CEEL – ISSN 2595-2221
Universidade Federal de Uberlândia
06 a 10 de Novembro de 2023

IMPLEMENTAÇÃO E APLICAÇÃO DE UMA REDE NEURAL EM UM FPGA PARA AGRICULTURA

Nercílio A. U. C. Júnior[1], Antônio M. M. Medeiros[1], Bruno Q. Oliveira[1],

PUC-Goiás-Pontifícia Universidade Católica de Goiás[1]

Resumo **- A produção agropecuária deixou o mundo rústico para um novo e tecnológico, na qual cada vez mais a agricultura vem utilizando tecnologias de precisão com a ideologia de melhorar a competitividade. São inúmeras as possibilidades de aplicação de inteligência artificial na agricultura. O estudo realizado na pesquisa constata que é possível a implementação de portas logicas programáveis (FPGA) em conjunto com a inteligência artificial dentro de hardware dedicado, reduzindo os custos dos produtores e melhor produtividade. O objetivo do trabalho é apresentar uma análise da aplicação do FPGA e de inteligências artificiais na agricultura, a fim de construir e implementar uma inteligência artificial com FPGA para o controle de sistemas de irrigação e pulverização.**

Palavras-Chave – agricultura, automação, baixo custo, FPGA, inteligência artificial, irrigação.

IMPLEMENTATION AND APPLICATION OF A NEURAL NETWORK IN AN FPGA FOR AGRICULTURE

Abstract **- Agricultural production left the rustic world for a new and technological one, in which agriculture is increasingly using precision technologies with the ideology of improving competitiveness. There are countless possibilities for applying this technology, the research results show that the FPGA together with artificial intelligence can reduce the costs of producers and improve productivity. The objective of the work was to present a deep analysis of the application of FPGA and artificial intelligence in agriculture with the purpose of facilitating agriculture from a technological point of view, the construction and implementation of an artificial intelligence together with an FPGA for the control of irrigation and spraying systems.**

Keywords – agriculture, automation, low cost, FPGA, artificial intelligence, irrigation.

I. INTRODUÇÃO

Nos últimos 40 anos, o Brasil saiu da condição de importador de alimentos para se tornar um grande provedor para o mundo. Foram conquistados aumentos significativos na produção e na produtividade agropecuárias. O preço da cesta básica, no Brasil, reduziu-se consideravelmente e o país se tornou um dos principais *players* do agronegócio mundial. Hoje, se produz mais em cada hectare de terra, aspecto importantíssimo para a preservação dos recursos naturais [1].

Em breve a população mundial chegará a 8,5 bilhões de pessoas em 2030, ou seja, 16% a mais que em 2016. O Brasil deve chegar a marca de 230 milhões de pessoas nos próximos 12 anos e a população da Ásia com quase 5 bilhões de pessoas terá aproximadamente 58% da população mundial. Estima-se que em 2023 a Índia ultrapasse a China como país mais populoso do mundo. [3].

As próximas décadas devem ser de mudanças importantes na distribuição espacial da população global. Até 2030, mais de 90% da população dos países em desenvolvimento, sobretudo na África Subsaariana e na Ásia, terá se urbanizado o que trará implicações importantes em termos de consumo de alimentos, água e energia [2].

Projeções indicam ainda forte expansão da classe média na população mundial, estando a maior porção nos países da Ásia. Em 2030, 60% da população mundial deverá estar no estrato da classe média, um crescimento de 15 pontos percentuais em comparação com 2016. O aumento da renda implica mudanças nos padrões de consumo, o que resulta na expansão da demanda por carne, frutas e vegetais, na redução do consumo de alimentos básicos, na diversificação da cesta de consumo, bem como no aumento da demanda por produtos mais elaborados. [4].

O comércio mundial de soja deverá crescer 25%, ou 36 milhões de toneladas, em 10 anos. A China será responsável por 85% desse aumento. A Índia, por sua vez, será a principal responsável pelo crescimento da demanda por óleo de soja (27%). Milho e algodão também serão demandados em maior quantidade. [4].

