UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIA E TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

LABORATÓRIO DE CIRCUITOS

BOA VISTA / RR
2024

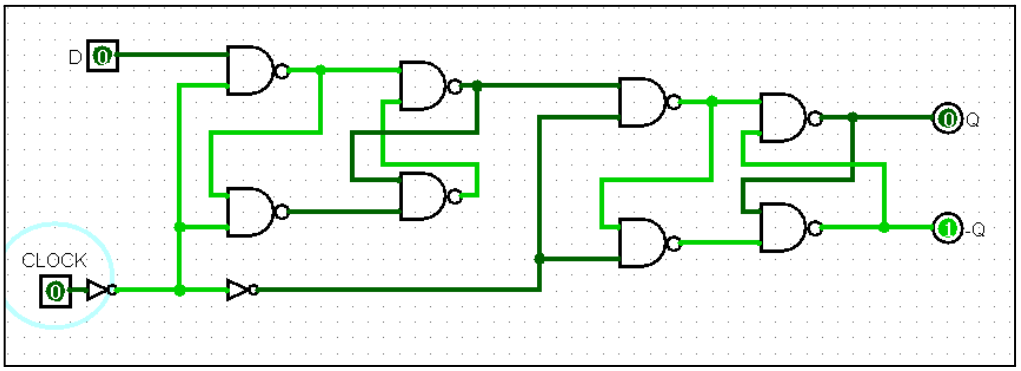
FÁBIO AURÉLIO BARROS ALEXANDRE
JONATHAN EMERSON BRAGA DA SILVA

LABORATÓRIO DE CIRCUITOS

Relatório apresentado na disciplina de Arquitetura e Organização de Computadores (DCC301) pelo Prof. Dr. Herbert Oliveira Rocha, na Universidade de Federal de Roraima, com objetivo de obtenção de nota parcial e apresentação de circuitos lógicos

BOA VISTA / RR
2024

01. Registrador Flip-Flop D



O circuito flip-flop tipo D é um elemento fundamental em eletrônica digital, atuando como um dispositivo de memória capaz de armazenar um único bit de informação. Ele possui uma entrada de dados (D) e uma saída (Q), além de um sinal de clock que sincroniza as mudanças de estado. A configuração mestre-escravo, utilizando portas NAND, é uma implementação comum e robusta desse tipo de flip-flop.

- **Funcionamento**

 # Estrutura Mestre-Escravo:

Parte Mestre: Durante o meio-ciclo em que o clock está baixo, a parte mestre do circuito é sensível à entrada D. O valor de D é armazenado temporariamente em um conjunto de portas NAND, preparando o circuito para a próxima mudança de estado.

Parte Escrava: Quando o clock muda para nível alto, a parte escrava é ativada. O valor armazenado na parte mestre é transferido para a saída Q e sua complementar ~Q. A partir desse momento, a saída Q manterá o valor até a próxima borda de subida do clock.

Papel das Portas NAND:

As portas NAND são utilizadas para implementar a lógica do flip-flop, tanto na parte mestre quanto na parte escrava. A combinação de portas NAND permite criar as funções lógicas necessárias para armazenar e transferir os dados. A porta NOT é

utilizada para gerar a saída complementar ~Q, que é essencial para muitas aplicações.

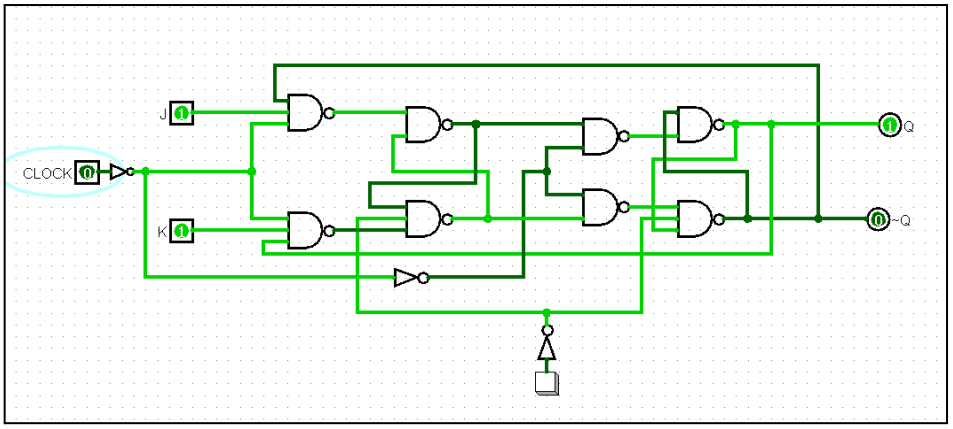
Sinal de Clock:

O sinal de clock sincroniza as mudanças de estado do flip-flop. A borda de subida do clock é o momento em que o valor armazenado na parte mestre é transferido para a saída. A utilização de um clock manual permite controlar o momento exato em que as mudanças de estado ocorrem, facilitando a análise e o debug do circuito.

