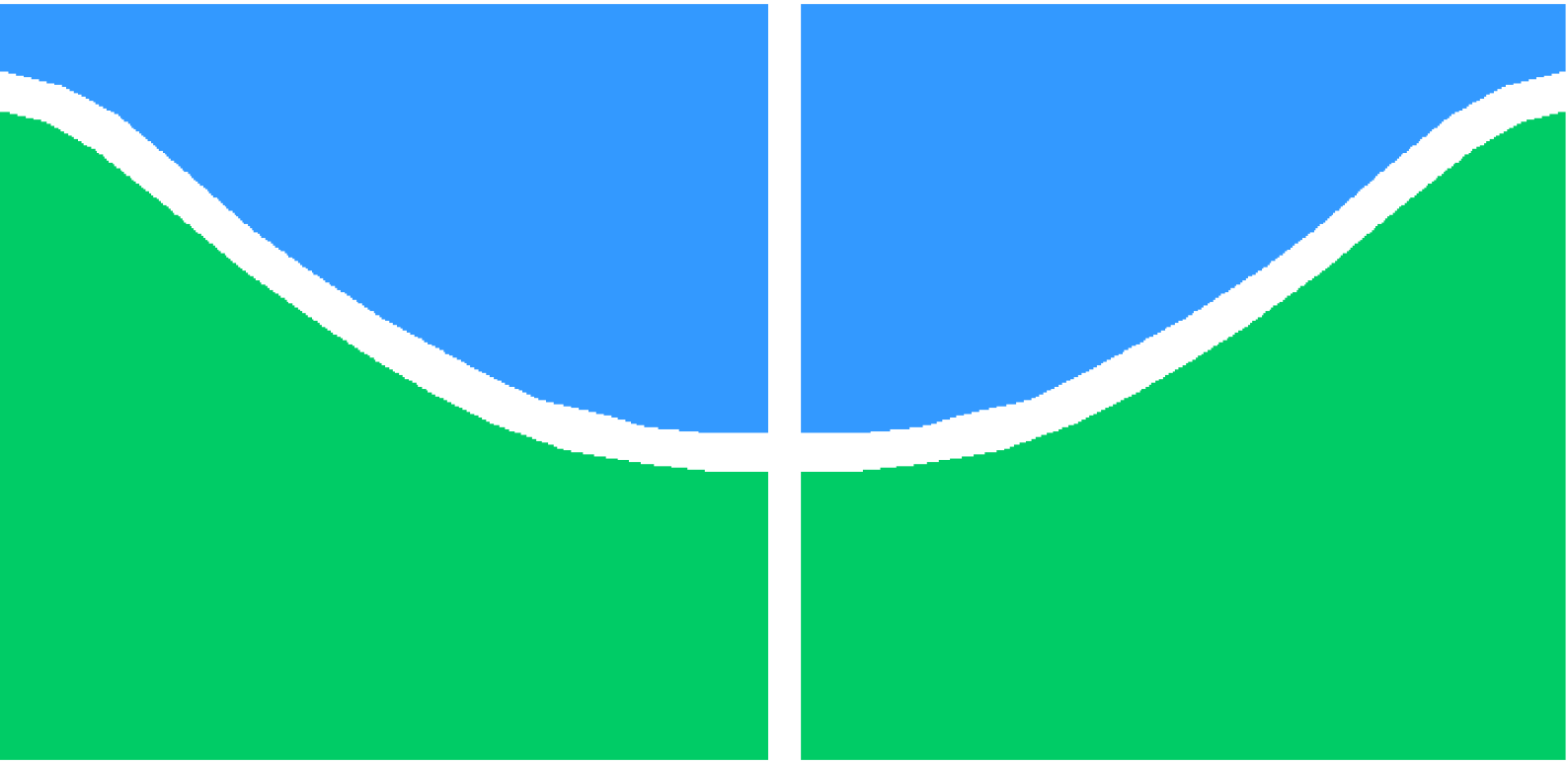


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia Eletrônica

Arquitetura de hardware dedicada de uma rede neural perceptron para reconhecimento de terreno aplicado a robótica móvel

Autor: Rafael Tolentino Rabelo

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Brasília, DF

2014

Rafael Tolentino Rabelo

Arquitetura de hardware dedicada de uma rede neural perceptron para reconhecimento de terreno aplicado a robótica móvel

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Brasília, DF

2014

Rafael Tolentino Rabelo

Arquitetura de hardware dedicada de uma rede neural perceptron para reconhecimento de terreno aplicado a robótica móvel/ Rafael Tolentino Rabelo. – Brasília, DF, 2014-

39 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2014.

1. Palavra-chave01. 2. Palavra-chave02. I. Prof. Dr. Daniel Mauricio Muñoz Arboleda. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Arquitetura de hardware dedicada de uma rede neural perceptron para reconhecimento de terreno aplicado a robótica móvel

CDU 02:141:005.6

Rafael Tolentino Rabelo

Arquitetura de hardware dedicada de uma rede neural perceptron para reconhecimento de terreno aplicado a robótica móvel

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Trabalho aprovado. Brasília, DF, 24 de junho de 2014:

Prof. Dr. Daniel Mauricio Muñoz Arboleda
Orientador

Prof. Dr. Marcelino Monteiro de Andrade
Convidado 1

Prof. Dr. Gilmar Silva Beserra
Convidado 2

Brasília, DF
2014

Resumo

Este trabalho visa à implementação em FPGAs (*Field Programmable Gate Arrays*) de uma rede neural perceptron multicamadas para a classificação de terreno. Para tal, utilizou-se um acelerômetro de 3 eixos para medir as variações de aceleração que um robô sofre em quatro tipos de terreno: arenoso, asfalto, grama e terra. Uma rede neural do tipo perceptron multicamada foi treinada para poder realizar o processo de classificação. Após o treinamento da rede, obteve-se os pesos e os bias da rede para realizar a descrição em *hardware* e implementação do modelo matemático da rede em FPGAs. Resultados experimentais demonstraram que o desempenho em termos do erro de classificação é melhorado quando os dados de entrada são uma medida estatística da aceleração. Foi usada a média de 32 amostras para compor cada conjunto de entrada da rede. Adicionalmente, os resultados demonstraram que a rede treinada com 8 neurônios na camada escondida alcança o melhor fator custo-benefício. Comparações numéricas entre os resultados obtidos em *software* e *hardware* foram realizados para validação da arquitetura, demonstrando a corretude da implementação. Finalmente, os circuitos desenvolvidos foram caracterizados em termos do consumo de recursos, frequência de operação e consumo de potência. O tempo de execução em diversas plataformas embarcadas foi estimado, demonstrando que a arquitetura proposta alcança fatores de aceleração de três ordens de magnitude se comparado com processadores de software embarcados MicroBlaze e Atmel, e quatro vezes se comparado com um processador Intel Core i7.

Palavras-chaves: Redes Neurais. Perceptron Multicamas. FPGA. Circuitos Reconfiguráveis. Robótica Móvel. Sistemas Embacados. Classificação de Terreno.

Abstract

This work proposes an FPGA (Field Programmable Gate Arrays) implementation of a multilayer perceptron neural network for terrain classification. A 3-axis accelerometer was used for acquiring the acceleration variation that a robot suffers when moving on four different terrains: sand, asfalt, grass and soil. A multilayer perceptron neural network was trained in order to perform the classification process. Afterwards, the trained weight and bias were used to implement in FPGAs a hardware mathematical model of the proposed network. Experimental results have demonstrated that the network performance in terms of classification error was improved when using statistical values of the acceleration as input data. Thus, the mean value of 32 samples was computed in order to compose the input data set of the proposed neural netwrok. Numerical comparisons between hardware and software results, using the Matlab as statistical estimator, were used for validating the hardware implementation. Finally, the implemented circuits were characterizaed in terms of the consumption of hardware resources, operational frequency and power consumption. The execution time using three software-based embedded platforms were estimated. The proposed architecture achieves speed-up factors of three order of magnitude in comparison with the MicroBlaze and Atmel software processors, as well as, four times in comparision with an Intel Core i7 solution.

Key-words: Neural Network. Perceptron Multilayer. FPGA. Reconfigurable circuits. Embedded Systems. Terrain Classification.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

FPGAs	Field Programmable Gate Arrays
ASICs	Circuitos Integrados de Aplicação Especifica
VHDL	Very high speed integrated circuits Hardware Description Language
RNAs	Redes Neurais Artificiais
PLDs	Programmable Logic Devices
CPLDs	Complex Programmable Logic Devices
HDL	Hardware Description Language
CLBs	Configurable Logic Blocks
LUTs	Look Up Tables
RAM	Random Access Memory
DSPs	Digital Signal Processing
IOB	Input/Output Bound
PSM	Programable Switch Matrix
FFs	Flip-Flops
ROM	Read Only Memorie
WOM	Write Only Memorie
MSE	Mean Square Error

Sumário

1 Introdução

Um robô móvel é um dispositivo com capacidade de locomoção capaz de interagir e atuar no ambiente em que está inserido. Atualmente a robótica móvel tem ganhado muito espaço, tanto na área de pesquisas quanto nas aplicações no dia a dia. Grande parte desse ganho de espaço é devido aos avanços tecnológicos na área de automação (COSTA; GOMES; BIANCHI, 2003). Entre as diversas aplicações que se podem encontrar na literatura, pode-se destacar a sua aplicação em diversos contextos, como no uso doméstico, na utilização de aspiradores de pó e cortadores de gramas robóticos e na indústria, onde são utilizados para tarefas de transportes e veículos de cargas autônomos, assim como também para tarefas de monitoramento, exploração e mapeamento em ambientes de difícil acesso a um ser humano (WOLF et al., 2009).

1.1 Descrição do problema

Em robótica móvel, às vezes é necessário fazer com que o robô se adapte ao ambiente, como por exemplo, em missões de exploração fora do planeta, onde as informações sobre o ambiente são incertas, havendo diversos fatores que podem dificultar o desempenho do robô. Um desses fatores é o tipo de terreno em que o robô esta se movendo, que podem reduzir a eficiência e a velocidade de progresso de uma tarefa. Neste contexto, é necessário um ajuste no controle de movimentação do robô de acordo com o tipo de terreno em que o robô se encontra, evitando que o robô fique inoperante. Segundo (DUPONT et al., 2008), um sistema de controle dependente do terreno está atualmente disponível para o carro Land Rover LR3, onde é possível ajustar o sistema de controle do veículo, assim como o controle de tração, resposta da aceleração, e relação de marchas, para melhorar a performance do veiculo em um conjunto de terreno.

Robôs móveis podem ser classificados como sistemas embarcados com restrições de baixo consumo de energia, portabilidade, assim como limitações na capacidade de processamento computacional. Portanto, um sistema de classificação de terreno de alto desempenho em termos de tempo de execução é necessário no intuito de não comprometer a tarefa a ser realizada.

