Manipulações de Álgebra Booleana no simulador Logisim-evolution

José Willames Sabino de Melo Júnior[1], **Alexandre de Andrade Barbosa**[1]

[1]Universidade Federal de Alagoas (UFAL) – Arapiraca – AL

jose.sabino@ufal.arapiraca.br, alexandre.barbosa@ufal.arapiraca.br

Abstract. *Digital circuit simulators are constantly used in the teaching and learning process. However, current simulators lack interactive features to assist in the learning process of boolean expressions simplification. Therefore, this work implemented modifications to the open source digital circuit simulator Logisim-evolution to enable students to learn through exploration with the feedback generated by the system.*

Resumo. *Simuladores de circuitos digitais são utilizados constantemente no processo de ensino e aprendizagem. Porém, os simuladores atuais não possuem nenhum recurso interativo para auxiliar o processo de aprendizagem de simplificação de expressões booleanas. Por isso, neste trabalho foram realizadas modificações no simulador de circuitos open source Logisim-evolution para permitir que alunos possam aprender através da exploração utilizando os feedbacks gerados pelo sistema.*

1. Introdução

Em cursos da área de computação, tais como Ciência da Computação e Engenharia da Computação, disciplinas fundamentais abordam conceitos essenciais sobre circuitos digitais e eletrônica. Os conceitos apresentados são essenciais para compreender o funcionamento dos computadores a partir de seus componentes básicos. Nestas disciplinas, a compreensão teórica é essencial, mas a prática desempenha um papel crucial no entendimento e na aplicação dos conceitos. É com esse objetivo que ferramentas que simulam o comportamento de circuitos digitais são utilizadas.

Para facilitar o aprendizado dos conceitos relacionados a circuitos digitais, é possível utilizar simuladores de circuitos no processo educativo [Prasad et al. 2014, Baneres et al. 2014, Nikolic et al. 2009]. Esses simuladores permitem que os estudantes construam e testem virtualmente diferentes circuitos, observando como os diversos componentes interagem entre si. Por serem puramente virtuais, os simuladores podem ser usados em situações onde o acesso a laboratórios é limitado, como no período de *lockdown* durante a pandemia de COVID-19 em 2020 [Zhu and Howell 2023]. Além disso, os simuladores oferecem uma solução acessível quando não existe material de *hardware* disponível ou adequado para todos os alunos.

Um dos conceitos importantes no estudo de circuitos digitais é a simplificação dos circuitos através de expressões booleanas. Simuladores de circuitos atuais e outras ferramentas oferecem a possibilidade do aluno visualizar a expressão mínima de um circuito. Algumas ferramentas também exibem os passos necessários em uma simplificação, mas

DOI: 10.5753/sbie.2024.242167

não permitem que o discente experimente e valide outras possibilidades de solução, e assim obtenha um maior aprendizado ao observar *feedbacks* [Metcalfe and Kornell 2007]. Com isso, o objetivo desse trabalho é adicionar uma nova funcionalidade ao simulador de circuitos *Logisim-evolution*, tal funcionalidade permite que o usuário possa testar e validar diferentes possibilidades para simplificar expressões booleanas recebendo *feedbacks* em casos de acertos e erros.

Este artigo está organizado da seguinte maneira, na Seção 2 são apresentados os fundamentos teóricos necessárias para compreensão do trabalho. Na Seção 3 são descritos os trabalhos relacionados ao contexto deste trabalho. A metodologia adotada, é apresentada na Seção 4. Na Seção 5 os resultados obtidos são exibidos. Por fim, na Seção 6 são descritas conclusões relativas a esse trabalho.

2. Fundamentação teórica

Este capítulo apresenta conceitos teóricos relacionados ao trabalho, incluindo o conceito de Sistemas Digitais, Álgebra Booleana e *Feedbacks*.

2.1. Sistemas Digitais

Sistema Digital (SD) é um sistema que manipula elementos discretos da informação representados internamente no formato binário [Mano and Ciletti 2018]. Para compreender o funcionamento de um SD e cada um dos seus módulos é necessário ter conhecimento sobre circuitos digitais. Os Computadores Pessoais, ou *Personal Computer (PCs)*, modernos são os melhores exemplos de sistemas digitais na atualidade.

