ENE0040 - Laboratório de Sistemas Digitais - Turma 07

Relatório do Experimento 1

Autor: Henrique Morcelles Salum
Matrícula: 232003008

Universidade de Brasília - UnB
Departamento de Engenharia Elétrica - ENE

Sumário

Questão 1

As portas lógicas AND, OR e NOT implementam, respectivamente, as funções booleanas AND, OR e NOT, que seguem, também, respectivamente, as tabelas-verdade a seguir:

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 1: Tabela-verdade da função booleana AND

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 2: Tabela-verdade da função booleana OR

A	$\neg A$
0	1
1	0

Tabela 3: Tabela-verdade da função booleana NOT

a) Implementar uma porta AND usando somente portas OR e NOT

Do Primeiro Teorema de De Morgan obtemos o seguinte resultado:

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$
$$\Rightarrow A \cdot B = \overline{\overline{A} + \overline{B}}.$$

Utilizando a notação da Lógica Proposicional, temos:

$$A \wedge B = \neg(\neg A \vee \neg B),$$

ou seja, o AND entre dois bits é o mesmo que o inverso (NOT) do OR entre os inversos desses bits. A implementação desse circuito no Logisim é como segue:

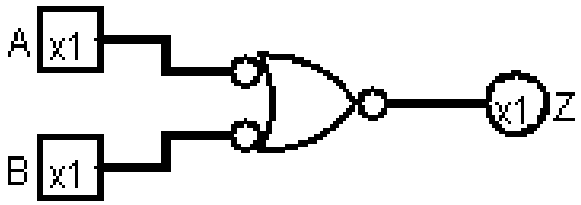


Figura 1: AND implementado apenas com OR e NOT

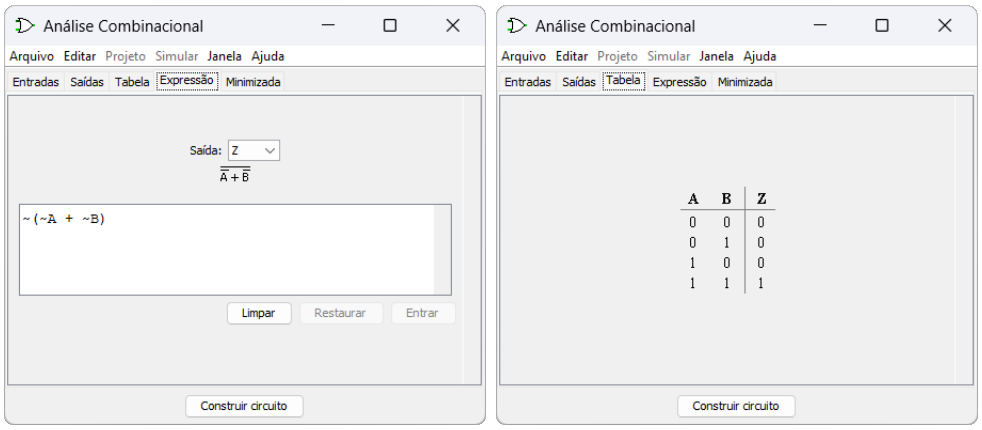


Figura 2: Expressão e tabela-verdade gerados pelo Logisim para o circuito da figura 2

b) Implementar uma porta OR usando somente portas AND e NOT

Do Segundo Teorema de De Morgan obtemos o seguinte resultado:

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$
$$\Rightarrow A + B = \overline{\overline{A} \cdot \overline{B}}.$$

Utilizando a notação da Lógica Proposicional, temos:

$$A \vee B = \neg(\neg A \wedge \neg B),$$

isto é, o OR entre dois bits é o mesmo que o inverso do AND entre os inversos dos dois bits. A implementação desse circuito no Logisim é como segue:

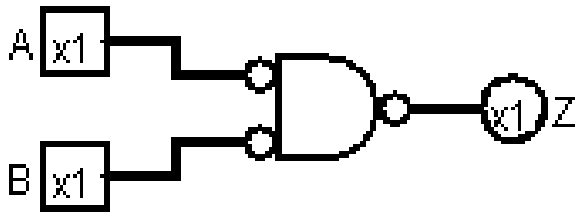


Figura 3: OR implementado apenas com AND e NOT

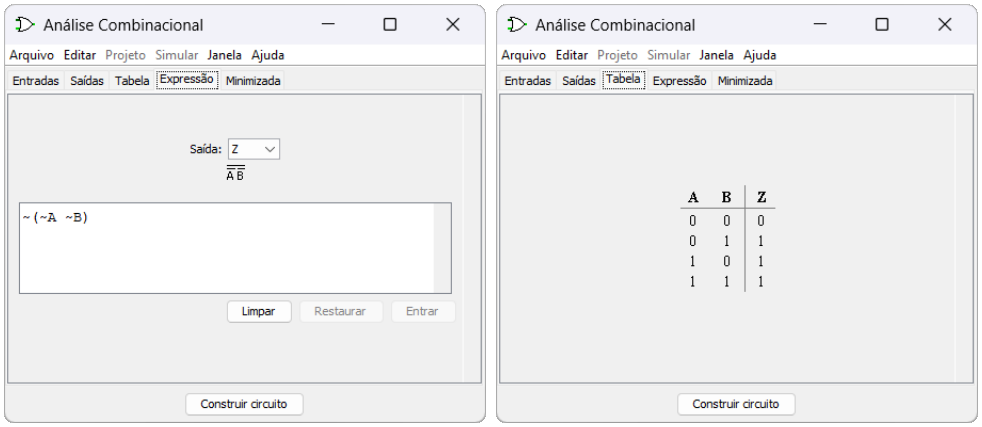


Figura 4: Expressão e tabela-verdade gerados pelo Logisim para o circuito da figura 3

Questão 2

Nessa questão, implementamos um somador completo. Um somador completo é um circuito lógico combinacional que realiza a soma de três bits: dois bits de entrada, A e B, e um bit adicional, C (tipicamente chamado de C_{in}, carry-in), que representa o "vai-um"(carry-out) de uma operação anterior. Ele gera duas saídas: S, o resultado da soma dos três bits, e T (tipicamente chamado de C_{out}, carry-out), o "vai-um"que pode ser gerado por essa soma.

As saídas do somador completo, T e S, podem ser implementadas, respectivamente, pelas funções booleanas

$$T = AB + AC + BC$$

e

$$S = \overline{A}\ \overline{B}\ C + \overline{A}\ B\ \overline{C} + A\ \overline{B}\ \overline{C} + A\ B\ C = A \oplus B \oplus C.$$

A tabela-verdade a seguir explicita a relação entrada-saída dessas duas funções.

Entradas			**Saídas**	
A	B	C	T	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tabela 4: Tabela-verdade das funções T e S

a) Implementar a função T utilizando somente AND e OR

A implementação dessa função é bastante direta: onde na expressão há multiplicação, utilizamos a porta AND e onde há soma, a porta OR. O circuito é o que segue.

