IMPLEMENTAÇÃO DE CIRCUITOS NEUROMÓFICOS DIGITAIS UTILIZANDO A ARQUITETURA EXTREME LEARNING MACHINE

Lucas Aparecido Silvino Toledo[1] (EG), Gabriel Antonio Fanelli de Souza (PQ)[1]
[1]*Universidade Federal de Itajubá - UNIFEI*

Palavras-chave: Asic. Elm. Fpga. Nios II.

Introdução

O estudo de redes neurais tem ganhado destaque com o avanço tecnológico e a disponibilidade de grandes volumes de dados, permitindo que essas redes aprendam padrões complexos. As Redes Neurais Artificiais buscam replicar a capacidade de aprendizagem e resolução de problemas do cérebro humano, sendo aplicáveis em diversas áreas. Entre os diferentes tipos de redes, a Extreme Learning Machine (ELM), introduzida por Huang em 2004, o modelo se destaca principalmente por sua estrutura diferenciada das demais, eliminando a necessidade de iterações complexas para o ajuste dos pesos da rede neural. Além disso, sua velocidade de treinamento é consideravelmente superior, devido a sua única camada oculta como é ilustrado na Figura 1, os cálculos dos pesos da rede acontecem através de métodos como mínimos quadrados ou pseudoinversas. O objetivo deste trabalho é utilizar o soft-core Nios II para realizar a implementação do algoritmo de pseudoinversa hardware através de uma FPGA, isso oferece vantagens como reconfigurabilidade e alta precisão nos cálculos, permitindo ajustes arquiteturais sem a necessidade de novo hardware.

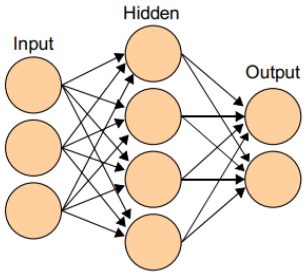

Figura 1: Topologia Extreme Learning Machine

Metodologia

O algoritmo a ser implementado foi proposto por Villora em 2016, ele descreve uma sequência de operações matemáticas para o treinamento de uma rede neural do tipo ELM, é possível verificar os passos necessários para os cálculos na Tabela 1.

Tabela 1: Passos para realizar os cálculos

PASSOS	ELM.V1
1º	$G = F(W, x)$
2º	$\mathbf{Z} = G^t \cdot \mathbf{G}$
3º	QRD(Z)
4º	R^{-1} by TMI
5º	$Z^{-1} = R^{-1} \cdot Q$
6º	$G^{\dagger} = Z^{-1} \cdot Gt$
7º	$h = G\dagger \cdot o$

Foram estudadas 3 versões distintas, todas compartilham a mesma base de arquitetura e objetivo, elas se diferem principalmente na forma como balanceiam desempenho, consumo de recursos e complexidade computacional. A versão ELM.V1 se distingue por ter menos paralelismo nos cálculos, resultando em uma implementação mais simples em relação as demais. A seguir é descrito a sequência dos passos de maneira simples:

1. Multiplicação Inicial: Multiplica-se a matriz de entrada de dados (W) pelos pesos da camada oculta (x), resultando na matriz G, que contém as ativações da camada oculta.
2. Produto com a Transposta: A matriz G é transposta (troca de linhas por colunas), e o resultado é multiplicado por G novamente, criando a matriz **Z.**
3. Decomposição QR: A matriz **Z é** decomposta em duas matrizes: uma ortogonal Q e uma triangular superior R, usando o método de rotações de Givens.
4. Inversão da Matriz R: A matriz triangular R é invertida, processo necessário para calcular a pseudoinversa.
5. Cálculo da Inversa de Z: Neste passo, a inversa da matriz Z é calculada multiplicando a inversa de R
pela matriz ortogonal Q.
6. Cálculo da Pseudoinversa: A pseudoinversa da matriz G é obtida multiplicando-se a inversa de R pela transposta de G.

7. Cálculo dos Pesos de Saída: Finalmente, os pesos da camada de saída são calculados multiplicando a pseudoinversa de G pelos valores de saída desejados.