2.2. Álgebra Booleana

Álgebra Booleana é um sistema algébrico criado por George Boole em 1854, mais tarde Claude E. Shannon em 1938 propôs a utilização do sistema criado por Boole para representar circuitos de sistemas elétricos [Barnett 2011]. As operações fundamentais da álgebra boolena são *AND*, *OR* e *NOT* representadas respectivamente por $\cdot$, $+$, $\sim$. Para a aplicação em circuitos digitais as operações booleanas são definidas em um conjunto fechado $B = \{0, 1\}$. As regras para os dois operadores binários $+$ e $\cdot$ e do operador $\sim$ são definidas na Tabela 1.

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	$\sim x$
0	1
1	0

Tabela 1. Regras para operações booleanas

A partir das regras básicas e das identidades é possível criar um conjunto de leis que podem ser usadas no processo de simplificação de uma expressão booleana. O conjunto de leis consideradas nesse trabalho estão expostas na Tabela 2.

Nome	*AND*	*OR*
Identity (Identidade)	$1 \cdot A = A$	$0 + A = A$
Null (Elemento Nulo)	$0 \cdot A = 0$	$1 + A = 1$
Idempotent (Idempotência)	$A \cdot A = A$	$A + A = A$
Inverse (Elemento Inverso)	$A \cdot \sim A = 0$	$A + \sim A = 1$
Commutative (Comutativa)	$A \cdot B = B \cdot A$	$A + B = B + A$
Associative (Associativa)	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	$(A + B) + C = A + (B + C)$
Distributive (Distributiva)	$A + (B \cdot C) = (A + B) \cdot (A + C)$	$A \cdot (B + C) = A \cdot B + A \cdot C$
Absorption (Absorção)	$A \cdot (A + B) = A$	$A + (A \cdot B) = A$
De Morgan's	$\sim (A \cdot B) = \sim A + \sim B$	$\sim (A + B) = \sim A \cdot \sim B$

Tabela 2. Leis da álgebra booleana

2.3. *Feedbacks*

Feedbacks são normalmente usados como forma de apresentar uma informação ao aprendiz em resposta a alguma ação realizada por ele. Eles podem ser apresentado de diferentes maneiras, variando tanto no momento em que são fornecidos (imediatamente após a ação ou após um tempo determinado) quanto no formato e conteúdo (como a oferta de sugestões, explicações da resposta correta, ou a verificação das respostas fornecidas pelo aprendiz, etc.) [Shute 2008].

O processo de tentativa e erro permite que o estudante explore soluções de forma independente, mas sem orientação externa, ele pode ter dificuldade em confirmar se suas respostas estão corretas. Embora esse método seja útil para aprofundar o aprendizado, há o risco de reforçar estratégias incorretas caso não haja um *feedback* adequado [Metcalfe and Kornell 2007].

Retornos imediatos após a ação do estudante se mostram efetivos de forma geral e no ensino de disciplinas como matemática e programação [Anderson et al. 1995, Dihoff et al. 2003]. Pela natureza do problema abordado neste trabalho se torna quase impossível construir *feedbacks* específicos para os diversos tipos de erros que um estudante pode cometer durante o processo de simplificação de uma expressão booleana, já que é permitido que o estudante realize a simplificação de qualquer circuito.

3. Trabalhos relacionados

Nesta seção alguns simuladores de circuitos serão analisados considerando principalmente se possuem pelo menos uma ferramenta útil para o ensino e aprendizagem de cir-

cuitos digitais, como visualização de tabelas verdade, mapas de Karnaugh ou exibição da simplificação dos circuitos. Além disso, os simuladores analisados são gratuitos e possuem código aberto disponível em algum repositório online que permite o controle de versão, armazenamento e visualização do código fonte.

TkGate [Hansen 2016] é um programa focado somente na criação de circuitos digitais. Funcionalidades básicas para um simulador estão presentes, como salvar um circuito para reaproveitar em circuitos maiores. Porém não oferece nenhuma das funcionalidades para ajudar no processo de ensino que foram apresentadas anteriormente.

O simulador *Qucs (Quite universal simulator)* [Jahn 2023], oferece diversas ferramentas para simulação de circuitos, porém ele foge do escopo deste trabalho, pois, como o próprio nome sugere, ele também oferece suporte a simulação de circuitos analógicos. Por conta disso, o programa acaba deixando de lado ferramentas como a simplificação do circuito com mapa de karnaugh, mas é possível gerar a tabela verdade de um circuito.