Levando em consideração essas questões acima citadas, o presente trabalho visa a implementação de um hardware dedicado de uma rede neural perceptron para reconhecimento de terrenos aplicado a robótica móvel.

1.2 Justificativa

Devido à falta de recursos computacionais que alguns desses robôs apresentam, se faz necessário a busca de soluções que atendam às restrições de um sistema embarcado para solucionar o problema de classificação de terreno. Entretanto, na literatura científica a maioria dos trabalhos usam sistemas microcontrolados para resolver o problema de classificação de terreno. Essa abordagem aumenta a carga computacional que o robô deve realizar, reduzindo o desempenho de outras tarefas que devem ser realizadas.

Uma revisão no estado da arte demonstrou que poucos trabalhos apresentam soluções para acelerar o processo de classificação de terreno usando FPGAs (*Field Programmable Gate Arrays*), atendendo ao mesmo tempo requisitos de alto desempenho e portabilidade. Soluções baseadas em FPGAs são úteis, pois geralmente permitem implementar arquiteturas paralelas que trabalham com baixa frequência de relógio, o que a torna uma solução viável para projeto de sistemas embarcados de alto desempenho, onde o consumo de energia é um fator crítico.

Uma vantagem adicional na utilização de FPGAs é a possibilidade de prototipagem de circuitos integrados digitais, possibilitando a fabricação de Circuitos Integrados de Aplicação Especifica (ASICs), podendo-se explorar processamento de alto desempenho e baixo consumo de energia (MEYER-BAESE, 2004).

1.3 Objetivos

O objetivo geral deste trabalho consiste em desenvolver, implementar e validar uma arquitetura de *hardware* dedicada utilizando FPGAs para a solução de classificação de terrenos em robótica móvel.

Os objetivos específicos deste trabalho estão relacionados a seguir:

- Revisão do estado da arte sobre modelos de classificação de terreno;
- Coleta de dados a partir de medições de aceleração de um sistema robótico em diversos tipos de terreno;
- Implementação e treinamento no Matlab de uma rede neural para classificação de terreno;
- Descrição de *hardware* de uma arquitetura paralela da rede neural obtida para classificação de terreno.
- Caracterização do circuito obtido e validação da arquitetura proposta.

1.4 Aspectos Metodológicos

No desenvolvimento deste trabalho foram utilizadas duas metodologias bem conhecidas pela comunidade acadêmica que implementa de circuitos integrados digitais. No inicio do trabalho foi utilizado uma metodologia *top-down*, que permite analisar o problema desde um alto nível de abstração, assim como para analisar a arquitetura de *hardware* proposta em FPGA. Durante o processo de desenvolvimento da arquitetura de *hardware* foi utilizada uma metodologia *bottom-up*, na qual módulos internos da arquitetura foram implementados e validados e posteriormente foram integrados, obtendo-se assim a arquitetura final.

As fases de desenvolvimento deste trabalho são as seguintes:

Na fase 1, utilizou-se um arduino e um acelerômetro de três eixos para coletar dados em quatro tipos de terreno: (a) arenoso, (b) asfalto, (c) grama, e (d) terra.

Na fase 2, foi feita uma análise e tratamento dos dados coletados. Foram traçados alguns gráficos no Matlab que levaram a escolha de medidas estatísticas para compor os valores de entrada da rede. Adicionalmente, foi aplicado um valor de offset aos dados coletados, fazendo com que os valores de entrada da rede ficassem entre -1.75 volts e +1.75 volts.

Na fase 3, foi realizada a implementação e treinamento de uma rede neural artificial tipo perceptron multicamada no Matlab. Em seguida, foi feita uma análise de convergência da rede para diferentes topologias e análise de desempenho a partir de gráficos de confusão. Foi realizada também uma análise de erro de classificação. Finalmente, com base nessas analises foi escolhida a melhor topologia da rede neural.

Na fase 4, foi feita a descrição em linguagem de descrição de *hardware* VHDL (*Very high speed integrated circuits Hardware Description Language*) dos componentes que formam a topologia escolhida da rede neural.

Na fase 5, foi feita a validação dos resultados, realizando a simulação do circuito e a caracterização do mesmo em termos do consumo de recursos de *hardware*, tempo de execução e consumo de potência.

Na fase 6, fez-se a estimação do fator de aceleração do tempo de execução a partir de implementações em software usando o Matlab e uma implementação em linguagem C, assim como usando duas soluções de software convencionais para sistemas embarcados (MicroBlaze e Arduino).

Na fase 7, foram feitas as discussões finais e a documentação do projeto.

1.5 Organização do trabalho

O presente documento está organizado da maneira descrita a seguir. O capítulo 2 descreve a fundamentação teórica sobre redes neurais artificiais e sobre hardware reconfigurável, assim como apresenta uma revisão do estado da arte sobre sistemas de classificação de terrenos aplicados em robótica móvel. O capítulo 3 apresenta as implementações de *hardware* e *software* realizadas no trabalho. O capítulo 4 reporta os resultados de síntese, simulação e execução. Finalmente, o capítulo 5 apresenta as conclusões finais do trabalho e as propostas para trabalhos futuros.

2 Fundamentação teórica

Neste capítulo, será apresentada à fundamentação teórica do trabalho. O capítulo está organizado da seguinte maneira: a seção 2.1 apresenta conceitos sobre redes neurais e suas características; e a seção 2.2 descreve aspectos relevantes sobre sobre hardware configurável.

2.1 Redes Neurais Artificiais

As redes neurais artificiais (RNAs) são modelos matemáticos bioinspirados capazes de modelar o funcionamento do cérebro. Segundo (HAYKIN, 1999), uma RNA é um processador paralelamente distribuído constituído de unidades mais simples, os neurônios, e tem a capacidade de armazenar conhecimento e disponibiliza-lo para uso. Elas se assemelham em dois aspectos com o cérebro humano. O primeiro aspecto está relacionando à aquisição de conhecimento, onde o conhecimento da rede é adquirido a partir de um processo de aprendizagem. O segundo está relacionado à fixação do conhecimento, onde as forças de conexões entre neurônios, conhecidas como pesos sinápticos, realizam essa fixação.

Devido ao processamento paralelo e à capacidade de armazenar conhecimento, as RNAs são consideradas aproximadores universais de funções não lineares, sendo capazes de resolver problemas complexos. Entretanto, as redes neurais comumente se integram a outros subsistemas. Em um problema complexo, a melhor saída seria dividir o problema em problemas menores, e atribuir à rede determinadas tarefas, de acordo com sua capacidade (HAYKIN, 1999).

2.1.1 Neurônio Artificial

Os neurônios artificiais são as unidades de processamento da rede neural, e são fundamentais para que a rede funcione. Na Figura 1, é mostrado um modelo de um neurônio artificial.

O neurônio artificial possui três elementos básicos: (a) os pesos sinápticos, (b) o somador e (c) a função de ativação (HAYKIN, 1999). Os pesos sinápticos (w_j) multiplicam os sinais de entrada (x_j). O somador faz a soma de todos os sinais de entrada ponderados pelos pesos sináptico, além de somar um *bias* (b) que tem a função de ativar o neurônio mesmo que as entradas sejam nulas. A função de ativação restringe o sinal de saída do neurônio.

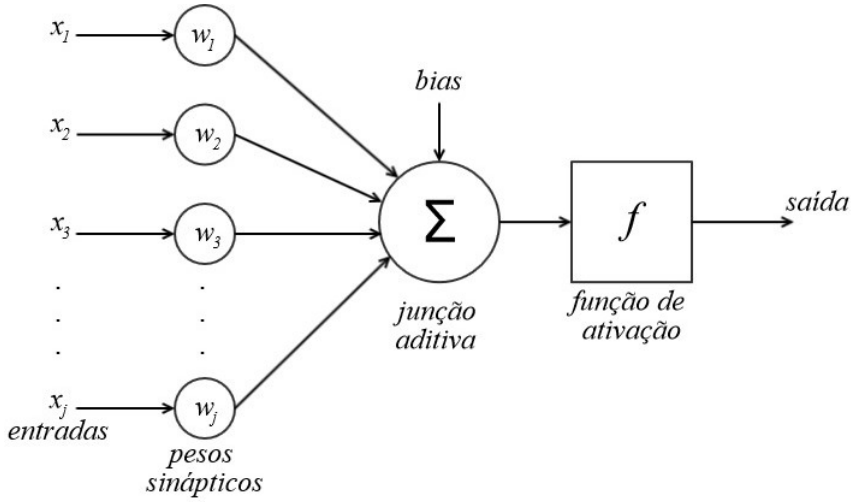


Figura 1 – Neurônio artificial

Matematicamente, o neurônio pode ser descrito como:

$$y = F(\sum_{j=1}^{m} w_j x_j + b) \tag{2.1}$$

sendo $x_1, x_2, \ldots, x_j$ os sinais de entrada; $w_1, w_2, \ldots, w_j$ os pesos sinápticos; b o *bias*; F a função de ativação; e y o sinal de saída do neurônio.

As três funções de ativações básicas podem ser visualizadas na Figura 2. Elas são a função limiar, a semi-linear e a sigmoid. Em algumas aplicações, utiliza-se a função tangente hiperbólica no lugar da função sigmoid.

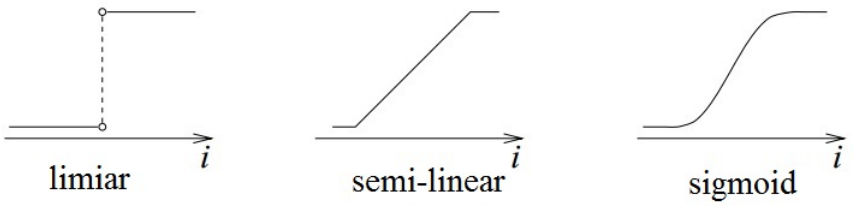


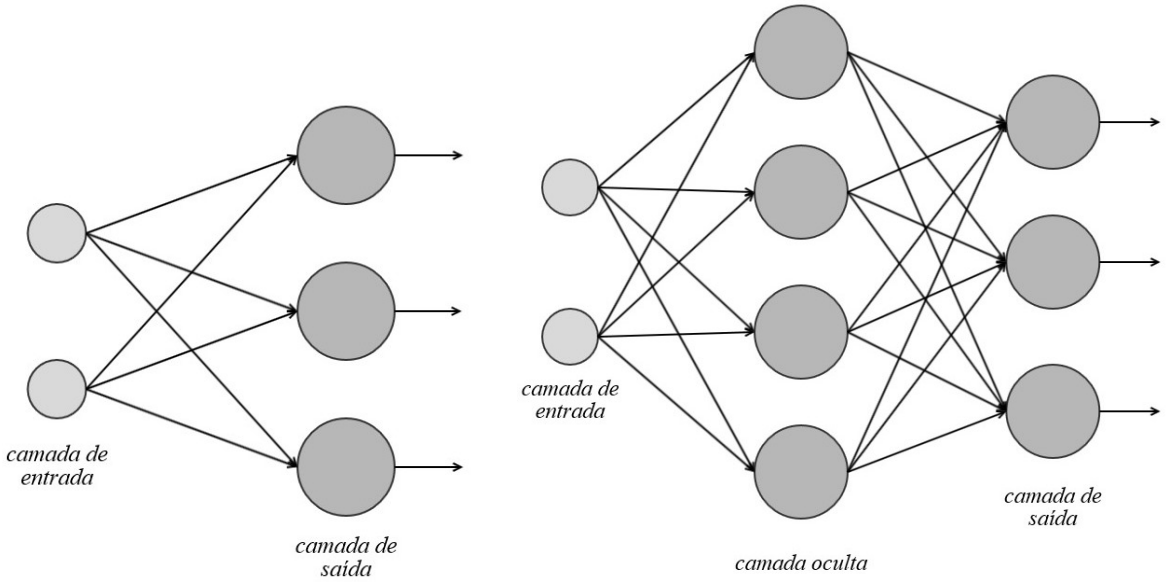
Figura 2 – Funções de ativação (B.KRÖSE; SMAGT, 1996)

2.1.2 Arquitetura da Rede

A arquitetura da rede representa a forma como os neurônios artificiais estão conectados e como os dados passam por eles (B.KRÖSE; SMAGT, 1996). Além disso, a forma como os neurônios estão conectados está diretamente ligada ao algoritmo utilizado para treinar a rede (HAYKIN, 1999). Em geral, podem-se destacar dois tipos de arquitetura.

