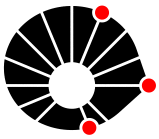
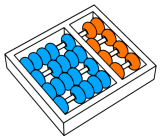




Executando um simulador de Modelo Ising em uma FPGA

André Luiz Vasconcelos Ferreira

Technical Report - IC-25-01 - Relatório Técnico
June - 2025 - Junho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Executando um simulador de Modelo Ising em uma FPGA

André Luiz Vasconcelos Ferreira*

Resumo

O modelo Ising é uma representação matemática de uma malha de spins cujo objetivo é representar o comportamento do ferromagnetismo em um sistema físico. Muitas pesquisas se beneficiam do método de Monte Carlo na implementação de um simulador que emula cenários diversos envolvendo o modelo Ising e suas variantes. Todavia, ao ignorar questões de hardware, o software criado pode não se aproveitar de todo seu potencial de simulação. Para superar essa questão, o uso de aceleradores, como um implementado por uma FPGA, se torna interessante. O presente projeto propõe o estudo da aplicação de um simulador de modelo Ising baseado no método de Monte Carlo em um simulador LiteX que emula o funcionamento de uma FPGA.

1 Introdução

O campo de Aprendizado de Máquina tem recebido destaque nos últimos anos e, naturalmente, suas técnicas são utilizadas em pesquisas de outras áreas do conhecimento além da computação, sendo uma delas a física [3, 4]. Um dos assuntos dessa ciência natural cujos estudos se beneficiam da inteligência artificial são os modelos Ising, esquemas matemáticos responsáveis pela descrição de propriedades do ferromagnetismo [5]. Pesquisas envolvendo tal representação numérica, e suas variantes, demonstram com seus resultados que o uso delas com auxílio do método Monte Carlo é eficiente na identificação da transição de fases de um sistema físico [2, 3, 14].

Muitas pesquisas foram desenvolvidas explorando o método Monte Carlo na investigação de modelos Ising, havendo inclusive aquelas realizadas nos anos 1980 [14]. Como explicado por Assis Elias et al. [3], existem duas formas principais de lidar com a questão. A primeira é tratando-a como um problema de parâmetro único [6, 9]. Nessa abordagem, apenas uma variável do sistema físico varia enquanto as outras permanecem fixadas. A vantagem de um sistema com essa característica é que ele permite encontrar valores específicos de componentes como, por exemplo, temperatura em sistemas físicos com características igualmente específicas [3]. A outra abordagem é por meio do aproveitamento da capacidade de reconhecimento de padrões do Aprendizado de Máquina para coletar informações de um sistema físico de forma mais geral, sem se limitar à variação de um único valor em um ambiente controlado [1, 2, 3, 7]. Com essa segunda forma de estudar o tema, é possível criar diagramas de fases que demonstrem o comportamento de um sistema físico como um todo, o que é útil quando o objeto de pesquisa de um cientista é um ambiente cujos fenômenos não possuem formas de agir previamente mapeadas [3].

Diante do histórico de trabalhos envolvendo o tema, é possível notar como o uso do método Monte Carlo é relevante na investigação de modelos Ising. Apesar disso, uma solução que se baseia apenas na aplicação do algoritmo pode não se aproveitar de seu potencial máximo ao desconsiderar questões de hardware. Por exemplo, Preis et al. [13] se aproveitam dos recursos de uma GPU para aumentar em 60 vezes a velocidade de execução do algoritmo de Monte Carlo para um modelo

*andreluizvf123@gmail.com

Ising. Pouco tempo depois, Block et al. [4] criam um sistema em que o tempo de computação tem seu desempenho melhorado quase que linearmente a cada adição de GPU na execução do simulador. Posteriormente, diversos pesquisadores continuaram a explorar o uso de GPUs para criar simuladores de modelos Ising. O uso de aceleradores implementados em FPGA também atrai a curiosidade de pesquisadores. Um trabalho que se destaca nesse sentido é o de Ortega-Zamorano et al. [12], em que os autores constroem uma circuito lógico capaz de executar o simulador de modelo Ising e se integrado a uma FPGA.

