



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

RYAN PIMENTEL DE OLIVEIRA
CLEILLYSON OSMAR SOUZA DINIZ DE ALMEIDA

LABORATÓRIO DE CIRCUITOS

BOA VISTA, RR

2024

RYAN PIMENTEL DE OLIVEIRA
CLEILLYSON OSMAR SOUZA DINIZ DE ALMEIDA

LABORATÓRIO DE CIRCUITOS

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	COMPONENTES	7
1.1	REGISTRADOR <i>FLIP-FLOP</i> DO TIPO D E DO TIPO JK.....	7
1.2	MULTIPLEXADOR	8
1.3	XOR	8
1.4	SOMADOR DE 8 <i>BITS</i> E SOMA MAIS 4	9
1.5	SOMADOR DE 8 <i>BITS</i>	10
1.6	MEMÓRIA <i>ROM</i>	10
1.7	MEMÓRIA <i>RAM</i> DE 8 <i>BITS</i>	11
1.8	BANCO DE REGISTRADORES DE 8 <i>BITS</i>	12
1.9	DETECTOR DE SEQUÊNCIA BINÁRIA.....	14
1.10	ULA DE 8 <i>BITS</i>	14
	1.10.1 Entradas	14
	1.10.2 Saídas.....	15
	1.10.2 Operações Disponíveis (Controladas pelo SEL)	15
1.11	EXTENSOR DE SINAL 4 <i>BITS</i> PARA 8 <i>BITS</i>	15
1.12	MÁQUINA DE ESTADOS: SEMÁFORO	15
1.13	CONTADOR SÍNCRONO	16
1.14	PARIDADE ÍMPAR	17
1.15	OTIMIZAÇÃO LÓGICA	18
1.16	DECODIFICADOR DE 7 SEGUIMENTOS.....	18
1.17	DETECTOR DE NÚMEROS PRIMOS	20

LISTA DE ILUSTRAÇÕES

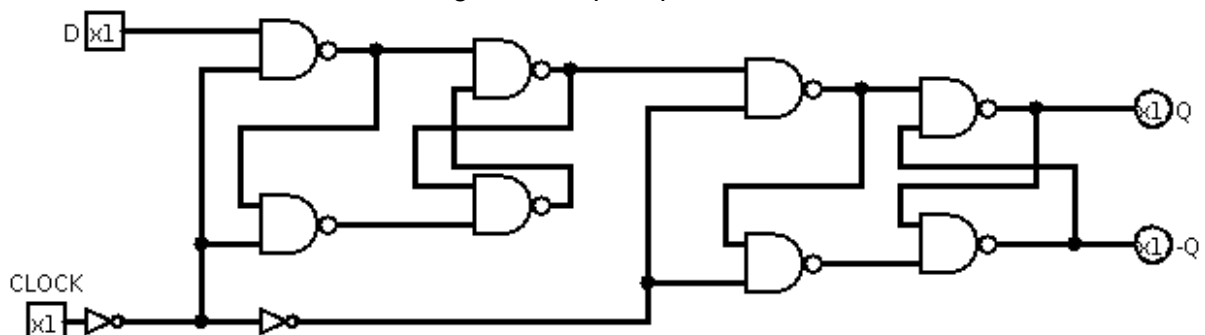
Figura 1 - Flip-Flop JK.....	7
Figura 2 - Flip-Flop D	7
Figura 3 - Multiplexador de 4 opções de entrada	8
Figura 4 - Porta logica XOR	8
Figura 5 - Somador de 1 <i>bit</i>	9
Figura 6 - Somador de 8 <i>bits</i> mais 4	10
Figura 7 - Somador de 8 <i>bits</i>	10
Figura 8 - Memória ROM.....	11
Figura 9 - Componente "palavra"	12
Figura 10 - Memória RAM	12
Figura 11 - Registradores.....	13
Figura 12 - Banco de registradores	13
Figura 13 - Detector de sequência binaria	14
Figura 14 - Extensor de sinal.....	15
Figura 15 - Semáforo de pedestres.....	16
Figura 16 - Contador síncrono.....	17
Figura 17 - Paridade ímpar.....	17
Figura 18 - Circuito simplificado	18
Figura 19 - <i>Display</i> hexadecimal	20
Figura 20 - Número primo	20

LISTA DE TABELAS

Tabela 1- Tabela verdade multiplexador	8
Tabela 2 - Tabela verdade somador 1 <i>bit</i>	9
Tabela 3 - Tabela dos endereços da ROM	11
Tabela 4 - Tabela de paridade ímpar	18
Tabela 5 - Tabela dos números primos.....	21

LISTA DE ABREVIATURAS E SIGLAS

FF	<i>Flip-Flop</i>
RAM	<i>Random Access Memory</i>
ROM	<i>Read-Only Memory</i>
ULA	Unidade Lógica e Aritmética



1.2 Multiplexador

O multiplexador apresenta 4 entradas de dados, 2 de controle e uma saída, as entradas de dados controlam qual vai ser a saída que será exibida.

