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

Fabrício Cauã Pereira Silvino

DARCKISON ALMEIDA TRAJANO

LABORATÓRIO DE CIRCUITOS – CODIFICAÇÃO E SIMULAÇÕES

BOA VISTA-RR
2024

FABRÍCIO CAUÃ PEREIRA SILVINO

DARCKISON ALMEIDA TRAJANO

LABORATÓRIO DE CIRCUITOS – CODIFICAÇÃO E SIMULAÇÕES

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

Sumário

12. Contador Síncrono.

-Não implementado!

13. Combine portas AND, OR e NOT para criar a lógica de um detector de paridade ímpar (entrada com número ímpar de 1s)

-Não implementado!

14. Resolva um problema de otimização lógica utilizando mapas de Karnaugh e implemente o circuito otimizado.

-Não implementado!

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

- 15.1 Estrutura do Circuito
- 15.2 Funcionamento do Circuito
- 15.3. Construção do circuito:

16. Detector primo.

- 16.1 Funcionamento do Circuito

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

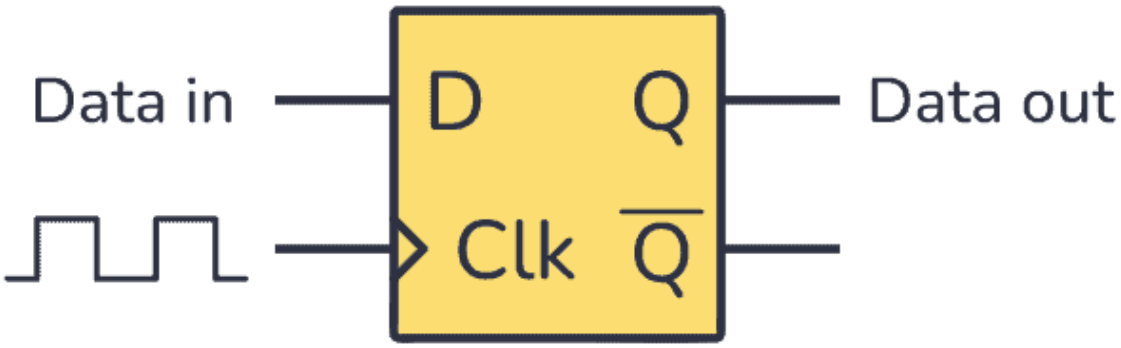
Os **Flip-Flops** são circuitos sequenciais fundamentais em sistemas digitais, projetados para armazenar um bit de informação. Eles são usados para criar elementos de memória, permitindo que dispositivos digitais retenham estados ou dados ao longo do tempo. Diferentemente de circuitos combinacionais, onde a saída depende apenas das entradas atuais, os Flip-Flops têm comportamento dependente de entradas, estados anteriores e de um sinal de controle chamado **clock**. Isso os torna cruciais em aplicações síncronas, como registradores, contadores e memórias.

Existem vários tipos de Flip-Flops, cada um com características e aplicações específicas, como os tipos **JK, SR, T e D ,** etc. Entre eles, o Flip-Flop do Tipo D (ou *Data Flip-Flop*) é o mais simples e amplamente utilizado devido à sua funcionalidade clara e direta.

1.1. Registrador Flip-Flop do tipo D

O Flip-Flop do Tipo D é uma variante que se destaca pela simplicidade e eficiência em armazenar dados. Ele possui duas entradas principais:

- **D (Data):** Contém o valor que será armazenado.
- **Clock (CLK):** Controla quando o valor da entrada será capturado.

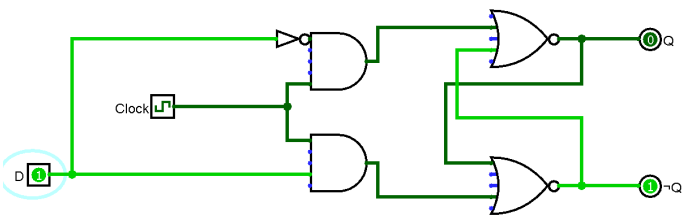


Seu funcionamento é baseado em capturar o valor da entrada D durante uma transição específica do clock (geralmente na borda de subida ou descida, dependendo do design). Após essa transição, o valor capturado é mantido na saída Q até que uma nova transição do clock ocorra.

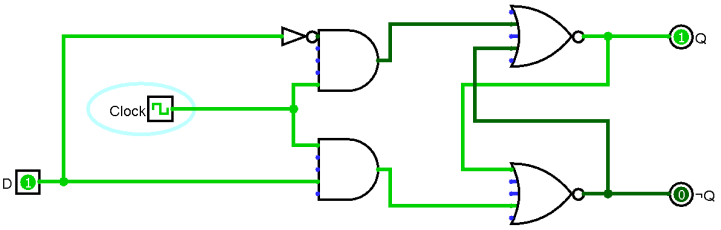
1.1.1. Construção

Esse é um modelo construído no logisim bastante simples, que a cada mudança do nível lógico(Clock) baixo para o alto produz uma saída em Q(Data out) de acordo com a entrada em D(Data in).

Nível lógico baixo



Nível lógico alto



Sua tabela verdade se comporta da seguinte forma:

Clock	D (Data in)	Q (Data out)	¬Q	Análise do caso
0	0	x	¬ x	Em nível lógico baixo a saída(Data out) permanece inalterada.
0	1	x	¬ x	
1	0	0	1	Mudança para o nível lógico alto, o valor armazenado em D será transferido para Q
1	1	1	0	

1.1.2. Características do Flip-Flop do Tipo D

1. **Operação Síncrona:**
   A atualização do estado do Flip-Flop é controlada exclusivamente pelo clock, garantindo que a saída Q só mude em momentos bem definidos.
2. **Memória Temporária:**
   O Flip-Flop pode armazenar um único bit de informação, que permanece disponível na saída Q até ser atualizado pelo clock.
3. **Simplicidade:**
   Diferentemente de outros tipos de Flip-Flops, como o JK, o Flip-Flop do Tipo D tem uma única entrada de dados, simplificando o design e a operação.

1.2. Flip-Flop do tipo JK

O Flip-Flop do tipo JK é um circuito digital sequencial usado como uma memória elementar em sistemas digitais. Ele é uma evolução do Flip-Flop do tipo RS (às vezes chamado de SR), projetado para resolver o problema de condição proibida (quando R e S estão ambos em 1). O Flip-Flop JK é amplamente utilizado em contadores, registradores e outros dispositivos que requerem armazenamento e processamento de informação binária.

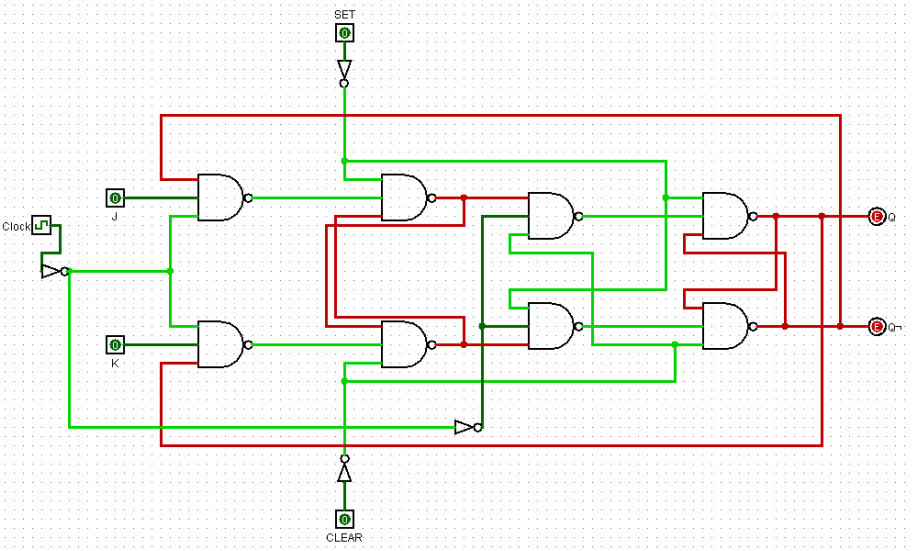
1.2.1. Funcionalidades

O Flip-Flop JK possui duas entradas principais, **J** e **K**, uma entrada de clock, e 1 botão de reset e clear . As funcionalidades são:

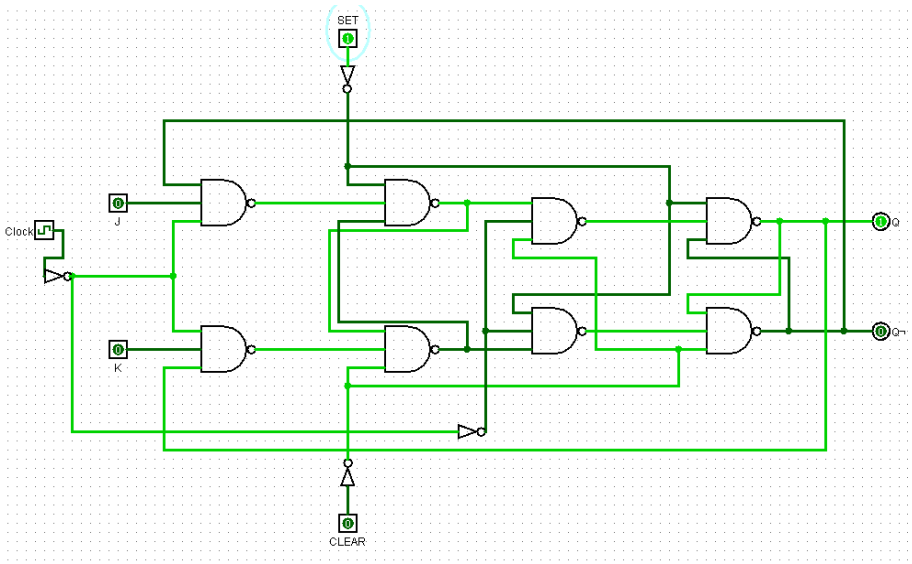
1. **Set = 1**: Permite o início da execução do circuito, para qualquer J = A qualquer e K = B qualquer e nível de clock, a saída é definida para 1.

Flip-flop JK em estado de oscilação.

1 - Caso



2 - Caso



Agora podemos analisar o comportamento do nosso circuito dado aa entradas e o sinal de clock.

Clock	J	K	Q	¬Q
0 — >1	0	0	Q(anterior)	¬Q(anterior)
0 — >1	0	1	0	1
0 — >1	1	0	1	0
0 — >1	1	1	alterna o estado a cada clock	alterna o estado a cada clock

A entrada de **clock** sincroniza o funcionamento do Flip-Flop, garantindo que as alterações ocorram apenas em bordas de subida ou descida do sinal de clock (dependendo do projeto).

A entrada CLEAR permite que seja limpo a saída de Q.

2. Multiplexador de quatro opções de entrada

Um multiplexador é um dispositivo digital utilizado para selecionar uma dentre várias linhas de entrada e direcionar o dado selecionado para uma única linha de saída:

S	X
0	A
1	B

a quantidade de entradas determina o número de seletores, sendo assim, é dada a fórmula:

log2(I) = S.

I: O número de entradas(Inputs), sendo ideal que I seja potência de 2.

S: O número de seletores.

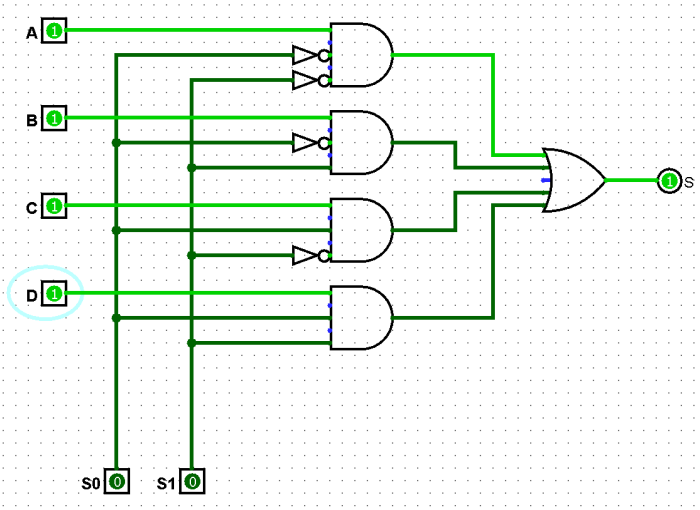
2.1. Construção de um multiplexador de 4 entradas.

O multiplexador de quatro opções de entradas possui quatro entradas, logo:

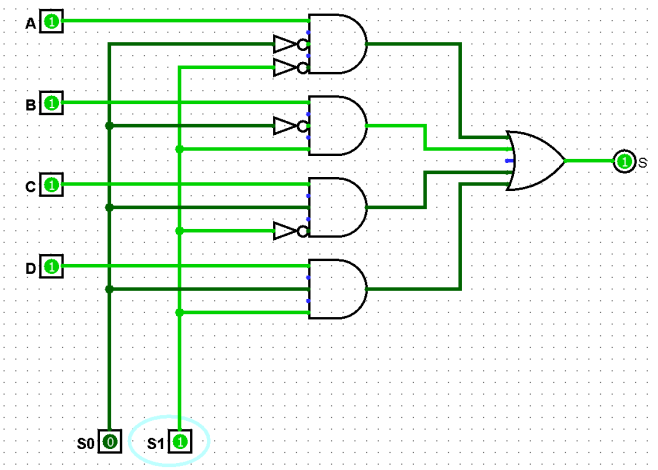
Log2(4)= 2 seletores .

Esse modelo construído no logisim utiliza de 4 portas END e um uma porta OR que direciona o nosso sinal para a saída.

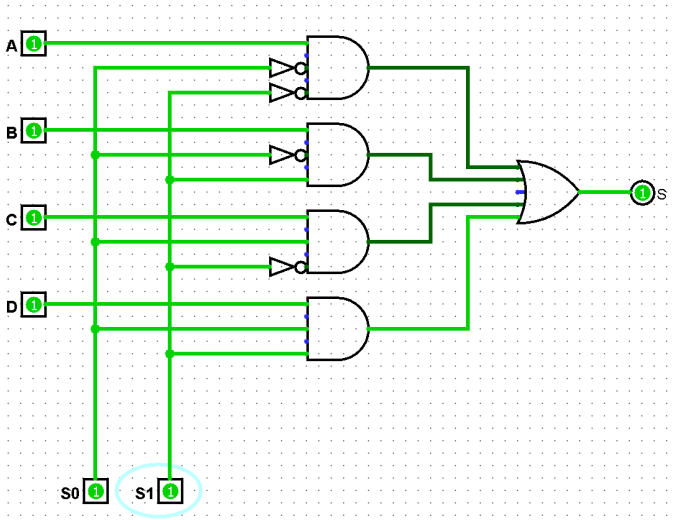
Caso A



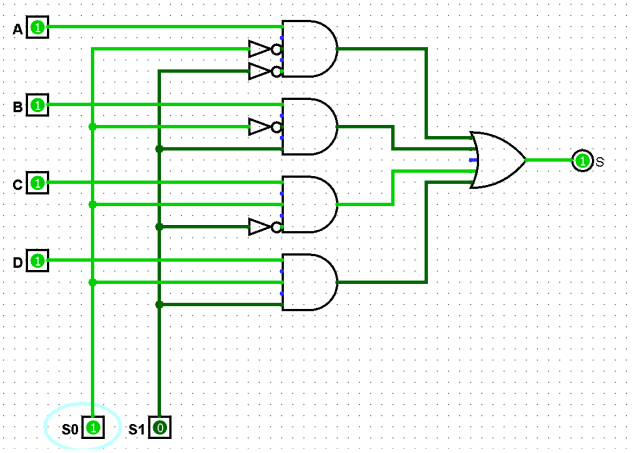
Caso B



Caso C



Caso D



Em cada caso, determinamos um padrão de bits nos seletores, assim construímos nossa tabela verdade:

S0	**S1**	**Saída(Output)**
0	0	A
0	1	B
1	0	C
1	1	D

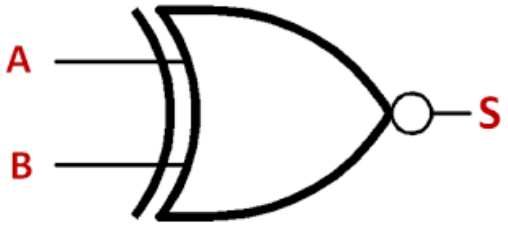
É importante mencionar que cada entrada precisa que seu valor seja igual a 1, pois é necessária a combinação com o valor 1 e os seletores para haver entrada selecionada.