O presente trabalho, portanto, detalha a implementação de um acelerador que auxilia a execução de um simulador de modelo Ising baseado no método Monte Carlo. O acelerador será executado no ambiente de simulação LiteX e sua performance será comparada com a de um outro simulador de modelo Ising implementado em C e executado em um desktop comum.

2 Fundamentação teórica

2.1 Modelo Ising 2D

Assis Elias [2] explicam que o modelo Ising 2D é uma representação matemática de um ferromagneto composto por uma malha de spins [2]. Nele, setas posicionadas para cima e para baixo indicam, respectivamente, os valores +1 e -1. Essas setas são dispostas em uma grade de duas dimensões, tal como na figura 1.

FIGURA 1

A energia do sistema é descrita por uma equação Hamiltoniana da Expressão 1.

$$H(x) = -J \sum_{<i,j>} \sigma_i \sigma_j \tag{1}$$

Em que J é a energia resultante da interação entre dois spins posicionados lado a lado quaisquer e a notação $< i, j >$ indica que o somatório se aplica aos pares de spins (σ_i, σ_j) do sistema físico que são adjacentes entre si.

A segunda derivada da expressão 1, quando aplicada ao modelo Ising 2D, indica uma descontinuidade que permite a identificação de mudança de um sistema físico para uma fase magnetizada ou contínua, situação que pode ser verificada experimentalmente [2].

A partir dessas informações é possível deduzir que essa representação matemática dos sistemas ferromagnéticos se mostra útil na investigação do fenômeno de transição de fases deles. A partir dos anos 1980, [14] mostraram que pesquisas desse tema podem se aproveitar dos métodos Monte Carlo.

2.2 Métodos Monte Carlo e o algoritmo de Metropolis

Os métodos Monte Carlo compõem um campo da matemática experimental que visa a realização de experimentos a partir de amostragens aleatórias [8]. Uma de suas formas de gerar dados é por meio da Cadeia de Markov Monte Carlo (MCMC), que se baseia no algoritmo de Metropolis-Hastings [10]. Como é explicado por Kroese et al. [10], suponha que é necessário criar amostragens de uma função $f(x)$ dada pela Expressão 2.

$$f(x) = \frac{p(x)}{Z} \tag{2}$$

Em que $p(x)$ é uma função conhecida e Z é uma constante de normalização. A aplicação do algoritmo de Metropolis se dá através da geração de uma série de T estados X_t, com $t \in$

$\{0, 1, \cdots, T-1\}$, que representam uma mudança de estados do modelo representado pela equação $f(x)$ ao longo do tempo. Dessa forma, primeiro é definido um estado X_0 e depois os seguintes passos são seguidos:

1. Criar um novo estado Y a partir do estado atual X_t. A probabilidade de X_t se transformar em Y é dada por uma equação $q(y|X_t)$;

2. Sortear um número U entre 0 e 1 e atualizar o valor do estado seguinte, X_{t+1}, a partir da Expressão 3

$$X_{t+1} = \begin{cases} Y & ,U \leq \alpha(X_t, Y) \\ X_t & ,U > \alpha(X_t, Y) \end{cases} \tag{3}$$

   em que α é dado pela Expressão 4.

$$\alpha(x, y) = \min\left(\frac{f(y) * q(y|x)}{f(x) * q(x|y)}, 1\right) \tag{4}$$

.

O algoritmo de Metropolis é utilizado em projetos de diversas naturezas, sendo um deles nas simulações de modelo Ising, como será explicado na seção 2.3.

2.3 Simulação de modelo Ising com método Monte Carlo

A simulação do modelo Ising com método Monte Carlo indica quando os seus spins mudam de sentido, baseando isso no objetivo de deixar o sistema físico em equilíbrio térmico. Para isso, é importante definir algumas igualdades, como será feito a seguir.