Tabela Verdade

Q (próximo estado	~Q (próximo estado invertido
0	1
1	0
Q (mantém)	~Q (mantém

02. Registrador Flip-Flop JK



O flip-flop JK é um circuito sequencial fundamental em eletrônica digital, amplamente utilizado em diversas aplicações devido à sua versatilidade. Ele possui três entradas principais: J, K e clock, e duas saídas: Q e Q'. A principal característica do flip-flop JK é sua capacidade de alternar o estado (toggle) quando ambas as entradas J e K estão em 1.

Funcionamento

Estrutura Mestre-Escravo:

Similar ao flip-flop D, o flip-flop JK frequentemente utiliza a configuração mestre-escravo para sincronizar as mudanças de estado. Isso garante que a saída mude apenas na borda de subida do clock, evitando problemas de rastreamento de clock.

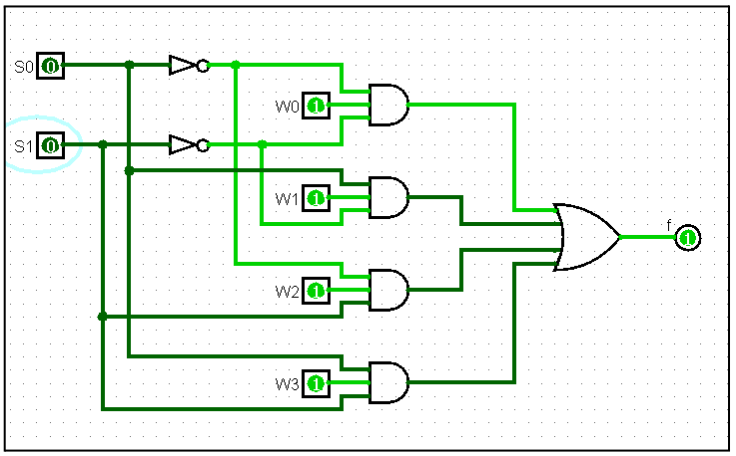
- **Parte Mestre:** Durante o meio-ciclo em que o clock está baixo, a parte mestre do circuito avalia as entradas J e K. O resultado dessa avaliação determina o próximo estado do flip-flop.
- **Parte Escrava:** Quando o clock muda para nível alto, a parte escrava é ativada, transferindo o valor armazenado na parte mestre para a saída Q e seu complemento ~Q.

Tabela Verdade:

J	K	Clock	Q (próximo estado)	~Q (próximo estado)
0	0	↑	Q (mantém)	~Q (mantém)
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	~Q (toggle)	Q (toggle)

03. Multiplexador de quatro opções de entrada.

O multiplexador é um circuito lógica que funciona apresentando saída de acordo com os sinais da chave de seleção. Ele recebe duas ou mais entradas e de acordo com a chave seletora, ele direciona para a saída. A chave de seleção é responsável por direcionar o fluxo de energia da entrada selecionada para sua saída. Quando as chaves estão configuradas, apenas a entrada correspondente às chaves percorre pelo circuito para a saída, não importando quantas outras entradas estejam em nível lógico alto.

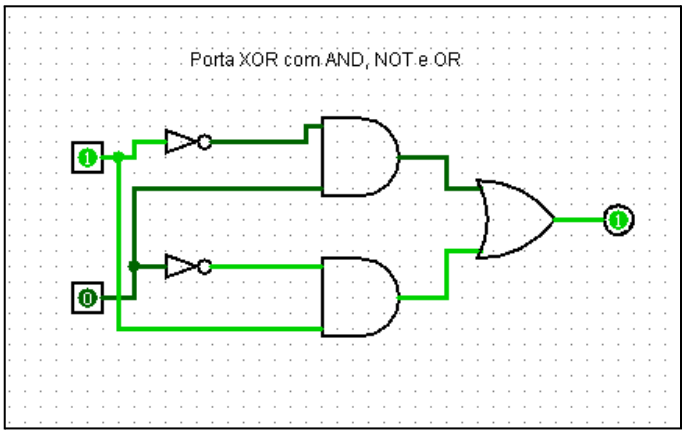


No caso do multiplexador de 4 entradas, é possível obter 4 combinações de acordo com a tabela verdade. O número de chaves seletoras é determinado a partir da quantidade de entradas de bits do circuito. Para um circuito de 4 entradas, temos $2^2 = 4$, ou seja, 4 saídas e 2 chaves de seleção. Para um circuito de 8 entradas, temos $2^3 = 8$, portanto, 8 saídas e 3 chaves seletoras.