2.2. Aplicações com os multiplexadores.

1. **Seleção de Sinais**: Multiplexadores são amplamente usados para selecionar um sinal específico dentre vários, como em sistemas de comunicação.
2. **Transmissão de Dados**: Permitem que dados de várias fontes compartilhem um único canal de transmissão, otimizando o uso de recursos.
3. **Controle de Sistemas**: São empregados em sistemas de controle onde diferentes variáveis ou sensores precisam ser monitorados de forma alternada.
4. **Implementação de Lógica Combinacional**: Utilizados para realizar funções lógicas específicas em circuitos digitais.

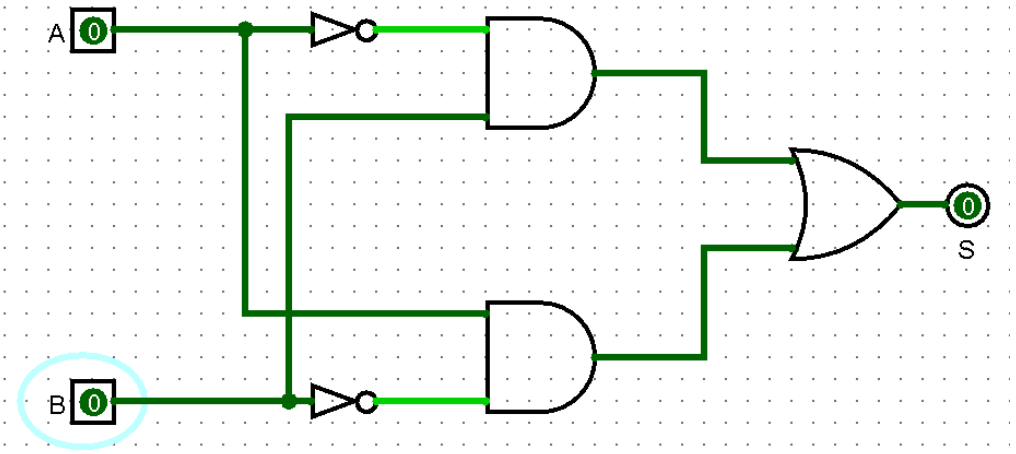
3. Porta lógica XOR usando os componentes: AND, NOT, e OR

A porta lógica XOR (OU Exclusivo) é uma função digital essencial nos sistemas combinacionais, caracterizada por produzir uma saída em nível alto (1) quando as entradas são diferentes e uma saída em nível baixo (0) quando as entradas são iguais. Essa funcionalidade a torna indispensável em diversas aplicações de lógica digital.



3.1. Construção da porta Xor

Podemos construir uma porta XOR utilizando as portas: AND, NOT e OR



Assim obtemos a seguinte tabela:

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

3.2. Funcionalidade e aplicações

O comportamento exclusivo da porta XOR resulta de sua capacidade de detectar diferenças entre os sinais de entrada. Quando apenas uma das entradas é ativada (1), a saída reflete esse estado. Em casos de concordância total entre as entradas (ambas 0 ou ambas 1), a saída é 0. Essa propriedade faz com que a XOR seja amplamente empregada em aplicações que requerem a distinção de estados ou eventos binários.

Algumas de várias aplicações:

1. **Comparadores Digitais**: A porta XOR é usada para comparar sinais binários, identificando diferenças entre bits correspondentes. Por exemplo, em comparadores de magnitude para verificar igualdade ou desigualdade entre números binários.
2. **Somadores Binários**: Em circuitos aritméticos, a XOR desempenha um papel crucial como componente do meio somador e somador completo, determinando o bit de soma sem transporte.
3. **Codificação e Decodificação**: Utilizada em sistemas de transmissão de dados para verificar e corrigir erros, como em códigos de paridade.
4. **Circuitos de Controle**: Em sistemas onde decisões lógicas dependem da diferença entre entradas binárias, a XOR simplifica a implementação de controles condicionais.

4. Somadores 8 bits.

Um somador é um circuito lógico digital fundamental que realiza a adição de números binários. Em termos simples, ele funciona como uma calculadora que soma números em formato binário, que é a linguagem utilizada pelos computadores.

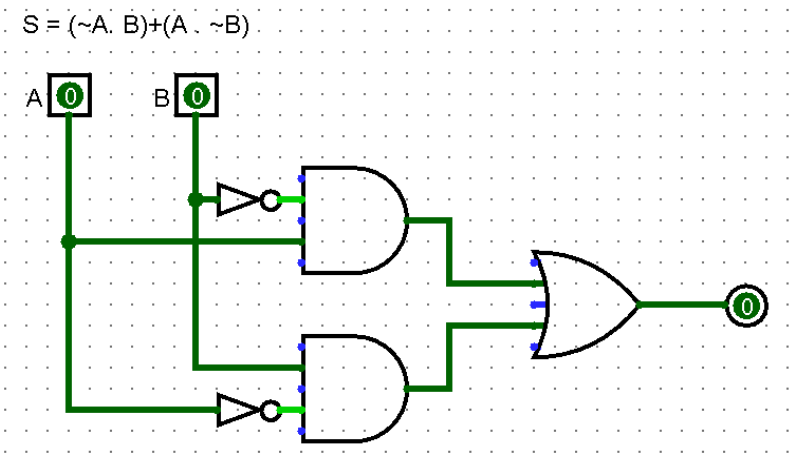
4.1. Por que os somadores são importantes?

- **Base da aritmética digital:** A soma é uma das operações aritméticas básicas. Somadores são os blocos construtores para realizar operações mais complexas como subtração, multiplicação e divisão.
- **Componentes de processadores:** Somadores são encontrados em grande quantidade dentro dos processadores, sendo essenciais para realizar cálculos numéricos.
- **Outras aplicações:** Além de processadores, somadores são utilizados em diversas outras áreas da eletrônica digital, como em circuitos de controle, comunicação de dados e processamento de sinais.

4.2. Tipos de Somadores

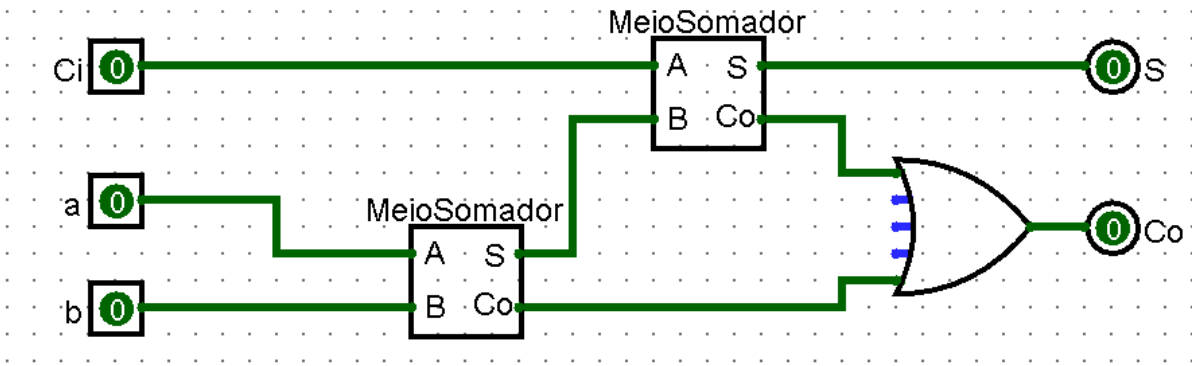
Existem diferentes tipos de somadores, cada um com suas características e aplicações:

- **Meio Somador (Half Adder):** Soma 2 bit e produz 1 bits de saída, o que limita a não representação do carry (vai um).



- **Somador Completo (Full Adder):** Soma três bits: dois bits de entrada e um carry de entrada, produzindo a soma e o carry de saída. Somadores completos são mais versáteis e são usados

para construir somadores de maior porte. Podemos construir o somador completo a partir do meio somador.

