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

ANDERSSON SILVA PEREIRA

ARTHUR CORREIA DE OLIVEIRA RAMOS

LABORATÓRIO DE CIRCUITOS - CODIFICAÇÃO E SIMULAÇÕES

BOA VISTA, RR

2024

ANDERSSON SILVA PEREIRA

ARTHUR CORREIA DE OLIVEIRA RAMOS

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

1 INTRODUÇÃO

Esse trabalho visa a aprendizagem, implementação e entrega de circuitos desenvolvidos e estudados para a disciplina de AOC (Arquitetura e organização de computadores), a estrutura do trabalho contém imagens, explicações e links correlacionadas com o funcionamento dos mesmos, documentado e enviado em repositório do GitHub onde os arquivos dos circuitos estão depositados. O trabalho proposto pelo professor contém 17 questões de componentes básicos de um computador, a turma foi dividida em duplas e usamos o software *Logisim* para simulação e construção dos circuitos.

A lista Lab_Circuitos_AOC tem os seguintes componentes:

COMPONENTE 01	Registrador Flip-Flop do tipo D e do tipo JK.
COMPONENTE 02	Multiplexador de quatro opções de entrada.
COMPONENTE 03	Porta lógica XOR usando os componentes: AND, NOT, e OR.
COMPONENTE 04	Somador de 8 bits que recebe um valor inteiro e soma com o valor 4.
COMPONENTE 05	Memória ROM de 8 bits.
COMPONENTE 06	Memória RAM de 8 bits.
COMPONENTE 07	Banco de Registradores de 8 bits.
COMPONENTE 08	Somador de 8 bits.
COMPONENTE 09	Construa um detector de sequência binária para identificar a sequência "101" em um fluxo de entrada.

COMPONENTE 10	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.
COMPONENTE 11	Extensor de sinal de 4 bits para 8 bits.
COMPONENTE 12	Implemente uma máquina de estados utilizando portas lógicas.
COMPONENTE 13	Contador Síncrono.
COMPONENTE 14	Combine portas AND, OR e NOT para criar a lógica de um detector de paridade ímpar (entrada com número ímpar de 1s)
COMPONENTE 15	Resolva um problema de otimização lógica utilizando mapas de Karnaugh e implemente o circuito otimizado.
COMPONENTE 16	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).
COMPONENTE 17	Detector de Número Primo: Crie um circuito que detecte se uma entrada binária de 4 bits representa um número primo. Utilize portas lógicas e mapas de Karnaugh para simplificar o circuito.

2 DESENVOLVIMENTO

Os circuitos integrados são os principais componentes para o bom funcionamento do computador, são eles que, através da eletricidade, fazem grandes transformações no cotidiano das pessoas.

2.1 CIRCUITO FLIP-FLOP D E JK

Flip-Flop é um circuito de registro de estado, baseado no *Latch SR* para armazenamento de estado, eles necessitam de duas entradas, no *Latch* entrada S (set) e entrada R (reset), no Flip-Flop JK tem as entradas J e K, no Flip-Flop D usamos o circuito anterior com a mesma entrada para as duas, porém uma será negada. Nos Flip-Flops é preciso uma atualização do clock para as saídas serem atualizadas, conforme as figuras a seguir. Nas tabelas verdades o sinal ‘!!’ Na saída é porque não houve atualização.

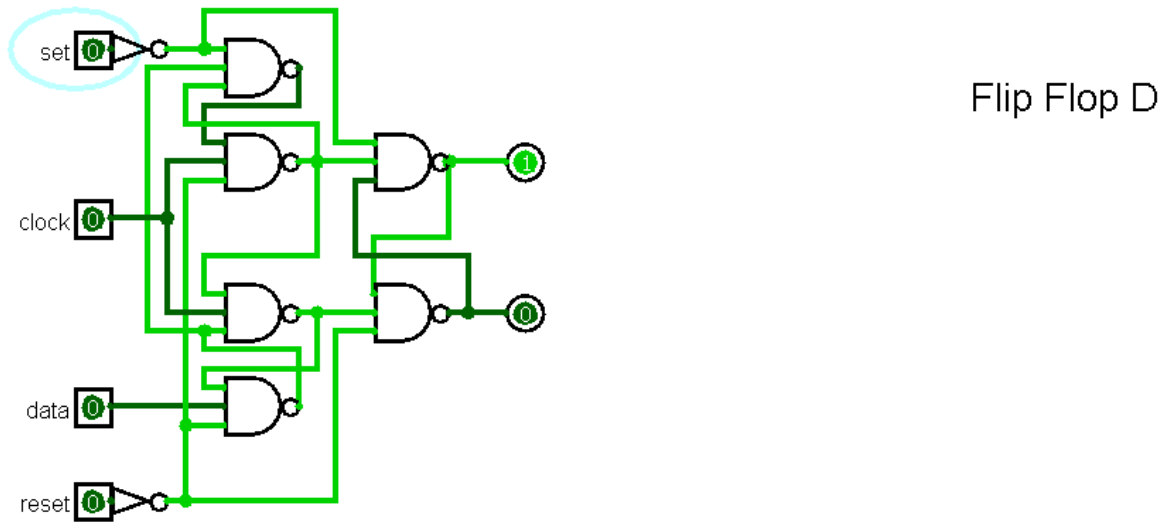


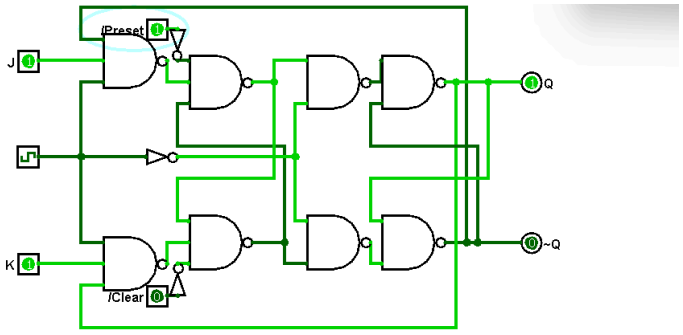
Imagem 1 - Diagrama do circuito Flip-Flop D.

A tabela verdade do Flip-flop D

Set	Clock	Data	Q	Q'
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	0
0	0	0	0	1
0	1	1	0	1

Imagem 2 - Tabela verdade do Flip-Flop D.

Já o flip -flop JK



Flip-Flop-JK

Este é um flip-flop jk, feito com preset e clear, aonde não podemos ligar ambos ao mesmo tempo.

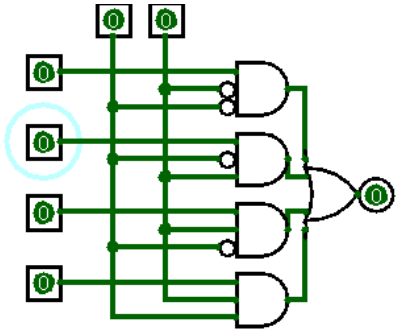
Imagem 3 - Flip-Flop JK.

A tabela verdade:

J	K	Clock	Q	Q'
0	0	0	Q	Q'
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	Q	Q'
1	0	1	1	0
1	1	0	0	1
1	1	1	T	T'

Imagem 4 - Tabela verdade do Flip-Flop JK.

2.2 MULTIPLEXADOR DE 4 ENTRADAS



Multiplexador De 4 opções de entrada

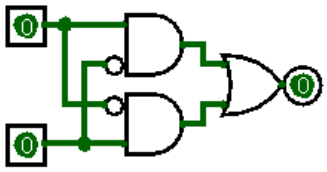
Imagem 5 - Multiplexador de 4 entradas

É um sistema que recebe 4 entradas de dados e tem apenas 1 valor de output onde na imagem acima os inputs a esquerda são as entradas de dados e os inputs acima são os sinais de controle que servem para determinar qual conexão será feita.Ele usa uma lógica combinacional (geralmente feita com portas lógicas) para ativar a entrada correspondente ao valor definido pelos sinais de controle.
Onde sua tabela verdade é:

S1	S0	Entrada Selecionada	Saída Y
0	0	I0	I0
0	1	I1	I1
1	0	I2	I2
1	1	I3	I3

Imagem 6 - Tabela verdade do multiplexador de 4 opções de entradas.

2.3 PORTA XOR



Porta XOR usando portas,OR,NOT e AND.