A implementação do algoritmo ELM em hardware foi realizada utilizando um soft-core Nios II em FPGA. O circuito foi construído utilizando a ferramenta Platform Designer, integrada ao Quartus, que facilita o desenvolvimento de circuitos com blocos de IP da Altera. A FPGA escolhida foi a DE2i-150, da família Cyclone IV, com ênfase na capacidade de memória, é possível observar o circuito projetado na Figura 2.

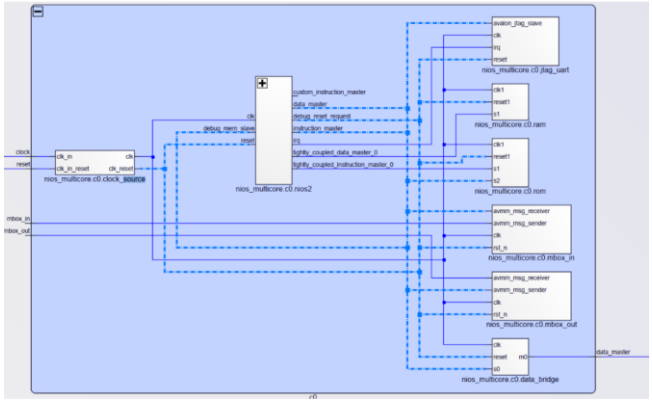

Figura 2: Construção do Nios II

O desenvolvimento incluiu a adaptação do projeto de Semenov, originalmente projetado para apenas multiplicar matrizes, para o uso com a arquitetura ELM. O código foi implementado em linguagem C e segmentado em funções responsáveis por operações como a multiplicação e transposição de matrizes, além da decomposição QR com rotações de Givens. Essas operações são cruciais para a inversão de matrizes durante o treinamento da rede.

Após a síntese do circuito no Quartus e a implementação do algoritmo. A descrição realiza uma série de operações matemáticas envolvendo álgebra linear baseados no método computacional. A função principal implementa a rotação de Givens afim de obter a decomposição QR, um método utilizado para transformar uma matriz *R* em uma matriz triangular superior, enquanto constrói simultaneamente a matriz ortogonal Q. Em paralelo, há funções auxiliares para calcular o determinante de uma matriz, os cofatores, e a transposição, que são etapas necessárias para encontrar a inversa da matriz. Além disso, há uma função dedicada à multiplicação de matrizes e à transposição de outra matriz, essencial para manipulação de dados em forma matricial.

Com o circuito sintetizado na FPGA e o algoritmo implementado em linguagem C, é necessário estabelecer a comunicação entre a FPGA e o computador por meio da interface UART. Para isso, utiliza-se o *Nios Command Shell*, uma interface de linha de comando essencial para interagir com o processador soft-core Nios II, desenvolvido pela Intel. Essa ferramenta, parte integrante do ambiente de desenvolvimento Quartus, permite aos desenvolvedores realizar uma série de operações, como depuração, controle do processador, diretamente a partir do terminal.

No entanto, o terminal apresenta uma limitação importante: ele exibe apenas valores em formato hexadecimal. Para contornar essa restrição e garantir a compatibilidade com o sistema, foi implementada na uma função que converte os valores de ponto flutuante para a representação hexadecimal, permitindo que os dados sejam exibidos corretamente no terminal.

A função converte um número de ponto flutuante em sua representação hexadecimal conforme o padrão IEEE 754, ela utiliza um ponteiro do tipo *unsigned int* para acessar a representação binária do número. Em seguida, imprime o valor hexadecimal correspondente, formatado como um inteiro de 8 dígitos.

Resultados e discussão

As operações necessárias para o treinamento da rede neural ELM foram implementadas com sucesso na FPGA DE2i-150 ilustrada na Figura 3. O código possui uma execução eficiente de multiplicação de matrizes, decomposição QR com transposição e inversão de matrizes, e o uso de comunicação UART entre a FPGA e o computador. Para validar o funcionamento correto do algoritmo, foi utilizado a interface *Nios Command Shell*, onde foi possível realizar o teste de cada etapa das operações e monitorar os resultados em tempo real. Através dessa interface, os valores processados foram exibidos em formato hexadecimal com padrão IEE 754 e verificados, garantindo que as funções de ponto flutuante e conversão para hexadecimal funcionassem conforme esperado.