A probabilidade ρ_μ de um sistema físico estar no estado μ possuindo uma energia E_μ é dada pela distribuição de *Boltzmann*, representada pela Expressão 5 [11].

$$\rho_\mu = \frac{e^{-E_\mu \frac{1}{Tk_b}}}{Z} \tag{5}$$

Em que T é a temperatura do sistema físico, k_b é a constante de *Boltzmann* e Z é uma constante de normalização. A Expressão 5 é a função relativa à Expressão 2 para o contexto do simulador de modelo Ising.

Para um sistema físico estar em equilíbrio térmico, a probabilidade de ele transitar de um estado μ para um estado υ deve ser igual à sua recíproca, sendo essa condição expressa pela equação 6 [2].

$$\rho_\mu q(\mu|\upsilon) = \rho_\upsilon q(\upsilon|\mu) \tag{6}$$

Outro detalhe importante é que a energia E do sistema físico muda com a mudança de direção de um de seus spins. O cálculo da energia do sistema é feito através da Expressão 1.

A partir dessas condições o algoritmo de Metropolis é programado para realizar a mudança de estados, definida pelo novo direcionamento de um spin, do simulador de modelo Ising. Isso é feito a partir das etapas a seguir:

1. Definir uma configuração de spins aleatória para o modelo Ising 2D e calcular sua energia com a Expressão 1. Essa configuração pode ser lida como o estado X_0 do algoritmo de Metropolis;

2. Modificar a direção de um spin aleatório. Isso denota um potencial novo estado Y do sistema físico simulado;

3. Calcular a energia do estado Y com a expressão 1 e sua diferença ΔE em relação à energia do estado atual X_t;

   (a) Se $\Delta E < 0$, $X_{t+1} = Y$;

   (b) Se $\Delta E \geq 0$, sortear um número U entre 0 e 1 e aplicar a mudança de estado do algoritmo de Metropolis com as equações 7 e 8;

$$X_{t+1} = \begin{cases} Y & ,U \leq \alpha_{Ising}(X_t, Y) \\ X_t & ,U > \alpha_{Ising}(X_t, Y) \end{cases} \tag{7}$$

$$\alpha_{Ising}(x, y) = \min\left(\frac{\rho_\mu q(\mu|\upsilon)}{\rho_\upsilon q(\upsilon|\mu)}, 1\right) = \min\left(e^{\frac{-\Delta E}{Tk_b}}, 1\right) \tag{8}$$

   Perceba que as equações 7 e 8 são relativas às equações 3 e 4 da Seção 2.2.

4. Repetir os passos 2 e 3.

Dessa forma é possível simular um sistema físico com modelo Ising através do algoritmo de Metropolis.

3 Metodologia

Para realização do presente projeto, primeiro foi implementado um simulador de modelo Ising 2D na linguagem C. Ele foi executado em um desktop comum e a partir de um *profiling* foi decidido qual parte de seu funcionamento seria otimizado por meio da implementação de um acelerador.

3.1 Aplicação do *profiling* sobre o simulador

O sistema implementado possui duas etapas principais, que são a geração de uma malha de spins com configuração aleatória e a aplicação do algoritmo de Metropolis sobre ela. O *profiling* permitiu a análise da porcentagem de tempo que essas partes ocupam na execução total do algoritmo. Os resultados do processo estão na Tabela 1.

Tabela 1: Resultados do profiling

	Tempo (segundos)	Porcentagem
Criação da malha de spins	0.001699	2%
Algoritmo de Metropolis	0.656738	98%
Tempo de execução total	0.671706	100%

A partir dessa análise, conclui-se que o acelerador deve ser implementado para executar o algoritmo de Metropolis.

3.2 Implementação do acelerador

O acelerador desenvolvido foi implementado com a linguagem de descrição de hardware Verilog. Ele foi baseado no trabalho de Ortega-Zamorano et al. [12] e tem como principal objetivo paralelizar a atualização dos spins. Dessa forma, ao invés de atualizar apenas um spin por iteração, o sistema atualiza um número consideravelmente maior, o que acelera a evolução do modelo Ising no objetivo de atingir o equilíbrio térmico. Além disso, o uso de *lookup tables* (LUT) acelera o tempo de

execução do algoritmo ao poupar tempo na execução de determinados cálculos cujos valores não variam tanto a ponto de exigirem a execução de suas equações em tempo real.