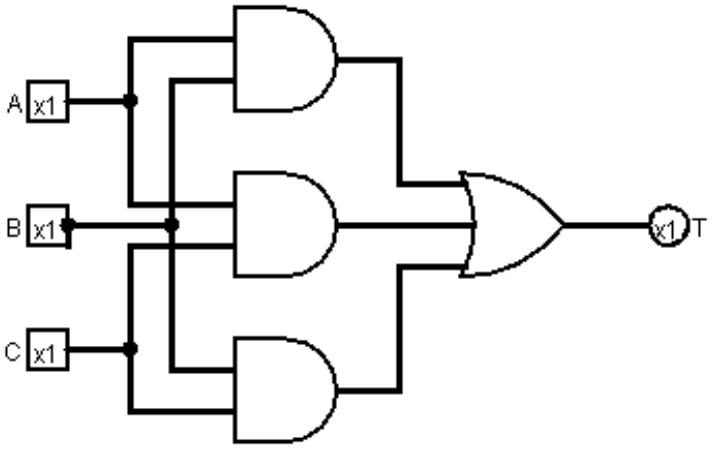


Figura 5: Implementação da função T apenas com AND e OR

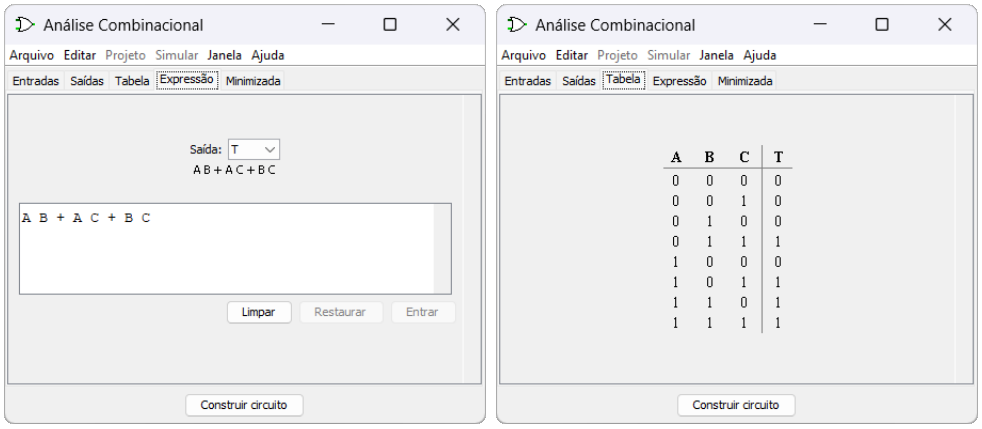


Figura 6: Expressão e tabela-verdade gerados pelo Logisim para o circuito da figura 5

b) Implementar a função S utilizando somente AND e OR

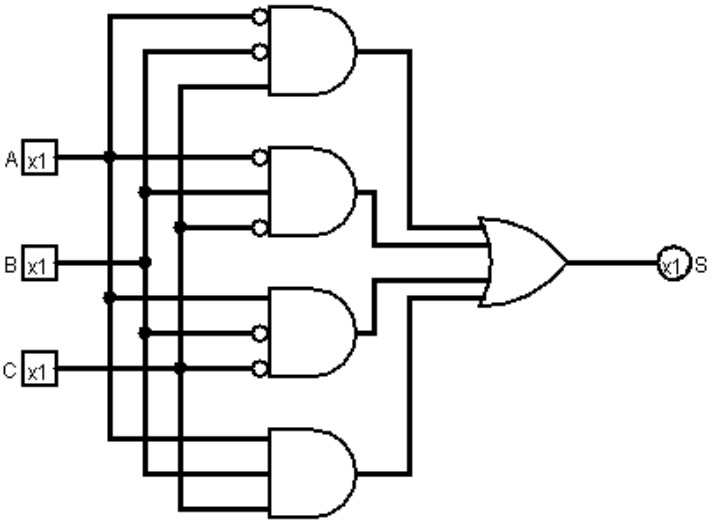


Figura 7: Implementação da função S apenas com AND e OR

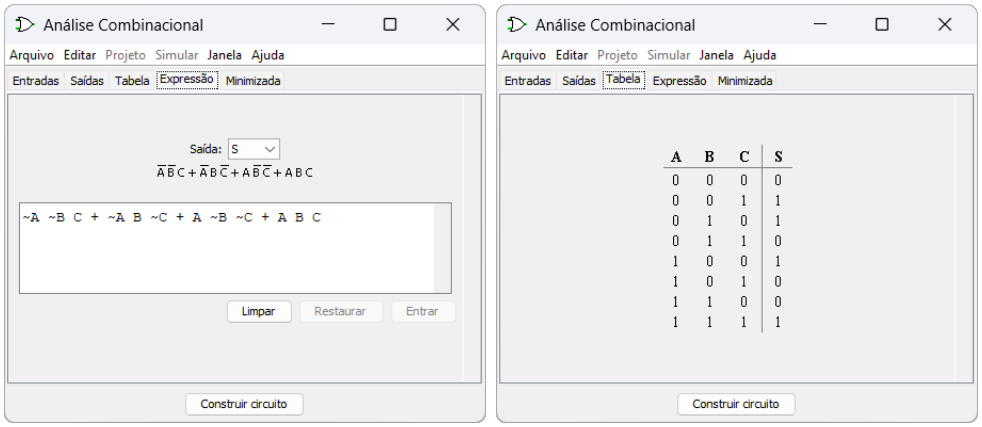


Figura 8: Expressão e tabela-verdade gerados pelo Logisim para o circuito da figura 7

Questão 3

Nessa questão, considera-se as funções T e S do item anterior, porém, implementá-las-emos utilizando apenas a porta NAND.

Para entender o porquê de podermos implementar essas (e quaisquer outras) funções booleanas com a porta NAND, devemos primeiro conhecer o teorema da completude funcional de Sheffer.