A primeira é conhecida como *feedforward*, onde os dados fluem somente da entrada para a saída. Este tipo de rede pode ainda ser de camada única ou de múltiplas camadas. Redes de camada única são aquelas em que a rede possui somente a camada de saída

(Fig. 3a). Na arquitetura de múltiplas camadas, além da camada de saída há a presença de uma ou mais camadas ocultas (Fig. 3b) (HAYKIN, 1999).



(a) (b)

Figura 3 – Arquiteturas alimentadas adiante: (a) Arquitetura de camada única; (b) Arquitetura de multplas camadas.

O segundo tipo de rede é chamada de rede *recorrente*, a qual diferencia da *feed-forward* por possuir conexões de realimentação (HAYKIN, 1999).

2.1.3 Aprendizado

Uma rede neural deve ser configurada de tal forma que, aplicando-se um conjunto de entradas, são obtidos valores desejados de saída. Para isso, é necessário ajustar os pesos sinápticos da rede. Uma forma seria escolher os pesos baseados em um conhecimento prévio. Outra forma é treinando a rede, informando padrões de entrada e recalculado os pesos segundo uma regra de aprendizado.

Com base nisso, há duas formas de estabelecer o aprendizado de uma rede neural. Uma utiliza um processo *supervisionado* onde um professor confere o quanto a rede está próxima da solução, adaptando os pesos entre os neurônios, de forma a reduzir o erro da saída. A outra utiliza um processo *não supervisionado*. Neste caso não existe um professor para verificar os resultado, então a rede é treinada para responder a um conjunto de critérios ou padrões.

2.1.4 Rede Perceptron Multicamadas

A rede perceptron é uma rede baseada em um neurônio não-linear e tem por objetivo classificar padrões mediante um conjunto de sinais de entrada. Porém, a rede

perceptron de camada única consegue resolver somente problemas linearmente separáveis (HAYKIN, 1999).

Para resolver problemas que não sejam linearmente separáveis, utiliza-se então a rede perceptron multicamada, que pode-se dizer que é uma generalização da perceptron de camada única. Ela é uma rede feedforward de múltiplas camadas, possuindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída.

O algoritmo de treinamento mais utilizado nesse tipo de rede é um algoritmo treinamento supervisionado chamado *back-propagation*, ou *retropropagação*, que será explicado na próxima sessão.

2.1.5 Back-Propagation

O *back-propagation* é um algoritmo de treinamento supervisionado que funciona de modo iterativo, podendo ser divido em dois passos. No primeiro passo, *passo para frente*, os sinais de entrada atravessam a rede, onde os pesos sinápticos são iniciados com valores aleatórios. O conjunto de entrada é aplicado à rede, e se propaga através das camadas até a saída da rede. Na saída da rede, então, são comparados os valores de saída com os valores esperados, verificando-se um erro. No segundo passo,*passo para trás*, o erro então é retro propagado da saída para as demais camadas. Os pesos então são reajustados, de acordo com uma regra de correção de erro, para que se obtenha o valor desejado na saída.

Segundo (JR.; CRUZ, 1998)pode-se dizer que o erro gerado é o somatório dos erros quadráticos, definido pela expressão (2.2):

$$E = \sum_{j,p} E_i^p = \frac{1}{2} \sum_{i,p} (d_i^p - O_i^p)^2 \tag{2.2}$$

onde E_i^p representa o erro no i_ésimo elemento neural, para o p_ésimo padrão de entrada, d_i^p é a saída esperada no i_ésimo elemento neural, pra o p_ésimo padrão de entrada e O_i^p é a saída produzida pelo neurônio:

$$O_i^p = F(y_i^p) = F(\sum_j (w_{ij} x_j^p - b_i)) \tag{2.3}$$

onde x_i^p é a j_ésima entrada.

Na técnica do gradiente decrescente, os valores dos pesos são modificados proporcionalmetne ao oposto da derivada do erro, de acordo com (2.4), sendo η a taxa de aprendizado:

$$\Delta w_{ij} = -\eta \frac{\partial E_i^p}{\partial w_{ij}} \tag{2.4}$$

Adotando $d_i^{p,k}$ como saída esperada na i_ésima unidade da k_ésmia camada, quando a p_ésima entrada é inserida, a função erro pode ser escrita como

$$E^k = \frac{1}{2}\sum_{j,p}(d_i^{p,k} - O_i^{p,k})^2 \tag{2.5}$$

.

Então, a correção dos pesos na camada de saída K é dada, aplicando-se a regra da cadeia em (2.4):

$$\Delta w_{ij}^K = -\eta\frac{\partial E^K}{\partial w_{ij}^K} = -\eta\sum_p \frac{\partial E^K}{\partial O_i^{p,K}}\frac{\partial O_i^{p,K}}{\partial y_i^{p,K}}\frac{\partial y_i^{p,K}}{\eta w_{ij}^K} \tag{2.6}$$

A correção para os pesos das camadas intermediaria k é feito de forma similar. e é dada por

$$\Delta w_{ij}^k = -\eta\frac{\partial E^k}{\partial w_{ij}^k} \tag{2.7}$$

2.2 Hardware Reconfigurável

Em sistemas digitais, existem três tipos básicos de dispositivos: memórias, microprocessadores e lógica (XILINX, 2014). As memórias são utilizadas para armazenamento de dados, os microprocessadores executam instruções de *software* e os lógicos fornecem funções específicas.

Os dispositivos lógicos podem ser classificados em duas categorias: fixos ou programáveis. Os dispositivos com lógica fixa, como o nome sugere, realizam funções fixas, que não se alteram depois de fabricados. Por outro lado, os dispositivos lógicos programáveis, ou PLDs (*Programmable Logic Devices*), podem ser alterados a qualquer momento para executar qualquer número de funções, oferecendo uma ampla capacidade lógica (XILINX, 2014).

A vantagem da utilização dos PLDs é que os projetistas conseguem desenvolver, simular e testar seus projetos de forma mais rápida, segura e barata e, caso seja necessário alguma alteração no projeto, a alteração é feita de uma forma mais rápida já que os PLDs são baseados na tecnologia de memória regravável. Outro ponto importante é que o PLD utilizado para prototipar o produto será o mesmo utilizado no produto final (XILINX, 2014).

Os dois principais tipos de dispositivos lógicos programáveis são os FPGAs e os CPLDs (*Complex Programmable Logic Devices*).Dos dois, os FPGAs oferecem a maior quantidade de densidade lógica, a maior quantidade de recursos e o mais alto desempenho. Os FPGAs possuem uma grande variedade de aplicações, que variam de processamento e armazenamento de dados, à instrumentação de telecomunicações e de processamento

digital de sinais. Já os CPLDs possuem uma quantidade muito menor de lógica, porém eles oferecem uma características de tempo muito previsíveis, sendo ideal para aplicações de controle críticos (XILINX, 2014).

Para a programação desses dispositivos PLDs é utilizado uma linguagem de descrição de hardware, ou HDL (*HardwareDescription Language)*, como o VHDL e o Verilog. Esta linguagem faz uma descrição de como o *hardware* deve se comportar.

2.2.1 FPGAs

FPGAs são dispositivos semicondutores que são baseados em torno de uma matriz de blocos lógicos configuráveis (CLBs) conectados via interconexões programáveis. Os blocos lógicos configuráveis possuem diversos elementos lógicos, chamados slices, sendo cada slice constituido de um número determinado de LUTs(*Look Up Tables*), registradores e multiplexadores. As LUTs permitem implementar uma funções lógicas simples. Dependendo da arquitetura, as LUTs podem ter 4,5,6 ou 8 entradas. As FPGAs ainda possuem alguns dispositivos de aplicação específica, como memorias RAM (*Random Access Memory*), DSPs (*Digital Signal Processing*), processadores, como o MicroBlaze, entre outros.

É importante ressaltar que todos os elementos internos da FPGA podem ser acessados de forma paralela no intuito de acelerar a execução dos algoritmos implementados. A Figura 4 mostra a arquitetura interna de uma FPGA, mostrando os CLBs, os IOB (*Input/Output Block*) que são a interface de entrada e saída dos CLBs e PSM (*Programable Switch Matrix*) que são as interconexões programáveis.

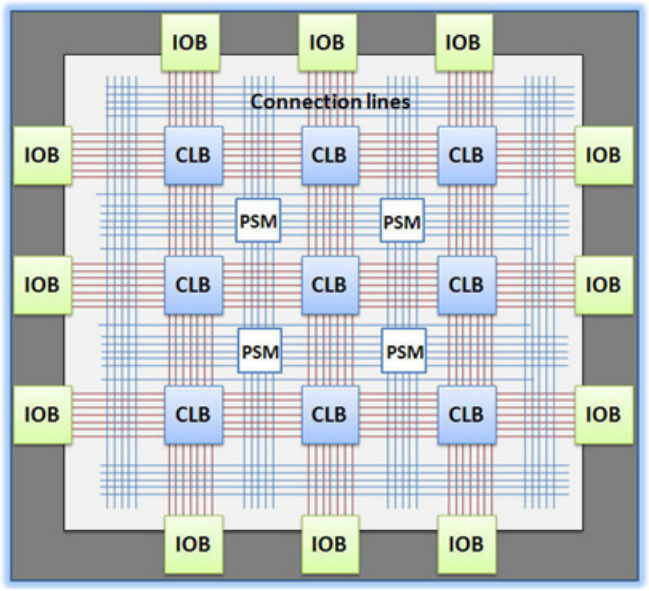


Figura 4 – Estrutura de uma FPGA (DENMARK, 2014).