Logisim [Burch 2002] surgiu com o propósito de ser usado no estudo de conceitos básicos de circuitos digitais. Por isso, o projeto possui opções para gerar mapas de Karnaugh para cada *output*, além de mostrar a expressão booleana que representa o circuito atual, permitindo que o usuário edite a expressão e gere um novo circuito a partir dela. O autor original do projeto acabou suspendendo por tempo indefinido o desenvolvimento e manutenção do projeto em 2014 [Burch 2014], desde então o projeto não recebe atualizações.

Como o *Logisim* possui código aberto, outros projetos foram criados a partir dele e um dos mais utilizado atualmente, considerando os dados encontrados no *GitHub*, é o *Logisim-evolution* [Evolution 2023]. *Logisim-evolution* é um *fork* do projeto original, isso significa que o projeto utiliza o código fonte original como base para implementar novas funcionalidades e realizar correção de *bugs*. Dentre as principais contribuições deste *fork* estão: Atualização dos elementos da interface gráfica, atualização da versão do Java utilizada, correção de *bugs* [Yakoh 2018], expressões booleanas podem ser visualizadas em diferentes formatos (matemático, lógico, programação). Além disso, o projeto continua recebendo atualizações, atualmente a última versão estável foi lançada em 2022.

A descontinuação do projeto *Logisim* motivou a criação do *Digital* [Neemann 2023] em 2016. O objetivo do *Digital* é resolver problemas na própria construção do *Logisim*, que ainda não tinham sido resolvidos pelos outros *forks* na época de sua criação. Por esse motivo, *Digital* não é um *fork* do *Logisim*, ele é um programa escrito do zero que resolve problemas existentes no projeto original especialmente na performance de simulações mais complexas. Apesar de focar na performance, o *Digital* não deixa de oferecer funcionalidades para exibir os mapas de Karnaugh, tabela verdade e expressão simplificada de um circuito. Além disso é possível criar um circuito a partir de uma expressão booleana, mas não é possível visualizar o circuito atual como uma expressão booleana sem simplificação.

Existem também ferramentas mais simples disponíveis na *internet* que podem ser usadas apenas para realizar apenas o processo de simplificação de expressões booleanas. Algumas dessas ferramentas apresentam quais propriedades foram aplicadas, mas, assim como os simuladores apresentados, não permitem que o usuário teste diferentes propriedades durante o processo para que erre ou acerte gerando aprendizado

[eMathHelp 2024, Symbolab 2024, Algebra 2024].

A Tabela 3 apresenta um resumo dos simuladores analisados considerando os seguintes parâmetros:

a) Popularidade: Medida pela quantidade de estrelas no *GitHub*. Esse parâmetro é essencial para garantir que a ferramenta possui uma comunidade ativa. Uma comunidade ativa significa que mais pessoas estão ajudando na resolução de problemas e implementação de novas funcionalidades na aplicação. O único simulador que não possui página oficial no *Github* é o *TkGate*, por isso esse campo possui o valor N/A (Não Aplicável).
b) Última *release*: A data da última *release* estável nas páginas oficias. Uma *release* recente ajuda a definir se as mudanças propostas pela comunidade estão sendo implementadas e disponibilizadas em versões estáveis para o usuário final.
c) Ferramentas para aprendizado: O programa deve possuir ferramentas que auxiliem o aprendizado de conceitos básicos de circuitos digitais, como visualização de tabela verdade, mapa de Karnaugh e expressão booleana de um circuito.

Nome	Popularidade	Última *release*	Ferramentas para aprendizado
Qucs	1029	22/05/2019	Sim
TkGate	N/A	17/02/2016	Não
Digital	3191	03/02/2023	Sim
Logisim-evolution	3564	02/10/2022	Sim

Tabela 3. Simuladores de circuitos

4. Metodologia

Nesta seção é apresentada a descrição das novas funcionalidades implementadas, a descrição geral do funcionamento do sistema e o processo de testes realizados para validação das funcionalidades propostas.

4.1. Descrição das novas funcionalidades

O objetivo principal das modificações adicionadas é permitir que o usuário possa testar se uma expressão é equivalente ou não a expressão original do circuito. A Figura 1 demonstra o processo de interação do usuário com o sistema. Cada nova expressão válida informada pelo usuário ficará registrada na tela, dessa forma ele sabe todos os passos realizados durante o processo de simplificação.