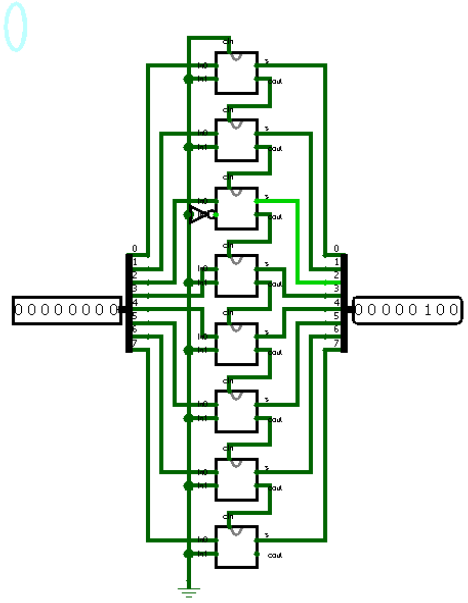
Imagem 7 - Porta XOR usando portas OR,AND e NOT

A porta XOR retorna um valor 1 apenas quando somente uma das entradas é 1 onde a mesma é construída utilizando apenas portas or, and e not. Onde a a entrada A e seu oposto se juntam por meio de portas and a /B e B e após isso são postos em uma porta ou gerando essa tabela verdade.

A	B	¬A	¬B	A · ¬B	¬A · B	A ⊕ B
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

Imagem 8 - Tabela verdade da porta XOR

2.4 SOMADOR DE 8 BITS QUE RECEBE UM VALOR INTEIRO E SOMA COM O VALOR 4.



Somador de 8bits que soma 4

Imagem 9 - Somador de 8 bits a uma constante

Nesse caso foi feita a passagem de um fio-terra em todos os bits com exceção do 3 bit que está setado para dar o valor 100(4 em binário) também o carry in inicial foi considerado 0 e está ligado ao fio terra. Cada circuito encapsulado equivale a um somador de 1 bit que recebe dois valores soma e gera um carry-out o qual este é interligado a o próximo bit e assim por diante até ter-se 8 bits. Somador de 1 bit que é composto por duas entradas o carry-in do último bit é a saída é o carry-out e a soma . A soma é feita por duas portas XOr e o carry-out é feita por 3 portas AND e uma porta OR.

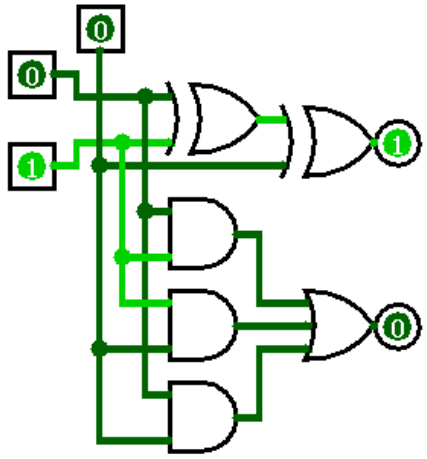


Imagem 10 - Somador de 1 bit

2.5 MEMÓRIA ROM DE 8 BITS

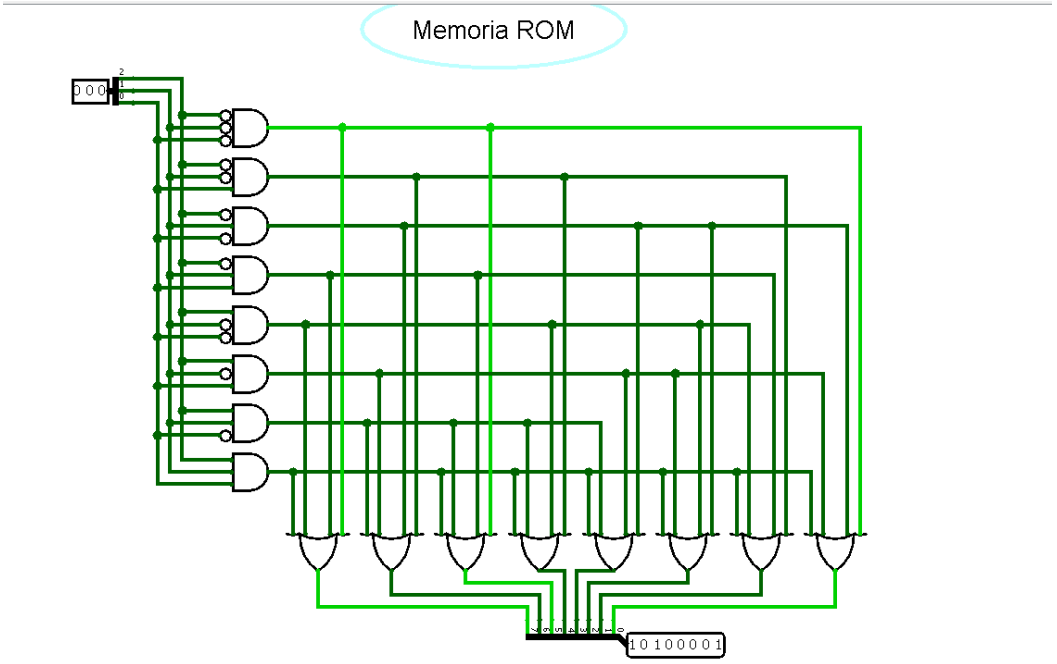


Imagem 11 - Memória ROM

Esse circuito representa uma memória ROM (Read-Only Memory), utilizada para armazenar dados fixos. Ele possui entradas de endereço (na esquerda), que passam por um decodificador formado por portas AND. O decodificador ativa uma única linha correspondente ao endereço selecionado. Essa linha atravessa uma matriz de conexões programadas, onde cada conexão define o valor (1 ou 0) das saídas para aquele endereço. Na parte inferior, as saídas exibem os dados armazenados, que são fixos e previamente programados. A ROM é usada para leitura de dados permanentes, como firmware ou tabelas de consulta.Onde os endereços fixos dela será nesse caso:

Endereço (Decimal)	Endereço (Binário)	Bits de Saída
0	000	10100001
1	001	11001100
2	010	11110000
3	011	00001111
4	100	01010101
5	101	00110011
6	110	11111100
7	111	00000011

Imagem 12 - Tabela de endereçamento da memória ROM

2.6 MEMÓRIA RAM

A memória RAM tem duas principais funções, sendo essa a principal diferença em relação à memória ROM. Enquanto a ROM tem apenas a operação de leitura, a RAM realiza tanto a leitura quanto a escrita. Os principais componentes da memória RAM são a entrada de dados, o endereço e a saída. A entrada de dados define os valores que serão armazenados na memória, enquanto o endereço, representado na imagem como "chave", determina o local exato na memória onde esses dados serão armazenados ou lidos. Outros elementos importantes incluem o controle de escrita e leitura, que define o tipo de operação a ser realizado: se os dados serão armazenados na célula de memória (escrever) ou enviados para a saída (ler). Além disso, o reset é usado para limpar ou inicializar os registradores da memória, e o clock sincroniza todas as operações, garantindo que as mudanças aconteçam de maneira precisa.

Funcionamento:

A operação de escrita na memória ocorre quando o controle de escrita está ativado. Neste caso, os dados presentes na entrada são armazenados no registrador correspondente ao endereço selecionado. Cada endereço está associado a um registrador encapsulado que funciona como uma célula de memória capaz de armazenar os dados. O decodificador de endereço assegura que somente o registrador correspondente ao endereço fornecido seja ativado para armazenar os

dados. O clock desempenha um papel essencial nessa etapa, garantindo que a operação de escrita seja realizada de forma sincronizada, geralmente em uma borda específica do sinal do clock.

Já a leitura da memória acontece quando o controle de leitura é ativado. O endereço fornecido determina qual registrador terá seus dados lidos, e o decodificador seleciona o registrador correspondente. O conteúdo armazenado neste registrador é então enviado para a saída. Como na escrita, o clock também sincroniza a operação de leitura, garantindo que os dados corretos sejam transferidos no momento adequado.

Os registradores encapsulados mostrados na imagem representam as células de memória individuais. Cada registrador possui uma entrada para os dados que serão armazenados, uma saída para leitura e uma conexão com o controle de escrita e leitura. Eles são organizados e controlados pelo decodificador de endereço, que ativa apenas o registrador correspondente durante as operações. O reset, quando acionado, zera todos os registradores, limpando os dados armazenados e iniciando a memória.

Assim, a memória RAM combina diferentes componentes, como registradores encapsulados, controle de leitura e escrita, reset e clock, para realizar de maneira eficiente as operações de leitura e escrita, sendo uma peça fundamental nos sistemas digitais e computacionais.