Figura 3 - Multiplexador de 4 opções de entrada

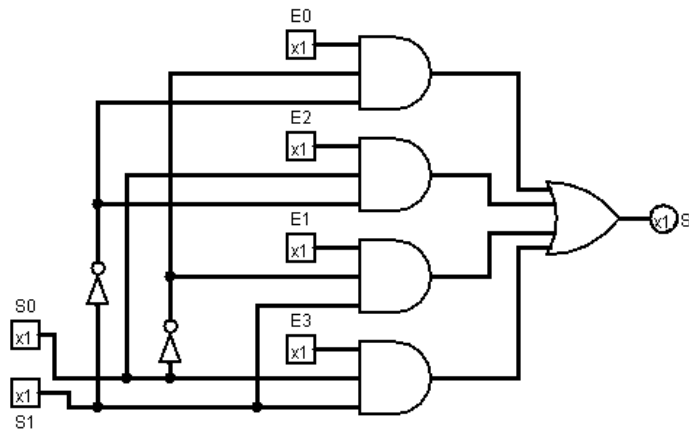


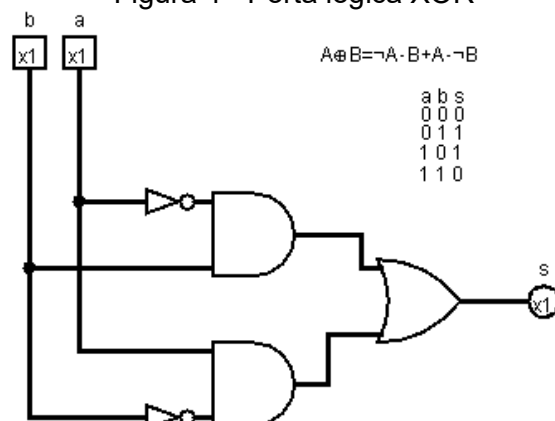
Tabela 1- Tabela verdade multiplexador

Controle		Entrada ativa
S0	S1	S
0	0	E0
0	1	E1
1	0	E2
1	1	E3

1.3 Xor

Essa porta logica apesar de simples é de extrema importância para simplificar vários circuitos, é o “ou exclusivo” onde a saída é 1 quando apenas uma das entradas é 1.

Figura 4 - Porta logica XOR



1.4 Somador de 8 *bits* e soma mais 4

O somador tem 2 partes fundamentais o somador geral e o somador *bit a bit*, o somador *bit a bit* tem 3 entradas: A, B são os números originais e *carry in* que é o resto da soma do *bit* anterior, as três entradas são somadas. O componente tem 2 saídas: S que é o resultado da soma que fica e o *carry out* que vai ser somado com o próximo *bit*.

Figura 5 - Somador de 1 *bit*

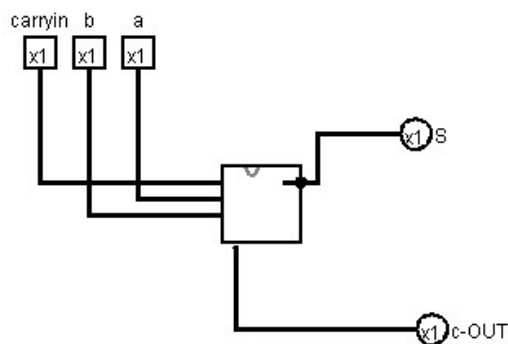
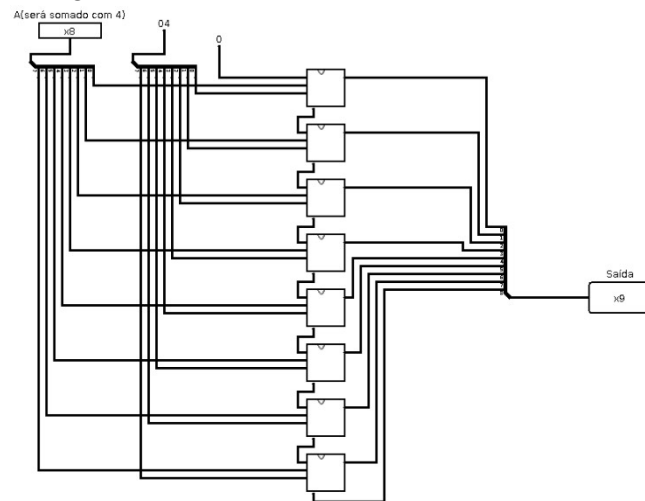


Tabela 2 - Tabela verdade somador 1 *bit*

A	B	carryin	S	c-OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A segunda parte o somador de 8 *bits* usa 8 somadores de *bit a bit*, esse componente tem uma entrada de 8 *bits*, pois o número somado será o 4, e uma saída de 9 *bits* para não dar *overflow*