S0	S1	Saída
0	0	W0
0	1	W1
1	0	W2
1	1	W3

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

O circuito da figura apresenta uma equivalência da porta XOR, por meio da combinação das portas AND, NOT e OR. A porta XOR funciona quando seus sinais de entrada são diferentes, ou seja, a saída é alta (1), quando suas entradas possuem sinais diferentes. A porta é composta por duas entradas e uma saída.



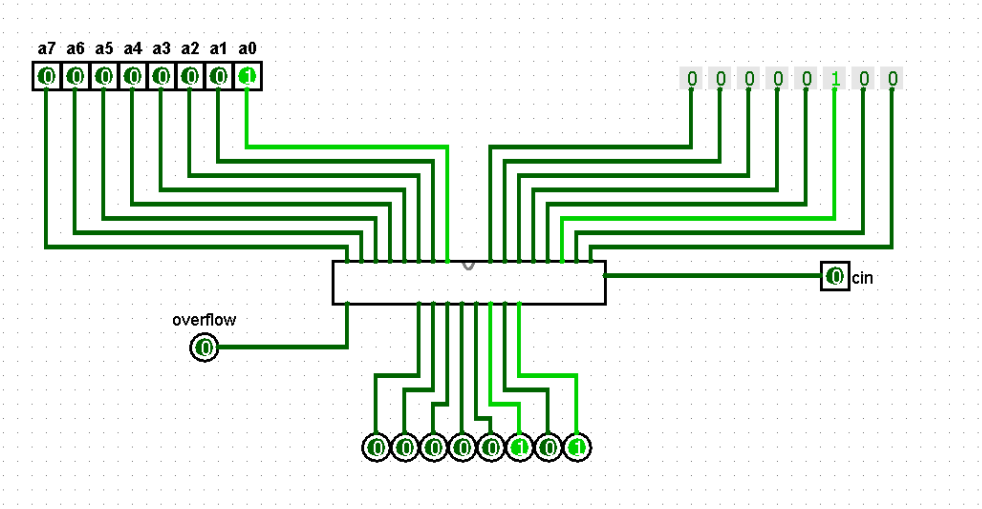
Podemos fazer testes a partir da construção da tabela verdade que possui 4 combinações possíveis, pois $2^2 = 4$. Como podemos notar:

Quando A = 0, B = 0 e A = 1, B = 1 (sinais iguais): a saída é 0;

Quando A = 1, B = 0 e A = 0, B = 1 (sinais diferentes a saída é 1.

A	B	A XOR B
0	0	0
0	0	1
1	0	1
1	1	0

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



- **Objetivo do Circuito**

O objetivo do circuito é realizar a adição de dois valores binários de 8 bits:

- O primeiro valor é uma entrada binária configurável (indicado como a7 a a0).
- O segundo valor é o número inteiro 4 (fixo), representado em binário como 00000100.

Esse circuito pode ser utilizado para operações aritméticas básicas em sistemas digitais, como incrementos fixos ou ajustes de valores em registradores.

- **Estrutura do Circuito**

Entradas

- **Entradas A (a7 a a0):** São 8 bits de entrada binária, representando um número configurável. Na imagem fornecida, o valor de entrada é 00000001 (1 em decimal).

- **Entradas B (Valor 4):** O valor fixo 4 é conectado como a segunda entrada ao somador, em formato binário (00000100).