Memória RAM:

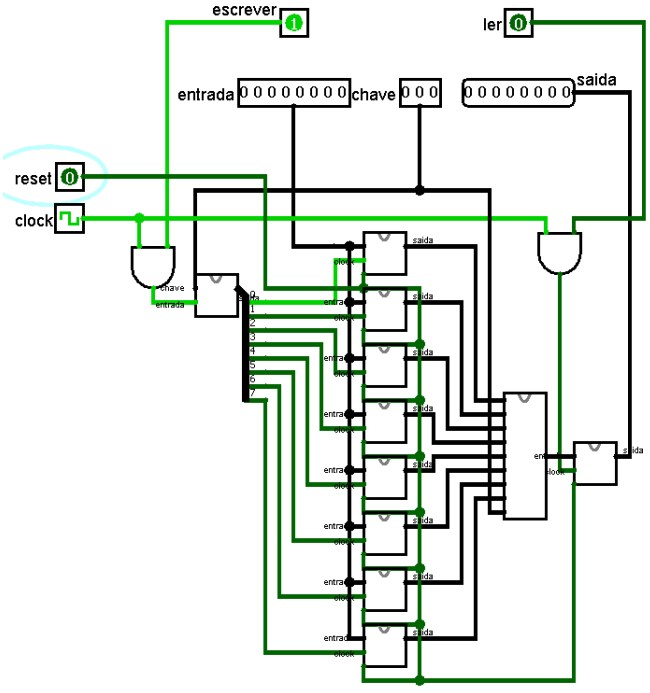


Imagem 13 - Memória RAM de 8 bits

Registrador:

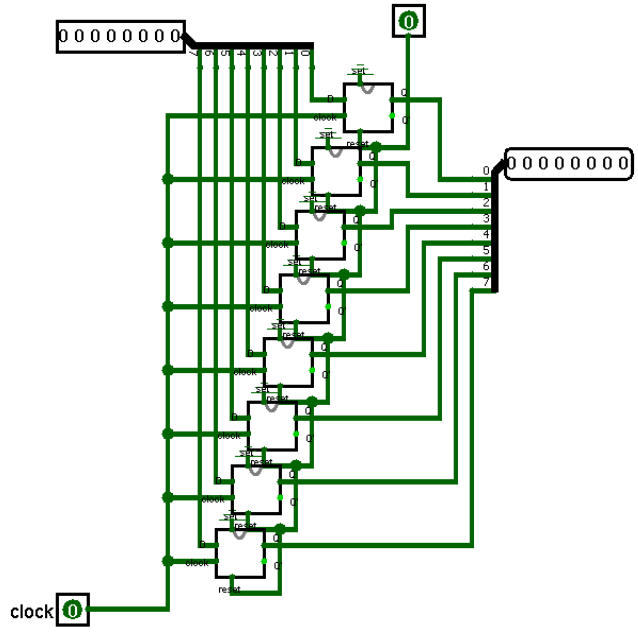


imagem 14 - Registrador

Flip-flop D utilizado no registrador

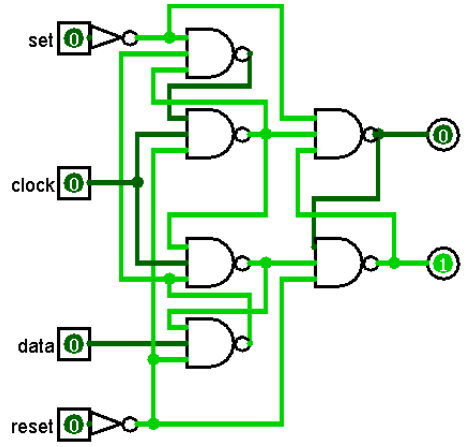


Imagem 15 - Flip Flop d utilizado na memória RAM

2.7 BANCO DE REGISTRADORES DE 8 BITS

O circuito apresentado implementa um banco de registradores, que é uma estrutura digital organizada composta por registradores, utilizada para armazenar e manipular dados em sistemas computacionais. Este circuito utiliza Flip-Flops D, demultiplexadores e multiplexadores para realizar operações de leitura e escrita de maneira controlada. Ele foi projetado para suportar até 16 registradores de 8 bits cada.

O banco de registradores é composto por componentes principais que desempenham funções específicas. Os registradores armazenam os dados de 8 bits e são implementados por Flip-Flops D, que garantem a retenção dos valores até que sejam atualizados. Cada Flip-Flop armazena um único bit, e um conjunto de 8 Flip-Flops forma um registrador. O circuito total possui 16 registradores, oferecendo um total de 128 bits de capacidade de armazenamento.

Para gerenciar a escrita nos registradores, o circuito utiliza um demultiplexador adaptado para lidar com 16 registradores. Ele direciona o sinal de escrita exclusivamente para o registrador selecionado pelo endereço fornecido no sinal A3. O registrador escolhido armazena os dados no próximo pulso de clock, desde que o sinal de habilitação de escrita (WE) esteja ativado.

A leitura dos valores armazenados nos registradores é feita de forma simultânea para dois registradores distintos. Isso é possível graças aos multiplexadores, que permitem selecionar os dados de dois registradores ao mesmo tempo, baseando-se nos endereços fornecidos nos sinais A1 e A2. Cada multiplexador direciona os valores dos registradores selecionados para as saídas RD1 e RD2, possibilitando a leitura simultânea de dois dados diferentes.

O circuito opera sob o controle de entradas e sinais de controle específicos. O clock sincroniza todas as operações do banco de registradores, garantindo que os dados sejam armazenados ou transferidos no momento correto. O sinal reset pode ser ativado para zerar todos os valores armazenados nos registradores. O barramento de dados de 8 bits conecta os registradores ao circuito externo, permitindo tanto a entrada quanto a saída de dados.

O funcionamento do circuito pode ser dividido em dois fluxos principais: escrita e leitura. Durante a escrita, o endereço fornecido em A3 seleciona o registrador, e os dados presentes no barramento de entrada são armazenados no registrador selecionado no próximo pulso de clock, desde que o sinal WE esteja ativado. Para a leitura, os endereços nos sinais A1 e A2 determinam quais registradores terão seus valores transferidos para as saídas RD1 e RD2, através dos multiplexadores.

A interface do circuito é organizada para facilitar o controle e o acesso aos dados. As entradas incluem A3, A1, e A2 (endereços para seleção dos registradores), o barramento de dados de 8 bits, o clock, o reset, e o sinal WE. As saídas, por sua vez, são RD1 e RD2, que fornecem os valores dos registradores selecionados.

Este circuito apresenta diversas vantagens. Ele permite a leitura simultânea de dois registradores, otimizando o acesso aos dados. O uso de endereços compactos para seleção dos registradores torna o circuito eficiente em termos de

controle e espaço. Além disso, os sinais separados para escrita e leitura garantem que não ocorram conflitos durante as operações.

O banco de registradores desempenha um papel crucial em sistemas digitais, sendo amplamente utilizado em processadores para armazenar e manipular dados de forma eficiente. Sua implementação com Flip-Flops D, demultiplexadores e multiplexadores demonstra como a lógica combinacional e sequencial pode ser integrada para criar circuitos versáteis e funcionais.

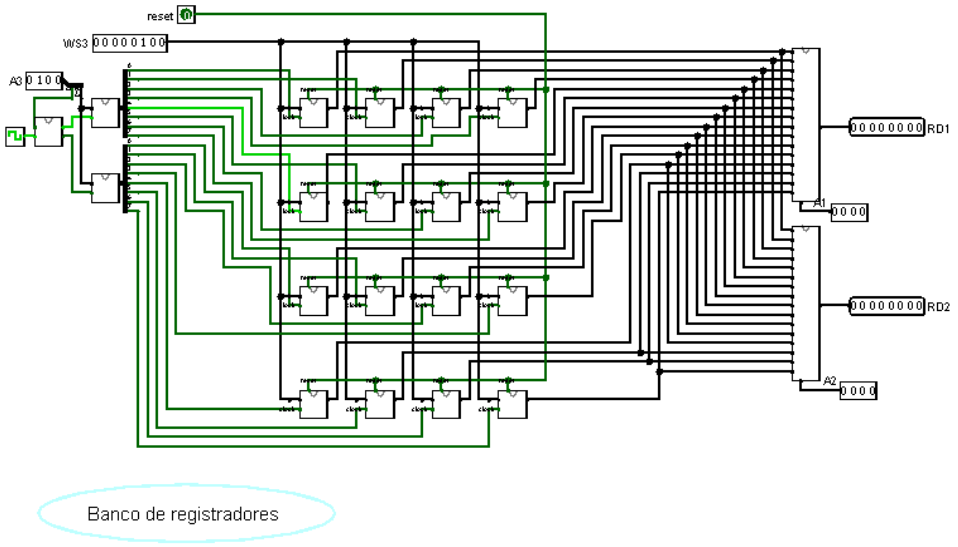


Imagem 16 - Banco de registradores

2.8 SOMADOR DE 8 BITS

O sistema é composto por 8 somadores de 1 bit interligados em cascata, e as entradas consistem em dois números binários de 8 bits que serão somados bit a bit. Além disso, o sistema também recebe como entrada o carry-in. No somador encapsulado, os bits correspondentes são somados.

