Universidade Federal de Viçosa
Campus Florestal
CCF 251 – Introdução aos Sistemas Lógicos Digitais
Prof. José Augusto Miranda Nacif

Trabalho Prático 01 - Circuitos Combinacionais

Gabriel Rodrigues Marques - 5097
Alan Araújo dos Reis - 5096
Rodrigo dos Santos Miranda - 5090
Edgar Alves de Jesus - 5087

23 de outubro de 2022

1. Objetivos

Para esse Trabalho Prático, foi pedido o desenvolvimento de um encriptador de números binários, utilizando a técnica de substituição da Cifra de César. Além de exibir os números binários criptografados em um display de 7 segmentos. O trabalho foi dividido em simulações em ambiente virtual e físico.

2. Desenvolvimento e Resultados

Primeiramente, antes de iniciar o desenvolvimento do trabalho, a equipe precisou obter uma chave de decodificação. A chave escolhida pelo grupo foi a chave Echo, de valor 5. A partir da escolha da chave, iniciamos a modelagem do trabalho, simulação e sua implementação.

Para iniciar, precisava-se codificar um número binário de 4 bits, com valores de 0 a 9. A partir dessa ideia, a primeira etapa foi construir duas tabelas, que ajudaram na execução do trabalho como um todo. Nas tabelas, os números só vão de 0 a 9, pois a partir do número 10 o sistema foi definido para reiniciar a contagem. Assim sendo, o número 10 é interpretado como 0, 11 como 1, 12 como 2, 13 como 3, 14 como 4 e o 15 como 5. A partir disso é feita sua codificação e exibição no display quando a entrada do número binário é superior ao valor 9.

A primeira tabela (Tabela 1), é uma tabela de codificação do número binário de valor 0 a 9 com sua respectiva associação a coleção de números, que estava disponível em uma tabela na especificação do trabalho.

Tabela de Codificação - Chave Echo 5

Número	Representação Binária (4 bits)	Número Codificado	Representação Binária Codificada (5 bits)
0	0 0 0 0	5	1 1 1 1 1
1	0 0 0 1	6	0 1 1 1 1
2	0 0 1 0	7	0 0 1 1 1
3	0 0 1 1	8	0 0 0 1 1
4	0 1 0 0	9	0 0 0 0 1
5	0 1 0 1	0	0 0 0 0 0
6	0 1 1 0	1	1 0 0 0 0
7	0 1 1 1	2	1 1 0 0 0
8	1 0 0 0	3	1 1 1 0 0
9	1 0 0 1	4	1 1 1 1 0

Tabela 1

A segunda tabela (Tabela 2), refere-se ao modo como cada número binário codificado seria mostrado no display de 7 segmentos.

Número / Binário Codificado						Display						
N°	A	B	C	D	E	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0	1	1	0	1	1
1	0	1	1	1	1	1	0	1	1	1	1	1
2	0	0	1	1	1	1	1	1	0	0	0	0
3	0	0	0	1	1	1	1	1	1	1	1	1
4	0	0	0	0	1	1	1	1	1	0	1	1
5	0	0	0	0	0	1	1	1	1	1	1	0
6	1	0	0	0	0	0	1	1	0	0	0	0
7	1	1	0	0	0	1	1	0	1	1	0	1
8	1	1	1	0	0	1	1	1	1	0	0	1
9	1	1	1	1	0	0	1	1	0	0	1	1