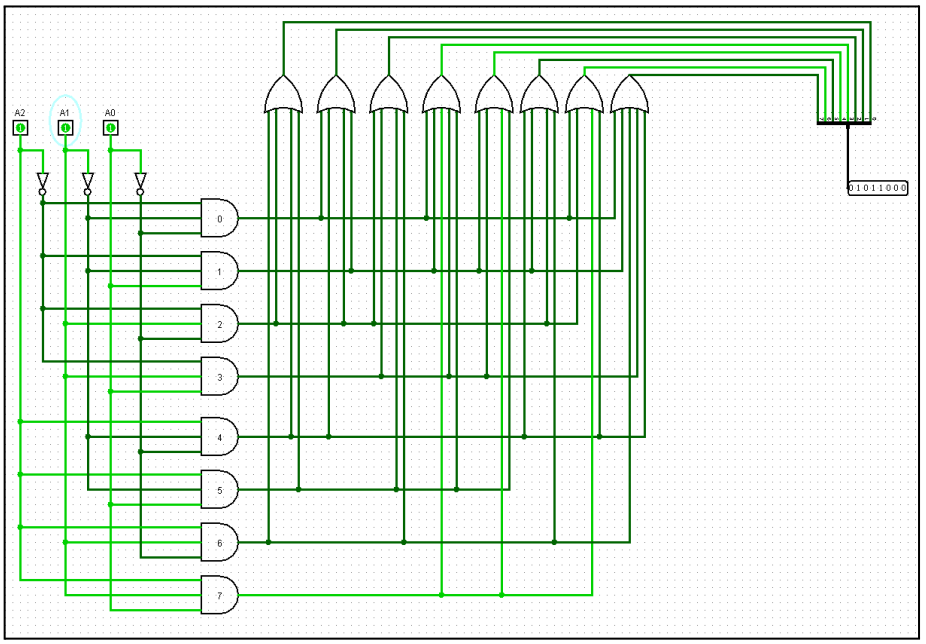
Saída

- A saída do somador é composta por 8 bits, representando o resultado da soma binária. Na imagem, o resultado é 00000101 (5 em decimal).

Componentes Adicionais

- **Carry-In (CIN):** Um bit de entrada adicional para operações com transporte inicial (carry-in). Neste circuito, ele está configurado como 0, indicando que não há transporte inicial.
- **Overflow:** Um indicador que monitora se houve overflow (estouro) durante a operação de soma. Neste caso, ele está desativado (0), indicando que o resultado está dentro do intervalo de 8 bits.

06. Memória ROM de 8 bits.



- **Objetivo do Circuito**

A ROM é um tipo de memória não volátil, ou seja, ela mantém os dados armazenados mesmo quando a energia é desligada. Ela é utilizada para armazenar informações fixas que não precisam ser alteradas durante a operação do sistema, como o código de inicialização de um microcontrolador ou tabelas de referência.

- **Estrutura do Circuito**

O circuito ROM apresentado parece ser composto por:

- **Entradas de Endereço (A0 a A7):** Essas entradas são utilizadas para selecionar uma palavra específica de memória. No caso, com 8 bits de endereço, o circuito pode armazenar 2^8 = 256 palavras de 8 bits cada.
- **Saídas de Dados (D0 a D7):** Essas saídas fornecem os 8 bits de dados correspondentes à palavra de memória selecionada.
- **Circuito Lógico:** A parte central do circuito é composta por portas lógicas (AND, OR, NOT) que implementam a função de leitura da memória. A configuração específica dessas portas determina o conteúdo da memória.

- **Funcionamento**
    1. **Seleção da Palavra:**

        Quando um endereço específico é aplicado nas entradas A0 a A7, o circuito lógico decodifica esse endereço e seleciona a linha de dados correspondente.

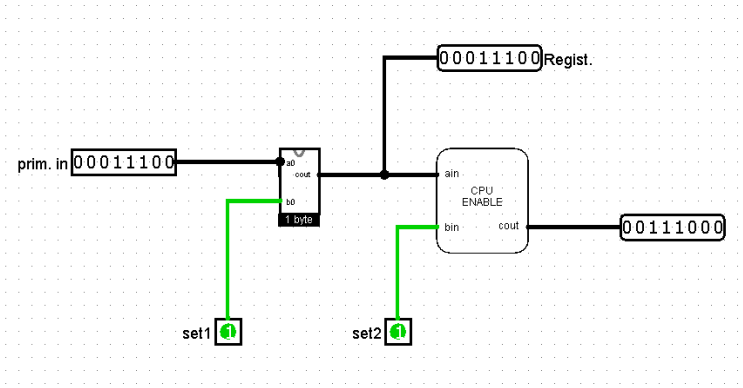
    2. **Leitura dos Dados:**

        Os dados armazenados na palavra selecionada são então colocados nas saídas D0 a D7.

    Características da ROM Implementada

    - **Capacidade:** 256 palavras de 8 bits.
    - **Tipo:** A ROM parece ser uma ROM máscara, onde o conteúdo da memória é definido durante o processo de fabricação e não pode ser alterado posteriormente.
    - **Organização:** A organização da memória é de 256x8, o que significa 256 linhas e 8 colunas.

07. Banco de Registradores de 8 bits.



Um banco de registradores é um componente fundamental em arquiteturas de computadores, atuando como uma memória de alta velocidade para armazenar dados que serão frequentemente acessados pela Unidade Lógica Aritmética (ULA) ou outras unidades do processador.