Figura 3: FPGA DE2I-150

Na Figura 4, são apresentadas todas as operações realizadas em uma matriz de dimensões reduzidas, com o intuito de facilitar a análise detalhada dos cálculos relacionados à topologia de rede estudada. Essa abordagem permite uma compreensão mais clara dos processos envolvidos, destacando a eficiência e a eficácia das operações implementadas no ambiente do soft-core Nios II. Os resultados obtidos foram verificados por meio da interface no terminal,

```
----------------------------
Matriz W:
|        0||3F800000||40000000|
|40400000||40800000||40A00000|
|40C00000||40E00000||41000000|
----------------------------
Matriz x
|3F800000||       0||       0|
|       0||3F800000||       0|
|       0||       0||3F800000|
----------------------------
W*x
|        0||3F800000||40000000|
|40400000||40800000||40A00000|
|40C00000||40E00000||41000000|
----------------------------
Matriz Transposta (W*x)^t:
|        0||40400000||40C00000|
|3F800000||40800000||40E00000|
|40000000||40A00000||41000000|
----------------------------
Z = G^t * G
|42340000||42580000||427C0000|
|42580000||42840000||429C0000|
|427C0000||429C0000||42BA0000|
----------------------------
QRD
Matrix Q:
|3EF41620||BF474BB5||BED105EC|
|3F1273AD||BD9F6FC4||3F5105EC|
|3F2ADC49||3F1F6FC4||BED105EC|
Matrix R:
|42BCC91D||42E71E8D||4308B9FE|
|       0||3FB35DBC||40335DBC|
|       0||       0||282C0000|
----------------------------
R1^-1
Matriz inversa
|3C2D9293||BF5FA773||56BE82FA|
|80000000||3F36B011||D73E82FA|
|       0||80000000||56BE82FA|
----------------------------
Matrix Q transposta:
|3EF41620||3F1273AD||3F2ADC49|
|BF474BB5||BD9F6FC4||3F1F6FC4|
|BED105EC||3F5105EC||BED105EC|
```

Figura 4: Implementação as operações da ELM.v1

Conclusões

Com base nos resultados obtidos através do Nios Command Shell, pode-se afirmar que a precisão nos testes realizados com diferentes matrizes pelo soft-core Nios II é satisfatória. Os cálculos são efetuados utilizando valores declarados como *double* na codificação em C, o que assegura uma precisão de aproximadamente 15 a 17 dígitos significativos, conforme a especificação IEEE 754 em formato hexadecimal. Essa abordagem permite a representação de uma ampla gama de valores, garantindo a integridade dos resultados e a confiabilidade das operações executadas.

Portanto, a implementação dos cálculos dos pesos de uma rede com topologia ELM em um soft-core em uma FPGA se apresenta como uma excelente alternativa para o cálculo dos pesos de saída da rede neuromórfica. Fatores como a reconfiguração da FPGA e sua alta precisão são aspectos que fazem ela ter vantagem em relação aos circuitos de aplicação específica (ASIC). Isso se deve ao fato de que o tempo de desenvolvimento para ASICs é geralmente superior ao da implementação em FPGA.

Agradecimentos

Gostaria de expressar minha profunda gratidão a Unifei pelo suporte financeiro essencial concedido durante todo o processo de pesquisa, o auxílio financeiro foi crucial para a execução e continuidade dos trabalhos. Além disso, gostaria de manifestar minha imensa gratidão aos professores associados ao grupo de microeletrônica.

Referências

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: Theory and
applications. Neurocomputing, v. 70, n. 1–3, 2006.

FRANCES-VILLORA, J. V. et al. Hardware implementation of real-time Extreme Learnin
Machine in FPGA: Analysis of precision, resource occupation and performance. Computer
& electrical engineering: an international journal, v. 51, 2016.

IEEE-754 floating point converter. Disponível em: <https://www.hschmidt.net/FloatConverter/IEEE754.html>. Acesso em: 21 set. 2024.

SEMENOV, I. Multicore-nios. Disponível em:
< https://github.com/Goshik92/multicore-nios>. Acesso em: 21 set. 2024.