2.3 Revisão do estado da arte

Para fins de referência, fez-se um levantamento de publicações encontradas em literatura científica sobre sistemas de classificação de terreno.

(DAVIST et al., 1995) analisa três classificadores de incremento de dimensionalidade e descreve sua aplicabilidade a diferentes aspectos do problema de padronização de terreno.

(LARSON; DEMIR; VOYLES, 2005) realiza a classificação de terreno tanto para locomoção eficaz de um robô articulado aplicado em tarefas de exploração e mapeamento. Nesse trabalho foram usadas medidas do impacto de marcha a partir de estimativas de erros do sistema de servovisão resultante da interação entre o robô e o terreno durante a locomoção.

(DUPONT et al., 2005) centra seu trabalho no desenvolvimento e aplicação de um algoritmo de detecção de terreno com base no terreno induzida pela vibração do veículo. As frequências de vibração dominantes são extraídas e utilizadas por uma rede neural probabilística para identificar o terreno.

(OJEDA et al., 2006) introduz novos métodos de classificação e caracterização de terreno com um robô móvel. A classificação do terreno visa associar terrenos com algumas categorias pré-definidas, comumente conhecidas, como cascalho, areia ou asfalto.

(KURBAN; BEşDOK, 2009) apresenta uma comparação de algoritmos de treinamento de redes neurais de função base radial (RBF) para fins de classificação de terreno.

(WALAS et al., 2013) implementou algoritmos de classificação do solo em FPGA. A classificação do tipo de terreno é baseada nos sinais adquiridos com sensor de força/torque montados no robô com patas.

(NGUYEN, 2013) apresenta sete novas contribuições para duas tarefas de percepção: detecção de vegetação e classificação do terreno, para sistemas autônomos de navegação.

Durante as pesquisas não foram encontrados na literatura científica trabalhos prévios que implementem em FPGAs sistemas de classificação de terreno usando redes neurais artificiais perceptron multicamada a partir de medições de aceleração.

3 Implementação

Neste capítulo, serão apresentadas as implementações realizadas neste projeto. O capitulo está organizado da seguinte forma: a seçao 3.1 fala sobre as ferramentas de *software* e *hardware* utilizadas; a Seção 3.2 descreve o processo de aquisição de dados de aceleração; a Seção 3.3 apresenta a implementação no Matlab da rede neural escolhida e as Seções 3.4 a 3.8 apresentam as arquiteturas implementadas.

3.1 Descrição das ferramentas de hardware e software

Para a implementação das arquiteturas de hardware foi utilizado um kit Digilent Nexys 4 (Fig.5),que possui um dispositivo FPGA Artix-7 cuja a tecnologia possui 15850 slices com 4 LUTS de 6 entradas e 8 Flip-Flops (*FFs*) cada um, além de possuir 240 DSPs, um conversor analógico-digital de 12 bits, entre outros. O kit ainda conta com um acelerômetro de três eixos (x,y e z) de 12 bits de resolução, capaz de medir acelerações na faixa de ±2g, ±4g e ±8g.

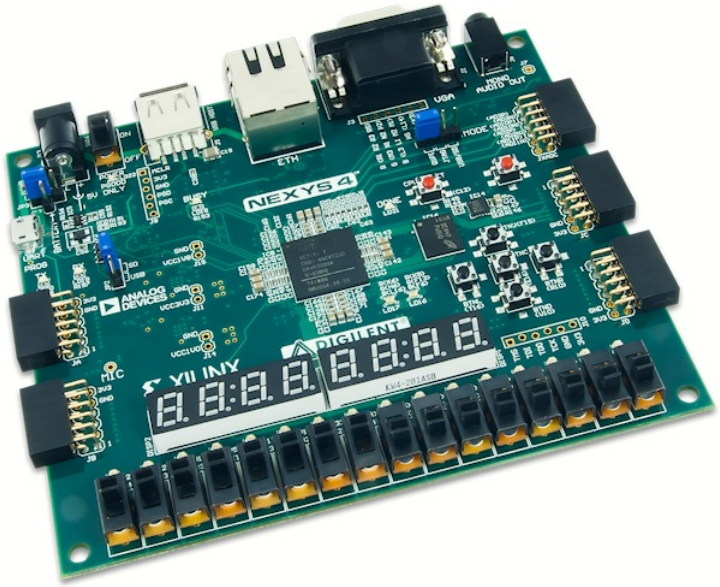


Figura 5 – Kit Nexys 4 (DIGILENT, 2014).

Para utilização de ferramentas de software foi utilizado um notebook com processador Intel core i3 2.2 GHz, com 4 GB de RAM, rodando um sistema operacional Windows 7.

As ferramentas de software utilizadas neste projeto foram o *ISE Design Tools 14.7*, utilizado para escrever, sintetizar os circuitos em VHDL, o *CORE Generator*, utilizado para importar blocos de propriedade intelectual em VHDL, o simulador *ISim*, os pro-

gramas *Xilinx Platform Studio* e *Xilinx Software Development Kit* para implementar e programar o microprocessador MicroBlaze em FPGA. Foram utilizados o Matlab para treinamento da rede neural e a IDE do arduíno para programar o arduíno. Para a documentação foi utilizada uma ferramenta online chamada *WriteLatex* (www.writelatex.com).

3.2 Coleta de dados

Para a coleta de dados foi utilizado um robô simples, sem amortecimento e com pouco controle de velocidade (aceleração máxima ou zero). Foi utilizado também um acelerômetro de eixos (x, y e z) MMA7361 (Fig. 7), capaz de medir acelerações na faixa de $\pm$1.5g e $\pm$6g, ligado a um arduino UNO.

Antes de realização da coleta de dados foi feita a calibração do sensor.A calibração foi feita via *software*, através de uma função da biblioteca do sensor para o arduino, sendo necessário que o eixo z do sensor esteja perpendicular à superfície da terra.

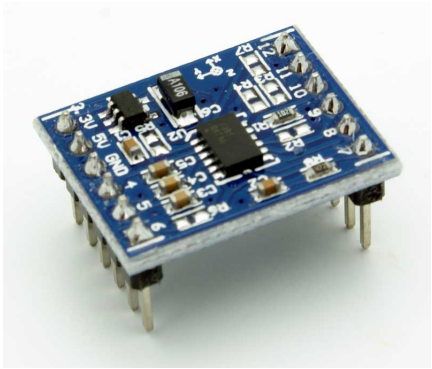
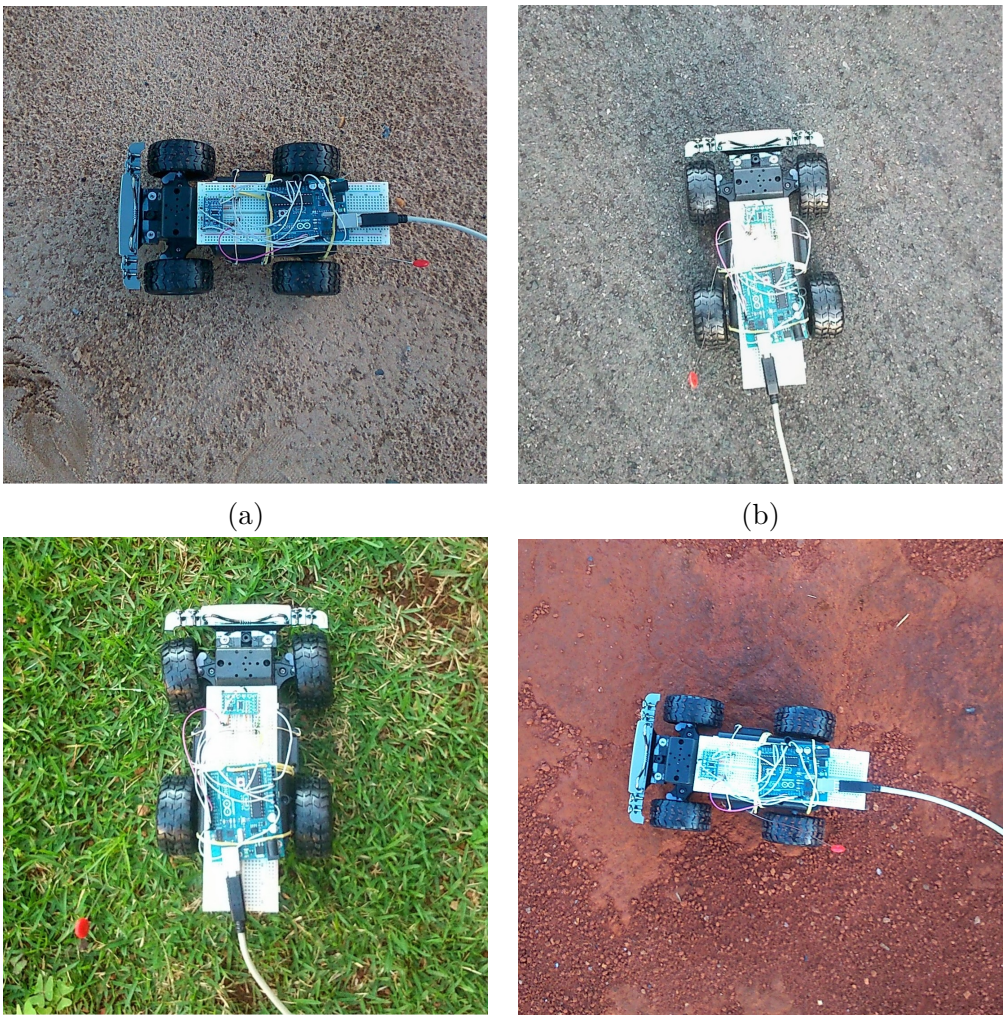


Figura 6 – Acelerômetro MMMA7361 (ELETRONICS, 2014).

O acelerômetro foi configurado para medir acelerações de $\pm$6g, e a taxa de amostragem foi de 100 Hz. A coleta de dados foi feita em quatro tipos diferentes terrenos: arenoso, asfalto, grama e terra como mostrado na Figura 7. A aquisição foi feita durante aproximadamente 5 metros, em linha reta.

Antes de realizar o treinamento da rede, foi feito um tratamento nos dados adiquiridos. Aplicou-se um valor de offset de -1.75, para se obter valores de entrada entre $\pm$1.75, e utilizou-se uma média de 32 amostras para compor os dados de entrada da rede. Na Figura 8 são mostrados os dados de aceleração no eixo x e o respectivo cálculo da média.



(a) (b)

(c) (d)

Figura 7 – Coleta de dados nos terrenos : (a) Arenoso; (b) Asfalto; (c) Grama; (d) Terra.