Teorema da Completude Funcional de Sheffer: *Dado um conjunto de operadores lógicos, diz-se que esse conjunto é funcionalmente completo se todas as funções booleanas podem ser expressas usando apenas esses operadores.*

O conjunto formado pelos operadores AND, OR e NOT, por exemplo, é completo funcional, i.e., qualquer função booleana pode ser implementada por essas três portas. Ainda mais incrível é que, como podemos implementar qualquer uma dessas portas apenas com a porta NAND - como está provado a seguir - concluímos que essa porta, sozinha, forma um conjunto funcionalmente completo (o mesmo é verdade para a porta NOR, mas isso não é relevante aqui).

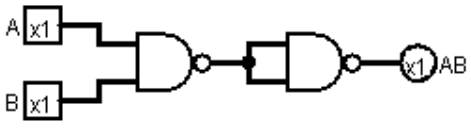
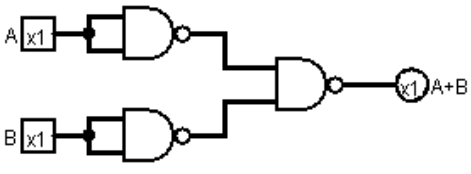
Porta Lógica	Implementação com NAND
NOT	
AND	
OR	

Tabela 5: Prova de que é possível implementar as portas AND, OR e NOT apenas com a porta NAND

Note que uma forma de implementar as funções da questão 2 apenas com portas NAND é simplesmente substituir as portas AND, OR e NOT dos circuitos dessa questão pelas suas respectivas implementações com a porta NAND. A abordagem que faremos na presente questão, porém, é mais eficiente (utiliza menos portas lógicas): utilizaremos o primeiro teorema de De Morgan para manipular as expressões de cada função e chegar à sua implementação canônica com NANDs.

Provado que é possível implementar qualquer função booleana apenas com a porta NAND, resta a dúvida: por que fazer isso? A resposta está nas características físicas de cada porta lógica: a porta NAND é implementada com menos transistores do que as outras, por isso, é mais eficiente e barato produzir circuitos lógicos com essa porta.

a) Implementar a função T utilizando somente NAND

Para saber como implementar a função T apenas com portas NAND, utilizaremos o primeiro teorema de De Morgan:

$$T = AB + AC + BC = \overline{\overline{AB} \cdot \overline{AC} \cdot \overline{BC}},$$

ou seja, T é igual ao NAND entre os NANDs de A e B, A e C, B e C. Precisamos somente aplicar essa lógica no Logisim e montamos o circuito abaixo.