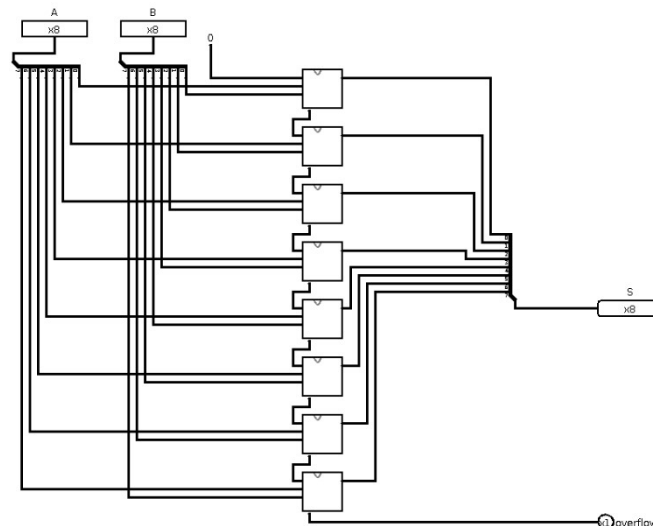
Figura 6 - Somador de 8 *bits* mais 4



1.5 Somador de 8 *bits*

O somador de 8 *bits* é idêntico ao somador de 8 *bits* anterior com exceção de que ele tem duas entradas de 8 *bits* ao invés de apenas uma e apresenta duas saídas uma de 8 *bits* e outra de 1 *bit* para *overflow*.

Figura 7 - Somador de 8 *bits*



1.6 Memória ROM

A memória ROM é um tipo de memória não volátil, utilizada para armazenar dados permanentes e instruções essenciais para o funcionamento de sistema. A nossa ROM uma entrada de 3 *bits* e saída de 8 *bits*.

Figura 8 - Memória ROM

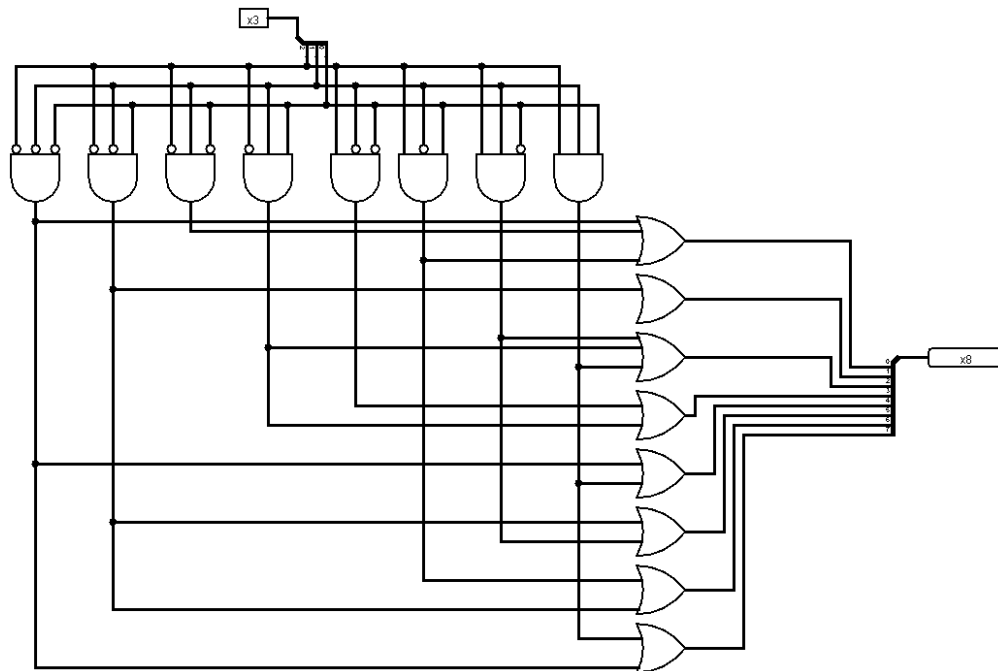


Tabela 3 - Tabela dos endereços da ROM

Entrada	Saída
000	10010001
001	01100010
010	00000001
011	00001100
100	00001000
101	01000001
110	00100100
111	10010100

1.7 Memória RAM de 8 bits

A Memória RAM é um tipo de memória que armazena dados de forma não sequencial, sendo o endereço de memória responsável por definir o local exato de armazenamento. A RAM projetada conta com 8 espaços de armazenamento, cada um capaz de guardar um valor de 8 bits. O seletor é conectado a um demultiplexador como entrada de seleção (*select*), enquanto a entrada do demultiplexador é o *clock* principal. Uma das oito saídas do demultiplexador atua como o *clock* para o componente "palavra", que é responsável por armazenar 8 bits de dados através de

oito FF do tipo D. O componente “palavra” possui uma entrada de 8 *bits*, duas de apenas um *bit* o *clock* e o *clear*. Esse componente possui apenas uma saída de 8 *bits*.

Figura 9 - Componente "palavra"