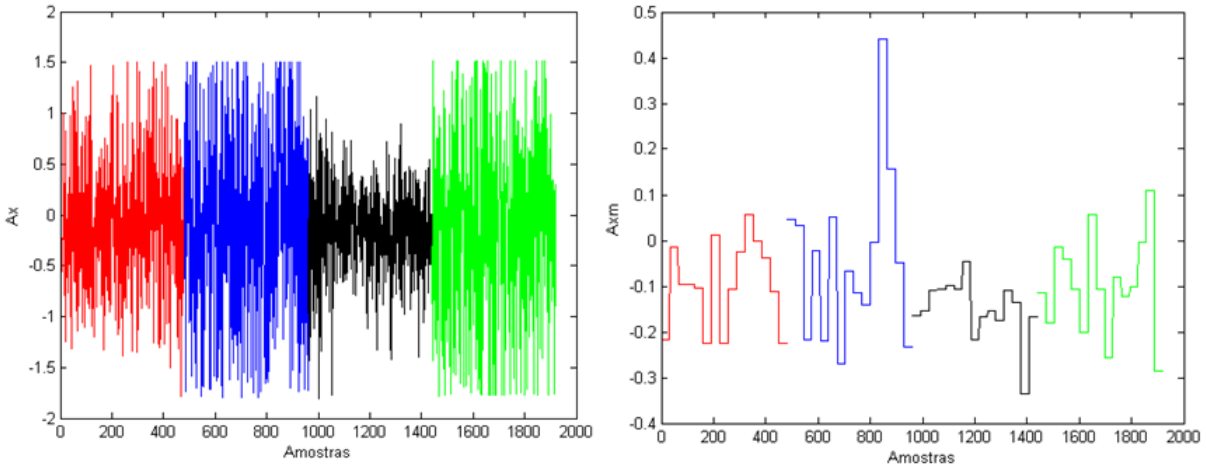


Figura 8 – Gráfico da esquerda representa as amostras e o da direita a média das amostras. As amostras em vermelho representam o terreno arenoso, em azul o asfalto, em preto representam a grama e em verde o terreno tipo terra.

3.3 Implementação em Matlab

O Matlab foi utilizado para o treinamento da rede neural. Para tal utilizou-se o *npr toolbox*, que é uma ferramenta para projeto de redes neurais artificias para reconhecimento e classificação de padrões.

A função de ativação escolhida foi a tangente hiperbólica (ou tangente sigmoide). Esta escolha se deve à sua facilidade de implementação em *hardware*, e pelo fato que foi aplicado um valor de offset nos valores de entrada obtendo valores entre ± 1.75, o que ajuda no treinamento da rede utilizando a tangente hiperbólica que está limitanda entre ± 1. O algoritmo utilizado para atualização dos pesos e dos bias foi o algoritmo *Levenberg-Marquardt backpropagation*, por ser o mais rápido, segundo o manual no Matlab, e por ter apresentado melhores resultados que o algoritmo *Gradiente Decrescente*. A definição da quantidade de neurônios na camada escondida foi definida a partir de gráficos de confusão, para 4, 6, 8 e 20 neurônios na camada escondida.

Além dos gráficos de confusão, foi feita uma análise de acerto que a rede apresentava, fazendo comparações das saídas e verificando se elas apresentava os resultados corretos. A Tabela 1 mostra a taxa de acerto da rede treinada para 4, 6, 8 e 20 neurônios.

	4 neurônios	6 neurônios	8 neurônios	20 neurônios
Arenoso	90,48%	85,71%	90,48%	90,48%
Asfalto	90,48%	100%	100%	100%
Grama	90,48%	100%	100%	100%
Terra	85,71%	85,71%	85,71%	90,48%

Tabela 1 – Porcentagem de acerto da rede para 336 amostras.

Então, a partir das análises feitas, optou-se por uma topologia com 8 neurônios na camada escondida, pois apresenta melhores resultados em termos de custo benefício. A Figura 9 mostram os gráfico de confusão obtidos no final do treinamento para redes com 4, 6, 8 e 20 neurônios na camada escondida. Na Figura 9c é possível verificar que para a saída 1, 32 amostras foram confundidas com a saída 3, e para a saída 4, 64 amostras foram confundidas com a saída 3.

3.4 Arquitetura geral da rede

A arquitetura geral da rede neural proposta é mostrada na Fig. 10 . Esta arquitetura faz uso dos módulos de *média*, *neuronio_h*, *neuronio_o* e *comparador*, que serão descritos nas seções seguintes.

O primeiro módulo é o módulo da média, onde será calculada uma média das amostras capturadas pelo acelerômetro. Em seguida, as médias de *ax*, *ay* e *az* entram nos

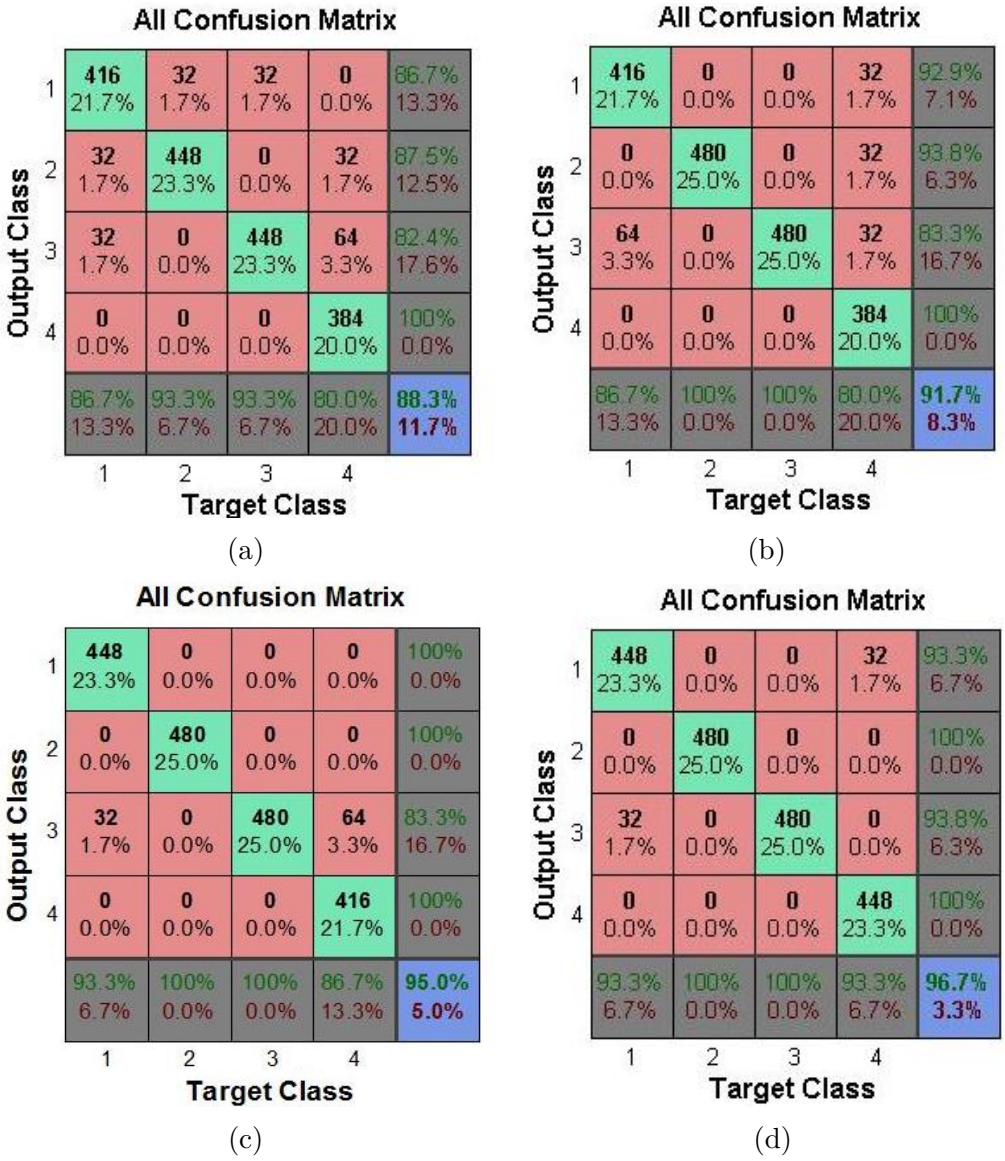


Figura 9 – Gráficos de confusão para redes com : (a) 4 neurônios na camada escondida (b) 6 neurônios na camada escondida (c) 8 neurônios na camada escondida (d) 20 neurônios na camada escondida

neurônios da camada escondida(*neuronio_h*), onde serão multiplicados pelos pesos, previamente calculados, e somados em um acumulador. O resultado do acumulador passa pela função de ativação tangente hiperbólica, a qual foi tabelada usando LUTs. A saída dos neurônios segue para os neurônios da camada de saída (*neuronio_o*), onde fará um processo semelhante ao do neurônio da camada oculta. Vale ressaltar que todos os neurônios, tanto da camada escondida quanto os de saída, foram implementados em uma abordagem paralela. Por último, a saída dos neurônios da camada de saída passam para o módulo *comparador* que verificará qual das saídas possui o maior valor.

Para sincronização de todos os módulos, foram utilizadas máquinas de estados finitos. A quantidade de estados varia de módulo para módulo.