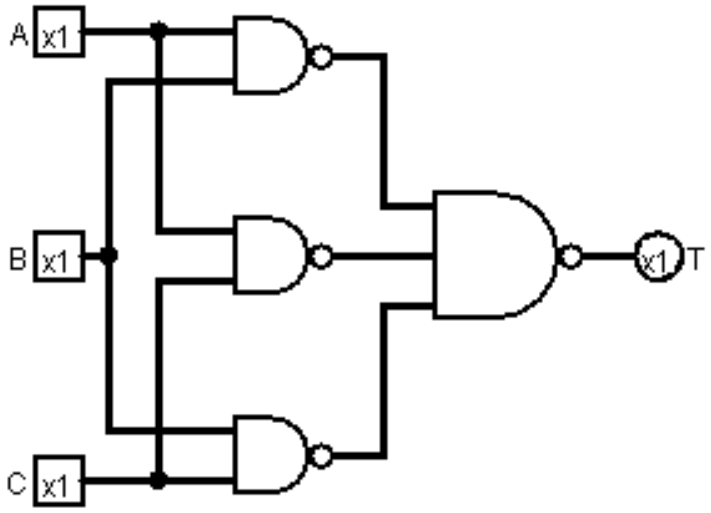


Figura 9: Função T implementada apenas com NANDs

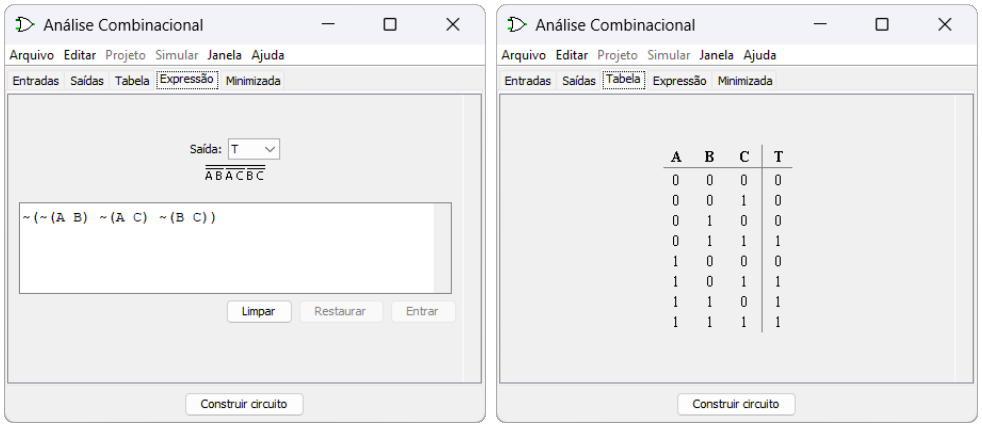


Figura 10: Expressão e tabela-verdade gerados pelo Logisim para o circuito da figura 9

b) Implementar a função S utilizando somente NAND

Para saber como implementar a função S apenas com portas NAND, utilizaremos, novamente, o primeiro teorema de De Morgan:

$$S = \overline{A}\ \overline{B}\ C + \overline{A}\ B\ \overline{C} + A\ \overline{B}\ \overline{C} + A\ B\ C = \overline{\overline{\overline{A}\ \overline{B}\ C} \cdot \overline{\overline{A}\ B\ \overline{C}} \cdot \overline{\overline{A}\ \overline{B}\ \overline{C}} \cdot \overline{A\ B\ C}}$$

ou seja, S é igual ao NAND entre os NANDs de $\overline{A}$, $\overline{B}$ e C; $\overline{A}$, B e $\overline{C}$; A, $\overline{B}$ e $\overline{C}$; A, B, C. Novamente, precisamos somente aplicar essa lógica no Logisim e montamos o circuito a seguir.

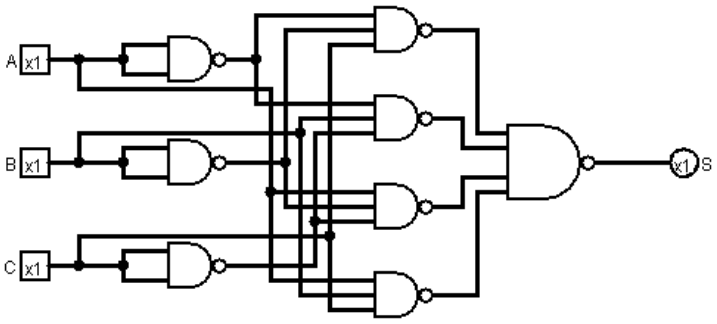


Figura 11: Função S implementada apenas com NANDs

Podemos deixar esse circuito um pouco mais bonito tratando as inversoras como caixas-pretas:

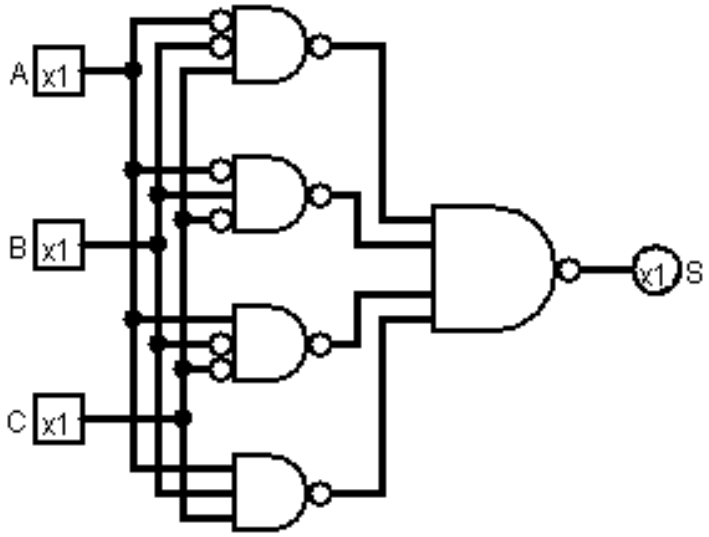


Figura 12: Função S simplificada implementadas com NANDs

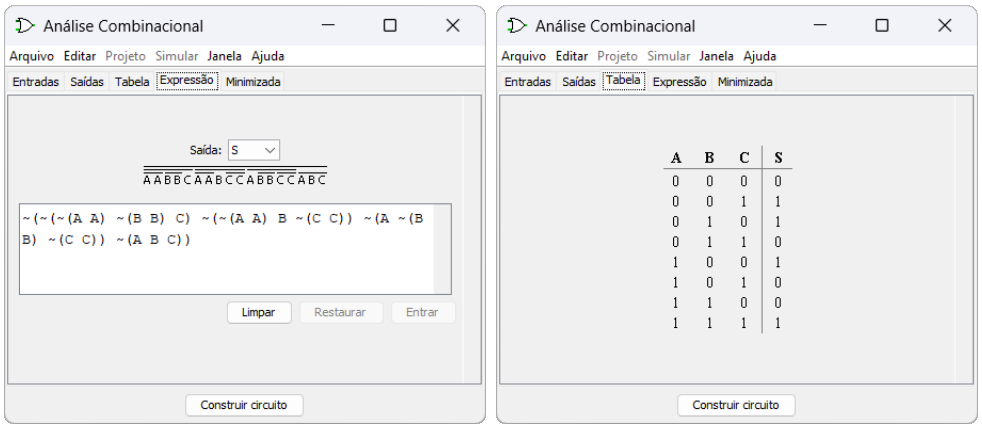


Figura 13: Expressão e tabela-verdade gerados pelo Logisim para o circuito da figura 11

Questão 4

Um multiplexador é um dispositivo digital que, dadas n entradas de dados (D), seleciona como saída uma delas por meio das suas $\log_2 n$ entradas de seleção (S). Nessa questão, implementamos o circuito lógico de um multiplexador 4-para-1, i.e., o multiplexador com 4 entradas de dados, modelado pela expressão

$$Y = D_0\ \overline{S_1}\ \overline{S_0} + D_1\ \overline{S_1}\ S_0 + D_2\ S_1\ \overline{S_0} + D_3\ S_1\ S_0.$$

A tabela-verdade desse dispositivo é a que segue.

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Tabela 6: Tabela-verdade multiplexador 4x1

Como devemos implementar Y apenas com NANDs, precisamos, novamente, usar De Morgan para chegar à expressão que nos fornece a implementação diretamente.

$$Y = D_0\ \overline{S_1}\ \overline{S_0} + D_1\ \overline{S_1}\ S_0 + D_2\ S_1\ \overline{S_0} + D_3\ S_1\ S_0$$
$$\Rightarrow Y = \overline{\overline{D_0\ \overline{S_1}\ \overline{S_0}} \cdot \overline{D_1\ \overline{S_1}\ S_0} \cdot \overline{D_2\ S_1\ \overline{S_0}} \cdot \overline{D_3\ S_1\ S_0}}.$$

O circuito que implementa essa função é o que segue:

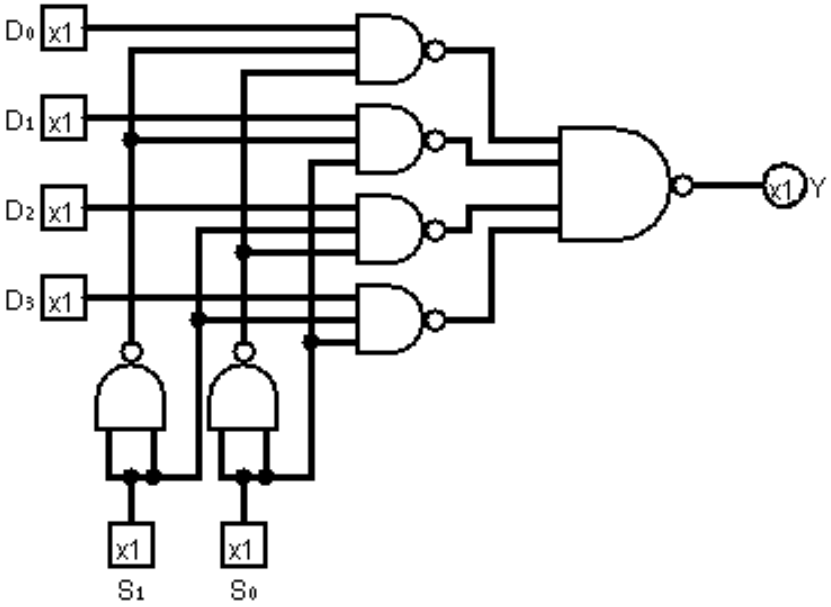


Figura 14: Implemetação da função Y apenas com NANDs

Novamente, para melhorar a visualização, podemos simplificar a implementação tratando as inversoras como caixas pretas. Essa implementação está apresentada abaixo.