O primeiro ponto a ser entendido na implementação do acelerador é a representação do direcionamento dos spins, que ao invés de ser 1 ou -1 agora é 1 ou 0. Isso permite o uso de operações lógicas, como *OR* e *XOR*, nas operações que envolvem a atualização desses elementos.

Em seguida é importante entender a implementação do módulo "spin". Ele tem como entradas o direcionamento do spin que representa, o direcionamento de seus vizinhos, dados pelas entradas "left","right","top" e "bottom", e um valor aleatório "random".

O cálculo da variação de energia dE se dá por meio de uma LUT e parte da noção de que seus potenciais valores se limitam a 0, 2, 4, -2 e -4. Quando um spin possui apenas dois vizinhos no sentido contrário ao seu, sua variação de energia é 0. Quando um spin possui 3 vizinhos no sentido contrário ao seu, sua variação de energia é -2. Quando todos os vizinhos do spin possuem um sentido contrário ao seu, sua variação de energia é -4. Por outro lado, quando todos os vizinhos possuem a mesma direção do spin central, a variação de energia é 4. Quando esse número é de três vizinhos a variação de energia é 2. Sendo assim, a Tabela 2 representa como alguns valores são mapeados baseando-se na concatenação do valor do direcionamento do spin central com o valor do direcionamento de seus quatro vizinhos.

Tabela 2: Lookup-table do dE

{spin,right,left,top,bottom}	dE
00000	4
00001	2
00010	2
00011	0
00100	2
00101	0
00110	0
00111	-2
...	...

De forma análoga, o cálculo do valor de α_{Ising}, da equação 8, se da por meio de uma *lookup-table*. Os valores da expressão $e^{\frac{-\Delta E}{Tk_b}}$ foram previamente calculados de acordo com os possíveis valores de dE e salvos na estrutura de dados. Nesse caso, sempre que o valor da expressão era maior que 1, seu valor se tornava 1 para ser coerente com a equação 8. Um detalhe importante a ser ressaltado sobre a implementação dessa parte do acelerador é que os valores salvos na *lookup-table* não estão entre 0 e 1, mas sim uma proporção do valor correto dentro do intervalo $[0, 4095]$. Isso ocorre pois o número aleatório utilizado para comparar se haverá *flip* ou não do spin é representado por um registrador de 12 bits, cujo maior valor possível é, em binário, 111111111111, ou, em decimal, 4095. Por fim, um outro detalhe importante é que os valores de α_{Ising} são engessados com a temperatura planejada para o sistema. Caso se deseje simular um sistema com outra temperatura, é necessário reimplementar este módulo do acelerador. A tabela 3 mostra a LUT para um sistema cuja temperatura tem valor 1.

O número aleatório que é comparado com o valor de α_{Ising} é calculado por meio de uma operação de *XOR* entre um número aleatório local obtido com um módulo de *Linear feedback shift register* (LFSR) de 12 bits e o valor da entrada "random", que consiste de um valor obtido com a execução de um LFSR de 32 bits. Esse valor é enviado para um módulo de comparação que retorna 1 se ele é menor que a saída da LUT α_{Ising} e 0 caso contrário.

Tabela 3: Lookup-table do α_{Ising}

dE	alpha (em binário)
0	111111111111
-2	111111111111
-4	111111111111
2	001000101010
4	000001001011

A saída da LUT de dE também é enviada para um módulo de comparação que retorna 1 caso a variação de energia seja negativa e 0 caso seja positiva. As saídas dos dois módulos de comparação são enviadas para um *OR*. A saída do operador lógico então é enviada para um *XOR*, que a compara com o valor inicial do spin. A Figura 1 representa o a estrutura do módulo que implementa o spin completo no acelerador.