O processo se inicia pelo bit menos significativo e avança até o mais significativo, garantindo que o carry-out gerado em uma operação seja considerado na soma da próxima posição. A saída do sistema inclui o resultado da soma e o carry-out (caso exista), que é propagado para o próximo somador. Caso ocorra overflow, caracterizado por um número que não pode ser representado corretamente com 8 bits, isso será indicado no carry-out final.Onde no somador de 1 bit utiliza-se uma porta XOR encapsulada representada na imagem abaixo.

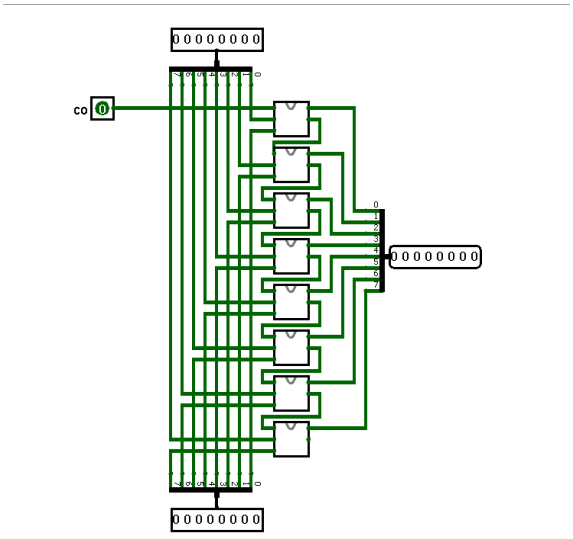


Imagem 17 - Somador de 8 bits

Somador de 1 bit

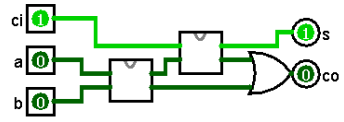
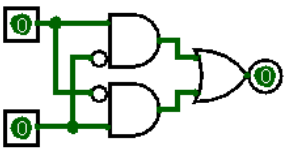


Imagem 18 - Somador de 1 bit

Porta XOR utilizada no somador de 1 BIT



Porta XOR usando portas,OR,NOT e AND.

Imagem 19 - porta Xor usada no somador

2.9 DETECTOR DE SEQUÊNCIA BINÁRIA

Esse circuito é formado por 8 bits de entrada que representam a sequência que será analisada em busca da combinação 101. As portas NAND são responsáveis por verificar, bit a bit, se os valores da entrada correspondem à sequência esperada. Para cada conjunto de 3 bits consecutivos (janelas deslizantes na entrada), uma porta AND é utilizada para combinar os resultados das portas

NAND correspondentes, verificando se o conjunto forma exatamente a sequência 101.A saída final do circuito é determinada por uma porta OR, que será ativada (nível lógico 1) caso qualquer uma das portas AND identifique a sequência "101". Dessa forma, o circuito consegue detectar a ocorrência da sequência esperada em qualquer posição da entrada de 8 bits.

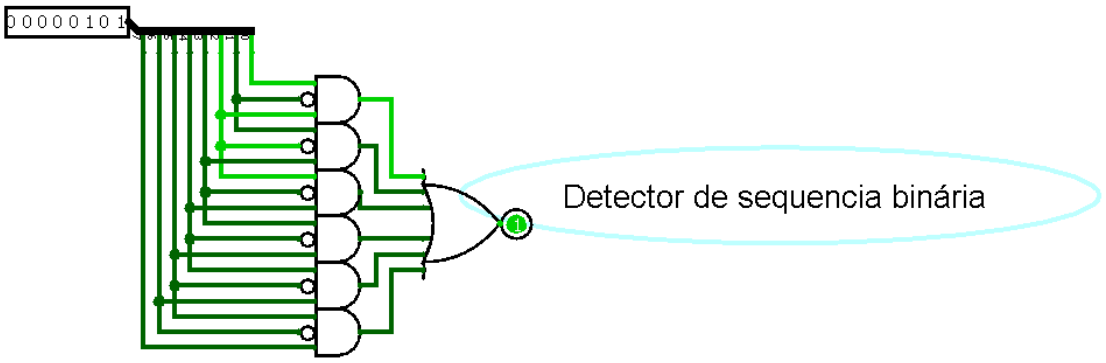


Imagem 20 - Circuito de detector de sequência binária

2.10 ULA

A Unidade Lógica e Aritmética (ULA) é um dos principais componentes de uma Unidade Central de Processamento (CPU), sendo responsável pela realização de operações aritméticas e lógicas essenciais para o funcionamento de sistemas digitais. Este circuito foi implementado no Logisim, com a capacidade de operar com dois conjuntos de 8 bits (entradas A e B) e realizar diversas operações, determinadas por uma entrada de controle chamada "CHAVE". A saída principal do circuito é o resultado da operação realizada, enquanto um bit adicional chamado "overflow" indica se houve estouro de capacidade em operações aritméticas.

As operações suportadas pelo circuito incluem tanto funções lógicas quanto aritméticas. Entre as funções lógicas estão: AND, OR, NOT (para A ou B), NOR, NAND e XOR. Cada uma dessas operações é implementada por portas lógicas específicas que processam os bits correspondentes das entradas A e B. Por exemplo, a operação AND realiza a conjunção lógica entre os bits das duas entradas, enquanto o XOR realiza a disjunção exclusiva. Essas funções permitem manipular diretamente os dados binários.

Para as operações aritméticas, a ULA é capaz de realizar soma e subtração. Ambas as operações utilizam um circuito compartilhado de somador/subtrator, no qual uma chave específica determina se o circuito irá somar ou subtrair os valores de A e B. Além disso, este circuito verifica a ocorrência de overflow, que ocorre quando o resultado de uma operação excede o intervalo que pode ser representado pelos 8 bits do sistema.

Outra funcionalidade importante implementada na ULA é o deslocamento de bits, que inclui o deslocamento para a esquerda e para a direita para as entradas A e B. Esses deslocadores são circuitos dedicados que movem os bits de um lado para o outro, preenchendo com zeros os espaços resultantes. Este recurso é útil para operações de multiplicação e divisão binária, além de manipulação de dados.

A operação realizada pela ULA é selecionada pela entrada "CHAVE", que é composta por 4 bits. Cada combinação de bits na chave corresponde a uma operação específica. Por exemplo, "0000" representa uma operação AND, enquanto "1011" representa a soma. Com base na chave, o circuito ativa os componentes necessários para realizar a operação desejada, enquanto desativa os demais. O resultado é então direcionado para a saída principal.

O circuito interno da ULA é formado por componentes como um somador/subtrator para operações aritméticas e deslocadores de bits para manipulações específicas. Esses blocos são conectados a um conjunto de portas lógicas que implementam as operações lógicas. A combinação desses elementos permite que a ULA execute um conjunto diversificado de operações com eficiência.

Por fim, a ULA também é responsável por sinalizar condições especiais, como o overflow. Isso é particularmente importante em operações aritméticas, onde o estouro de capacidade pode indicar um erro ou uma condição que precisa ser tratada no sistema. A flexibilidade e eficiência desse design tornam a ULA um componente essencial em processadores, controladores e sistemas digitais em geral.