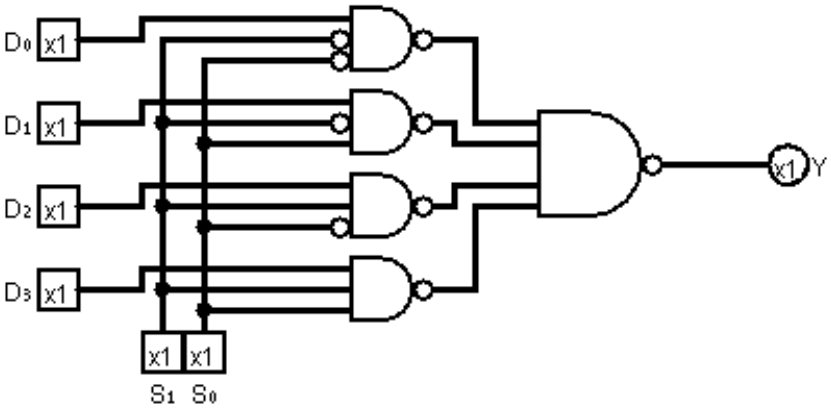


Figura 15: Implemetação da função Y simplificada

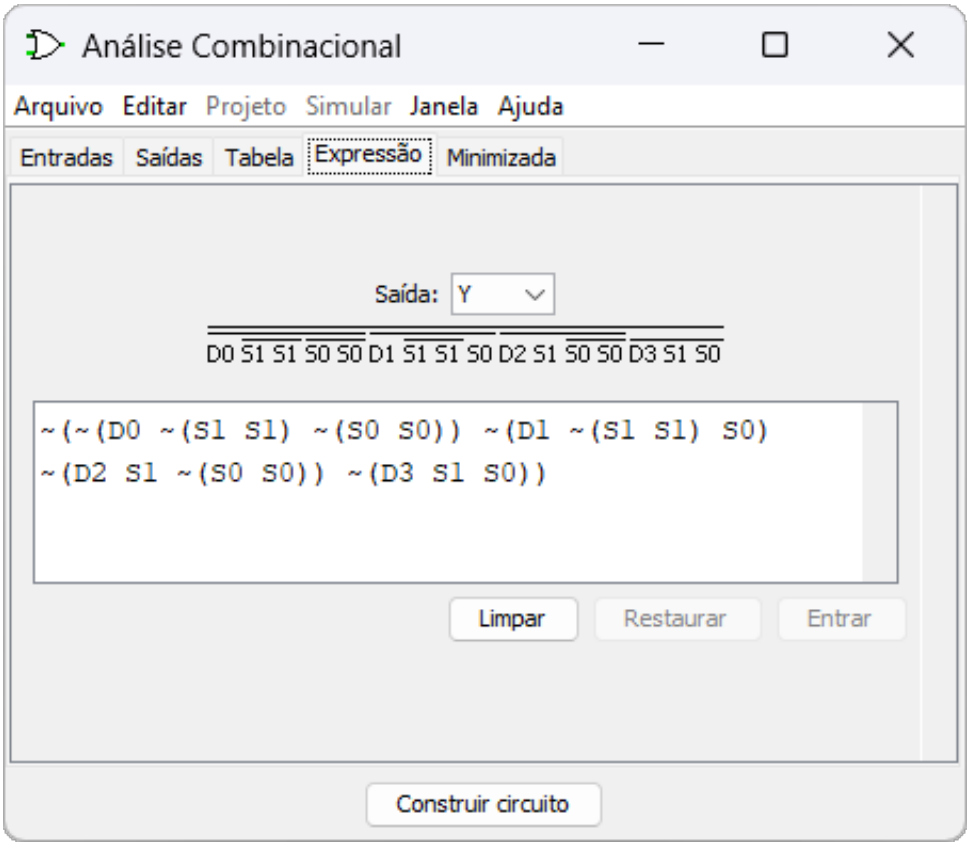
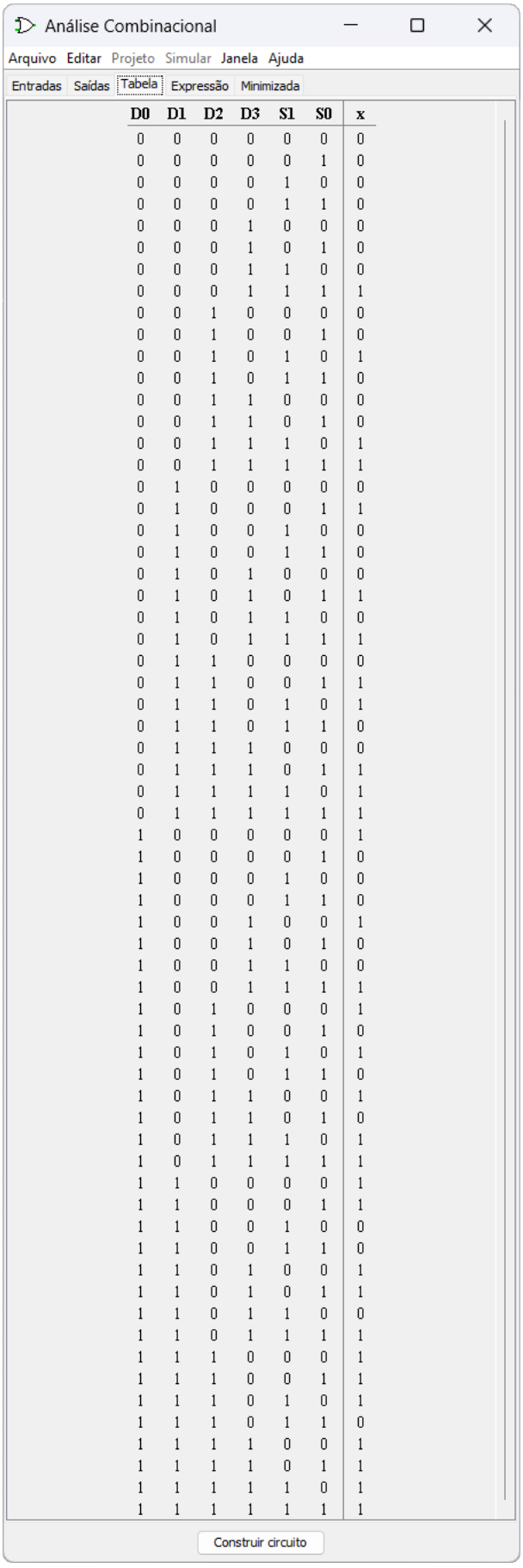