Elementos do Circuito

- **Registradores:** Os elementos retangulares rotulados como "Registr" representam os registradores individuais. Cada registrador pode armazenar uma quantidade específica de bits (neste caso, 8 bits, ou 1 byte).
- **Entradas e Saídas:** As linhas rotuladas como "prim_in" e "cout" representam as entradas e saídas dos registradores, respectivamente.
- **Sinais de Controle:** Os sinais "set1" e "set2" provavelmente são sinais de controle que determinam qual registrador será acessado ou qual operação será realizada no banco de registradores.
- **Bloco CPU ENABLE:** Este bloco representa a interface com a Unidade Central de Processamento (CPU), indicando quando o banco de registradores está habilitado para operações de leitura ou escrita.

Funcionamento

1. **Seleção do Registrador:**

   Os sinais de controle "set1" e "set2" (e possivelmente outros não mostrados no diagrama) são utilizados para selecionar o registrador específico que será acessado.

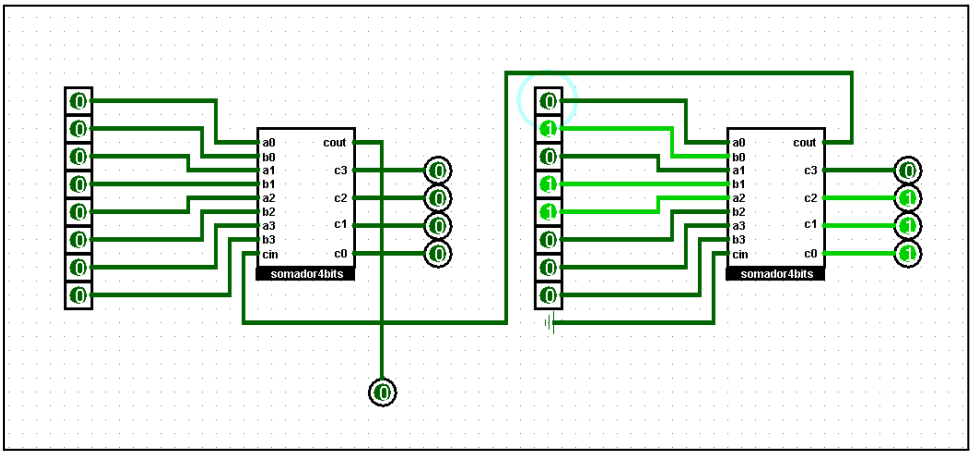
2. **Operação de Leitura:**

Quando a CPU precisa ler um dado de um registrador, ela envia o sinal de controle apropriado para selecionar o registrador desejado. O dado armazenado no registrador selecionado é então colocado na saída "cout".

3. **Operação de Escrita:**

Para escrever um dado em um registrador, a CPU envia o dado para a entrada "prim_in" e o sinal de controle apropriado para selecionar o registrador de destino. O dado é então armazenado no registrador selecionado.

08. Somador de 8 bits.



Essa é uma estrutura fundamental em eletrônica digital, utilizada para realizar a operação de adição entre dois números binários de 8 bits cada. Somadores são componentes essenciais em processadores, calculadoras e outros dispositivos digitais.

Análise do Circuito

Componentes Principais:

- **Registradores:** Os elementos retangulares representam os registradores que armazenam os números binários de 8 bits a serem somados. Cada registrador possui 8 bits, representados por a0 a a3 e b0 a b3.
- **Somadores de 4 bits:** Os blocos rotulados como "somador 4bits" são responsáveis por somar 4 bits de cada vez. Eles recebem duas entradas de 4 bits (a0-a3 e b0-b3) e um bit de carry-in (cin) e produzem uma saída de 4 bits (soma) e um bit de carry-out (cout).
- **Carry-in e Carry-out:** O bit de carry-in (cin) é utilizado para conectar os dois somadores de 4 bits, permitindo que o carry gerado pela soma dos 4 bits menos significativos seja propagado para a soma dos 4 bits mais significativos. O bit de carry-out (cout) do segundo somador de 4 bits representa o carry final da operação de soma.