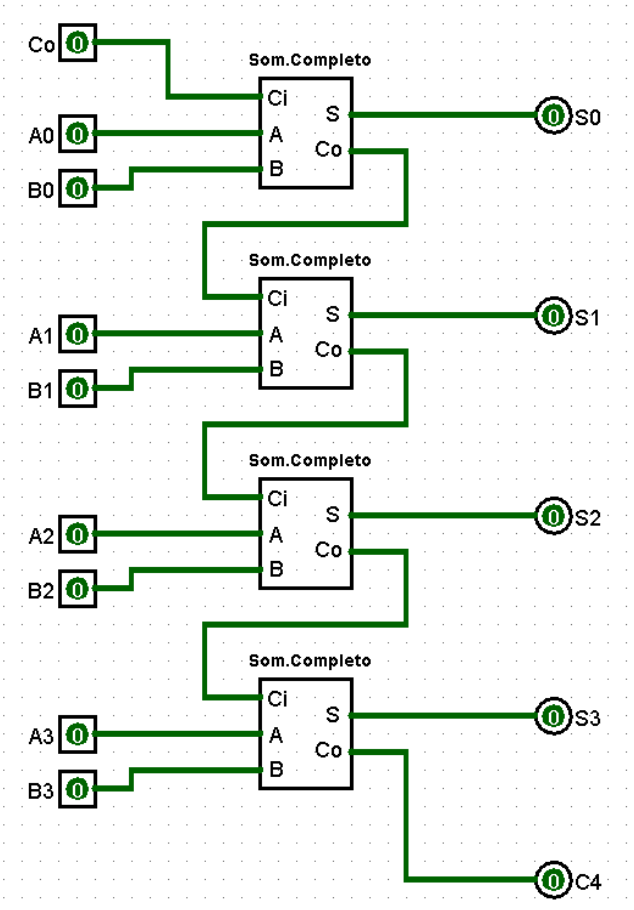


Assim podemos obter sua tabela verdade para entradas A e B:

A	B	SOMA	C0(Carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

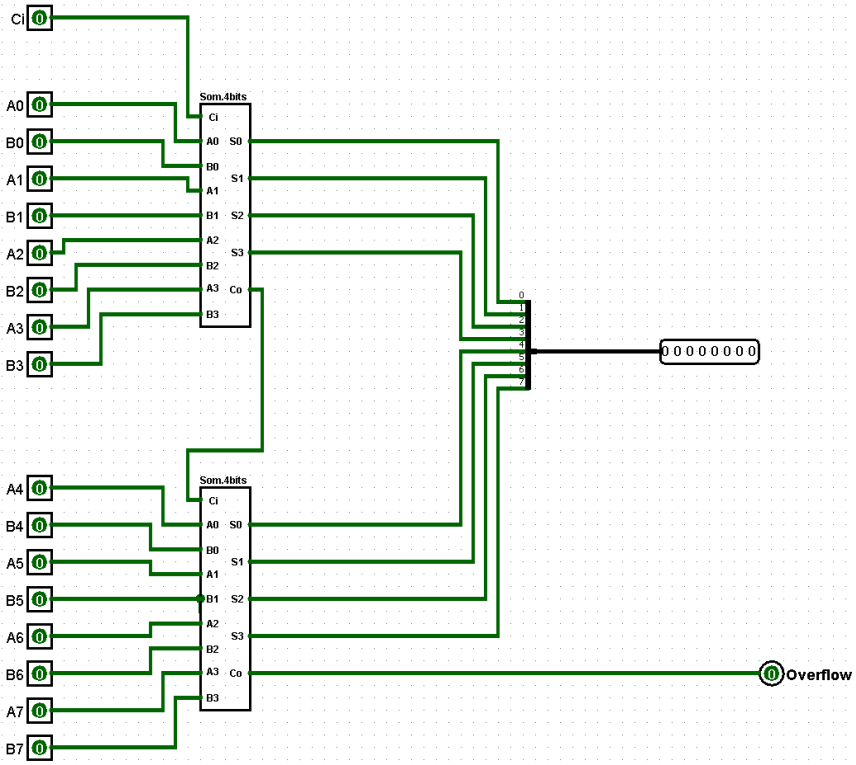
somador de 4 bits

- O **somador de 4 bits** estabelece a base do nosso somador de 8 bits, sendo construído a partir de somadores completos:



4.2.1. somador de 8 bits

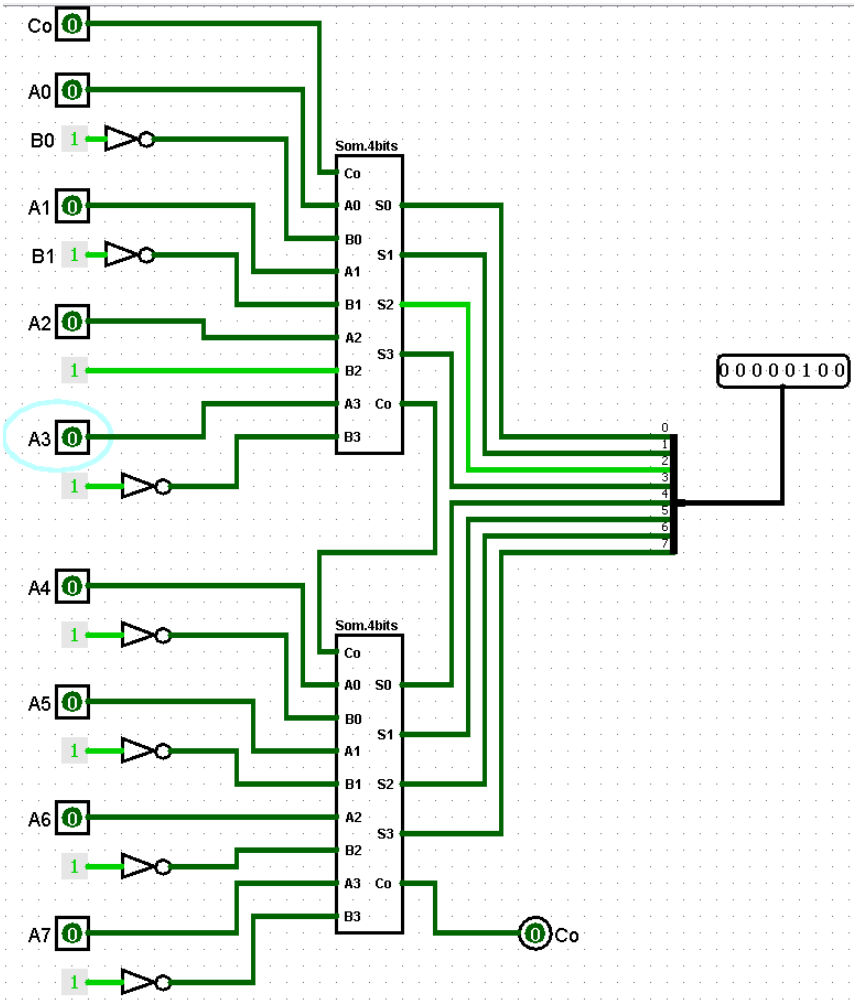
- Por fim, temos o nosso somador de 8 bits que recebe duas entradas A = A0,...,A7 e B = B0,..,B7 cada uma representando o número máximo de 8 bits:



- A última saída chamada overflow representa que o valor da soma dos valores digitados pelo usuário ultrapassou o limite de bits suportado.

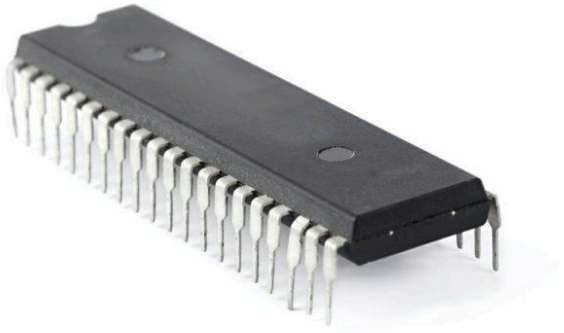
4.2.2 Somador de 8 bits que recebe um valor inteiro e soma com o valor 4

Podemos construir um somador que recebe um valor digitado pelo usuário e que some que um valor constante pré definido: vejamos com o valor 4 representado em bits



5. Memória ROM de 8 bits.

A memória ROM (Read-Only Memory), ou memória de somente leitura em português, é um tipo de circuito integrado não volátil que armazena dados de forma permanente. Isso significa que os dados armazenados na ROM não são apagados quando o dispositivo é desligado. A ROM é programada durante o processo de fabricação ou uma única vez após a fabricação, e seu conteúdo geralmente não pode ser alterado pelo usuário.



5.1. Funções da Memória ROM

A ROM desempenha um papel crucial em diversos dispositivos eletrônicos, sendo utilizada para armazenar:

- Firmware: Software embutido que controla o hardware básico do dispositivo, como BIOS em computadores, firmware de roteadores e microcontroladores.
- Tabelas de referência: Dados de referência, como tabelas de caracteres, códigos de cores ou valores de calibração.
- Programas de inicialização: Instruções iniciais para iniciar o sistema operacional ou um aplicativo específico.
- Dados de configuração: Parâmetros de configuração do dispositivo que não precisam ser alterados com frequência.