De fato, a agricultura passa por profundas transformações econômicas, culturais, sociais, tecnológicas, ambientais e mercadológicas – que ocorrem em alta velocidade e em diferentes direções, as quais impactam de forma substancial o mundo rural. Dessa forma, para as próximas décadas, uma questão primordial relacionada ao planejamento estratégico das organizações públicas e privadas de ciência, tecnologia e inovação (CT&I) é analisar os principais sinais e tendências, antever transformações e contribuir para o delineamento estratégico da programação de pesquisa, desenvolvimento e inovação (PD&I). Isso é imprescindível para definir o

ambiente e o foco de atuação para os próximos anos no intuito de elevar ainda mais o protagonismo da agricultura brasileira. Com a preocupação de estar constantemente conectada a essas transformações e suas implicações em CT&I para a agricultura, a Embrapa tem aprofundado estudos de futuro por meio de uma rede interna de especialistas, vinculados ao Sistema de Inteligência Estratégica.

O objetivo do projeto é desenvolver uma inteligência artificial dentro de uma porta logica programável (FPGA ou *field-programmable gate array*)[5], usando a linguagem VHDL e portas lógicas, para o controle de sistemas de irrigação e de pulverização, que serão ativados em épocas específicas do ano.

II. FUNDAMENTOS TEÓRICOS

O objetivo da inteligência artificial é substituir os controladores por regras de produção e supervisores inteligentes. Inteligência artificial (IA) é uma tecnologia programada para simular a inteligência humana e, assim, ter algum nível de autonomia para tomar decisões e resolver problemas lógicos. A ideia de uma máquina que "pensa" nasceu com o matemático e criptógrafo Alan Turing, em 1950, que propôs um dispositivo que manipularia símbolos de acordo com uma série de regras, algo que perdura até os computadores atuais, já que todos funcionam com base no paradigma de Turing [6].

A completude de Turing trata-se um conjunto de regras para manipulação de dados (semelhante a uma linguagem de programação, um autómato celular, um conjunto de instruções) que pode ser usado para resolver qualquer problema de computação (simula a lógica de qualquer algoritmo de computador). Tal completude é completa ou universal se e somente se puder ser usados para controlar a máquina de Turing (a máquina digital primitiva e universal), e assim podendo controlar qualquer computador. Um exemplo clássico é o cálculo lambda (um sistema formal que estuda funções recursivas computáveis).

Além disso, Turing propôs o Jogo da Imitação, em 1950. Este jogo questiona a capacidade de um computador de enganar um terço dos jogadores. O jogo segue três regras simples:

- 3 jogadores. A, B, C.
- Eles se comunicam por apenas mensagens escritas.
- Jogador C não pode ver os outros dois e deve descobrir quem é o homem e quem é a mulher.
- Jogador A é um homem e deve confundir o jogador C para levá-lo ao erro.
- Jogador B é uma mulher e deve confundir o jogador C para levá-lo ao acerto.

Na proposta de Turing, o jogador A é substituído por um computador para determinar o sucesso da máquina em comparação aos resultados do homem. A premissa é que, se o computador tiver resultados positivos próximos aos do homem, pode-se dizer que o computador tem um determinado nível de inteligência. Essa teoria do Teste de Turing ainda é amplamente discutida e faz parte da área de filosofia da inteligência artificial (IA).

A IA irá substituir um controlador por regras de produção e supervisores inteligentes. Um exemplo de um controlador por regras de produção é uma versão proposta por Nascimento e Yoneyama [5], onde se considera uma planta descrita pela equação diferencial ordinária e invariante no tempo:

$$\frac{dx1(t)}{dt} = x2(t) \quad \frac{dx2(t)}{dt} = -\beta[x2(t)]^3 + u(t) \quad (1)$$

Onde β = 1 é uma constante e, no instante inicial t=0, o sistema está em repouso ($x_1(t)=0$ e $x_2(t) = 0$). Tome-se como MV o sinal u(t) e como PV o sinal $y = x_1(t)$. Suponha um sinal do tipo degrau para a variável de referência, $y_r(t) = 1$, $\forall_t \epsilon [0, \infty)$, ou seja, $SP = y_r(t)$.