Figura 16: Expressão gerada pelo Logisim para o circuito da figura 14



D0	D1	D2	D3	S1	S0	x
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	1	1	0
0	0	0	1	0	0	0
0	0	0	1	0	1	0
0	0	0	1	1	0	0
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	0	0	1	0
0	0	1	0	1	0	1
0	0	1	0	1	1	0
0	0	1	1	0	0	0
0	0	1	1	0	1	0
0	0	1	1	1	0	1
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	0	0	1	1
0	1	0	0	1	0	0
0	1	0	0	1	1	0
0	1	0	1	0	0	0
0	1	0	1	0	1	1
0	1	0	1	1	0	0
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	0	0	1	1
0	1	1	0	1	0	1
0	1	1	0	1	1	0
0	1	1	1	0	0	0
0	1	1	1	0	1	1
0	1	1	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	0	0	1	0
1	0	0	0	1	0	0
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	0	1	0	1	0
1	0	0	1	1	0	0
1	0	0	1	1	1	1
1	0	1	0	0	0	1
1	0	1	0	0	1	0
1	0	1	0	1	0	1
1	0	1	0	1	1	0
1	0	1	1	0	0	1
1	0	1	1	0	1	0
1	0	1	1	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	0	1
1	1	0	0	0	1	1
1	1	0	0	1	0	0
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	0	1	0	1	1
1	1	0	1	1	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	1
1	1	1	0	0	1	1
1	1	1	0	1	0	1
1	1	1	0	1	1	0
1	1	1	1	0	0	1
1	1	1	1	0	1	1
1	1	1	1	1	0	1
1	1	1	1	1	1	1

Figura 17: Tabela-verdade gerada pelo Logisim para o circuito da figura 14