Caso o aluno informe uma expressão inválida, além de receber a informação de que a expressão não é válida são também mostradas as tabelas verdades da expressão informada e da expressão original. Ao exibir as tabelas verdades o aluno percebe o motivo da expressão ser inválida, já que somente avisar que a resposta é incorreta pode não ajudar no processo de aprendizado [Pashler et al. 2005].

Outra funcionalidade adicionada é a possibilidade do aluno poder a qualquer momento pressionar um botão de ajuda que exibe na tela possíveis propriedades que podem ser aplicadas na expressão que foi digitada e o estado da expressão após a aplicação da propriedade. Não necessariamente a aplicação das propriedades exibidas resulta

em uma expressão simplificada, mas podem servir como um passo para o processo de simplificação.

Durante a fase de desenvolvimento das novas funcionalidades, o código fonte original foi analisado a fim de verificar quais partes do projeto original poderiam ser usadas para realizar o processo de simplificação e quais deveriam ser adicionadas ou modificadas para atender aos requisitos das novas funcionalidades. Toda as partes relacionadas a representação das expressões foi reaproveitada sendo necessário somente adicionar as regras para verificar as propriedades que podem ser aplicadas em uma expressão e a interface gráfica para o usuário.

4.2. Verificação e validação

Uma parte essencial das novas funcionalidades propostas é a correta simplificação das expressões booleanas para que seja possível mostrar para o usuário as propriedades que podem ser aplicadas na expressão. Portanto, foram realizados testes a fim de garantir que as manipulações algébricas realizadas estejam corretas e sigam as leis da álgebra booleana consideradas na Tabela 2.

Para conseguir abordar a maior quantidade possível de casos de simplificação foram considerados esquemas de partição em que são considerados a lei que deve ser aplicada, a forma *OR* e forma *AND* da lei. Para isso foram passados ao método responsável pela sugestão de simplificação a expressão original e após isso o retorno do método foi analisado para garantir que a lei sugerida e a simplificação realizada estão corretas. A Tabela 4 demonstra os testes realizados para os casos bases com suas respectivas expressões originais, lei aplicada e expressão resultante. Para garantir que todos os casos descritos estavam funcionando também para o usuário final os testes foram repetidos através da interface de usuário digitando a expressão original e verificando a simplificação sugerida

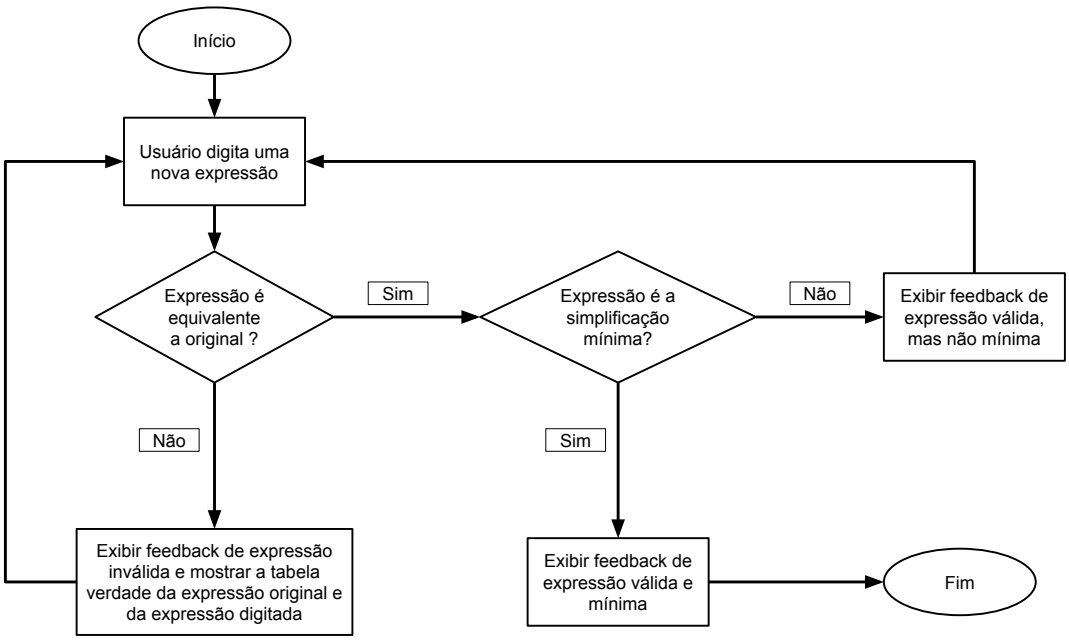

Figura 1. Fluxograma descrevendo a interação do usuário no sistema.

pelo sistema na interface. Todos os testes descritos na tabela foram executados corretamente em ambos os casos.