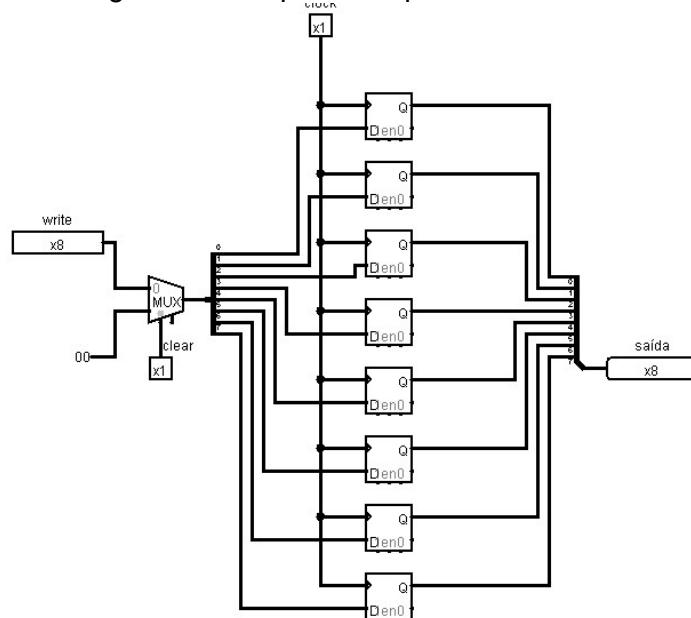
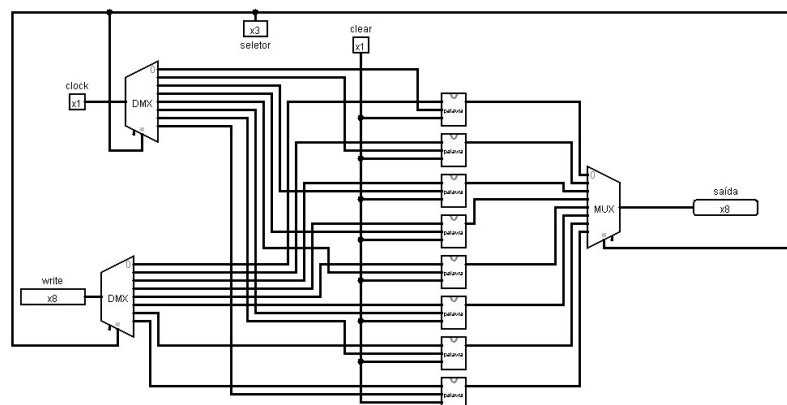


Figura 10 - Memória RAM



1.8 Banco de registradores de 8 *bits*

Um banco de registradores é um componente digital formado por um conjunto de registradores organizados para acesso organizado. Ele permite a realização de operações de leitura, para recuperar dados previamente armazenados, e de escrita, para atualizar ou modificar as informações internas.

O nosso banco de registradores possui 4 entradas:

- *data_in* (8 bits): Representa o valor a ser escrito no registrador selecionado.
- *write_select* (4 bits): Controla qual dos 16 registradores receberá o valor de *data_in*. Ele está conectado a um demultiplexador que direciona o sinal de escrita para o registrador correspondente.
- *rx_select* (4 bits): Determina qual registrador será lido para a saída *rx*.
- *ry_select* (4 bits): Determina qual registrador será lido para a saída *ry*.

E apresenta duas saídas:

- *rx* (8 bits): Fornece o valor armazenado no registrador selecionado por *rx_select*.
- *ry* (8 bits): Fornece o valor armazenado no registrador selecionado por *ry_select*.

Sua arquitetura interna possui 16 registradores de 8 bits cada um com entradas para dados, a entrada *write_select* é conectada a um demultiplexador que direciona o sinal de escrita para o registrador apropriado. O valor em *data_in* é armazenado no registrador selecionado, por fim dois multiplexadores para Leitura um é controlado pelo *rx_select* que conecta o registrador a saída *rx* e o *ry_select* que faz o mesmo, mas para a saída *ry*.

Existem duas operações principais: a escrita e a leitura. Na operação de escrita, o valor presente em *data_in* é armazenado no registrador indicado pelo sinal *write_select*. Apenas o registrador especificado é atualizado com o novo valor, enquanto os demais registradores mantêm seus conteúdos inalterados. E na de leitura, os registradores a serem lidos para *rx* e *ry* são determinados por *rx_select* e *ry_select*, respectivamente. Cada seletor escolhe um registrador cuja saída será enviada ao barramento correspondente.