Funcionamento:

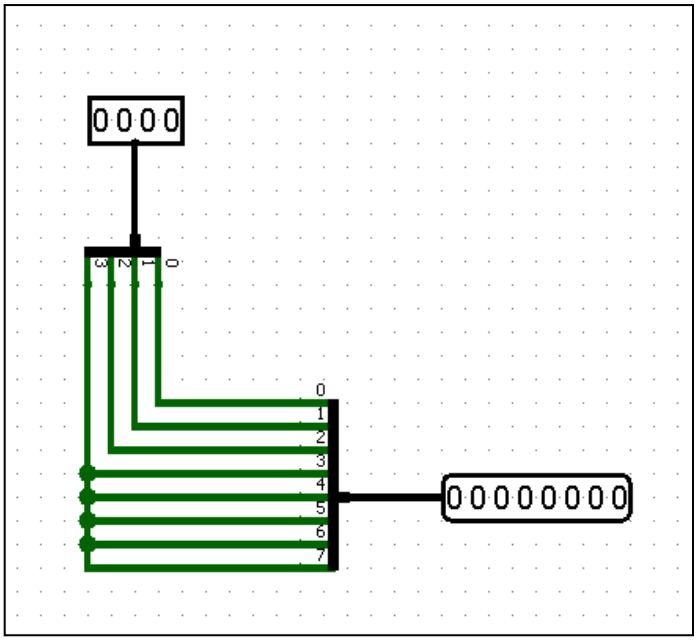
1. **Entrada de Dados:** Os números binários a serem somados são armazenados nos registradores.

2. **Soma dos 4 bits menos significativos:** Os 4 bits menos significativos de cada registrador são enviados para o primeiro somador de 4 bits. Este somador gera uma soma de 4 bits e um carry-out.

3. **Soma dos 4 bits mais significativos:** Os 4 bits mais significativos de cada registrador, juntamente com o carry-out do primeiro somador, são enviados para o segundo somador de 4 bits. Este somador gera a soma final de 4 bits e o carry-out final.

4. **Saída:** A saída do circuito é a soma dos dois números de 8 bits, representada pelos bits de saída dos dois somadores de 4 bits.

09. Extensor de sinal de 4 bits para 8 bits



Objetivo do Circuito

O circuito em questão é um **extensor de largura de dados**, que transforma uma entrada de 4 bits em uma saída de 8 bits. Esse tipo de circuito é amplamente utilizado em sistemas digitais para expandir a representação de dados, especialmente em aplicações que lidam com números binários e precisam ajustar os tamanhos das palavras para operações subsequentes.

- **Estrutura do Circuito**

- **Entradas**

  Há uma entrada principal de 4 bits, composta por sinais binários (cada bit é representado por 0 ou 1). Essa entrada está conectada ao lado esquerdo do circuito, identificada como "0000" na parte superior da interface gráfica.

- **Saídas**

  O circuito fornece uma saída de 8 bits. Os 4 bits superiores (mais significativos) são preenchidos com zeros e os 4 bits inferiores correspondem diretamente à entrada de 4 bits.

- **Conexões**

  Cada bit de entrada está conectado diretamente às 4 linhas inferiores da saída (bits menos significativos).

As 4 linhas superiores da saída são configuradas para manter sempre o valor lógico "0", realizando a extensão por zero-padding.

- **Funcionamento**

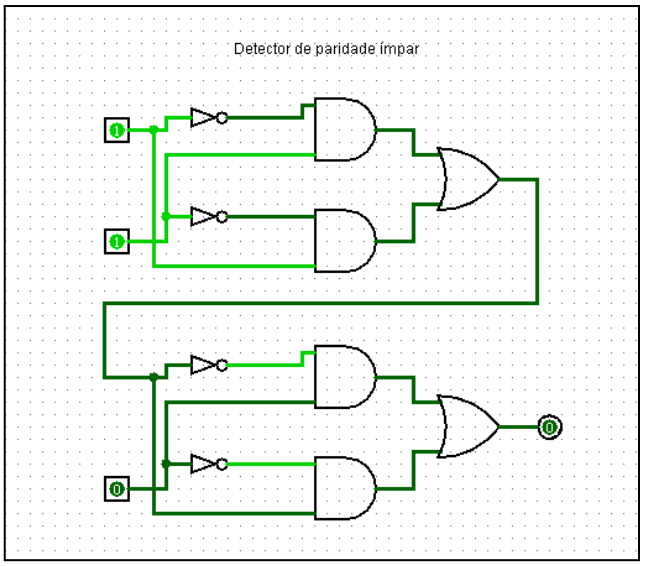
1. **Entrada de Dados:** Quando um valor binário de 4 bits é fornecido à entrada, o circuito copia esse valor para os 4 bits inferiores da saída.
2. **Extensão para 8 Bits:** Os 4 bits superiores da saída são automaticamente definidos como "0", resultando em um valor de saída de 8 bits. Por exemplo:

   Entrada: 1010 (4 bits)

   Saída: 00001010 (8 bits)
3. **Finalidade:** Esse processo é útil quando o sistema precisa padronizar os tamanhos das palavras binárias para operações ou armazenamento, garantindo compatibilidade com barramentos de dados ou unidades de processamento maiores

10. Detector de paridade ímpar

O circuito da figura abaixo configura a lógica para um detector de paridade ímpar, que detecta quando o número das entradas for um número ímpar. A saída desse circuito será verdadeira quando, de acordo com as combinações das entradas, a quantidade de números 1 for ímpar.