Nessas condições, um controlador que pode ser utilizado para essa planta é da forma:

$$u(t) = -k \times sat[K, y(t) - y_r(t)] \quad (2)$$

Onde a função sat[.,.] dada por:

$$sat[M, z] = \begin{cases} z, \ se \ |z| \leq M \\ M \ sing(z), \ se \ |z| > M \end{cases} \quad (3)$$

De modo heurístico, foram propostas as seguintes regras para o ajuste *on-line* de k e K, admitindo que o nível de saturação do controle é K = 5,0 e velocidades adequadas estão na faixa de V = 0,18:

REGRA 1: Se $(x_2(t) \leq V) \wedge (|\ y(t) - y_r(t)\ | < 0{,}2) \wedge (x_1(t) \leq 1{,}0)$ Então: $(k \leftarrow 50{,}0)$ (4)

REGRA 2: Se $(x_2(t) > V) \wedge (|\ y(t) - y_r(t)\ | < 0{,}2) \wedge (x_1(t) > 1{,}0)$ Então: $(k \leftarrow -50{,}0)$ (5)

REGRA 3: Se $(x_2(t) \leq -V) \wedge (|\ y(t) - y_r(t)\ | < 0{,}2) \wedge (x_1(t) > 1{,}0)$ Então: $(k \leftarrow -50{,}0)$ (6)

REGRA 4: Se $(x_2(t) \leq -V) \wedge (|\ y(t) - y_r(t)\ | < 0{,}2)$ Então: $(k \leftarrow 1{,}0)$ (7)

REGRA 5: Se $(x_2(t) \leq -V) \wedge (x_1(t) < 1{,}0)$ Então: $(k \leftarrow 1{,}0)$ (8)

Uma vez que as regras 1 a 5 promovem um controle com saída levemente oscilatória, é introduzida uma regra adicional:

REGRA 6: Se $(t > 2{,}0)$ (9)

Então:

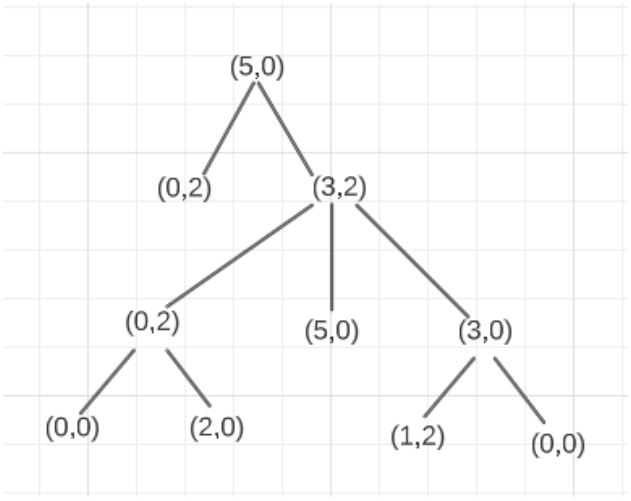
$$(aux \leftarrow aux + 0{,}05) \wedge (V \leftarrow V/aux)$$

Onde aux é uma variável que permite ajustar dinamicamente a faixa de variação V da velocidade.

Para o desenvolvimento da inteligência artificial, primeiro devem ser estabelecidos alguns métodos de busca. Muitos problemas quem envolvem IA necessitam de métodos de casamento (*matching*) e busca (*search*) durante o processo de solução [7].

Considere-se o seguinte problema: Querem-se obter exatamente 1 litro de água em uma das jarras, a partir de uma jarra com 5 litros de capacidade, inicialmente cheia, e uma de 2 litros, inicialmente vazia. Representando as duas garrafas por um par ordenado (*x*,*y*), onde *x* é o número de litros de agua na primeira jarra e *y* na segunda, os passos para a solução do problema podem ser apresentados na seguinte forma gráfica, Figura 1.

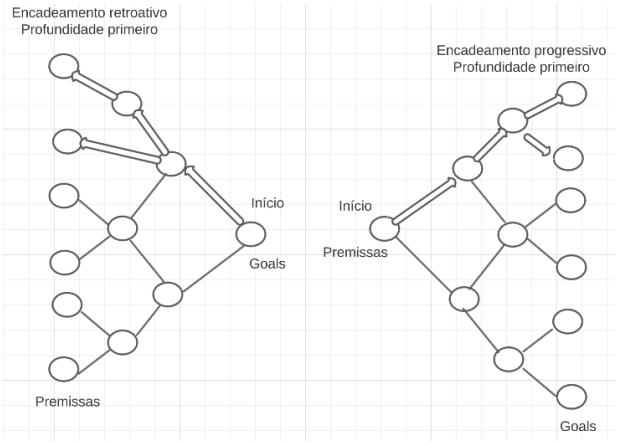
Figura 1: Árvore correspondente ao problema dos jarros d'água.