5.2 Construindo uma memória rom de 8 bits

Existem diversos tipos de memória ROM, cada um com suas características e aplicações:

- **ROM mascarada:** Programada durante o processo de fabricação e não pode ser alterada posteriormente.

5.2.1 Pergunta 01. Como funciona a memória rom programada pelo usuário?

1. A **PROM (Programmable Read-Only Memory)** é um tipo de memória de somente leitura programável pelo usuário uma única vez após a fabricação. Inicialmente, todas as células estão em estado lógico 1, e, por meio de um **programador de PROM**, fusíveis específicos são queimados para alterar bits para o estado lógico 0. Esse processo é irreversível.

Como funciona?

1. **Estrutura Interna:** Composta por uma matriz de células (fusíveis ou anti fusíveis) organizadas em linhas e colunas.
2. **Programação:** Fusíveis são queimados por tensões altas, mudando os bits de 1 para 0.
3. **Leitura:** Os endereços são acessados para ler o estado dos bits sem alterar os dados.
4. **Imutabilidade:** Após programada, a PROM não pode ser reescrita ou apagada.

Vantagens:

- **Personalização:** Ideal para necessidades específicas.
- **Confiabilidade:** Dados imutáveis após gravação.
- **Custo:** Mais acessível para pequenos volumes comparada a memórias ROM fixas.

Desvantagens:

- **Programação única:** Não permite reescrita.
- **Requer equipamento especial:** Necessidade de um programador de PROM.

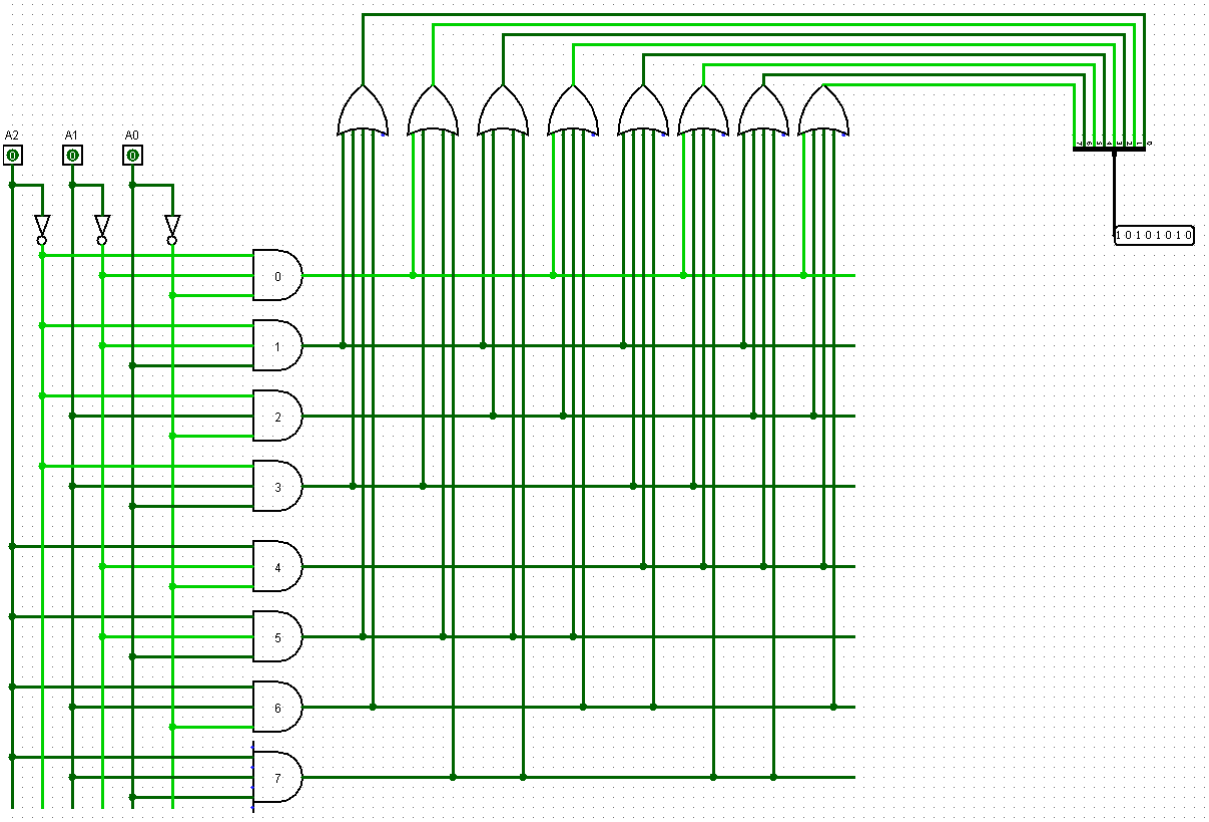
- **EPROM (Erasable Programmable Read-Only Memory):** Pode ser apagada com luz ultravioleta e reprogramada.
- **EEPROM (Electrically Erasable Programmable Read-Only Memory):** Pode ser apagada e reprogramada eletricamente, sem a necessidade de luz ultravioleta.
- **Memória Flash:** Um tipo de EEPROM que permite apagar e reescrever dados em blocos, sendo amplamente utilizada em dispositivos como pendrives e cartões de memória.

Este componente é uma aplicação que utiliza **todos os 8 bits** ou seja, cada número tenha um padrão único em que todos os bits sejam usados, sem serem fixos ou repetitivos dado a seguinte tabela:

Endereço de Mem.	A2	A1	A0	Dados de saída
0x00	0	0	0	10101010
0x01	0	0	1	010101010
0x02	0	1	0	11001100
0x03	0	1	1	00110011
0x04	1	0	0	11110000
0x05	1	0	1	00001111

0x06	1	1	0	10011001
0x07	1	1	1	01100110

Assim com cada valor de entrada montados o valor da nossa saída



O padrão de armazenamento segue da seguinte forma:

Número Decimal	**Representação binária**
0	10101010
1	10101011
2	10101100
3	10101101
4	10101110
5	10101111
6	10110000
7	10110001

6. Memória Ram.

7. Banco de Registradores de 8 bits

Um registrador é um pequeno elemento de armazenamento de alta velocidade localizado dentro de uma CPU ou outro componente eletrônico. Ele armazena dados temporariamente, como números, instruções ou endereços, que são usados diretamente durante a execução de operações pelo processador.

Principais características:

1. Alta velocidade: Muito mais rápido que a memória RAM.
2. Capacidade limitada: Geralmente armazena poucos bits, como 8, 16, 32 ou 64.
3. Uso direto: Interage diretamente com a Unidade Lógica e Aritmética (ALU) para realizar operações como soma, subtração ou movimentação de dados.

Funções:

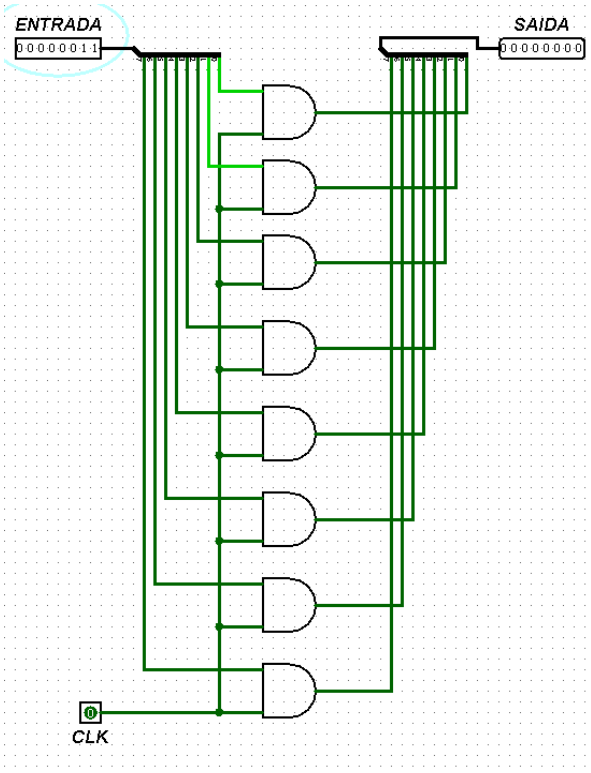
- Armazenar operandos para operações aritméticas/lógicas.
- Guardar endereços de memória para leitura ou escrita.
- Controlar o fluxo de execução, como o contador de programa (PC).