Figura 11 - Registradores

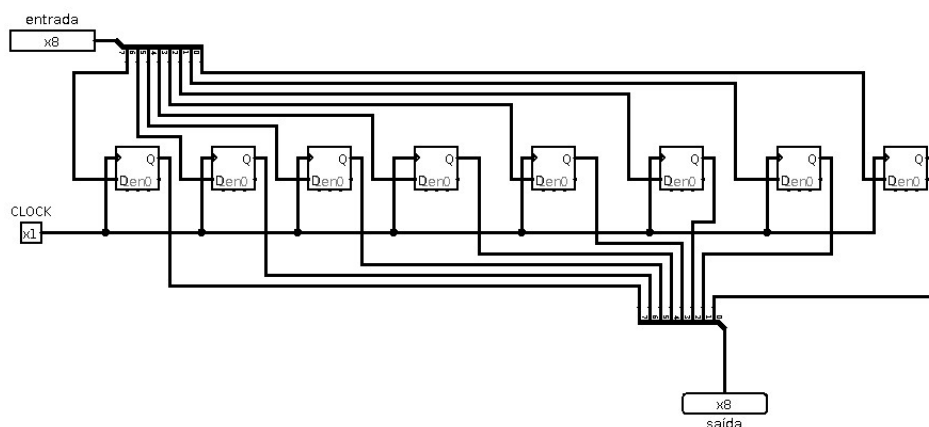
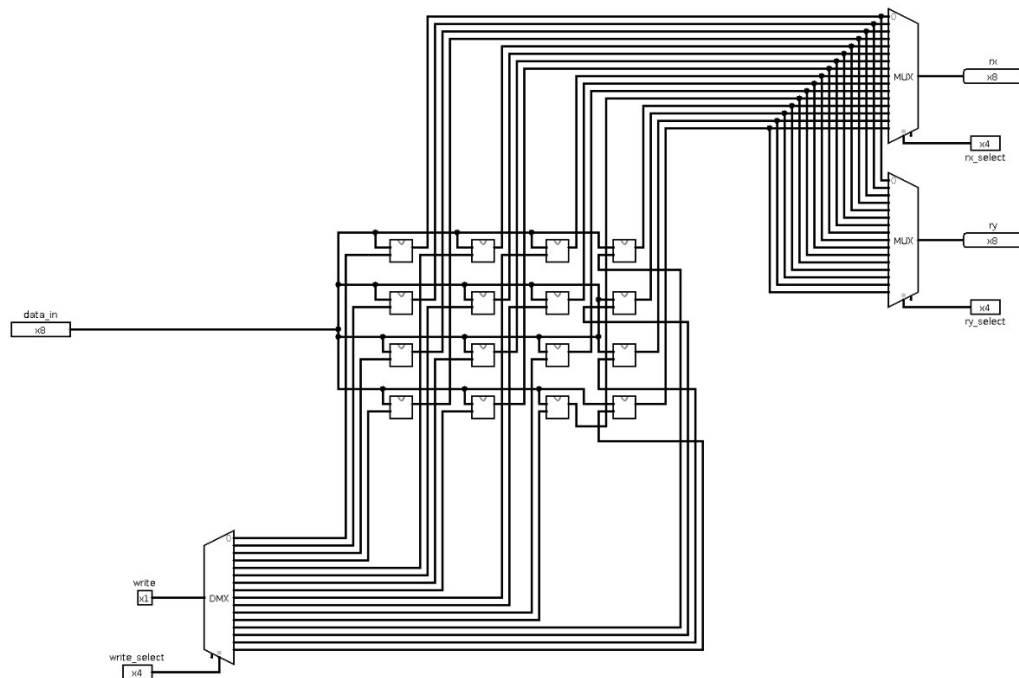


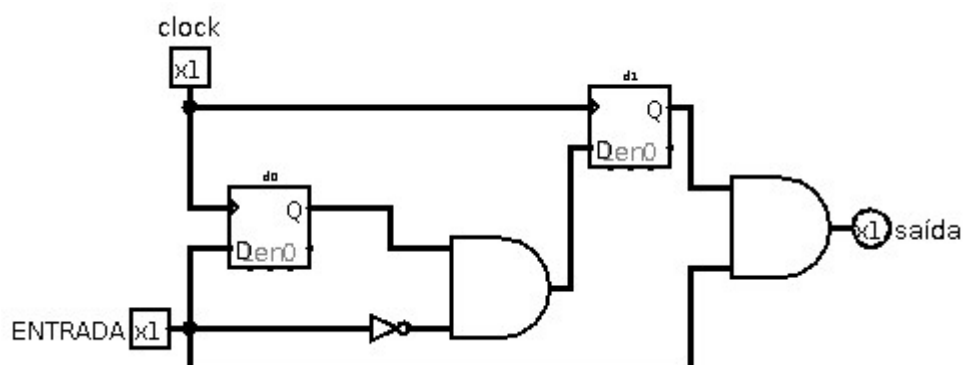
Figura 12 - Banco de registradores



1.9 Detector de sequência binária

Para criar o componente de sequência binária, que reconheça a sequência "101", foi criado primeiro uma máquina de estados finitos que o aceite, essa máquina está representada nessa imagem, a saída dela é 1 quando detecta a sequência 1 0 1, a partir dela se cria tabela verdade e os consequentes máquina de karnaugh e operação lógica

Figura 13 - Detector de sequência binária



1.10 Ula de 8 bits

A ULA é um componente digital responsável por realizar operações lógicas e aritméticas básicas. Esta ULA possui as seguintes características e funcionalidades.

1.10.1 Entradas

- A (8 bits): Operando de entrada principal.

- B (8 *bits*): Operando de entrada secundário.
- SEL (4 *bits*): Sinal de seleção que determina a operação a ser executada.

1.10.2 Saídas

- *RESULT* (8 *bits*): Resultado da operação selecionada. 2.
- *overflow_sub* (1 *bit*): *overflow* do sub 3.
- *overflow_sum* (1 *bit*): *overflow* de soma.