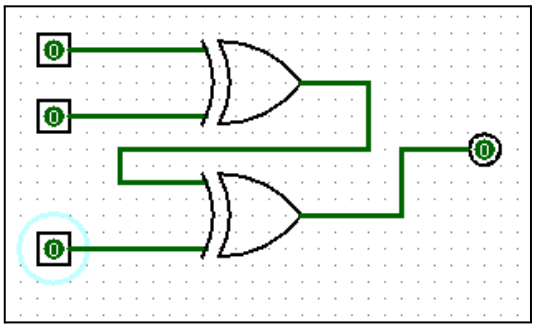


É constituído pela combinação das portas AND, OR e NOT. O detector implementado da figura possui 3 entradas, 1 saída e 8 combinações possíveis, pois $2^3 = 8$, ou seja, conseguimos verificar os números de 0 a 7 e assim podemos construir a tabela verdade. A saída será 1 quando a quantidade de números 1s (as entradas que estão altas) for um número ímpar, no caso do exemplo, os números 1 e 3.

A	B	C	Quantidade de 1s	Saída
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	2	0
1	0	0	1	1
1	0	1	2	0
1	1	0	2	0
1	1	1	3	1

Por exemplo: na 4ª linha temos A = 0, B = 1, C = 1, ou seja, temos 2 números 1, logo a saída será 0, pois 2 é número par. Já na 8ª linha, temos A = 1, B = 1, C = 1. Três números 1, logo a saída = 1, pois 3 é número ímpar.

Se observarmos o circuito, podemos perceber que o circuito é equivalente a composição interna da porta “ou exclusivo”, então podemos construir o circuito dessa maneira com as portas XOR em série:

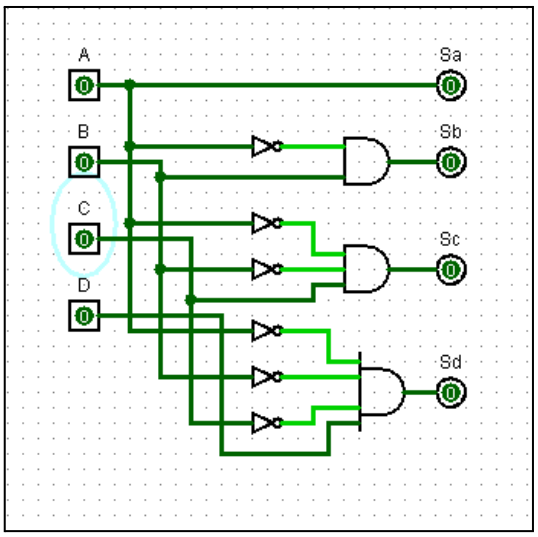


A saída da porta XOR somente é verdadeira quando existe um número ímpar verdadeiro nas suas entradas, ou seja, não são iguais entre si. No exemplo de 3 entradas, podemos fazer a seguinte verificação:

- Comparamos os primeiros termos, A e B, e verificamos se são iguais. Caso sejam, a saída será 0, caso contrário, 1.

- Depois disso, pegamos esse resultado e comparamos com a próxima entrada, o C. Isso pode ser repetido de acordo com a quantidade de entradas a serem analisadas.

11. Otimização de problemas e circuitos lógicos a partir dos mapas de Karnaugh



O mapa de Karnaugh é uma técnica utilizada em otimização de circuitos, para reduzir o seu tamanho e conse O circuito lógico da figura configura um sistema de prioridade, que define qual chave tem prioridade sobre outra.

Nesse exemplo, o circuito possui 4 entradas de dados e é organizado com 6 portas NOT, 3 portas AND, 4 entradas (2^4 = 16 combinações) e 4 saídas correspondentes às entradas, após a otimização.

Seguindo essa lógica, a tabela verdade de 16 linhas pode ser montada da seguinte forma, onde a ordem de prioridade é A → B → C → D. Porém, devemos assegurar que nas situações onde mais de uma chave é acionada, apenas a saída correspondente a entrada de maior prioridade pode ser ativada.