7.1. Construção

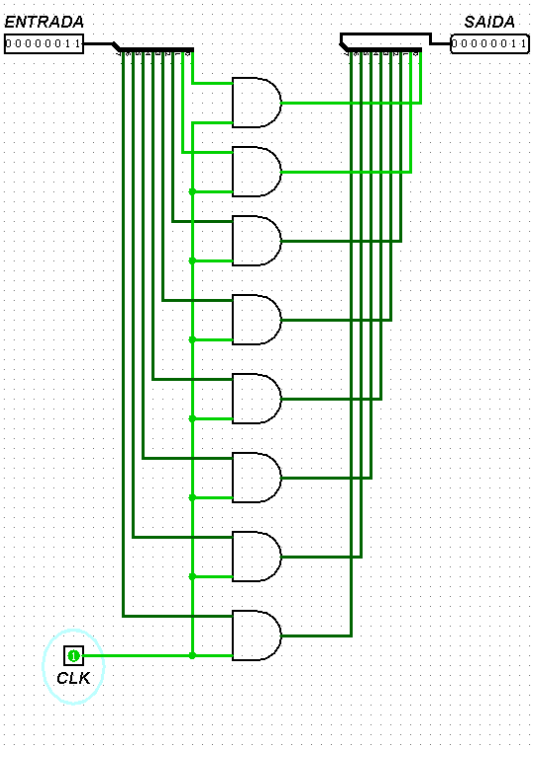
Para sua construção é necessário dois componentes:

1. O circuito que permite a passagem de dados a cada subida de clock:

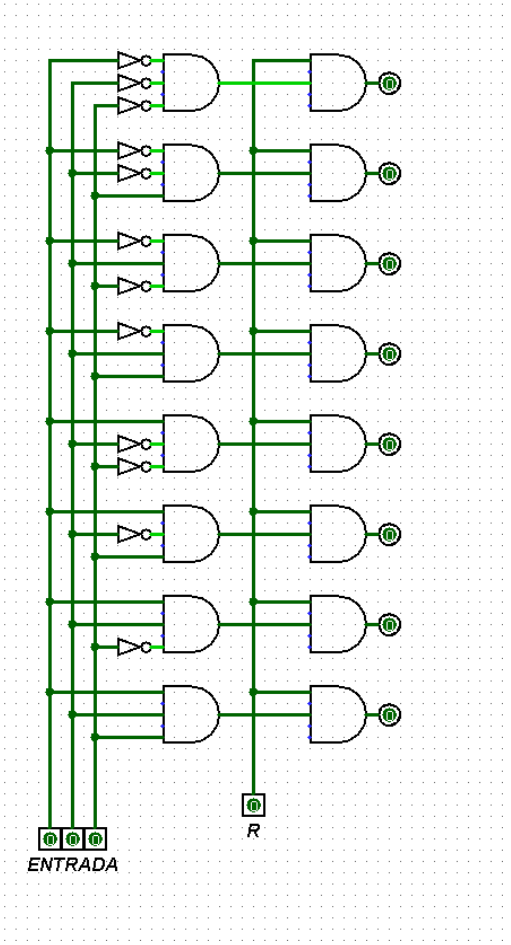
A-Dado à espera de permissão de passagem



B-Permissão concedida

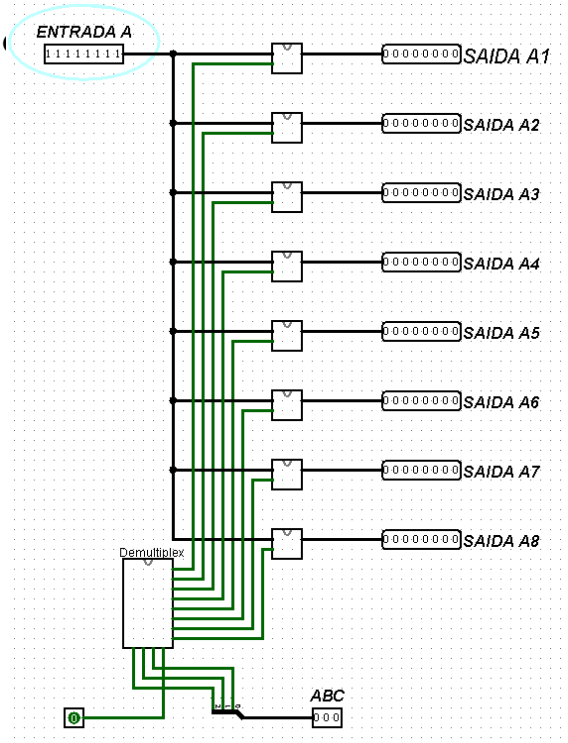


2. O demultiplexador que direciona os dados a 1 das 8 saídas:

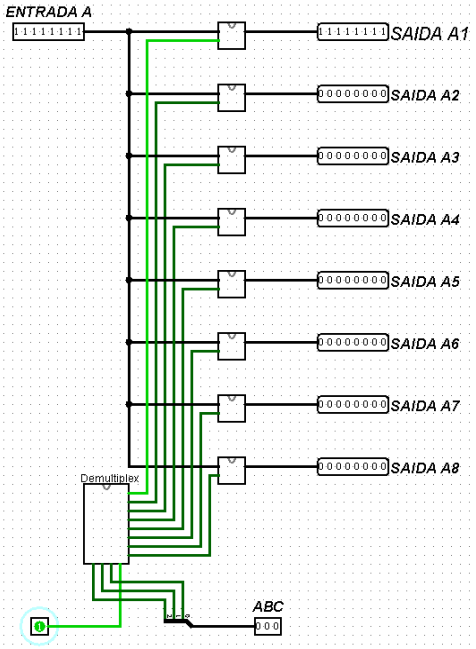


Por fim, podemos montar o nosso circuito:

1 - valor ‘11111111’ digitado a espera de confirmação para a saída A1.



2 - valor ‘11111111’ direcionado a saída A1 após subida de Clock.



7.2. Características de construção

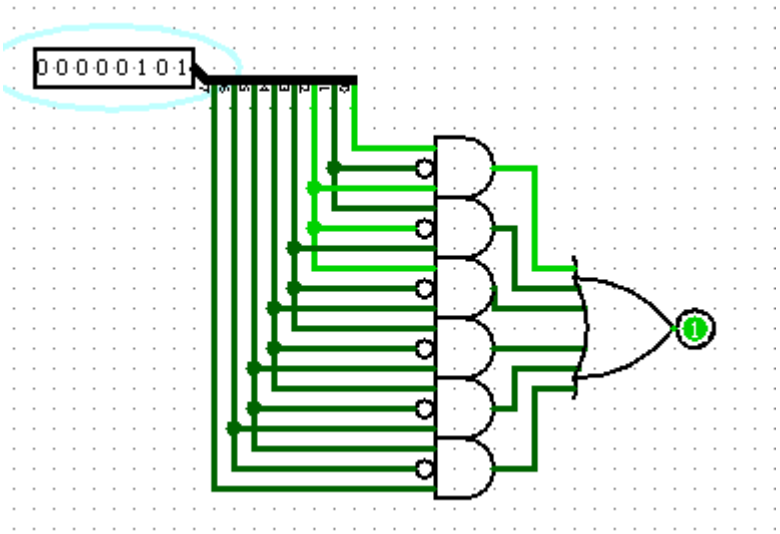
1. Uma entrada A de 8 bits;
2. Uma entrada que representa o ciclo de clock;
3. 8 saídas que são representadas pela combinação de bits digitadas em ABC.

A	B	C	Saídas
0	0	0	Saída A1
0	0	1	Saída A2
0	1	0	Saída A3
0	1	1	Saída A4
1	0	0	Saída A5
1	0	1	Saída A6
1	1	0	Saída A7
1	1	1	Saída A8

8. Detector de sequência de bit 101

Um **detector de bits** é um circuito digital que analisa um conjunto de bits de entrada para identificar padrões ou condições específicas, como o número de bits em alto nível lógico (1), a presença de um padrão binário predefinido, ou se todos os bits estão no mesmo estado.

Aqui definimos a sequência ‘101’ para ser detectada pelo circuito que possui entrada de 8 bits.



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

10. Extensor de sinal de 4 bits para 8 bits

Um **extensor de sinal** é um circuito digital usado para ampliar a largura de um sinal binário sem alterar sua representatividade. Ele é frequentemente utilizado em processadores e sistemas digitais para adaptar sinais de menor bitagem (como 4 bits) para maiores (como 8 bits).

