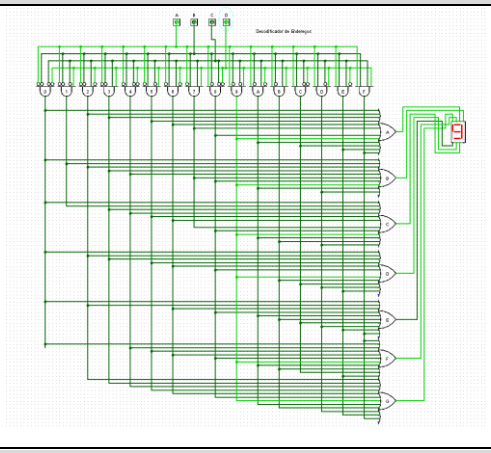
Caso 8



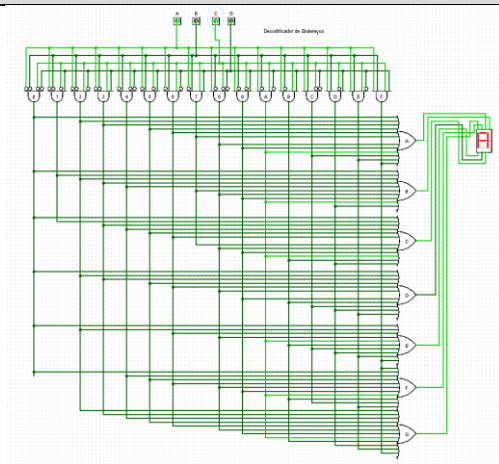
Caso 9



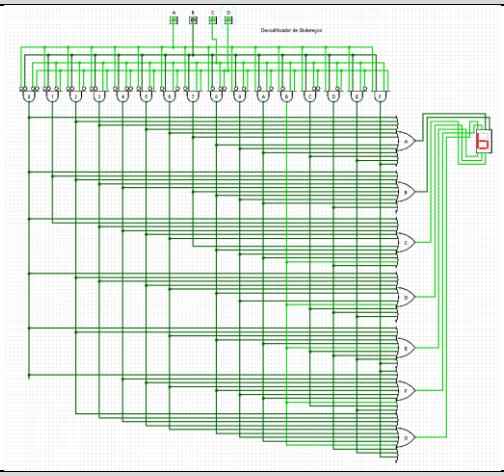
Caso A



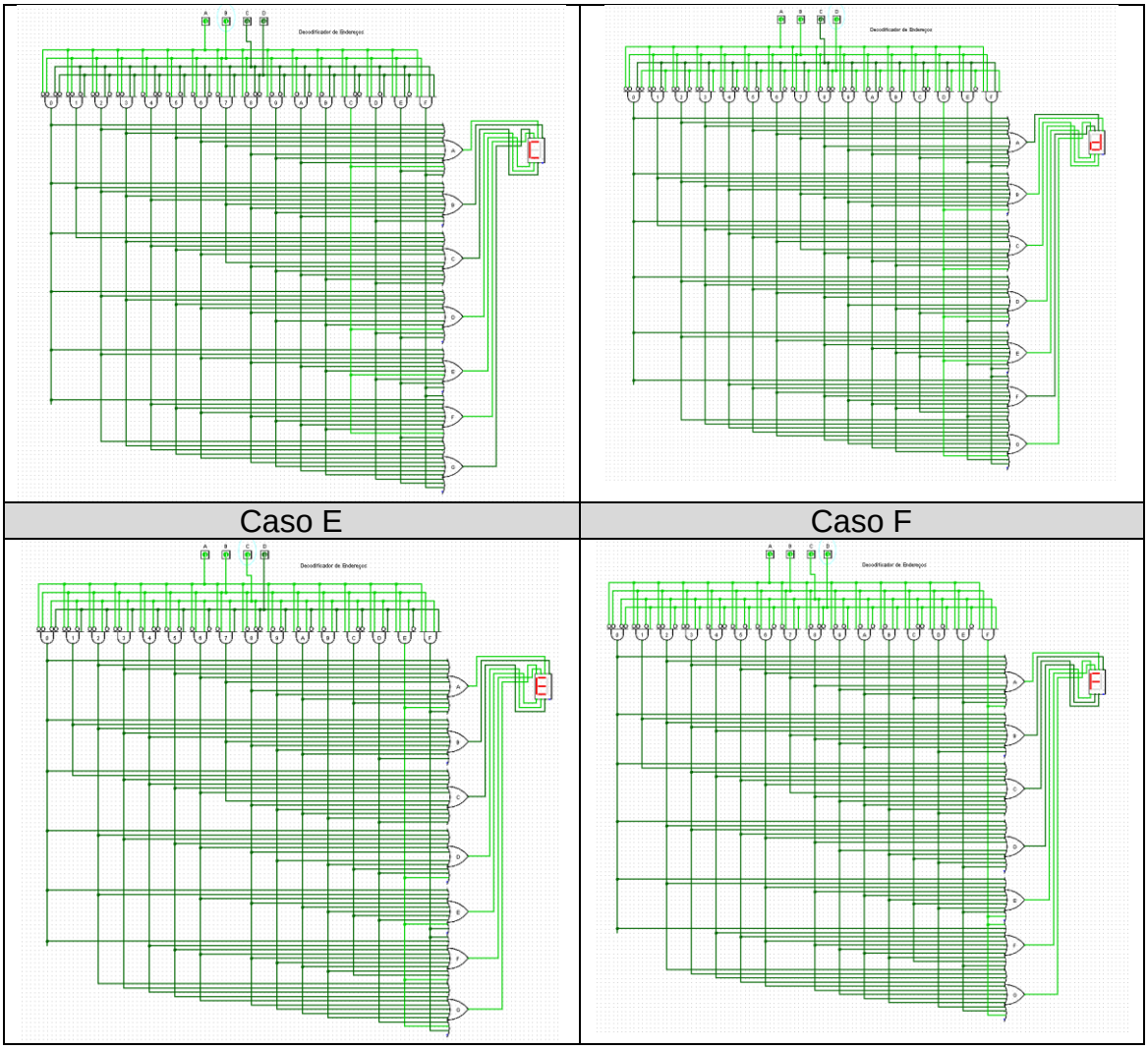
Caso B



Caso C



Caso D



Caso E	Caso F

2.17. Detector de Número Primo de 4 entradas com Mapa de Karnaugh

Um Detector de Número Primo de 4 entradas utilizando mapa de Karnaugh é um circuito lógico que identifica se um número de 4 bits (representado pelas entradas (A, B, C, D) é primo. Números primos são aqueles que possuem exatamente dois divisores 1 e ele mesmo. Onde nesse intervalo os números primos são 2, 3, 5, 7, 11 e 13. Quando há uma entrada em binário de um desses números primos a saída S = 1, caso contrário S = 0.

Tabela 10 - Tabela verdade do detector de número primo

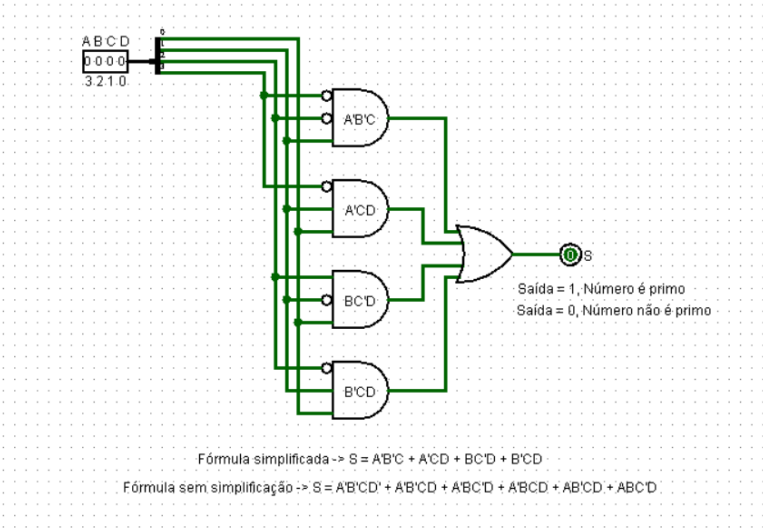
	A	B	C	D	Y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

Figura 78 - Mapa de Karnaugh do detector de número primo

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

Com a otimização usando o mapa de Karnaugh a expressão booleana foi bem simplificada, fazendo a construção do circuito ser mais simples e com menos componentes, como mostra a imagem a abaixo.

Figura 79 - Detector de Número Primo



O circuito acima é composto por uma entrada de 4 bits, distribuidor, quatro AND, quatro NOT, um OR e uma saída de 1 bit, onde cada bit de entrada tem sua posição no distribuidor, e vai para AND correspondente, como está no rótulo do AND, no final se entrada for um número binário correspondente os números primos 2, 3, 5, 7, 11 e 13, a saída S = 1, caso contrário S = 0.