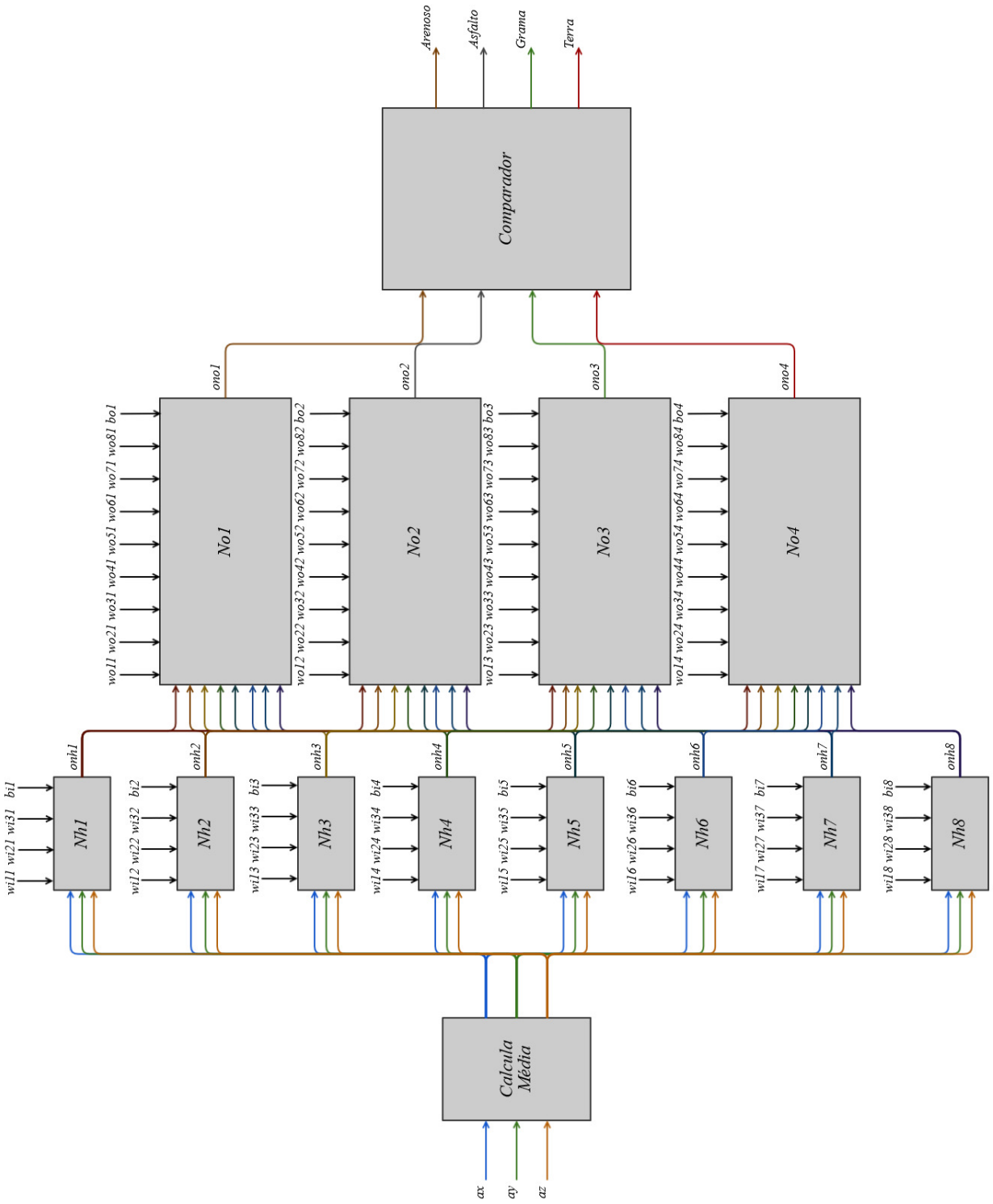


Figura 10 – Arquitetura geral.

3.5 Cálculo da média

A arquitetura do módulo que calcula a média é mostrada na Figura 11.

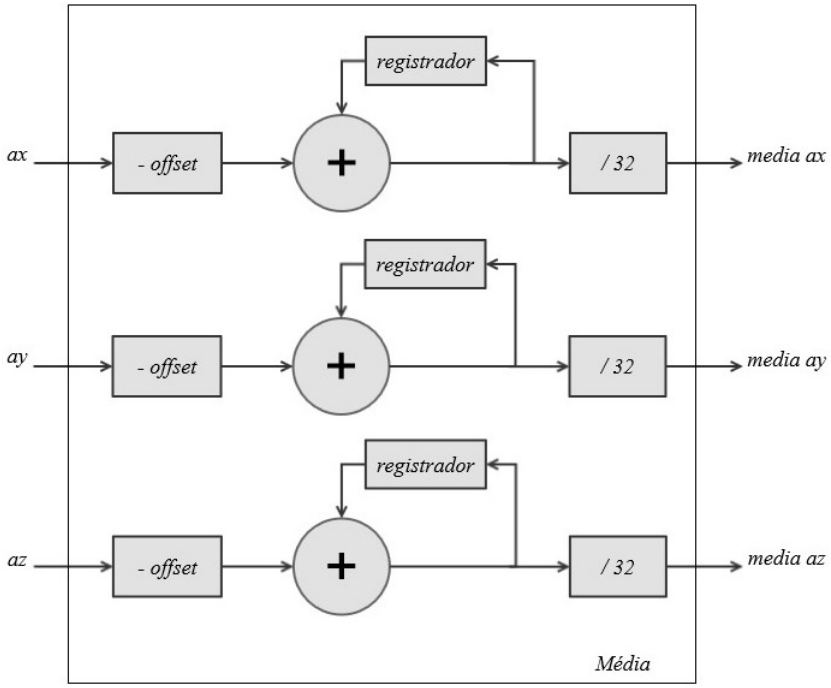


Figura 11 – Arquitetura do bloco que calcula a média.

As entradas deste bloco são provenientes do acelerômetro, que possuem palavras de 12 bits. Essas palavras são transformadas em palavras de ponto fixo com 9 bits de parte inteira e 9 bits de parte fracionária, concatenando-se zeros à esquerda da palavra de 12 bits.

Após a conversão da palavra, é subtraído um valor de offset de 1.75 do sinal de entrada. Então, é utilizado um acumulador para fazer a soma de 32 sinais de entrada. Em seguida, é feita a divisão por 32, para a qual o resultado do acumulador é deslocado cinco vezes à direita, obtendo-se assim a média dos sinais. Essa operação é feita em paralelo para as 3 entradas do bloco de média, de modo que as médias de ax, ay e az estejam prontas ao mesmo tempo.

3.6 Neurônio da camada escondida

A arquitetura do neurônio da camada escondida (*neuronio_h*) é mostrada na Figura 12. As entradas dos neurônios da camada escondida estão conectadas à saída do bloco que calcula a média. Os pesos sinápticos w são entradas constantes, definidas após o treinamento da rede neural.

As entradas e os pesos são multiplicados em paralelo, de tal modo que os resultados das multiplicações estão prontos simultâneamente. Os resultados das multiplicações

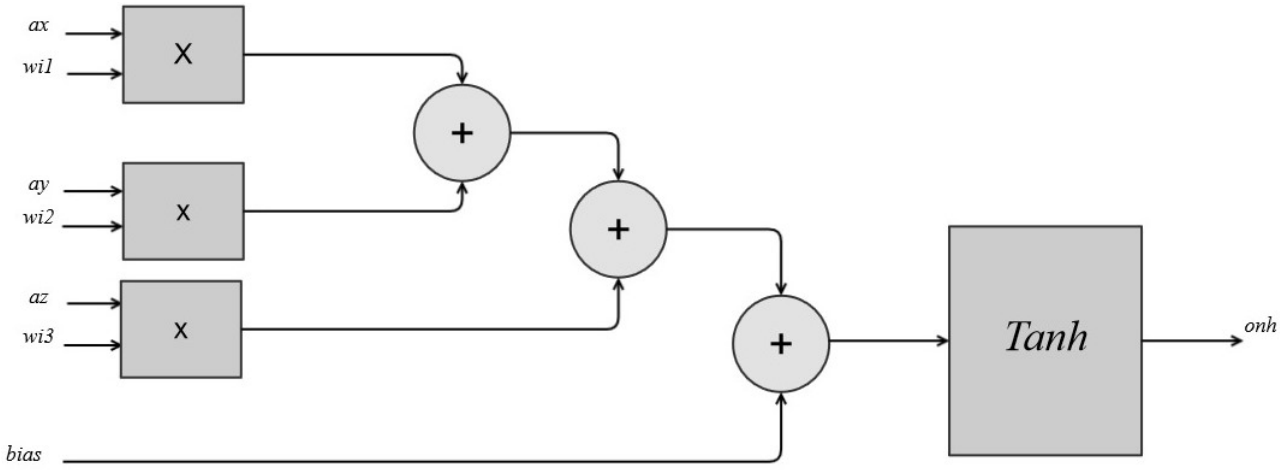


Figura 12 – Arquitetura do neurônio da camada oculta.

e o *bias* são somados em um acumulador, e passam pelo cálculo da função tangente hiperbólica. A tangente hiperbólica foi implementada através de uma LUT, utilizada para tabelar os valores da função. Foram tabelados 80 pontos da tangente hiperbólica, como mostrado na Figura 13.

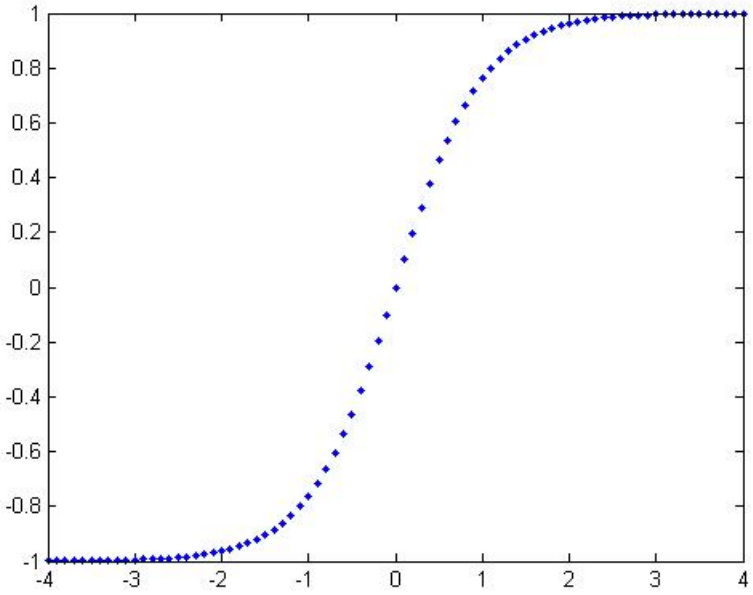


Figura 13 – Tangente hiperbólica tabelada.

3.7 Neurônio da camada de saída

A arquitetura do neurônio da camada de saída é mostrada na Figura 14. Esta arquitetura é semelhante à do neurônio da camada escondida. As entradas deste módulo são provenientes das saídas dos neurônios da camada escondida, que são multiplicados por

outros pesos sinápticos, e são somadas junto com um *bias* por um acumulador, passando em seguida pela função hiperbólica.