O extensor pode operar de duas formas principais:

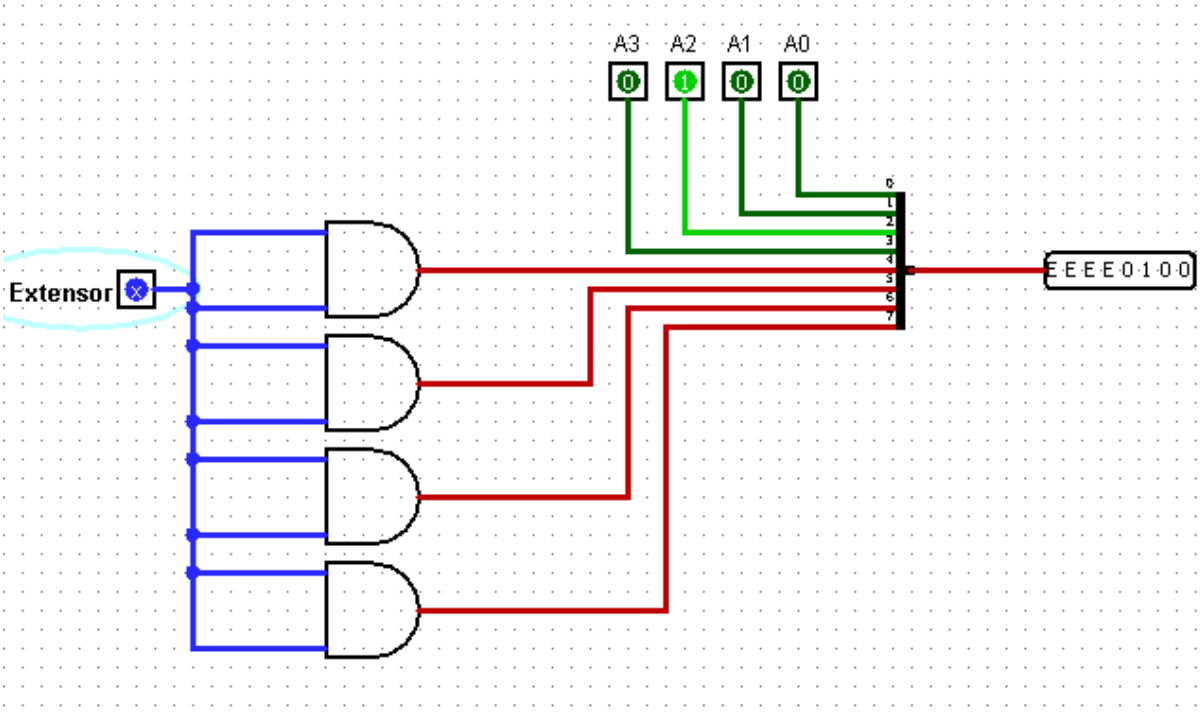
- **Extensão com sinal (Sign Extension):** Preserva o bit de sinal do número (para representação de valores negativos em complemento de 2).
- **Extensão sem sinal (Zero Extension):** Completa os bits mais significativos (MSB) com zeros, adequado para números inteiros não assinados.

10.1 Funcionamento do Extensor de 4 para 8 Bits

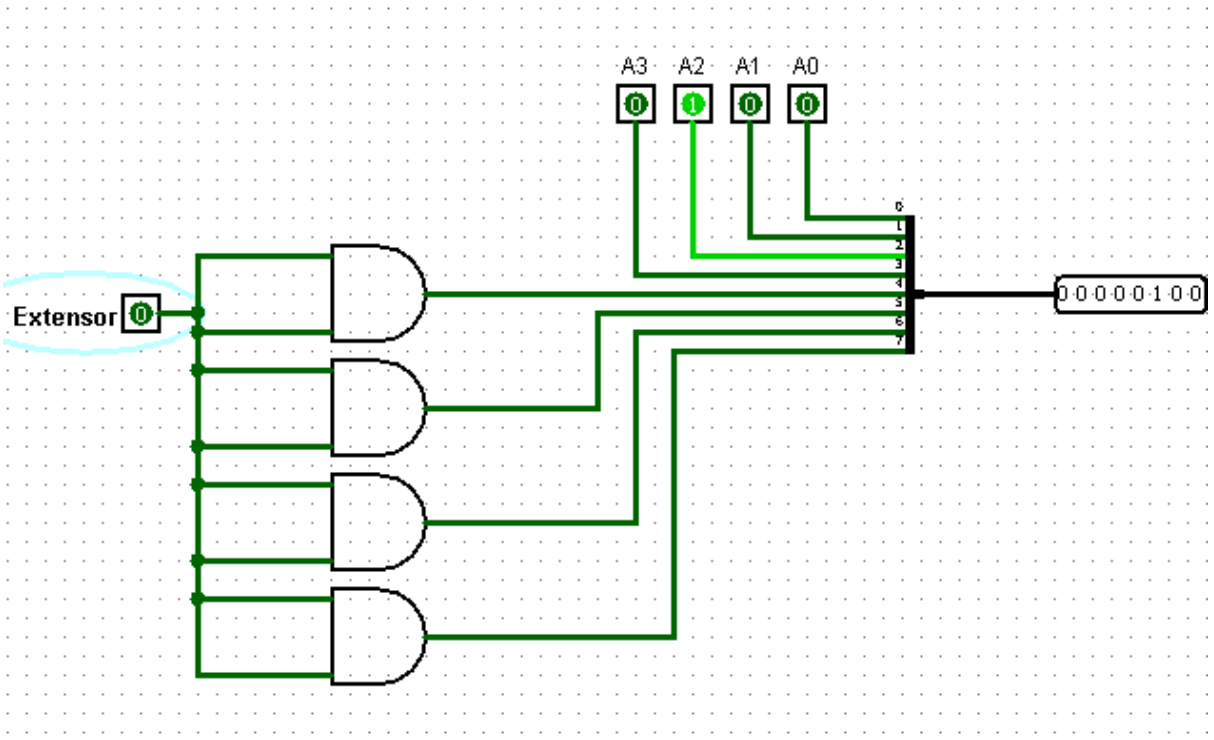
1. **Entrada:**
    - Sinal de 4 bits (D3,D2,D1,D0), onde D3 é o bit mais significativo (MSB).
2. **Saída:**
    - Sinal de 8 bits (Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0).
3. **Regras de Extensão:**
    - **Extensão com sinal:**
        - Os 4 bits mais significativos (Q7,Q6,Q5,Q4Q_7, Q_6, Q_5, Q_4Q7,Q6,Q5,Q4) são preenchidos com o valor do MSB (D3D_3D3).
    - **Extensão sem sinal:**
        - Os 4 bits mais significativos (Q7,Q6,Q5,Q4) são preenchidos com 0.
4. **Lógica Interna:**
    - Portas lógicas ou fios diretos são utilizados para replicar o MSB ou inserir zeros nos bits adicionais.

Vejamos o exemplo a seguir:

Temos uma entrada de 4 bits com valor “0100”



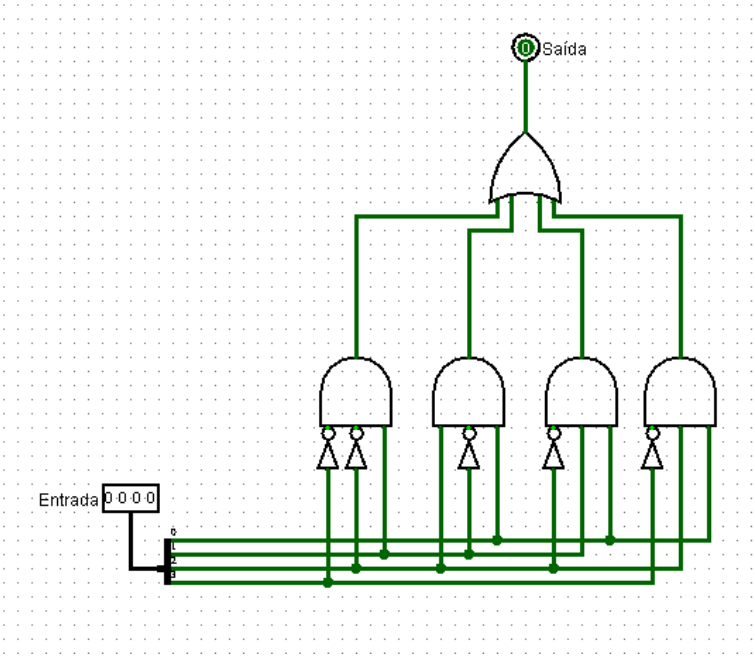
Ao acionar o nosso extensor com valor 0, aumentamos nossa largura de bits da entrada de 4 bits:



16. Detector primo.

O **detector de números primos** é um circuito lógico digital que analisa uma entrada binária e determina se o número correspondente (em decimal) é primo. Ele é útil em sistemas embarcados, circuitos de controle e lógica combinacional, onde é necessário realizar essa verificação de forma rápida e eficiente.