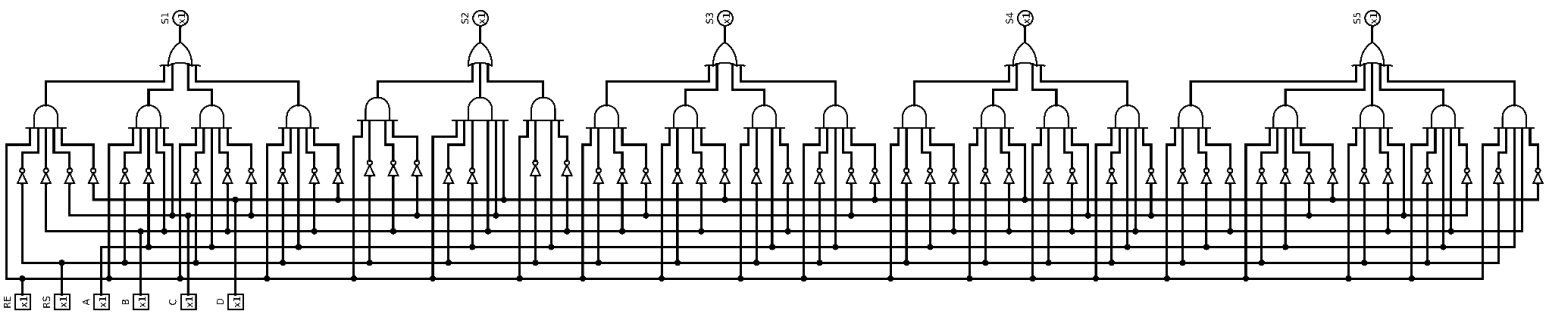
Tabela 2

Além do número binário de 4 bits como entrada, foi solicitado duas entradas de 1 bit cada: ready e reset. A codificação e exibição no display só é realizada quando o valor de ready é 1 e o de reset 0, caso contrário, no display será exibido 0.

Em seguida, através software Logisim, uma ferramenta para a concepção e simulação digital de circuitos lógicos, foram realizadas as seguintes etapas:

- Levantamento das equações booleanas, através das tabelas-verdade do codificador e do display (Tabela 1 e Tabela 2)
- Simplificação das equações booleanas geradas, utilizando lógica booleana e mapas de Karnaugh
- Elaboração do circuito simplificado com portas lógicas

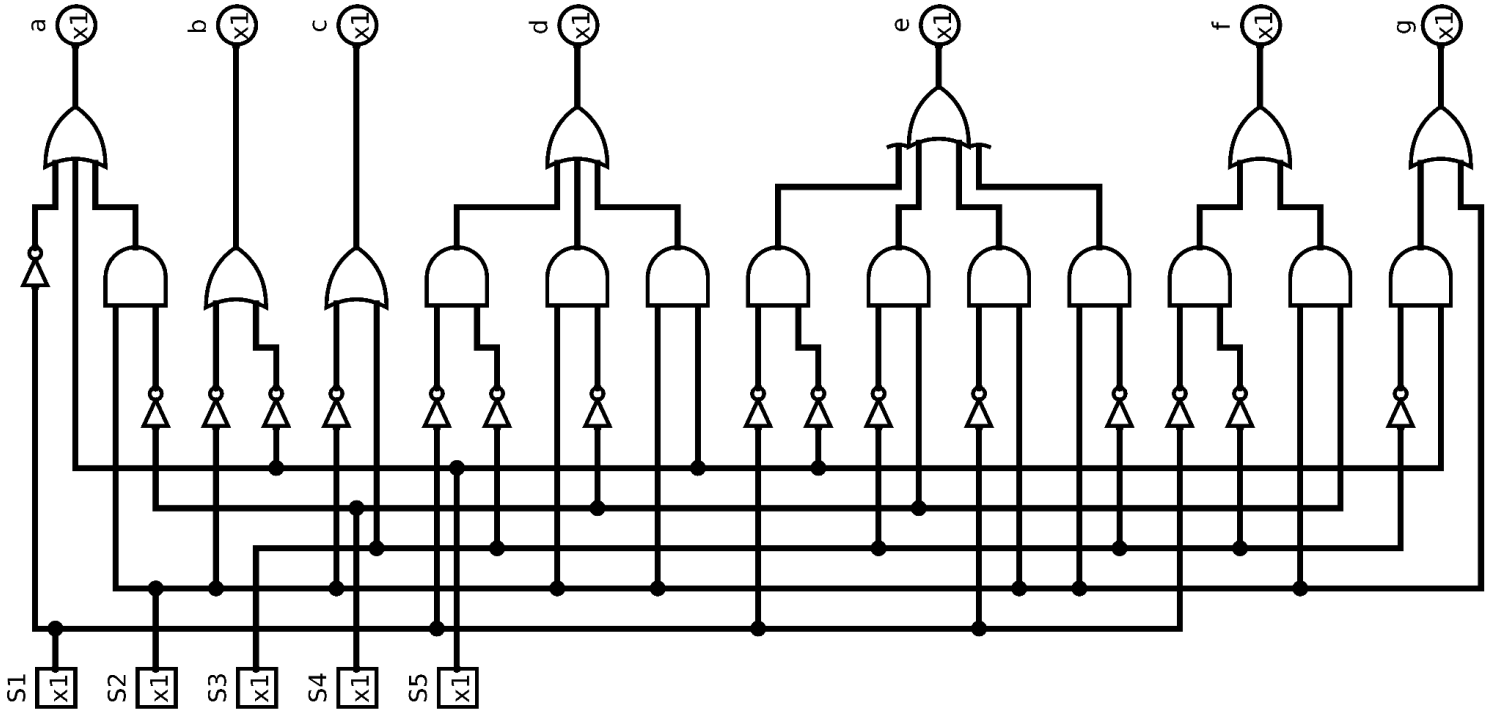
Todos esses processos podem ser verificados logo abaixo:



Circuito Simplificado do Codificador

```
S1 = ((RE & ~RS & ~B & ~C & ~D) | (RE & ~RS & ~A & B & C) | (RE & ~RS & A & ~B & ~C) | (RE & ~RS & A & ~B & ~D));
S2 = ((RE & ~RS & ~B & ~C) | (RE & ~RS & ~A & B & C & D) | (RE & ~RS & A & ~B));
S3 = ((RE & ~RS & ~B & ~C) | (RE & ~RS & ~B & ~D) | (RE & ~RS & A & ~B) | (RE & ~RS & A & ~C & ~D));
S4 = ((RE & ~RS & ~A & ~B) | (RE & ~RS & ~B & D) | (RE & ~RS & ~B & C) | (RE & ~RS & A & B & ~C));
S5 = ((RE & ~RS & ~A & ~B) | (RE & ~RS & ~A & ~C & ~D) | (RE & ~RS & ~B & C) | (RE & ~RS & A & B & ~C) | (RE & ~RS & A & B & ~D));
```

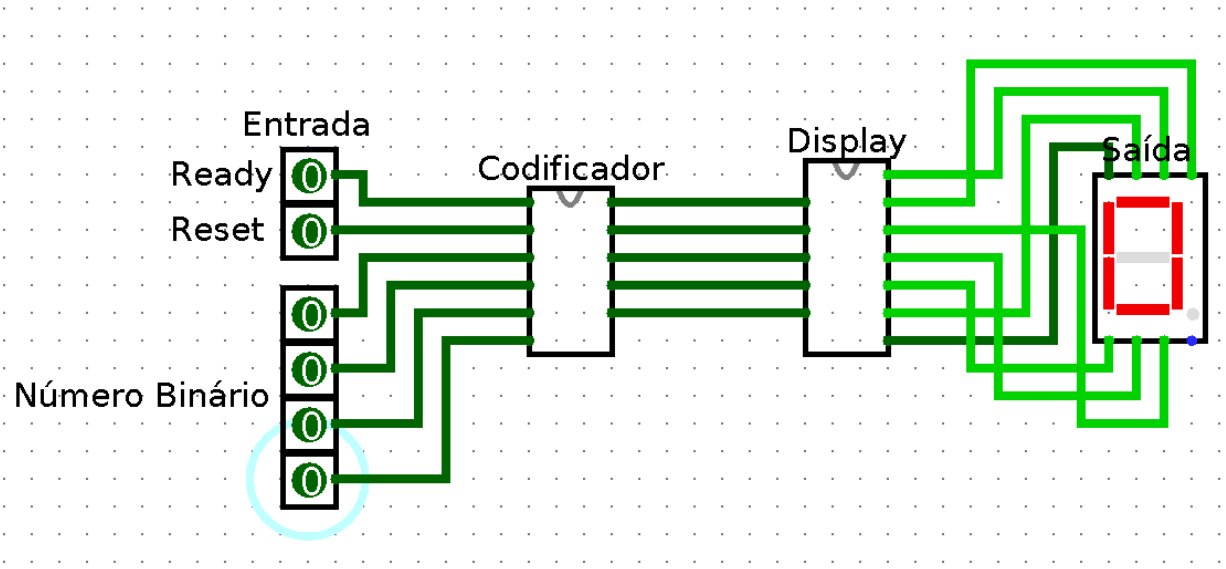
Equações Booleanas do Codificador



Circuito Simplificado do Display

```
a = (~S1 | S5 | (S2 & ~S4));
b = (~S2 | ~S5);
c = (~S2 | S3);
d = ((~S1 & ~S3) | (S2 & ~S4) | (S2 & S5));
e = ((~S1 & ~S5) | (~S3 & S4) | (~S1 & S2) | (S2 & ~S3));
f = ((~S1 & ~S3) | (S2 & S4));
g = ((~S3 & S5) | S2);
```

Equações Booleanas do Display



gif_circuito.gif

Exemplificação do Circuito Codificador/Display

Com toda a lógica construída no Logisim, foi utilizada a linguagem de descrição de hardware, Verilog, para descrever os módulos dos circuitos acima.

Dividimos o código em 3 módulos bases para ficar mais organizado: “echo_codificador.v”, “echo_display.v” e “echo_testbench.v”. Além de gerar os arquivos “echo_tesbench.vvp” e “echo_GTKWAVES.vcd”. Os detalhes do funcionamento do código e simulação estão no vídeo que foi gravado. Em