ULA:

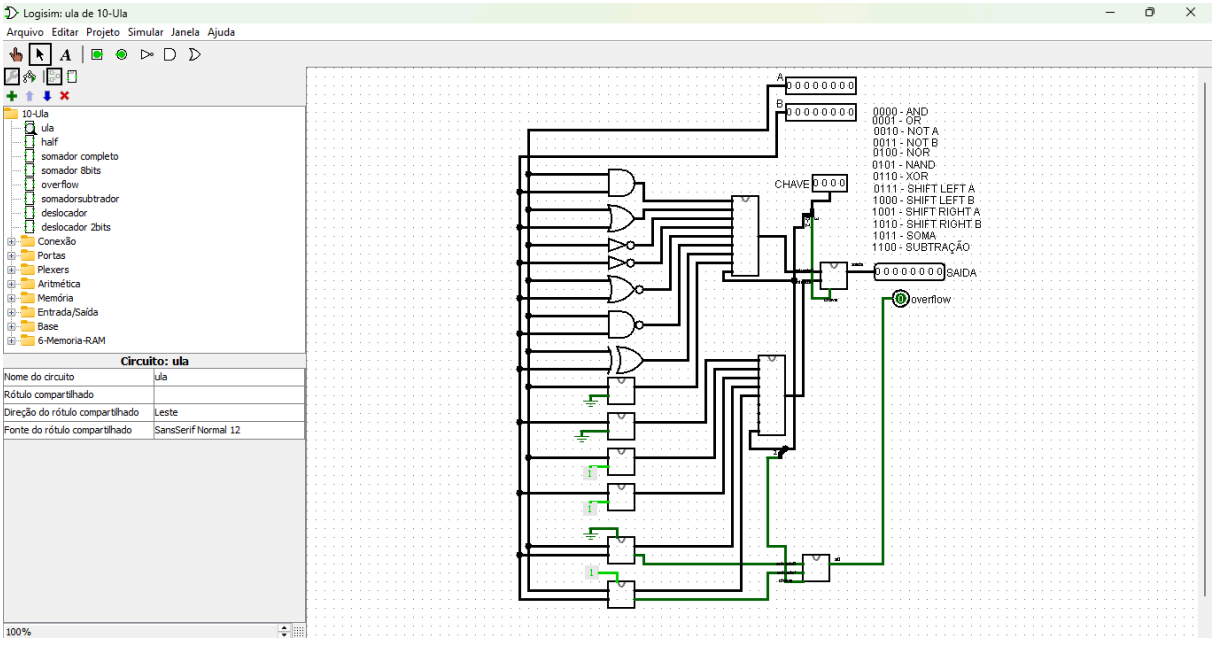


Imagem 21 - Circuito completo da ULA

Operações aritméticas na ULA:

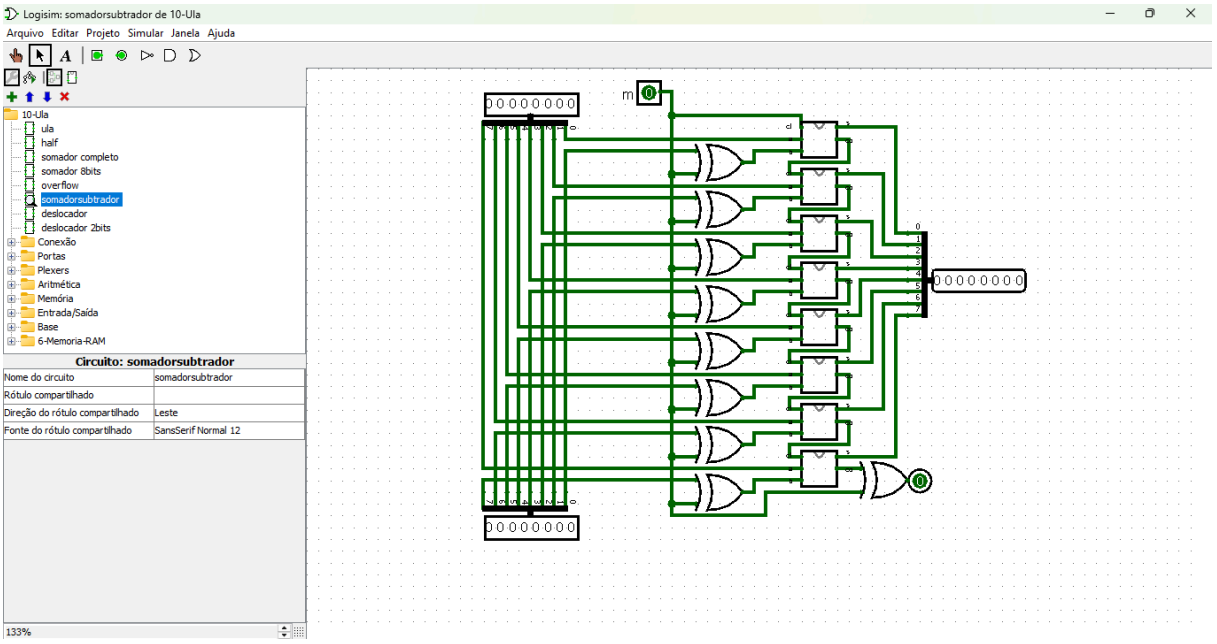


Imagem 22 - Circuito responsável pelas operações aritméticas da ULA

Operações de Deslocamento:

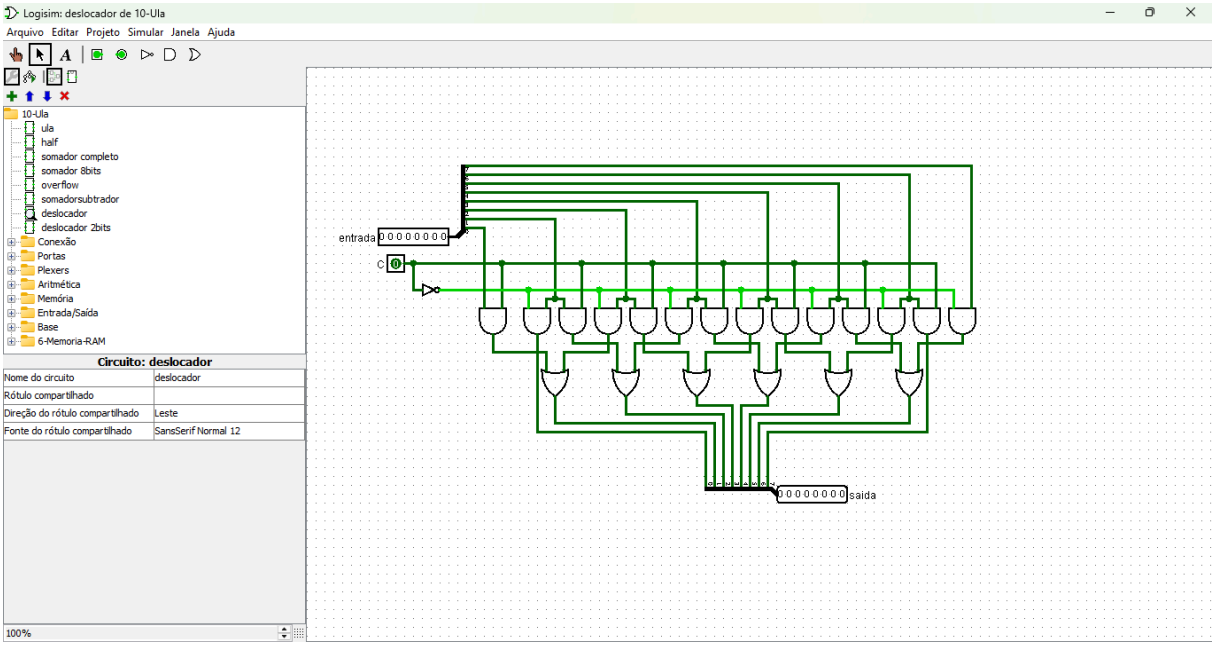


Imagem 23 - Circuito responsável por operações de deslocamento da ULA

E as tabelas das operações previamente descritas ficariam da seguinte forma:

Tabela - Operações de Deslocamento

Entrada A	Operação	CHAVE	SAIDA
01010101	Shift Left A	0111	10101010
01010101	Shift Right A	1001	00101010

Imagem 24 - Tabela de operações de deslocamento

Tabela - Operações Aritméticas

Entrada A	Entrada B	Operação	CHAVE	SAIDA	Overflow
01010101	00110011	Soma	1011	10001000	0
01010101	10101010	Subtração	1100	10101011	1

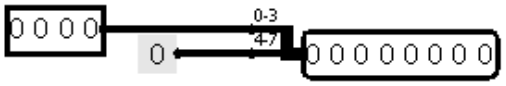
Imagem 25 - Tabela de operações aritméticas

Tabela - Operações Lógicas

Entrada A	Entrada B	Operação	CHAVE	SAIDA
11001100	10101010	AND	0000	10001000
11001100	10101010	OR	0001	11101110
11001100	N/A	NOT A	0010	00110011

Imagem 26 - Tabela de operações lógicas

2.11 EXTENSOR DE SINAL DE 4 PARA 8 BITS



Extensor de 4 para 8 bits

Imagem 27 - Extensor de 4 para 8 bits

Composto por uma entrada de 4 bits que será adicionada a 4 bits menos significativos totalizando assim um output de 8 bits, os bits de entrada são alocados nos primeiros slots da direita para a esquerda.