3. CONCLUSÃO

Concluir o desenvolvimento dos 17 componentes digitais no software Logisim foi uma experiência trabalhosa, mas que fortaleceu os conhecimentos de circuitos. Durante o trabalho, foi preciso usar os conhecimentos teóricos sobre circuitos digitais de forma prática, o que exigiu entender bem as lógicas envolvidas, além de planejar e resolver problemas. O prazo curto para concluir as tarefas tornou o processo mais desafiador, exigindo organização e foco para construir e testar todos os componentes corretamente. Apesar de trabalhoso, essa experiência trouxe muito aprendizado, mostrando como a prática ajuda a fixar a teoria. No final, o projeto foi importante para desenvolver habilidades técnicas e de análise, úteis tanto na vida acadêmica quanto no futuro profissional.

4. REFERÊNCIAS

CASTRO, Leonardo; ALVES, Álefe. **Repositório do projeto AOC**. Disponível em: https://github.com/thetwelvedev/AOC_LeonardoCastroAlefeAlves_UFRR_LabCircuitos_2024. Acesso em: 10 dez. 2024.

MAPA DE KARNAUGH. Disponível em: http://www.32x8.com/index.html. Acesso em: 9 dez. 2024.

STALLINGS, William; BOSNIC, Ivan; VIEIRA, Daniel. **Arquitetura e organização de computadores**. 8. ed. São Paulo: Prentice Hall, 2006.

PATTERSON, David A. **Organização e projeto de computadores**. 3. ed. Rio de Janeiro: Elsevier, 2005.