O problema das jarras pode, portanto ser resolvido mediante o uso de um algoritmo de busca em arvores. Aqui, a partir do nó (5,0), será feito uma busca que terminaria ao encontrar o nó objetivo (1,2).

Desta forma, os processos de busca podem ser de encadeamento progressivo (*data driven*) ou retroativo (*goal* driven). No primeiro caso, apresenta-se um conjunto de premissas que irão provar um objetivo (*goal*) como incorreto ou correto. Já no segundo caso, tem-se um conjunto de objetivos que buscam provar que o conjunto de premissas é correto ou incorreto. Ainda, dependendo de como se executa o controle sobre a busca, pode ser do tipo profundidade primeiro ou largura primeiro, Figura 2.

Figura 2: Ilustração das formas de encadeamento retroativo e progressivo para o caso de busca em profundidade.



III. DESENVOLVIMENTO

Com estes artifícios em mão, é possível estabelecer um modelo teórico da inteligência artificial a ser implementada. Conforme foram apresentados os processos de busca, para esta solução, utilizaremos um encadeamento progressivo.

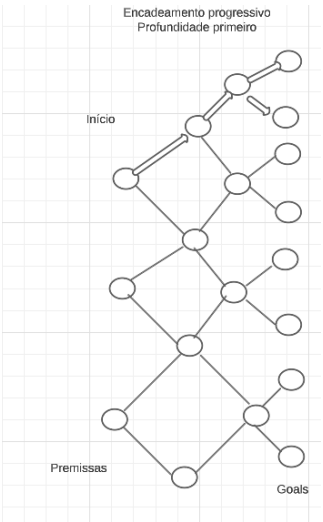
Como explicado anteriormente, um encadeamento progressivo trata-se de um agrupamento de ideias iniciais que irão provar uma intenção.

No caso deste projeto, as ideias iniciais são a seguinte:

- Data, mais especificamente o mês (para comparação com a base de dados);
- Plantação em questão (o que está sendo plantado);
- Tempo na propriedade agrícola (chuvoso, ensolarado etc.);
- Tempo em execução no sistema (quanto tempo os sistemas devem bombear água para a plantação);

Baseado nestas premissas, pode criar o seguinte grafo para a busca em encadeamento progressivo, Figura 3.

Figura 3: Possível grafo para o encadeamento progressivo.



Após a definição das ideias iniciais, serão projetadas as regras de controle. Tais regras determinarão quando e quanto o sistema será iniciado.

Como exemplo, implementaremos 6 regras genéricas para uma plantação composta por três tipos de plantas diferentes.

1. Regra de intervalo fixo:
    a. Irrigar todas as plantas a cada três dias.
    b. Verificar se a diferença entre a data atual e a data da última irrigação é igual a três dias.

2. Regra de intervalo variável por planta:
    a. Planta 1: irrigar a cada dois dias.
    b. Planta 2: irrigar a cada quatro dias.
    c. Planta 3: irrigar a cada cinco dias.
    d. Verificar a diferença entre a data atual e a data da última irrigação para cada planta e aplicar o intervalo correspondente.

3. Regra de horário específico por planta:
    a. Planta 1: irrigar pela manhã, entre as 6:00 e as 8:00.
    b. Planta 2: irrigar à tarde, entre as 14:00 e as 16:00.
    c. Planta 3: irrigar à noite, entre as 19:00 e as 21:00.
    d. Verificar o tempo atual e aplicar o horário de irrigação específico para cada planta.

4. Regra de umidade do solo por planta:
    a. Planta 1: irrigar quando a umidade do solo estiver abaixo de 30%.
    b. Planta 2: irrigar quando a umidade do solo estiver abaixo de 40%.
    c. Planta 3: irrigar quando a umidade do solo estiver abaixo de 50%.
    d. Medir a umidade do solo por meio de sensores e acionar a irrigação com base na umidade de cada planta.