Expressão original	Lei aplicada	Expressão resultante
$a \cdot 1$	*Identity*	a
$a + 0$	*Identity*	a
$a \cdot 0$	*Null*	0
$a + 1$	*Null*	1
$a \cdot a$	*Idempotent*	a
$a + a$	*Idempotent*	a
$a \cdot b \cdot a$	*Idempotent*	$a \cdot b$
$a + b + a$	*Idempotent*	$a + b$
$a \cdot \sim a$	*Inverse*	0
$a + \sim a$	*Inverse*	1
$a + (b \cdot c)$	*Distributive*	$(a + b) \cdot (a + c)$
$a \cdot (b + c)$	*Distributive*	$(a \cdot b) + (a \cdot c)$
$(a + b) \cdot (a + c)$	*Distributive*	$a + (b \cdot c)$
$(a \cdot b) + (a \cdot c)$	*Distributive*	$a \cdot (b + c)$
$(a + b) \cdot d \cdot (a + c)$	*Distributive*	$(a + (b \cdot c)) \cdot d$
$(a \cdot b) + d + (a \cdot c)$	*Distributive*	$(a \cdot (b + c)) + d$
$a \cdot (a + b)$	*Absorption*	a
$a + (a \cdot b)$	*Absorption*	a
$a \cdot d \cdot (a + b)$	*Absorption*	$a \cdot d$
$a + d + (a \cdot b)$	*Absorption*	$a + d$
$\sim (a \cdot b)$	*DeMorgan's*	$\sim a + \sim b$
$\sim (a + b)$	*DeMorgan's*	$\sim a \cdot \sim b$
$\sim a + \sim b$	*DeMorgan's*	$\sim (a \cdot b)$
$\sim a \cdot \sim b$	*DeMorgan's*	$\sim (a + b)$

Tabela 4. Descrição dos testes realizados

5. Resultados e discussão

Nesta seção será apresentado o funcionamento do sistema para o processo de simplificação da expressão $a \cdot \sim b + a \cdot b + a \cdot c$. que requer múltiplos passos para chegar a simplificação mínima. O processo será executado seguindo o fluxograma da Figura 1, iniciando com uma expressão inválida, prosseguindo usando a ajuda do sistema até chegar a expressão mínima.

Para acessar as funcionalidades implementadas o usuário deve criar um circuito qualquer na interface gráfica do *Logisim-evolution*, após isso ele terá que na barra superior clicar em "Projeto", depois "Analisar circuito" e por fim selecionar a guia "Algebra" na nova tela que irá aparecer. Na guia "Algebra"é onde ocorre o processo de simplificação interativa. Inicialmente o usuário irá visualizar a expressão original e sua respectiva tabela verdade com título "Expressão original" como demonstrado na Figura 2.

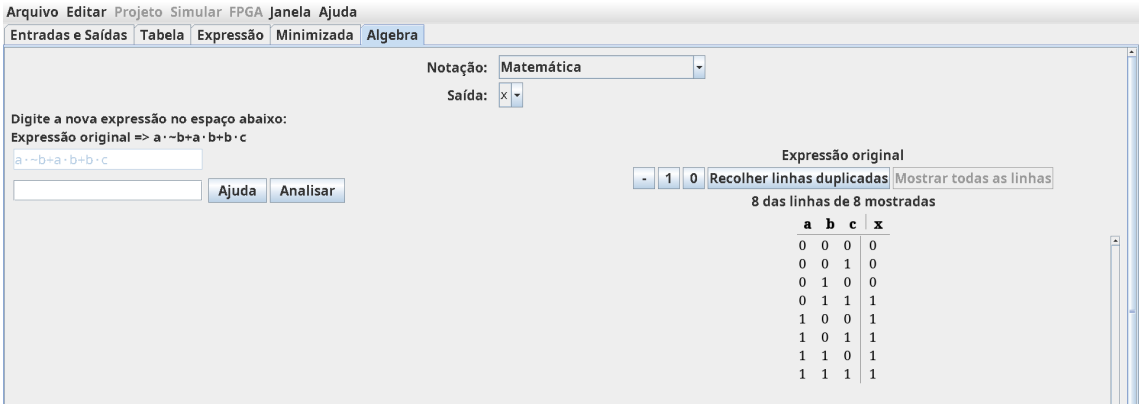

Figura 2. Tela inicial ao acessar a funcionalidade de simplificação.