Exemplos de uso:

Input (4 bits)	Output (8 bits)	Extension Type
1010	00001010	Zero-Extension
0111	00000111	Zero-Extension
1101	11111101	Sign-Extension

Imagem 28 - Exemplo de uso do extensor

2.12 MÁQUINA DE ESTADO

Utiliza um flip-flop para armazenar o estado atual, com a lógica de transição implementada através de portas lógicas (NAND e OR). A entrada do sistema, combinada com o estado atual armazenado no flip-flop, é processada pelas portas NAND para determinar o próximo estado. A porta OR combina os sinais das portas NAND e gera a entrada para o flip-flop, atualizando o estado na borda do clock.

Os componentes que foram utilizados além das portas not e and foi um circuito flip-flop D que serve para armazenar o estado atual. Já o clock serve para sincronizar a mudança de estados.

Imagem da máquina:

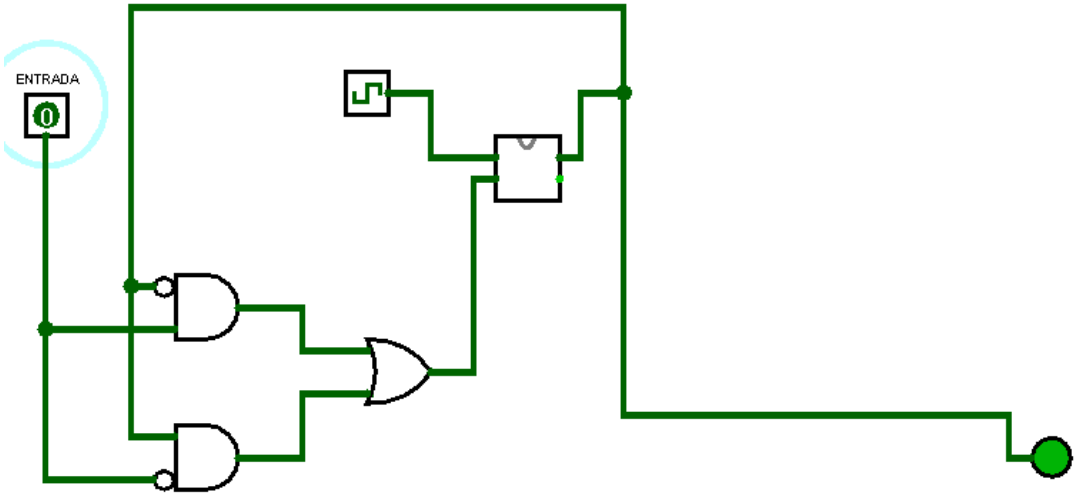


Imagem 29 - Máquina de estado

A Tabela com as principais informações seria

Entrada (E)	Estado Atual (Q)	Próximo Estado (Q+)	Saída (S)
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

Imagem 30 - Tabela dos estados da máquina de estado

2.13 CONTADOR SÍNCRONO

O contador síncrono apresentado é um circuito sequencial que utiliza flip-flops controlados pelo mesmo sinal de clock para contar pulsos em sequência binária. Ele possui 4 flip-flops, cada um representando um bit, permitindo uma contagem de 0 a 15 (em 4 bits). A cada pulso de clock, o flip-flop menos significativo (LSB) alterna seu estado, enquanto os flip-flops subsequentes mudam de estado com base em condições definidas por portas AND, que monitoram os estados dos flip-flops anteriores. Isso garante que o contador siga a contagem binária corretamente. Quando o contador atinge o valor máximo (1111), ele retorna ao estado inicial (0000) no próximo pulso, reiniciando a contagem. O design síncrono garante transições simultâneas, evitando atrasos

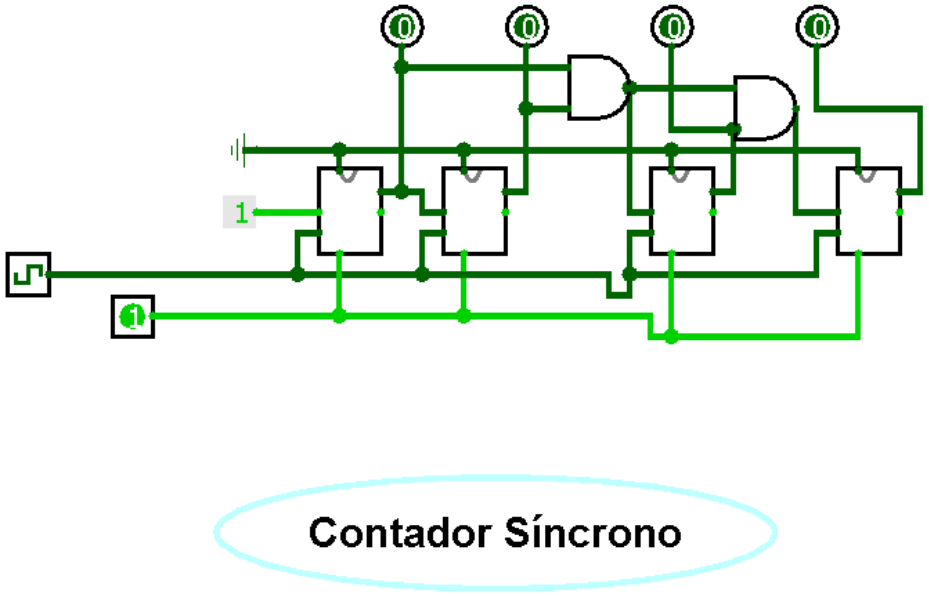


Imagem 31 - Circuito do contador síncrono

Flip flop JK encapsulado do contador

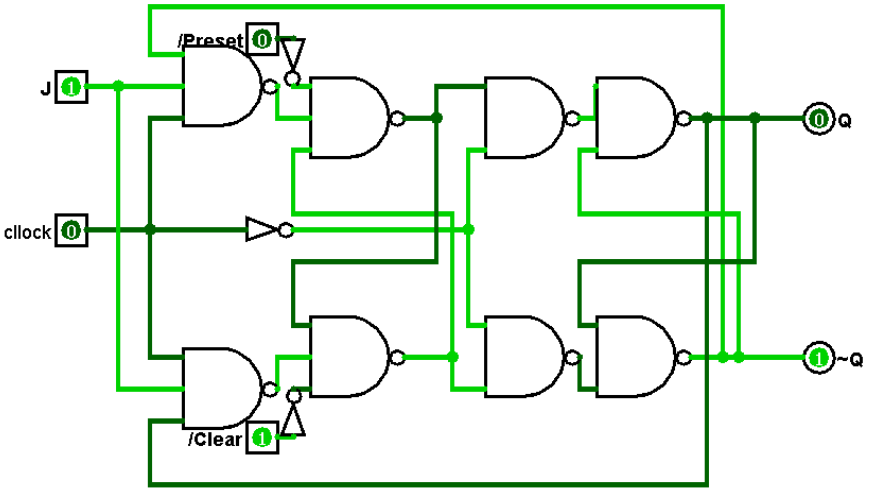


Imagem 32 - Circuito JK encapsulado no Contador síncrono

2.14 DETECTOR DE PARIDADE ÍMPAR

O detector de paridade ímpar verifica se o número de bits 1 em uma entrada de 8 bits é ímpar. Ele utiliza portas XOR, que acumulam a paridade dos bits em camadas. Se o número de bits 1 for ímpar, a saída será 1; caso contrário, será 0.