No circuito construído, temos um detector de números primos para números representados por 4 bits, ou seja, números no intervalo de 0 a 13(não foi implementado para identificar 15).



Estrutura do Circuito

1. **Entradas:**
   - Quatro linhas de entrada (D3,D2,D1,D0), representando números binários de 4 bits.
   - D3D é o bit mais significativo (MSB), e D0D_0D0 é o menos significativo (LSB).
2. **Saída:**
   - Uma única linha de saída que indica se o número é primo:
     - **Saída = 1**: O número é primo.
     - **Saída = 0**: O número não é primo.
3. **Lógica Interna:**
   - Utiliza portas lógicas **AND**, **NOT** e **OR** para verificar condições específicas.
   - Combinações de bits que representam números primos (2, 3, 5, 7, 11, 13) são identificadas por meio de portas lógicas.

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

Um decodificador de 7 segmentos é um circuito lógico usado para converter números binários em sinais capazes de controlar um display de 7 segmentos. Esse tipo de display é amplamente utilizado em dispositivos eletrônicos para exibir números e alguns caracteres. O circuito mapeia entradas binárias (4 bits) para acionar os segmentos do display de forma a representar números de 0 a 9 e letras de A a F (formato hexadecimal).

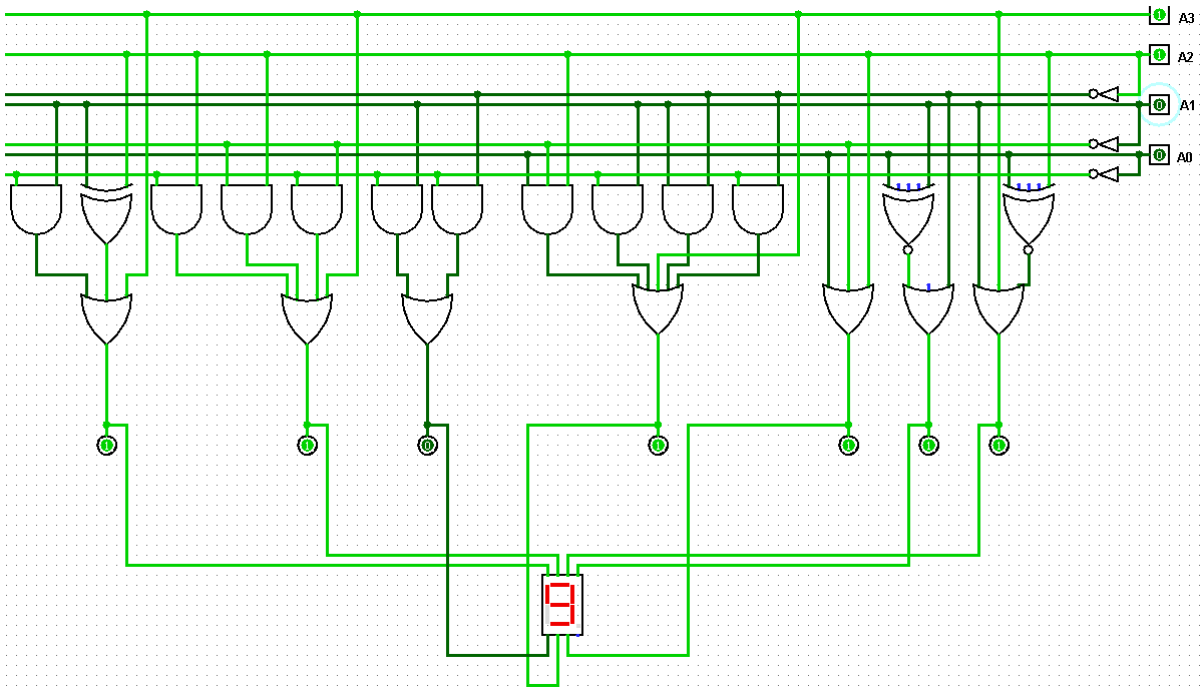
15.1 Estrutura do Circuito

1. **Entradas:**
    - Quatro linhas de entrada (A3,A2,A1,A0), representando números binários de 4 bits:
        - A3A_3A3: Bit mais significativo (MSB).
        - A0A_0A0: Bit menos significativo (LSB).
2. **Saídas:**
    - Sete saídas (Sa,Sb,Sc,Sd,Se,Sf,Sg) que correspondem aos segmentos do display:
        - Cada segmento acende (1) ou apaga (0) de acordo com a entrada binária.
3. **Lógica Interna:**
    - Portas lógicas **AND**, **OR**, **NOT** são utilizadas para criar as combinações necessárias que ativam os segmentos do display.
    - O circuito combina cada entrada binária com operações lógicas para determinar o estado de cada segmento.

15.2 Funcionamento do Circuito

1. **Conversão Binária para Padrão de Segmentos:**
    - O circuito decodifica a entrada binária de 4 bits e ativa os segmentos apropriados para exibir o caractere correspondente (0–9, A–F).
2. **Mapeamento de Segmentos:**
    - Cada número ou caractere hexadecimal tem uma combinação específica de segmentos acesos.
    - Por exemplo:
        - O número **0** acende Sa,Sb,Sc,Sd,Se,Sf.
        - A letra **F** acende Sa,Se,Sf,Sg e apaga Sb,Sc,Sd.

15.3. Construção do circuito:



Esse circuito apresenta as seguintes limitações:

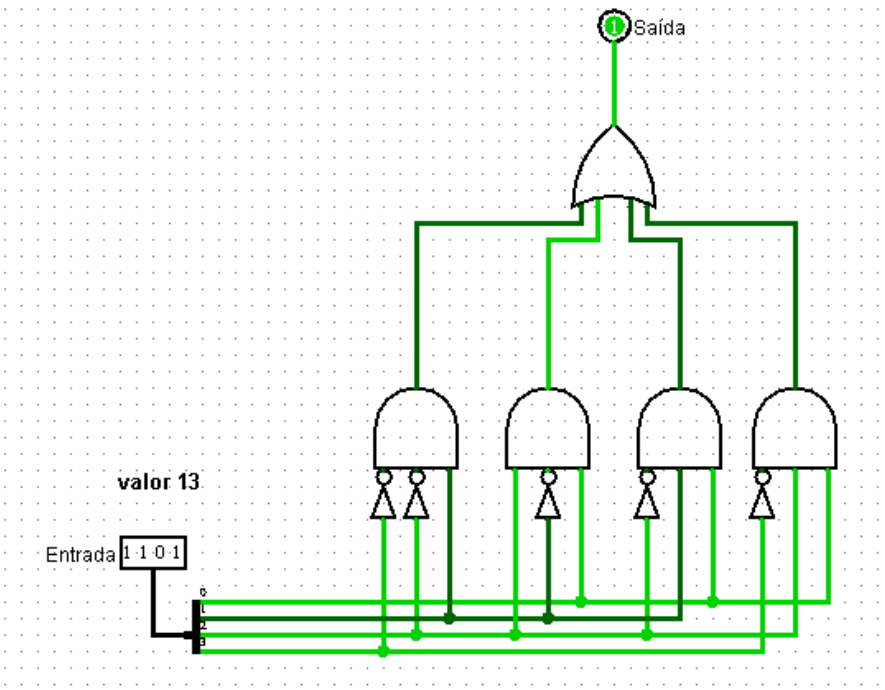
1. Não há representação de números maiores que 9.

2. Não é possível representar letras.

16.1 Funcionamento do Circuito

1. **Detecção de Padrões Primos:**
    - O circuito é projetado para ativar a saída (1) apenas para os números primos dentro do intervalo (2, 3, 5, 7, 11, 13).
    - Cada número primo é detectado por uma combinação específica de entradas.
2. **Implementação das Condições:**
    - Portas **NOT** invertem alguns bits de entrada para criar as combinações desejadas.
    - Portas **AND** verificam se o conjunto de bits corresponde a um número primo.
    - Portas **OR** agrupam os resultados das condições para ativar a saída.

Teste:



Sua tabela verdade se comporta da seguinte maneira:

Entrada (D3, D2, D1, D0)	Número Decimal	Saída(Primo)
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