Após digitar uma expressão inválida e pressionar o botão ”Analisar” o sistema irá exibir a mensagem ”A expressão digitada não é equivalente” e mostrar a tabela verdade da expressão original e da expressão digitada, Figura 3. Como é possível observer na figura a tabela verdade gerada pela expressão digitada não é igual a tabela verdade gerada pela expressão original, portanto as duas expressões não são equivalentes e a simplificação é inválida.

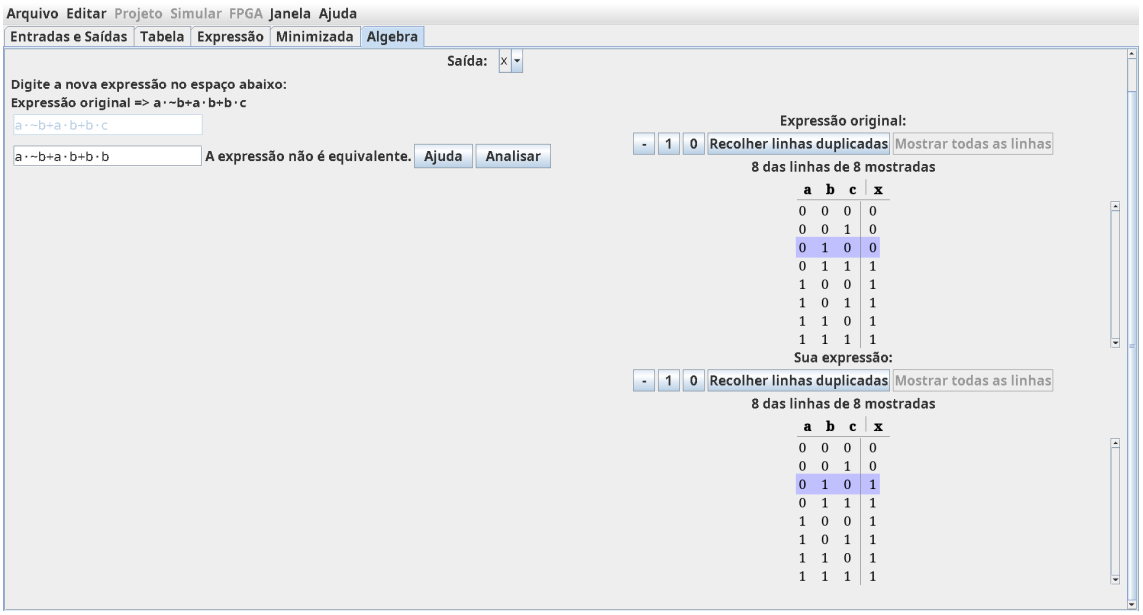

Figura 3. Tela com mensagem de erro ao digitar uma expressão que não é equivalente.

Caso o usuário pressione o botão ”Ajuda” serão mostradas as possíveis manipulações para a expressão digitada. Como é possível verificar na Figura 4, o sistema irá apresentar todas as manipulações possíveis, mas algumas delas não ajudariam no processo de simplificação. Cabe ao aluno decidir qual propriedade seria mais adequada em cada caso.