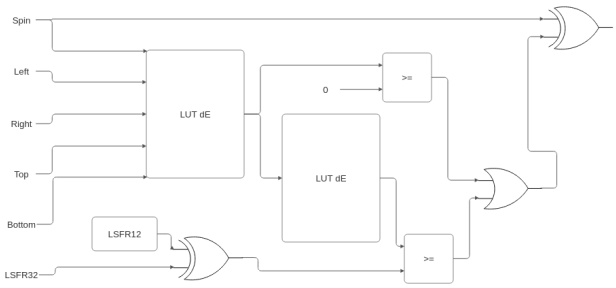


Figura 1: Módulo do acelerador que representa um Spin individual.

Entendido como é a implementação do módulo de um spin individual, agora é importante saber como a malha funciona como um todo. A malha do acelerador é representada por dois conjuntos de spins, o "conjunto branco", com todos os spins que não são vizinhos entre si, e o "conjunto cinza", com todos os spins restantes. Os spins do "conjunto branco" têm como vizinhos spins dentro do "conjunto cinza" e vice versa. Dessa forma, o acelerador atualiza todos os spins de um dos conjuntos a cada ciclo de clock, alternando qual conjunto é atualizado em cada momento. Isso permite a paralelização da atualização dos direcionamento dos spins. O módulo que representa a malha como um todo ainda possui um gerador de números aleatórios baseado no método LFSR com 32 bits. O valor aleatório é enviado a todos os spins a cada ciclo de clock.

4 Experimentos e resultados

Para o trabalho foram desenvolvidas duas aplicações, uma em C executada em um desktop e uma em C e Verilog executada no simulador do LiteX. As Figuras 2 e 3 representam os resultados das duas aplicações para as temperaturas de 0.5,1,5 e 10 em uma malha de spins de dimensão 32x32.

As imagens à esquerda representam a média das direções dos spins nas malhas ao longo do tempo. Isso significa que quanto mais para baixo se encontra a linha azul do gráfico, mais spins apontando para baixo estão presentes no sistema e quanto mais para cima se encontra a linha azul, mais spins positivos se encontram no sistema. Se a linha azul fica no centro do gráfico, quer dizer que o número de spins negativos e positivos do sistema é igual. Quanto menor é a temperatura de um sistema, mais este sistema tende a ter spins a uma das duas direções possíveis. Quanto maior

é a temperatura de um sistema, mais as quantidades de spins negativos e positivos tendem a se equilibrar.

As imagens à direita representam a variação de energia ao longo do tempo. Quanto menor é a temperatura do sistema, mais estável será a variação de energia. Quanto maior é a energia do sistema, mais caótica é a variação de energia.

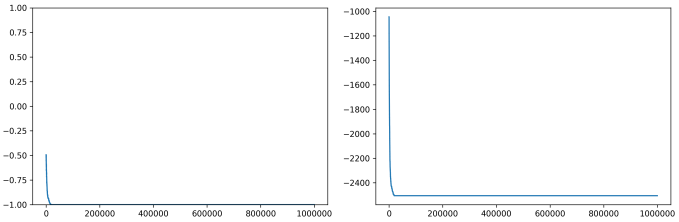
Os resultados de ambas as aplicações são coerentes com o que se espera do modelo Ising, apesar de a aplicação executada no simulador do LiteX apresentar uma sensibilidade maior à variação de temperatura.

A Tabela 4 mostra o tempo médio de execução, em segundos, das etapas do simulador de modelo Ising para as duas aplicações desenvolvidas. No caso, cada aplicação foi executada uma única vez para as temperaturas 0.5, 1, 5 e 10 e o tempo de execução delas foi somado e dividido por 4.