síntese, tem-se uma entrada de 6 bits (RE, RS, A, B, C, D), essa entrada passa pelo módulo de codificação para mascarar o número binário de acordo com a chave. Em seguida, as saídas do codificador (S1, S2, S3, S4, S5) são utilizadas como entradas no módulo de display, que gera suas saídas (a, b, c, d, e, f, g) para serem utilizadas de entrada no display de 7 segmentos. O funcionamento é exatamente como o gif, disponível no link, demonstra.

Ao compilar o código ele mostrará as seguintes informações:

- Entrada: RE RS A B C D (entrada testada)
- Codificado: S1 S2 S3 S4 S5 (número binário codificado)
- Display: a b c d e f g (segmentos que serão acesos no display)

3. Conclusão

Por fim com a utilização de ferramentas, como Icarus Verilog, GTKWave, Logisim e o Visual Studio Code, e dos artifícios e conhecimentos de lógica booleana e circuitos combinacionais que tínhamos em mãos, obtivemos um resultado, que ao nosso ver, atende às especificações solicitadas anteriormente.

4. Referências

[1] R. Katz, G. Borriello, Contemporary Logic Design, 2ª edição, Prentice Hall, 2004.
[2] TANENBAUM, A.S. Organização Estruturada de Computadores. 5. ed. Editora Pearson Prentice Hall, 2007.;
[3] Github. Disponível em: </https://github.com/>;
[4] Stack Overflow. Disponível em <https://stackoverflow.com/>;
[5] Geeks for Geeks. Disponível em <https://www.geeksforgeeks.org/>;
[6] Icarus Verilog;
[7] GTKWave;
[8] Logisim;
[9] Visual Studio Code;