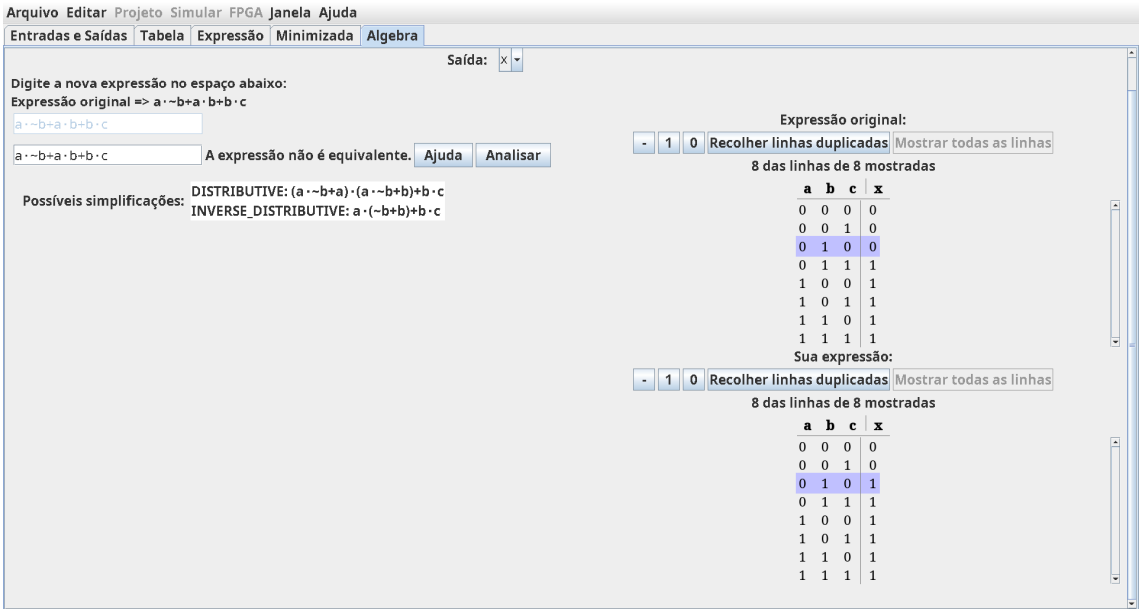

Figura 4. Sugestões exibidas para a expressão $a\cdot \sim b + a \cdot b + b \cdot c$ ao clicar no botão Ajuda.

Ao chegar na expressão mínima é exibida a mensagem ”Expressão valída e mínima” Figura 5. Caso a expressão digitada seja equivalente, mas ainda não é a expressão mínima é exibida a mensagem ”Simplificação válida, mas não é a mínima”. Após chegar na expressão mínima o aluno ainda pode continuar usando as outras funcionalidades normalmente para testar outras possibilidades de simplificação, repetindo o processo de através dos botões ”Analisar” e ”Ajuda”.

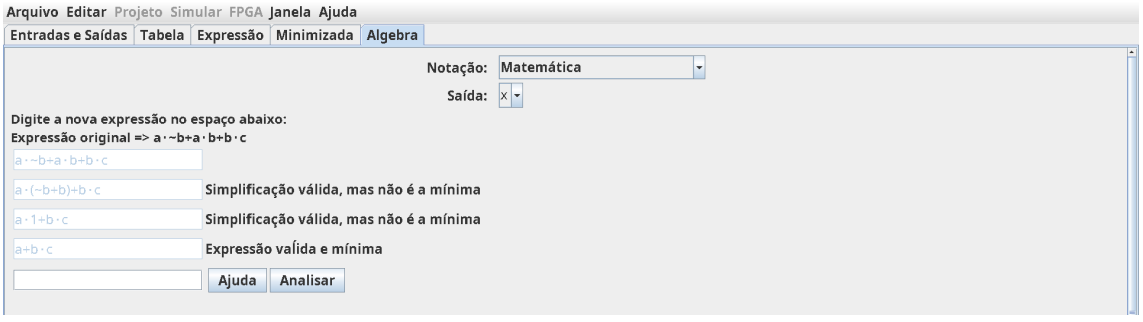

Figura 5. Estado da tela quando o usuário finaliza o processo de simplificação.

Funcionalidades que podem ser úteis durante o processo de simplificação e que já estavam presentes no sistema foram aproveitadas na guia ”Algebra”. É possível a qualquer momento alterar a representação das expressões dentro das opções fornecidas pelo sistema que são ”Matemática”, ”Lógico”, ”Lógica Alternativa”, ”Programação com Boolean’s”, ”Programação com bits”. Além disso é possível escolher qual a saída está sendo considerada no momento, para circuitos que possuem mais de uma saída, dessa forma é possível simplificar as expressões de cada saída individualmente.