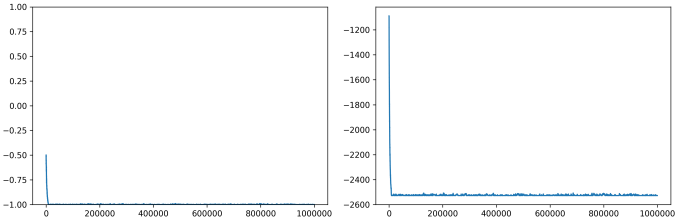
Tabela 4: Resultados do profiling

	Desktop C	LiteX_sim C + Verilog
Criação da malha de spins	0.000044	0.076916
Algoritmo de Metropolis	0.2739718	0.003725
Tempo de execução total	0.274919	0.083014

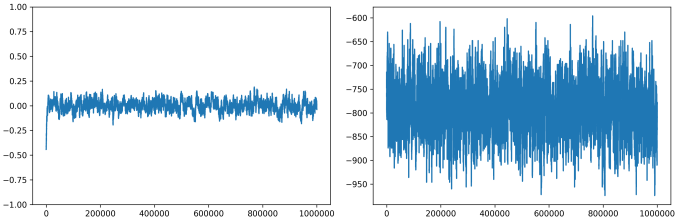
Nota-se que, com a implementação do acelerador, o simulador de modelo Ising teve um *speedup* da execução do algoritmo de Metropolis de aproximadamente 73.5 vezes e um *speedup* do tempo total de aproximadamente 3.3 vezes.



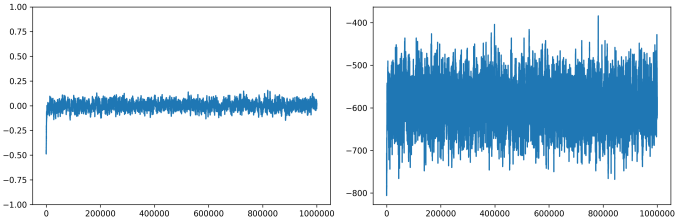
(a) Desktop C - Temperatura=0.5



(b) Desktop C - Temperatura=1

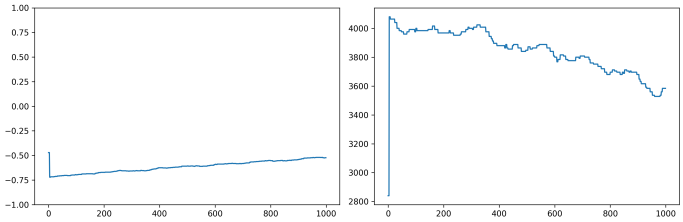


(c) Desktop C - Temperatura=5

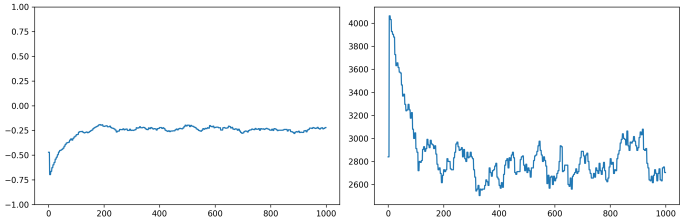


(d) Desktop C - Temperatura=10

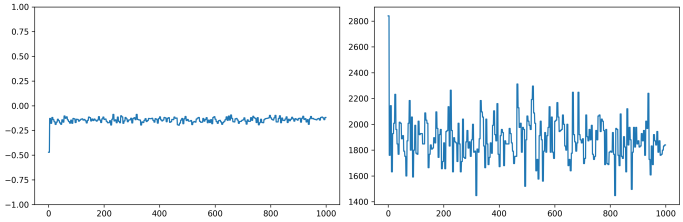
Figura 2: Resultado dos experimentos executados com o código em C, no desktop. A imagem da esquerda indica a média aritmética da soma do direcionamento dos spins do sistema. A imagem da direita indica a variação de energia ao longo do tempo.



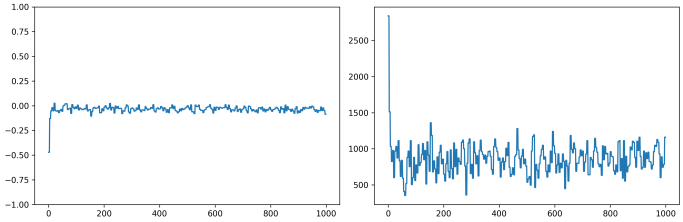
(a) LiteX Sim - Temperatura=0.5



(b) LiteX Sim - Temperatura=1



(c) LiteX Sim - Temperatura=5



(d) LiteX Sim - Temperatura=10

Figura 3: Resultado dos experimentos executados com o código em C e Verilog, no simulador do Litex. A imagem da esquerda indica a média aritmética da soma do direcionamento dos spins do sistema. A imagem da direita indica a variação de energia ao longo do tempo.