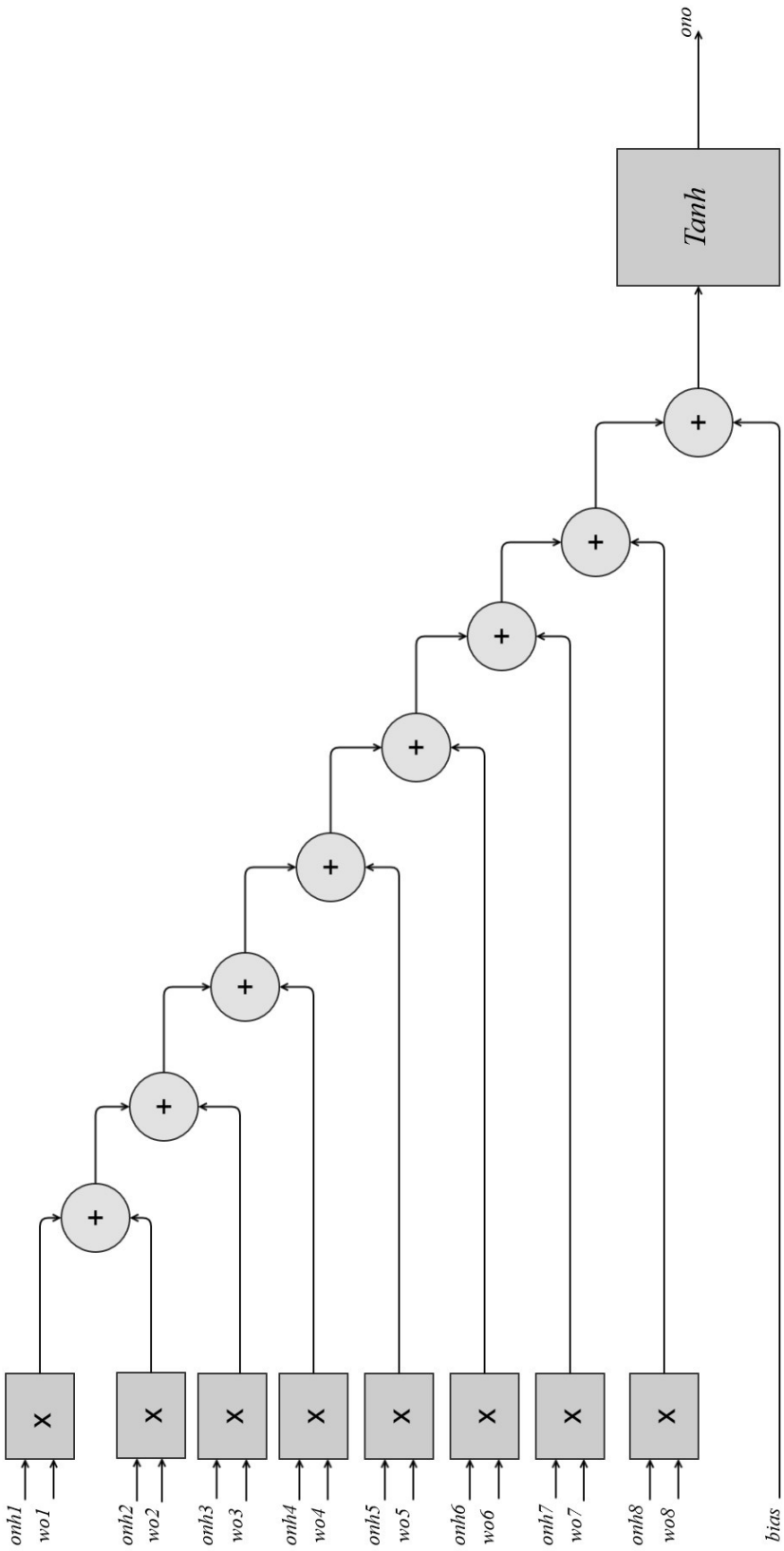


Figura 14 – Arquitetura do neurônio de saída.

3.8 Módulo de comparação

Este módulo verifica, através de comparações, qual das 4 saídas dos neurônios de saída possui o maior valor, identificando o tipo de terreno (vide Figura 15). Para verificar qual saída possui maior valor, as saídas são comparadas umas com as outras e, após verificar qual saída é maior, é enviado um nível lógico alto para uma das quatro saídas do comparador. Caso o primeiro neurônio possua maior valor, é enviado um nível lógico alto para a primeira saída do comparador indicando que o terreno identificado é o arenoso. Caso seja o segundo, o terreno identificado é o asfalto. Caso seja o terceiro, o terreno é grama e, caso seja o quarto, o terreno é terra.

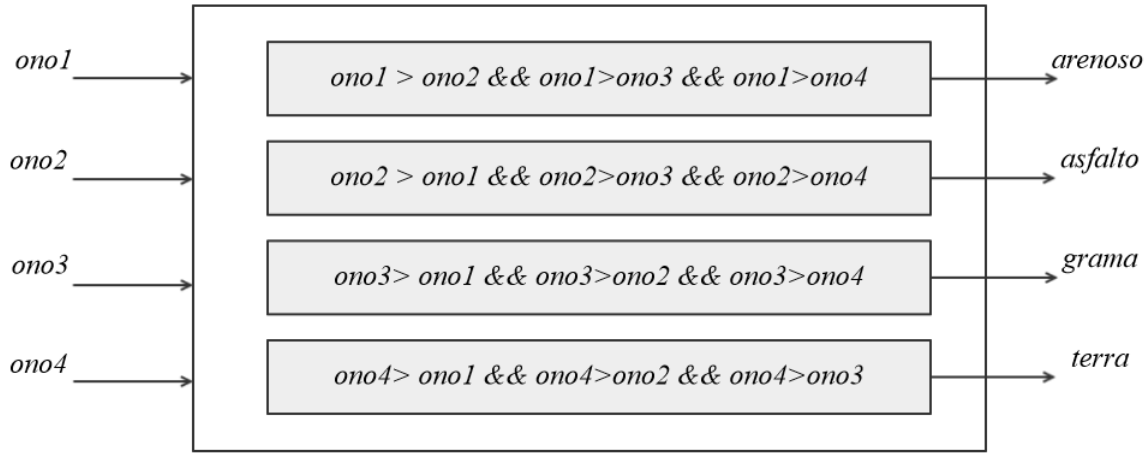


Figura 15 – Módulo comparador.

4 Resultados

Neste capítulo, serão apresentados os resultados obtidos das implementações. O capitulo está organizado da seguinte maneira: a seção 4.1 apresenta os resultados de consumo de recursos de *hardware* utilizados; a seção 4.2 mostra os resultados de validação dos circuitos propostos; e a seção 4.3 apresenta umaa comparação de tempo de execução entre implementações em *hardware* e *software*.

4.1 Análise de consumo de recursos

Os resultados de síntese lógica das arquiteturas desenvolvidas estão aprensentados na Tabela 2. Estes resultados permitem analisar o consumo de recursos de hardware dos circuitos desenvolvidos, assim como a frequência máxima de operação dos mesmos.

	LUTs (63400)	FFs (126800)	DSPs (240)	Freq (MHz)
Arquitetura Geral	3445	1579	56	205,645
Média	134	136	-	305,344
Neurônio escondido	284	212	3	237,925
Neurônio de saída	386	484	8	230,521
Comparador	158	-	-	-

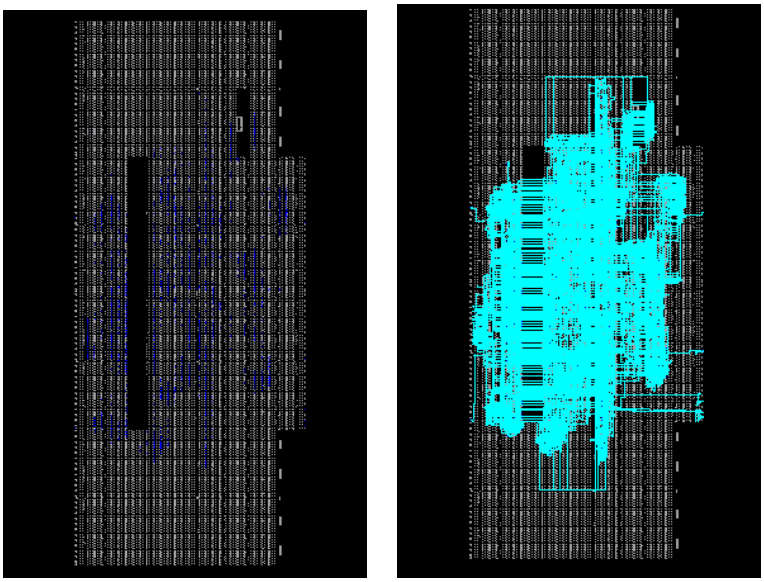
Tabela 2 – Resultado de síntese das arquiteturas implementadas.

Pode-se observar que a arquitetura geral consome aproximadamente 5.43% das LUTs, 1.24% de flip-flops e 23% de blocos DSPs. A frequência máxima de operação é de 205.6 MHz.

Após a etapa de mapeamento e roteamento da FPG, fez-se um levantamento de consumo de potência consumida pelo circuito da arquitetura geral, que é mostrado na Figura 16, onde é possivel verificar o consumo total de pontência de 101 mW, sendo $88mW$ de potência estática utilizada para ligar a FPGA e $13mW$ de potência dinâmica, associada ao funcionamento da lógica do circuito proposto. A Figura 17 mostra o resultado final do processo de mapeamento e roteamento do circuito proposto implementado no dispositivo FPGA.

	Total	Dynamic	Quiescent
Supply Power (W)	0.101	0.013	0.088

Figura 16 – Consumo de potência da arquitetura geral.



(a) (b)

Figura 17 – Área utlizada da FPGA: (a) Elementos utlizados; (b) Roteamento.

4.2 Validação da arquitetura

Para validação das arquiteturas implementadas, realizou-se simulações tanto em *hardware* quanto em *software*. Para as simulações em *hardware*, foi utilizada uma memória *ROM* (*Read Only Memory*) com 100 valores de entrada geradas aleatoriamente entre os valores de $\pm 1,75$ por um script em Matlab e uma memória *WOM* (*Write Only Memory*) onde os resultados obtidos foram armazenados. Na simulação em *software* foi implementado o modelo matemático das arquiteturas, utilizando-se as mesmas variáveis de entrada para a simulação em *hardware*. Com os resultado obtidos pela simulação em *hardware* e em *software*, foi calculado o erro quadrático médio (*MSE*) para avaliação das arquiteturas implementadas. A Tabela 3 mostra o MSE das arquiteturas dos neurônios das camadas escondida e de saída, assim como das 4 saídas da rede integrada.

Arquitetura	MSE
Neurônio escondido	4,07E-5
Neurônio de saída	1,30E-4
Rede Saída 1	3,1E-6
Rede Saída 2	1,21E-4
Rede Saída 3	6,1E-4
Rede Saída 4	3,7E-3

Tabela 3 – MSE das arquiteturas implementadas.

A Figura 18 mostra o resultado da simulação de *hardware* para uma frequência de relógio de 100 MHz. É possível verificar que o tempo de execução da rede, que começa quando o sinal de *start* vai para 1 e termina quando o *ready* vai para 1, é de 32 ciclos de relógio. A partir dos resultados da síntese lógica, pode-se observar que a frequência de

operação máxima do circuito é de 205,645 MHz e, portanto, o tempo de execução final seria de 155,6 ns.