6. Conclusões e trabalhos futuros

Neste trabalho foram apresentadas alterações realizadas no simulador de circuitos digitais *Logisim-evolution* para auxiliar o processo de aprendizado de simplificações de expressões booleanas aplicando as propriedades da álgebra booleana. Com as alterações, o discente pode não somente obter uma expressão minimizada ao final do processo, mas também tentar por conta própria simplificar a expressão passo a passo com a ajuda do sistema recebendo *feedbacks* para conseguir aprender através de tentativa e erro ou solicitando sugestões de possíveis aplicações de propriedades.

Com as funcionalidades apresentadas o simulador se mostra como uma ferramenta útil ao processo de ensino e aprendizagem de sistemas digitais. Para trabalhos futuros, as implementações realizadas devem ser validadas em um ambiente de ensino e aprendizagem real, onde os discentes e docentes possam fornecer suas opiniões sobre a ferramenta e os resultados do uso da ferramenta podem ser avaliados para verificar o seu impacto no processo de aprendizado.

Disponibilidade de artefatos

O código fonte do simulador Logisim-evolution junto com as alterações realizadas estão disponíveis no repositório *"logisim-evolution-edu"* no GitHub.

Referências

Algebra, B. (2024). Boolean algebra simplifier. `https://www.boolean-algebra.com/`. Acessado em: 01/06/2024.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207.

Baneres, D., Clariso, R., Jorba, J., and Serra, M. (2014). Experiences in digital circuit design courses: A self-study platform for learning support. *IEEE Transactions on Learning Technologies*, 7(4):360–374.

Barnett, J. H. (2011). Applications of boolean algebra: Claude shannon and circuit design. *Available from the webpage http://www. cs. nmsu. edu/historical-projects*.

Burch, C. (2002). Logisim: A graphical system for logic circuit design and simulation. *J. Educ. Resour. Comput.*, 2(1):5–16.

Burch, C. (2014). Logisim development officially suspended. `http://www.cburch.com/logisim/retire-note.html`. Acessado em: 01/06/2024.

Dihoff, R. E., Brosvic, G. M., and Epstein, M. L. (2003). The role of feedback during academic testing: The delay retention effect revisited. *The Psychological Record*, 53:533–548.

eMathHelp (2024). Boolean algebra calculator. `https://www.emathhelp.net/en/calculators/discrete-mathematics/boolean-algebra-calculator/`. Acessado em: 01/06/2024.

Evolution, L. (2023). Digital logic design tool and simulator. `https://github.com/logisim-evolution/logisim-evolution`. Acessado em: 01/06/2024.

Hansen, J. P. (2016). Tkgate download. https://sourceforge.net/projects/tkgate/. Acessado em: 01/06/2024.

Jahn, S. (2023). Qucs project official mirror. https://github.com/Qucs/qucs/. Acessado em: 01/06/2024.

Mano, M. and Ciletti, M. (2018). *Digital Design, Global Edition*. Pearson Education.

Metcalfe, J. and Kornell, N. (2007). Principles of cognitive science in education: The effects of generation, errors, and feedback. *Psychonomic Bulletin & Review*, 14:225–229.

Neemann, H. (2023). A digital logic designer and circuit simulator. https://github.com/hneemann/Digital. Acessado em: 01/06/2024.

Nikolic, B., Radivojevic, Z., Djordjevic, J., and Milutinovic, V. (2009). A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization. *IEEE Transactions on Education*, 52(4):449–458.

Pashler, H., Cepeda, N. J., Wixted, J. T., and Rohrer, D. (2005). When does feedback facilitate learning of words? *Journal of experimental psychology: Learning, Memory, and Cognition*, 31(1):3.

Prasad, P. C., Alsadoon, A., Beg, A., and Chan, A. (2014). Incorporating simulation tools in the teaching of digital logic design. In *2014 IEEE international conference on control system, Computing and engineering (ICCSCE 2014)*, pages 18–22. IEEE.

Shute, V. J. (2008). Focus on formative feedback. *Review of educational research*, 78(1):153–189.

Symbolab (2024). Boolean algebra calculator. https://www.symbolab.com/solver/boolean-algebra-calculator. Acessado em: 01/06/2024.

Yakoh, T. (2018). Inappropriate minimizing in combinational analysis. https://github.com/logisim-evolution/logisim-evolution/issues/124. Acessado em: 01/06/2024.

Zhu, Y. and Howell, S. (2023). Independent and creative learning in a digital electronics course using a web-based circuit simulator. *Computer Applications in Engineering Education*, 31(3):634–641.