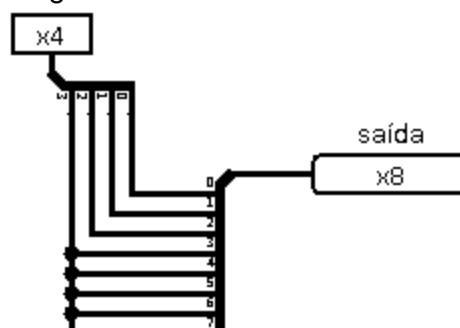
1.10.2 Operações Disponíveis (Controladas pelo SEL)

1. *AND*: Operação lógica *AND* entre A e B. *RESULT* = $A \& B$
2. *OR*: Operação lógica *OR* entre A e B. *RESULT* = $A | B$
3. *NOT* (A): Operação lógica *NOT* aplicada apenas ao operando A. *RESULT* = $\sim A$
4. *NOR*: Operação lógica *NOR* entre A e B. *RESULT* = $\sim (A | B)$
5. *NAND*: Operação lógica *NAND* entre A e B. *RESULT* = $\sim (A \& B)$
6. *XOR*: Operação lógica *XOR* entre A e B. *RESULT* = $A \oplus B$
7. *SUBTRAÇÃO*: Subtração entre os operandos A e B. *RESULT* = $A - B$
8. *SHIFT LEFT*: Deslocamento lógico do operando A para a esquerda em 2 *bits*. *RESULT* = $A \ll 2$
9. *SHIFT RIGHT*: Deslocamento lógico do operando A para a direita em 2 *bits*. *RESULT* = $A \gg 2$
10. *SOMADOR*: Somados operandos A e B, com detecção de "carry out". *RESULT* = $A + B$
overflow_sum = (Resultado > 255)

1.11 Extensor de Sinal 4 *bits* para 8 *bits*

Um extensor de sinal de 4 *bits* para 8 *bits* copia os 4 *bits* de entrada para os 4 *bits* menos significativos da saída. Para extensão de sinal, o *bit* mais significativo da entrada (*bit* 3) é replicado nos 4 *bits* mais significativos da saída. Isso mantém o valor correto em complemento de dois.

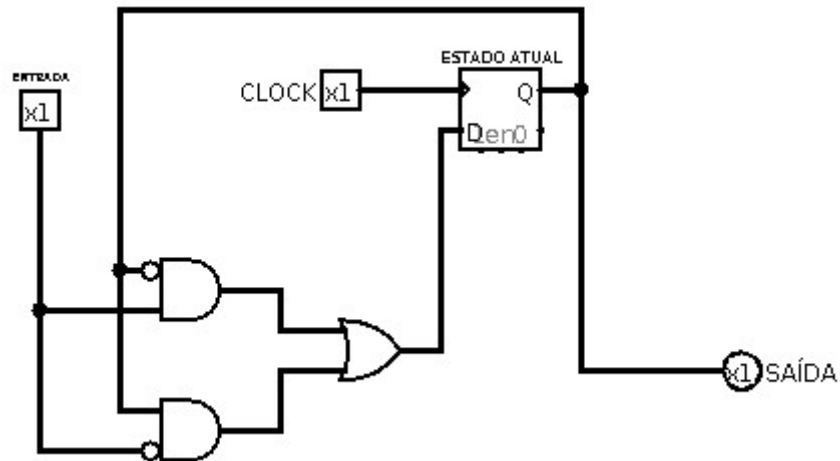
Figura 14 - Extensor de sinal



1.12 Máquina de estados: semáforo

Máquina de estados de um semáforo de pedestres, tem dois estados: verde e vermelho, a entrada é um botão que quando está em 0 não faz nada, e quando está em 1, muda o estado da máquina

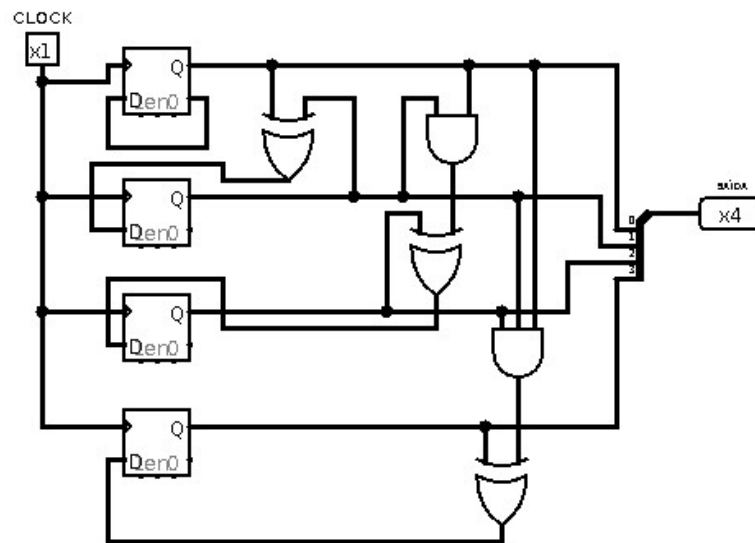
Figura 15 - Semáforo de pedestres



1.13 Contador síncrono

Esse componente conta de 1 em 1 conforme o *clock* muda, apresenta uma escalabilidade grande e simples utilizando apenas portas *and*, *xor* e FF do tipo D. O primeiro FF precisa apenas que a saída Q/ seja colocado a sua entrada e assim já varia a cada vez que acontece 1 *clock*. Para o segundo FF é adicionado uma porta *xor* e nessa porta é ligado a saída Q do primeiro e do segundo FF, após isso é ligada essa saída a entrada do FF D, dessa maneira ela varia com dois *clocks*. No terceiro FF é adicionado uma porta *and* e uma porta *xor*, na porta *xor* são ligadas as saídas Q do primeiro e do segundo FF, a saída dessa porta *and* é ligado a nova porta *xor* a na outra entrada da porta *xor* é colocado a saída de Q do terceiro FF e a saída da porta *xor* é colocada na entrada D do FF. A partir do terceiro é a mesma coisa, porta *xor* e *and* com todas as saídas Qs anteriores.