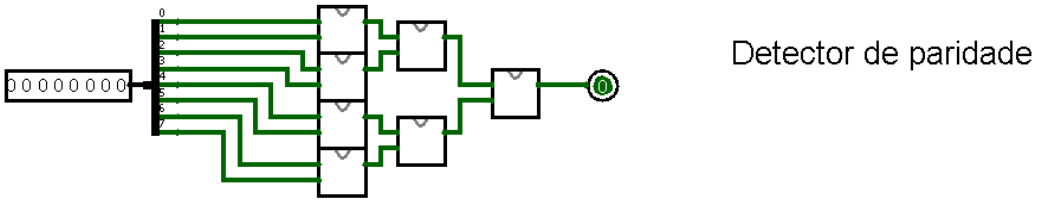


Imagem 33 - Circuito de detector de paridade

A porta XOR utilizada:

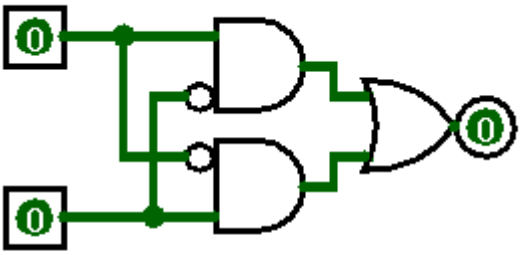


Imagem 34 - Porta XOR utilizada

Exemplos de utilização:

Entrada (8 bits)	Paridade Ímpar Detectada
11010011	Sim (1)
10101010	Não (0)
11111111	Sim (1)

Imagem 35 - Exemplo de saída do detector

2.15 OTIMIZAÇÃO LÓGICA COM MAPA DE KARNAUGH

Com base na equação original

S=A′B′C+ABC′+A′BC′+ABC

a simplificação da mesma foi feita da seguinte forma primeiramente agrupa-se os termos que possuem fatores comuns: S=C(A′B′+AB)+C′(A′B+AB′) . S = C(A'B' + AB) + C'(A'B + AB') . S=C(A′B′+AB)+C′(A′B+AB′).Após isso, aplica-se às propriedades de álgebra booleana o qual que para os termos A′B′+ABA'B' + ABA′B′+AB, aplica-se a propriedade do XOR: A′B′+AB=(A⊕B)′A'B' + AB = (A \oplus B)'A′B′+AB=(A⊕B) Para os termos A′B+AB′A'B + AB'A′B+AB′, aplica-se novamente o XOR: A′B+AB′=A⊕BA'B + AB' = A \oplus BA′B+AB′=A⊕BSubstituição na Equação: Substitui-se os resultados simplificados:S=C(A⊕B)′+C′(A⊕B)S = C(A \oplus B)' + C'(A \oplus B)S=C(A⊕B)′+C′(A⊕B)Propriedade do Multiplexador: O circuito agora tem a forma de um multiplexador controlado por CCC. Analisando a tabela verdade, percebe-se que o circuito sempre resulta em:
S=C ,S =C ,S=C.

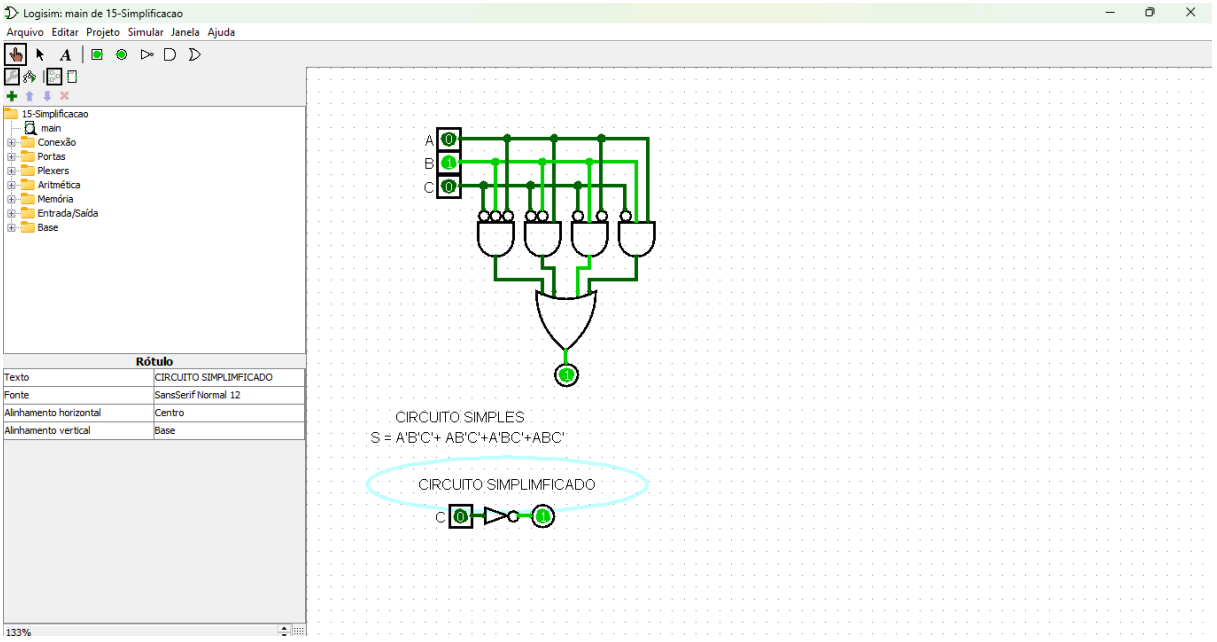


Imagem 36 - Circuito antes e após a simplificação

Onde a tabela verdade é :

A	B	C	Saída (S)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Imagem 37 - Tabela verdade do circuito

2.16 DECODIFICADOR DE 7 SEGMENTOS

O decodificador de 7 segmentos é um circuito digital utilizado para controlar displays de 7 segmentos, comuns em dispositivos eletrônicos para exibir números e caracteres alfanuméricos. Ele recebe como entrada um valor binário de 4 bits, representando números de 0 a 15 no formato hexadecimal, e traduz essas entradas em sinais que determinam quais segmentos do display serão acesos.

O display de 7 segmentos é composto por 7 LEDs dispostos em forma de "8". Cada LED, ou segmento, é identificado por uma letra de a a g. Para representar um número ou letra, o decodificador ativa (acende) os segmentos apropriados e desativa os demais. Por exemplo, para o número "2", os segmentos a, b, g, e e d são ativados, enquanto os segmentos c e f permanecem apagados.

O decodificador utiliza um conjunto de portas lógicas (AND, OR e NOT) para implementar a lógica combinacional necessária. Cada segmento possui um circuito lógico exclusivo que processa as entradas binárias e determina se o segmento será ativado ou não. Esses circuitos são projetados com base na tabela verdade do decodificador, que define como as entradas binárias devem ser traduzidas para ativar os segmentos correspondentes.

Internamente, o decodificador possui um bloco lógico que combina as entradas binárias (A, B, C e D) para gerar sinais de saída (a, b, ..., g). Esses sinais são então enviados para os segmentos do display, que exibem o número ou letra desejados. A lógica é repetida para cada segmento, garantindo que o padrão correto seja exibido para cada entrada.

Assim, o decodificador funciona como um tradutor entre o valor digital e a representação visual, sendo essencial em sistemas digitais que precisam exibir informações de maneira legível ao usuário.

Os componentes necessários para a sua montagem são:
Entradas Binárias:

- A, B, C, D: 4 linhas de entrada representando números ou letras no formato binário (0 a 15 em hexadecimal).

Saídas para os Segmentos:

- a, b, c, d, e, f, g: 7 saídas correspondentes aos segmentos do display de 7 segmentos.

Portas Lógicas:

- AND: Utilizadas para combinar entradas e gerar saídas baseadas em condições específicas.
- OR: Utilizadas para combinar múltiplas condições lógicas.
- NOT: Utilizadas para inverter sinais de entrada.

Tabela Verdade:

- Define a lógica combinacional necessária para ativar os segmentos corretos com base nas entradas binárias.

Bloco Lógico Interno:

- Circuito combinacional que implementa a lógica baseada na tabela verdade para cada segmento (a a g).

Display de 7 Segmentos:

- Dispositivo que recebe os sinais de saída do decodificador e exibe o padrão correspondente, formado pelos LEDs organizados de a a g.