5 Contratempos enfrentados

Apesar de os resultados terem sido bons, a aplicação em Verilog apresentou um comportamento estranho. O resultado da variação de energia e do direcionamento médio de spins oscila consistentemente de quatro em quatro iterações do simulador. Dessa forma, os resultados da Figura 3 apresentam o cálculo das duas medidas apenas para iterações divisíveis por 4. O motivo desse resultado ainda é desconhecido, porém existe a hipótese de o comportamento ser causado pelo paralelismo empregado na solução ou por algum erro de implementação não identificado.

Outro problema enfrentado na implementação do simulador de modelo Ising foi a geração de números aleatórios no ambiente do simulador do LiteX. De fato, é possível gerar números aleatórios, mas o uso de seeds diferentes para os métodos LFSR implementados, bem como para a função "rand" do C, sem auxílio do "time(NULL)" fizeram com que exista apenas uma aleatoriedade no sistema, uma vez que não existem seeds diferentes para ele.

6 Conclusão

Apesar dos contratempos citados anteriormente, o simulador de modelo Ising foi implementado com sucesso no ambiente do simulador do LiteX. Como projetos futuros, pretende-se investigar e corrigir as causas dos *bugs* do sistema atual bem como realizar testes em uma FPGA real.

Referências

[1] S Acevedo, M Arlego, and CA Lamas. Phase diagram study of a two-dimensional frustrated antiferromagnet via unsupervised machine learning. *Physical Review B*, 103(13):134422, 2021.

[2] D. R. Assis Elias. *Exploration of phase behavior of classical spin systems using machine learning: Exploração de comportamentos de fase em modelos de spin clássicos utilizando aprendizado de máquina*. PhD thesis, [sn], 2023.

[3] DR Assis Elias, E Granato, and M de Koning. Global exploration of phase behavior in frustrated ising models using unsupervised learning techniques. *Physica A: Statistical Mechanics and its Applications*, 589:126653, 2022.

[4] B Block, P Virnau, and T Preis. Multi-gpu accelerated multi-spin monte carlo simulations of the 2d ising model. *Computer Physics Communications*, 181(9):1549–1556, 2010.

[5] SG Brush. History of the lenz-ising model. *Reviews of modern physics*, 39(4):883, 1967.

[6] J Carrasquilla and RG Melko. Machine learning phases of matter. *Nature Physics*, 13(5): 431–434, 2017.

[7] C Casert, T Vieijra, J Nys, and J Ryckebusch. Interpretable machine learning for inferring the phase boundaries in a nonequilibrium system. *Physical Review E*, 99(2):023304, 2019.

[8] John Hammersley. *Monte carlo methods*. Springer Science & Business Media, 2013.

[9] W Hu, RP Singh, and RT Scalettar. Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination. *Physical Review E*, 95(6): 062122, 2017.

[10] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of monte carlo methods*. John Wiley & Sons, 2013.

[11] DA McQuarrie Statistical Mechanics. University science books. *Sausalito, CA*, 194, 2000.

[12] Francisco Ortega-Zamorano, Marcelo A Montemurro, Sergio Alejandro Cannas, José M Jerez, and Leonardo Franco. Fpga hardware acceleration of monte carlo simulations for the ising model. *IEEE Transactions on Parallel and Distributed Systems*, 27(9):2618–2627, 2015.

[13] T Preis, P Virnau, W Paul, and JJ Schneider. Gpu accelerated monte carlo simulation of the 2d and 3d ising model. *Journal of Computational Physics*, 228(12):4468–4477, 2009.

[14] W Selke. Finite-size behaviour of the two-dimensional annni model. *Zeitschrift für Physik B Condensed Matter*, 43(4):335–344, 1981.