Figura 16 - Contador síncrono



1.14 Paridade ímpar

Esse componente mostra quais dos sinais apresentam quantidades ímpares de 1s

Figura 17 - Paridade ímpar

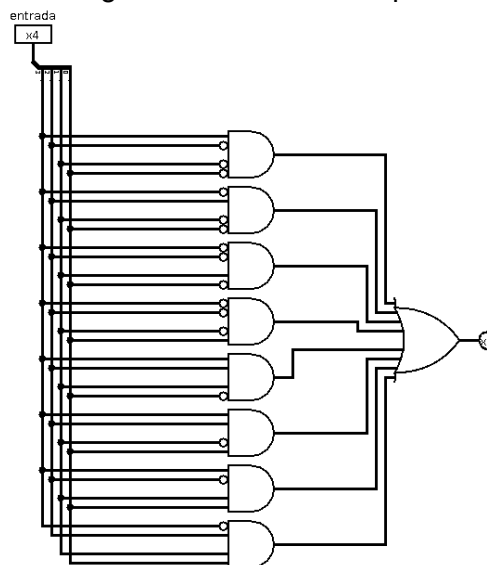


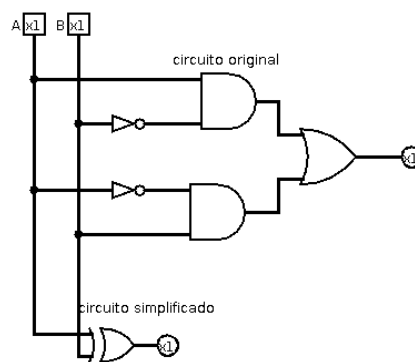
Tabela 4 - Tabela de paridade ímpar

ENTRADA				SAÍDA
0	0	0	1	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1

1.15 Otimização lógica

Foi escolhido o xor para otimizar, ao invés de usar um circuito grande foi reduzido a apenas uma porta logica.

Figura 18 - Circuito simplificado



1.16 Decodificador de 7 seguimentos

Para fazer esse componente foram feitos 8 componentes, sendo 7 deles os responsáveis por representar cada seguimento corretamente e 1 apenas para receber a entrada e a saída para o display.