Decodificador:

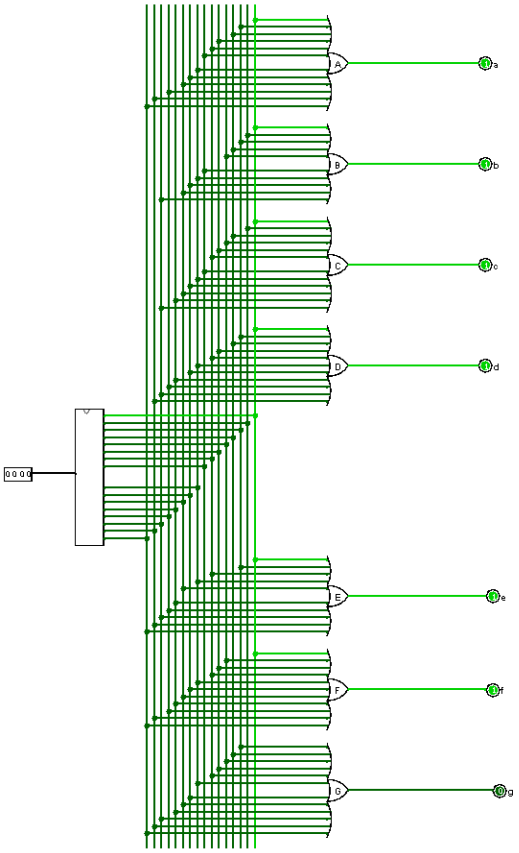


Imagem 38 - Decodificador de 7 segmentos

Circuito Interno encapsulado

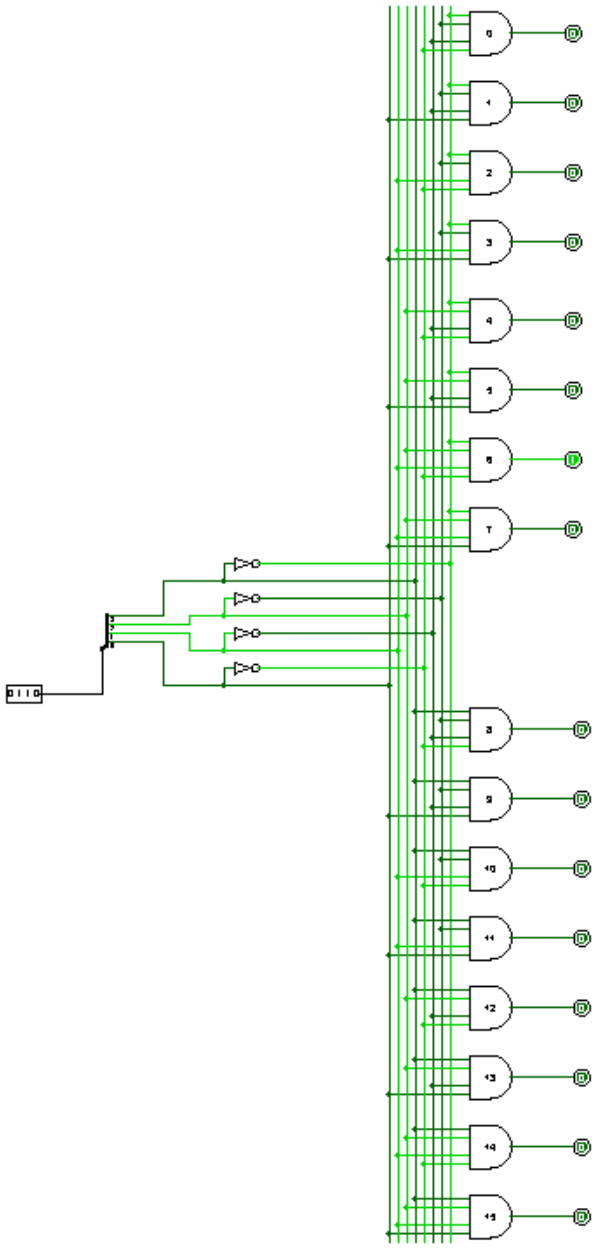


Imagem 39 - Circuito interno encapsulado no decodificador

Tabela verdade do circuito:

Entrada (Binário)	Hexadecimal	a	b	c	d	e	f	g
0000	0	1	1	1	1	1	1	0
0001	1	0	1	1	0	0	0	0
0010	2	1	1	0	1	1	0	1
0011	3	1	1	1	1	0	0	1
0100	4	0	1	1	0	0	1	1
0101	5	1	0	1	1	0	1	1
0110	6	1	0	1	1	1	1	1
0111	7	1	1	1	0	0	0	0
1000	8	1	1	1	1	1	1	1
1001	9	1	1	1	1	0	1	1
1010	A	1	1	1	0	1	1	1
1011	B	0	0	1	1	1	1	1
1100	C	1	0	0	1	1	1	0
1101	D	0	1	1	1	1	0	1
1110	E	1	0	0	1	1	1	1
1111	F	1	0	0	0	1	1	1

Imagem 40 - Tabela verdade do decodificador

2.17 DETECTOR DE NÚMERO PRIMO COM MAPA DE KARNAUGH

O circuito é composto por 4 entradas de 4 bits que representam um número binário de 0 a 15, portas lógicas AND e OR que dizem se o número é primo ou não.Cada termo da fórmula corresponde a uma condição que representa um número primo:A'B'C: Número 3.A'CD: Número 5.BCD': Número 7.B'CD: Número 11.
Ou seja , cada porta AND faz o teste de um desses números onde caso uma porta passe o valor de 1 então o número será primo

O circuito:

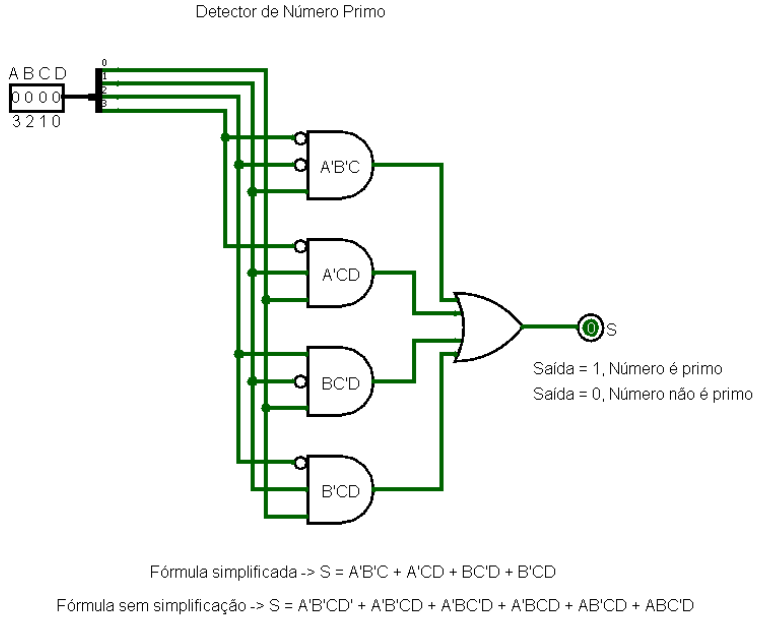


Imagem 41 - Circuito de detector de números primos

a tabela verdade:

Entrada (Binário)	Número Decimal	Saída (S)
0000	0	0
0001	1	0
0010	2	0
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	0
1110	14	0
1111	15	0

Imagem 42 - Tabela verdade do circuito de detecção de números primos

3 CONCLUSÃO

A construção e análise dos circuitos, como Flip-Flops D e JK, bem como o banco de registradores e a ULA, permite compreender profundamente a complexidade intrínseca desses sistemas digitais. Durante o processo de montagem, é possível observar como os diversos componentes interagem para executar funções específicas, desde operações lógicas e aritméticas até o armazenamento e manipulação de dados. A complexidade aparente desses circuitos se traduz em um comportamento lógico previsível e eficiente, demonstrando como a organização e integração de portas lógicas, multiplexadores, demultiplexadores e Flip-Flops podem ser utilizadas para criar soluções funcionais em sistemas computacionais. Essa prática proporciona um entendimento mais sólido da base da eletrônica digital e da construção de arquiteturas computacionais.

REFERÊNCIAS:

KUNZLE, A. **Mapa de Karnaugh**. Disponível em: https://www.inf.ufpr.br/kunzle/disciplinas/ci068/2019-2/slides/aula9_mapa_de_karnaugh.pdf. Acesso em: 10 dez. 2024.

LOGISIM EVOLUTION. **Logisim Evolution**. Disponível em: https://github.com/logisim-evolution/logisim-evolution. Acesso em: 10 dez. 2024.

PINHO, Bruno. **Sistemas Digitais - Aula 14: Mapas de Karnaugh (Parte 1)**. YouTube, 20 mar. 2020. Disponível em: https://www.youtube.com/watch?v=Dp32ur04XcQ&list=PLFnx8Vsp0KYkhig6WCHnU-kt-1bugtl7e. Acesso em: 10 dez. 2024.

Repositório contendo os circuitos citados : https://github.com/Moab76/AOC_AnderssonSilvaArthurRamos_UFRR_LabCircuitos_2024