A	B	C	D	Sa	Sb	Sc	Sd
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

A partir disso, podemos extrair as expressões lógicas correspondentes e simplificar por meio dos mapas de Karnaugh. Cada coluna de cada saída deve ser otimizada.

Saída A (Sa)

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

Resultado

$F = A$

Saída B (Sb)

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

Resultado

$F = \overline{A}\,B$

Saída C (Sc)

AB \ CD	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

Resultado

$F = \overline{A}\,\overline{B}\,C$

Saída D (Sd)

AB \ CD	00	01	11	10
00	0	1	0	X
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

Resultado

$F = \overline{A}\,\overline{B}\,\overline{C}\,D$

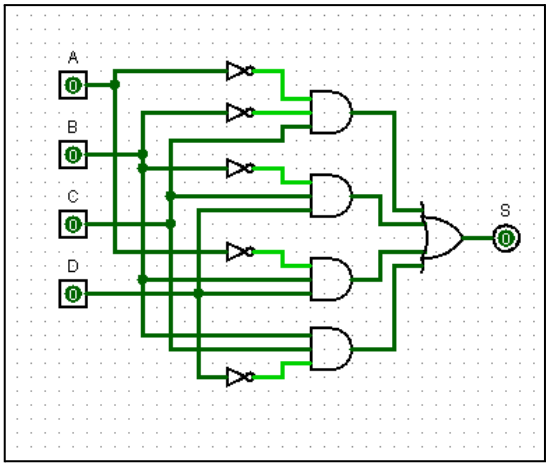
Com isso, obtemos a soma das expressões obtidas, e assim podemos verificar as combinações de acordo com a tabela verdade.

Observando a tabela montada, pode-se observar que existe somente uma saída para todos os casos, o que define a prioridade como mencionado anteriormente. Por exemplo, se a entrada A e B forem acionadas, a saída será Sa, por A tem prioridade sobre B. Agora, se B e C ou B e D forem acionadas, a saída será Sb, pois B tem prioridade sobre C e D. O mesmo acontece se as 4 entradas forem acionadas simultaneamente: A terá prioridade sobre todas as outras e a saída será Sa.

Nas figuras abaixo, podemos observar o circuito inicial e o mesmo depois de otimizado.

12. Detector de Número Primo

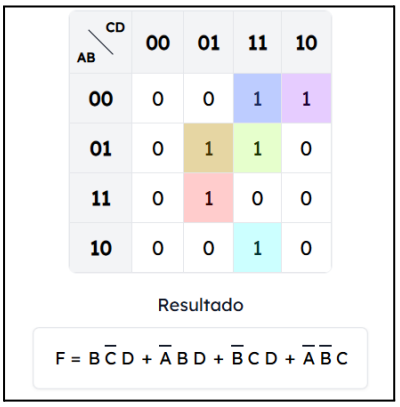
O detector de número primo consiste num circuito constituído por portas lógicas NOT, OR e AND, para verificar se um número das combinações de entrada é primo ou não. O circuito da figura possui 4 entradas e 1 saída, logo $2^4 = 16$ possibilidades de combinação (0 a 15).



Primeiro, vamos verificar quais números entre 0 a 15 são primos, no caso: 2, 3, 5, 7, 11 e 13. A saída será 1 quando o número for primo e caso contrário, 0.

Decimal	Entradas ABCD	Saída (primo)
0	0	0
1	1	0
2	10	1
3	11	1
4	100	0
5	101	1
6	110	0
7	111	1
8	1000	0
9	1001	0
10	1010	0
11	1011	1
12	1100	0
13	1101	1
14	1110	0
15	1111	0

Organizando as saídas em uma mapa de Karnaugh de 4 variáveis, é possível obter a expressão lógica já simplificada. No mapa, é possível obter 4 duplas que resultam em 4 expressões.



AB \ CD	00	01	11	10
00	0	0	1	1
01	0	1	1	0
11	0	1	0	0
10	0	0	1	0

Resultado

$$F = B\overline{C}D + \overline{A}BD + \overline{B}CD + \overline{A}\,\overline{B}C$$

Implementando a expressão em circuito, podemos verificar as combinações obtidas da tabela verdade.

Exemplo: 13 é número primo e seu correspondente em binário é 1101, ou seja, A = 1, B = 1, C = 0, D = 1. Logo, a saída será verdadeira (1). O número 9 não é um número primo, logo a saída será falsa (0), com entradas A = 1, B = 0, C = 0 e D = 1.