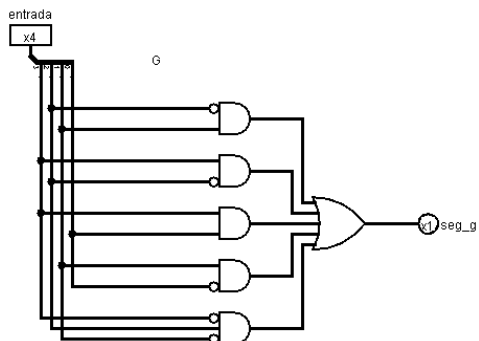
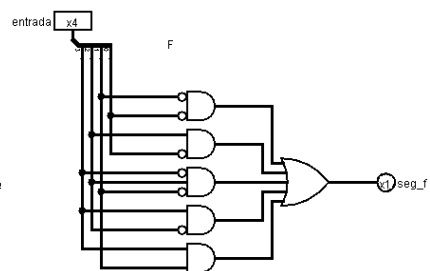
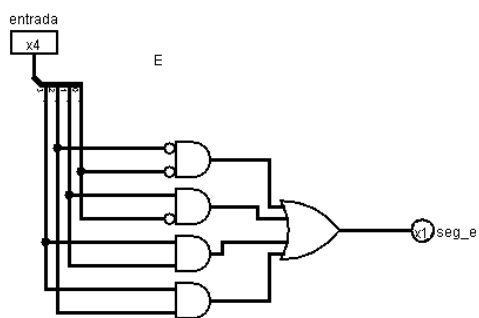
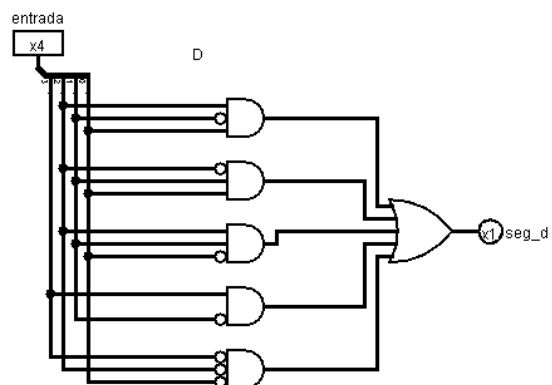
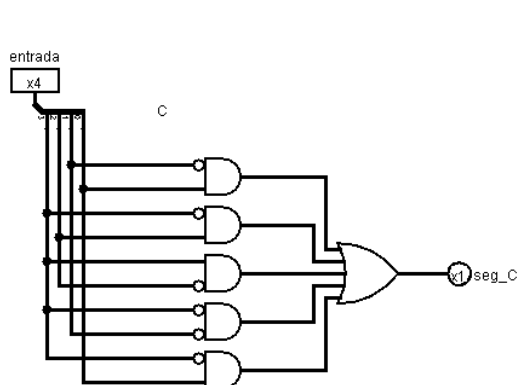
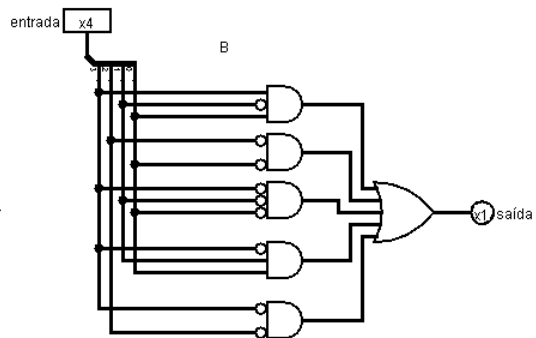
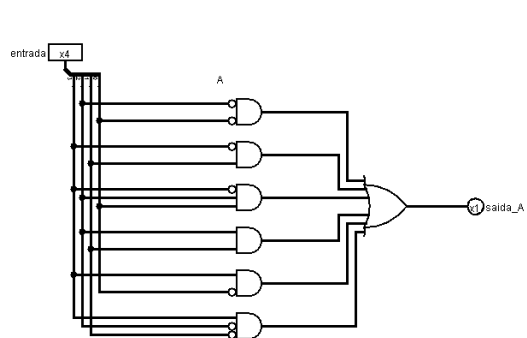
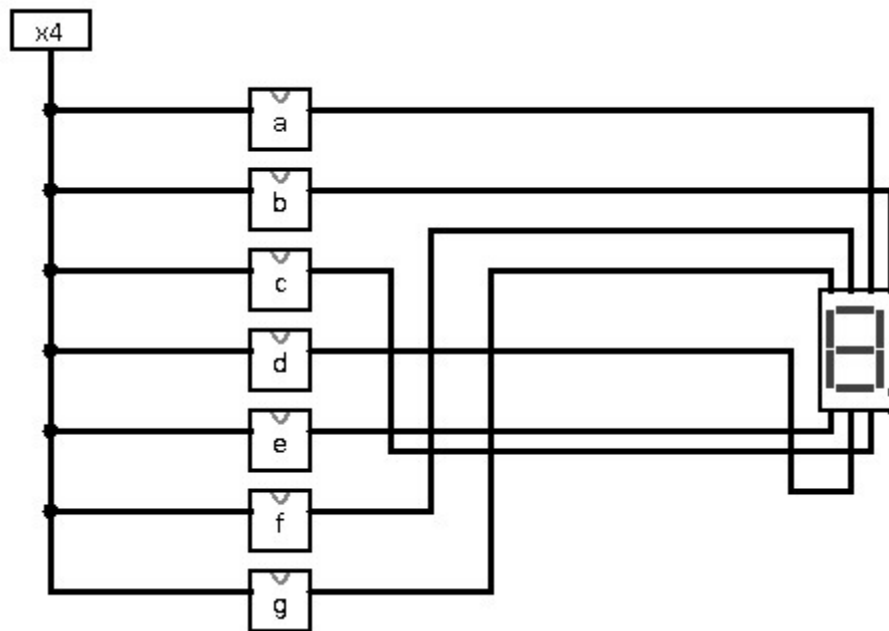


Figura 19 - *Display* hexadecimal



1.17 Detector de números primos

O componente através de portas lógicas recebe uma entrada de 4 *bits* e a saída é 1 apenas se o número for primo.

Figura 20 - Número primo

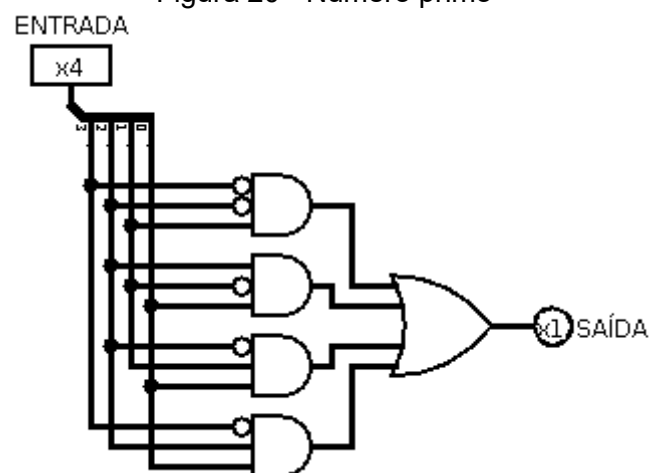


Tabela 5 - Tabela dos números primos

ENTRADA				SAÍDA
0	0	1	0	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	1	1	1
1	1	0	1	1

REFERÊNCIAS

Maganha, G. V. (7 de 12 de 2024). *GV ensino*. Fonte: Youtube:
<https://youtube.com/@gvensino?si=6hcXfaUM9wIMjL1r>

Souza, P. (7 de 12 de 2024). *Pedro Souza*. Fonte: Youtube:
<https://youtube.com/@pedrosouza-bu2dn?si=s4laZ1mUzetzPUC1>