# Registrador Flip-Flop do tipo D e do tipo JK:

Um registrador é um circuito de memória fundamental em qualquer processador ou sistema digital. Sua principal função é armazenar um conjunto de bits de dados. Ele é construído a partir de uma coleção de Flip-Flops, onde cada Flip-Flop armazena um único bit.

* Funcionamento do Registrador com Flip-Flop D:
  + Lógica: Ele tem uma entrada de dados D e uma saída Q. Quando o sinal de *clock* é ativado, o valor que está na entrada D é copiado para a saída Q e fica armazenado lá.
  + Aplicação: Um registrador é simplesmente um conjunto de Flip-Flops. Todos eles compartilham o mesmo sinal de clock. Quando o clock pulsa, os dados presentes nas entradas D são "capturados" e armazenados simultaneamente.
* Funcionamento do Registrador com Flip-Flop JK:
  + Lógica: O Flip-Flop JK é mais versátil. Ele tem duas entradas, J e K. Em vez de apenas copiar a entrada, ele pode manter o valor atual (J=0, K=0), forçar a saída para 1 (J=1, K=0), forçar a saída para 0 (J=0, K=1) ou inverter o valor atual (J=1, K=1).
  + Aplicação: Embora seja mais comum em contadores, para usá-lo como um registrador de armazenamento simples, ele é configurado para imitar um Flip-Flop D. Isso é feito conectando a entrada de dados D na entrada J e a versão invertida de D (usando uma porta NOT) na entrada K.

# Multiplexador de quatro opções de entrada:

Um multiplexador funciona como uma chave seletora digital. Ele permite que você escolha qual de várias fontes de entrada será conectada a uma única saída.

* Funcionamento:
  1. Entradas de Dados: Possui quatro linhas de entrada.
  2. Entradas de Seleção: Possui duas linhas de controle, ou seleção. Como, dois bits são suficientes para escolher entre quatro opções.
  3. Saída Única: Possui uma única linha de saída.
* Aplicação: É crucial para selecionar dados em uma ULA ou para rotear dados de diferentes partes de um sistema para um barramento comum.

# Porta lógica XOR usando AND, NOT e OR:

A porta XOR é um bloco lógico fundamental. Sua saída é '1' se, e somente se, suas entradas forem diferentes.

* Funcionamento:
  + Se as entradas A e B são 0 e 0, a saída é 0.
  + Se as entradas A e B são 1 e 1, a saída é 0.
  + Se uma entrada é 0 e a outra é 1, a saída é 1.
* Implementação com outras portas: A expressão booleana para a operação XOR é:

Isso se traduz diretamente no circuito:

* 1. Cria-se o termo (A E NÃO B): A entrada A é conectada a uma porta AND, e a entrada B é primeiro invertida por uma porta NOT e depois conectada à mesma porta AND.
  2. Cria-se o termo (NÃO A E B): A entrada B é conectada a uma segunda porta AND, e a entrada A é primeiro invertida por uma porta NOT e depois conectada a esta segunda porta AND.
  3. As saídas das duas portas AND são então conectadas a uma porta OR. A saída desta porta OR é o resultado da operação XOR.

# Somador de 8 bits que soma com o valor 4:

Este é um circuito aritmético especializado, projetado para uma única tarefa: adicionar o número 4 a qualquer número de 8 bits que ele receba.

* Funcionamento: Ele é construído a partir de um somador de 8 bits padrão.
  1. Entrada A: É um barramento de 8 bits que recebe o número variável (vamos chamá-lo de N).
  2. Entrada B: Em vez de ser uma entrada variável, ela é fixa. O número 4 em binário de 8 bits é 00000100. Portanto, os fios da entrada B são permanentemente conectados da seguinte forma: o terceiro fio (B2) é conectado à tensão alta, e todos os outros sete fios (B0, B1, B3-B7) são conectados ao terra.
  3. Saída: A saída do circuito será sempre o resultado de N + 4.

# 8. Somador de 8 bits:

Este é um dos circuitos aritméticos mais importantes. Sua função é receber dois números binários de 8 bits (A e B) e calcular a soma deles.

# 13. Contador Síncrono:

Um contador é um tipo especial de registrador que passa por uma sequência predeterminada de estados a cada pulso de clock. Sua função é contar eventos ou gerar sequências de controle.

* Funcionamento (Síncrono): A palavra "síncrono" é a chave. Ela significa que todos os Flip-Flops dentro do contador compartilham exatamente o mesmo sinal de clock.
  + Vantagem: Quando o clock pulsa, todos os bits do contador mudam de estado simultaneamente. Isso o torna preciso e confiável, especialmente em altas velocidades, pois evita os atrasos em cascata que ocorrem em contadores "assíncronos" (onde o clock de um flip-flop é a saída do anterior).
  + Lógica: Lógica combinacional adicional (portas AND, OR, etc.) é usada para determinar qual será o próximo estado do contador com base no estado atual. Por exemplo, em um contador crescente, a lógica garante que o número armazenado seja incrementado em 1 a cada pulso de clock.

# 14. Detector de Paridade Ímpar:

A paridade é uma técnica simples de verificação de erros. Um detector de paridade ímpar é um circuito que verifica se um conjunto de bits de entrada contém um número ímpar de '1's.

* Funcionamento:
  + O circuito recebe um conjunto de bits como entrada (por exemplo, 4 bits).
  + Ele conta o número de '1's nessa entrada.
  + Se a contagem for ímpar (1, 3, 5, etc.), a saída do detector é '1'.
  + Se a contagem for par (0, 2, 4, etc.), a saída do detector é '0'.
* Implementação: A função de paridade ímpar é logicamente idêntica à operação XOR em cascata. Para uma entrada de 4 bits A, B, C, D, a saída é simplesmente . O circuito é feito conectando a saída de uma porta XOR à entrada da próxima.

# 17. Detector de Número Primo:

Este é um circuito de lógica combinacional customizado para uma tarefa matemática específica.

* Funcionamento:
  1. Entrada: Recebe um número binário de 4 bits. Este número pode representar qualquer valor decimal de 0 a 15.
  2. Lógica Interna: O circuito é projetado com base em uma tabela verdade. Essa tabela define que a saída deve ser '1' apenas para as entradas que correspondem a números primos. Para 4 bits, os números primos são 2, 3, 5, 7, 11 e 13. Para todas as outras entradas (0, 1, 4, 6, 8, 9, 10, 12, 14, 15), a saída deve ser '0'.
  3. Saída: Produz um único bit de saída. A saída é '1' se o número de entrada for primo e '0' caso contrário.
* Implementação: A partir da tabela verdade, usa-se técnicas como Mapas de Karnaugh para derivar a expressão booleana mais simples possível. Essa expressão é então implementada fisicamente usando portas AND, OR e NOT.