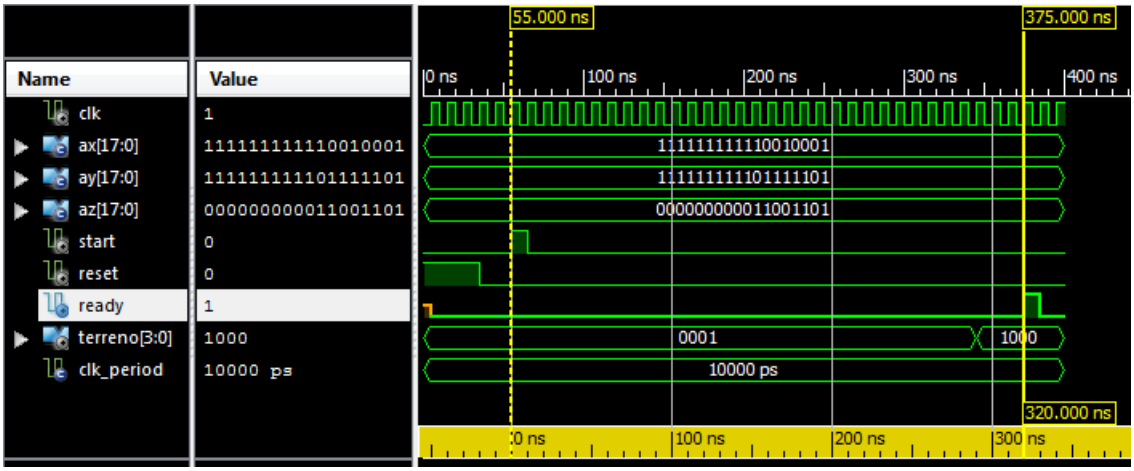
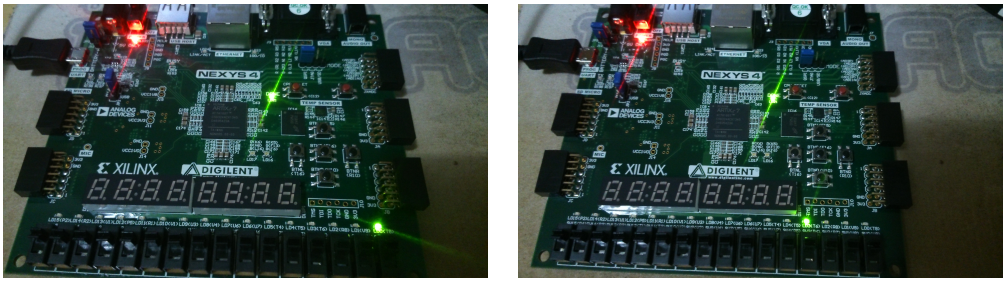


Figura 18 – Simulação da rede neural.

Foi implementado também o circuito da rede no kit Nexys 4, utilizando os mesmos valores de entrada utilizados na simulação da Figura 18, obtendo-se o mesmo resultado obtido na simulação (Fig. 19). Na Figura 19a, o circuito se encontra no estado inicial, onde o led mais da direita está ligado, devido como a lógica implementada.Após pressionar o botão start o circuito irá executar sua lógica e, como resultado, tem-se a Figura 19b, onde o quarto led da placa é aceso, indicando que foi identificado o terreno arenoso.



(a) (b)

Figura 19 – Implementação no kit Nexys 4: (a) Estado inicial; (b) Resultado final.

Outro teste foi realizado utilizando os 8 dados de cada terreno. Comparou-se então os resultados obtidos com a rede implementada na FPGA com os resultados do modelo matemático implementado no Matlab, verificando a taxa de acerto do hardware em relação ao software (Tab. 4).

4.3 Comparação HW/SW do tempo de execução

A comparação entre *hardware* e *software* foi feita comparando-se o tempo de execução da rede neural em diversas plataformas. Como base, utilizou-se o tempo de exe-

Terreno	Taxa de acerto (%)
Arenoso	100
Asfalto	70
Grama	100
Terra	100

Tabela 4 – Taxa de acerto do hardware.

cução do circuito implementado na FPGA. Implementou-se então o modelo matemático da rede em Matlab e em linguagem C. Em seguida, embarcou-se o código da rede em dois processadores de software para aplicações embarcadas diferentes. Escolheu-se o kit Arduino UNO e o MicroBlaze. O MicroBlaze é um processador de software que faz uso dos mesmos recursos de hardware (LUTs, flip-flops, blocos DSPs, etc) da FPGA em que foi implementada a arquitetura proposta. É importante salientar que, para estimar o tempo de execução no MicroBlaze foi usado um timer de 32 bits. Adicionalmente, o MicroBlaze, usa a mesma frequência de relógio usada para implementar o circuito final no dispositivo FPGA. Os resultados desta análise estão mostrados na Tabela 5.

Implementação	Ciclos de relógio	Tempo de execução (ms)	Fator de aceleração
Hardware (VHDL)	32	0,000320	1
MicroBlaze (C)	207707	-	7162,31
Arduino (C)	-	0,450	1406,25
Notebook (C)	-	0,00128	4,00

Tabela 5 – Resultado da análise do tempo de execução.

5 Discussão e Conclusões

Neste capítulo é apresentada a conclusão do trabalho. O capítulo está organizado da seguinte maneira: a seção 5.1 apresenta a discussão dos resultados obtidos e a seção 5.2 apresenta a conclusão final do trabalho.

5.1 Discussão

Na seção 4.1 é possível verificar o consumo de recursos utilizados da FPGA. Nota-se que foram utilizado 5.4% de LUTs, 1.23% de flip-flops e 23% de blocls DSPs. Adicionalmente, foi possível verificar que o circuito proposto alcança uma frequência de operação de 205 MHz e apresenta um baixo consumo de potência, aproximadamente 13 mW de potência dinâmica.

Na seção 4.2, pode-se verificar os testes de validação realizados. No primeiro teste verificou-se o erro quadrático médio para comparar o erro entre a implementação em *hardware* e *software*, já que na implementação de *hardware* aparecem problemas relacionados a arredondamentos, além do fato de que a função da tangente hiperbólica foi tabelada, aumentando a propagação do erro dos sinais. Felizmente, os erros apresentados nas saídas das arquiteturas implementadas foram relativamente baixos, não trazendo muitos problemas.

A comparação entre os resultados de *hardware* e de *software* foi de fundamental importância para a verificação do funcionamento correto do *hardware*, pois os valores encontrados estavam condizentes com valores obtidos em relação ao modelo matemático da rede implementado em *software*. Vale ressaltar que os resultados obtidos através da simulação na ferramenta *npr toolbox* estavam divergindo dos resultados encontrados tanto no modelo matemático implementado quanto na rede. Porém, verificando-se que o modelo matemático implementado estava correto, optou-se por escolher como referência o modelo matemático até que seja averiguado o porque da diferença dos resultados.

A comparação de tempo de execução mostrou que a arquitetura dedicada, mesmo com um ciclo de relógio mais baixo, pode ser mais rápida que um computador trabalhando na faixa de GHz. Algo que vale ser comentado é que a execução no arduino, que trabalha a 16 MHz, foi mais rápida que a execução no MicroBlaze, que trabalha a 100 MHz. Isso pode ter ocorrido pelo fato de que as funções utilizadas pelo MicroBlazer não estão otimizadas, fazendo com que ele gaste mais tempo para executa-las.

5.2 Conclusões Finais

Este trabalho tinha como objetivo densenvolver, implementar e validar uma arquitetura de *hardware* para solução de classificação de terreno. Sendo assim, é possivel concluir que os objetivos iniciais do trabalho foram alcançados, tendo como resultado um circuito para classificação de quatro tipos de terreno implementado em uma FPGA.

O hardware implementado apresentou resultados satisfatórios em relação à implementação realizado em software. Além disso, operaa de forma rápida, atendendo à necessidade de classificação rápida do terreno para não comprometer a integridade do robô enquanto o mesmo realiza suas tarefas.

Analisando a forma como circuito foi implementado, algumas alterações ainda podem ser feitas para uma melhora do circuito, como a implementação da função tangente hiperbólica, ao invés de usar valores tabelados. Outra proposta para o circuito seria torna-lo uma arquitetura genérica, possibilitando que a rede neural seja utilizada para outros fins além da classificação de terreno.

5.3 Proposta de trabalhos futuros

Algumas propostas para trabalhos futuros estão descritas abaixo:

- Realizar o algoritmo de treinamento online em hardware.
- Realizar a filtragem dos dados de aceleração usando outros tipos de filtro, como por exemplo o filtro Kalman.
- Fazer a implementação da função tangente hiperbólica usando métodos numéricos.
- Fazer a generalização da rede neural: permitindo ajustar a topologia da rede.
- Acrescentar um módulo de comunicação serial para envio dos pesos da rede treinada.
- Explorar outros tipos de redes neurais (p.ex: redes de Kohonen)
- Validação para outros tipos de terreno.
- Validação para diferentes tipos de plataformas móveis.

Referências

B.KRÖSE; SMAGT, P. van der. *An Introduction to Neural Network*. 8ª edição. ed. [S.l.: s.n.], 1996. Citado 2 vezes nas páginas 6 e 16.

COSTA, E. R.; GOMES, M. L.; BIANCHI, R. A. C. Um mini robÔ mÓvel seguidor de pistas guiado por visÃo local. Centro Universitário da FEI, 2003. Citado na página 11.

DAVIST, I. L. et al. Terrain typing for real robots. 1995. Citado na página 21.

DENMARK, U. of S. *spartan6*. 2014. Disponível em: <http://web.sdu.dk/jjm-/mmmi/Logic/PLDs/Spartan_6/Spartan6.htm>. Citado 2 vezes nas páginas 6 e 20.

DIGILENT. *Nexys 4*. 2014. Disponível em: <http://www.digilentinc.com/Products>. Citado 2 vezes nas páginas 6 e 22.

DUPONT, E. M. et al. Frequency response method for terrain classification in autonomous ground vehicles. 2008. Citado na página 11.

DUPONT, E. M. et al. Online terrain classification for mobile robots. 2005. Citado na página 21.

ELETRONICS, V. *3-Axis Accelerometer Board (MMA7361) for Arduino*. 2014. Disponível em: <http://www.vetco.net/catalog/product_info.php?products_id=12790>. Citado 2 vezes nas páginas 6 e 23.

HAYKIN, S. *Redes Neurais: Princípios e Prática*. 2ª edição. ed. [S.l.]: Bookman, 1999. Citado 4 vezes nas páginas 15, 16, 17 e 18.

JR., R. A. L. F. P. P. D. C.; CRUZ, F. R. B. D. *Minimização do erro no algoritmo back-propagation aplicado ao problema de manutenção de motores*. 1998. Citado na página 18.

KURBAN, T.; BEşDOK, E. A comparison of rbf neural network training algorithms for inertial sensor based terrain classification. 2009. Citado na página 21.

LARSON, A. C.; DEMIR, G. K.; VOYLES, R. M. Terrain classification using weakly-structured vehicle/terrain interaction. 2005. Citado na página 21.

MEYER-BAESE, U. *Digital Signal Processing with Field Programmable Gate Arrays*. 3ª edição. ed. [S.l.]: Springer, 2004. Citado na página 12.

NGUYEN, D.-V. Vegetation detection and terrain classification for autonomous navigation. 2013. Citado na página 21.

OJEDA, L. et al. Terrain characterization and classification with a mobile robot. 2006. Citado na página 21.

WALAS, K. et al. Hardware implementation of ground classification for a walking robot. 2013. Citado na página 21.

WOLF, D. F. et al. *Robótica Móvel Inteligente: Da Simulação às Aplicações no Mundo Real*. 2009. Citado na página 11.

XILINX. *What is Programmable Logic?* 2014. Disponível em: <http://www.xilinx-.com/company/about/programmable.html>. Citado 2 vezes nas páginas 19 e 20.