5. Regra de temperatura e umidade relativa do ar:
    a. Irrigar todas as plantas quando a temperatura ambiente estiver acima de 30°C e a umidade relativa do ar estiver abaixo de 40%.
    b. Medir a temperatura e a umidade relativa do ar por meio de sensores e ativar a irrigação quando as condições forem atendidas.

Agora, tais regras serão convertidas em lógica digital. Cada regra pode ser representada como uma combinação de portas lógicas para tomar decisões com base nas entradas.

Após isso, o programa será gravado em um FPGA Altera, modelo EPM7032SLC44-10 e conectado à uma base de dados que contém o nome das plantas, datas para pulverização e irrigação.

Em uma primeira versão do código, não era possível armazenar as datas anteriores, pois não havia nenhum tipo de memória. Porém, agora foram implementados dois códigos: um que utiliza uma memória ROM (*Read-Only-Memory*) e um que utiliza uma RAM (*Random Access Memory*).

IV. RESULTADOS

A memória ROM é um tipo de memória de computador que armazena dados permanentes que não são perdidos quando a energia é desligada. Os dados na memória ROM são gravados durante a fabricação e não podem ser alterados ou regravados após esse processo. Em outras palavras, a memória ROM é "somente leitura" em comparação com a memória RAM, que é "gravável" e "apagável". A memória ROM neste contexto é usada para armazenar os parâmetros treináveis da rede neural, como os pesos e biases, que são cruciais para o funcionamento da rede.

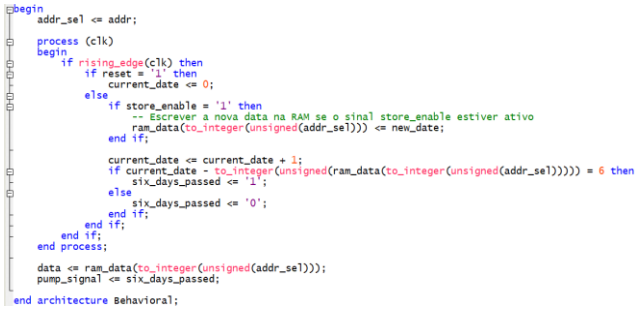


Figura 4: Exemplo do código de implantação da regra citada.

A Figura 5 motra as bibliotecas necessárias para o código. A biblioteca ieee.std_logic_1164 fornece o tipo de dados std_logic e operações relacionadas.

Figura 5: biblioteca inseria no programa.

```
1   library ieee;
2   use ieee.std_logic_1164.all;
```

Aqui é definida a entidade `ROM_with_DateComparison`, que representa um componente de ROM com uma comparação de datas. Ele possui parâmetros genéricos `D_Width` e `A_Width` que determinam o tamanho dos vetores de dados e endereço, respectivamente. Além disso, há portas de entrada (`clk`, `addr`, `reset`) e portas de saída (`data`, `pump_signal`) definidas para a interface do componente.

Figura 6: Declaração da entidade.

```
entity ROM_with_DateComparison is
    generic (
        D_Width : integer := 8;
        A_Width : integer := 3
    );
    port (
        clk         : in std_logic;
        addr        : in std_logic_vector(A_Width-1 downto 0);
        data        : out std_logic_vector(D_Width-1 downto 0);
        pump_signal  : out std_logic;
        reset       : in std_logic
    );
end entity ROM_with_DateComparison;
```

Esta seção define a arquitetura `ROm_Arch` para a entidade `ROM_with_DateComparison`. Ela declara sinais internos que serão usados dentro desta arquitetura, incluindo `rom_d` e `data_reg` para armazenar dados da ROM, `addr_sel` para seleção de endereço, `current_date` e `stored_date` para datas, `difference` para armazenar a diferença entre datas e `six_days_passed` para indicar se seis dias se passaram.

Figura 7:

```
architecture ROm_Arch of ROM_with_DateComparison is
    signal rom_d, data_reg : std_logic_vector(D_Width-1 downto 0);
    signal addr_sel : std_logic_vector(2 downto 0);
    signal current_date, stored_date : integer := 0;
    signal difference : integer;
    signal six_days_passed : std_logic := '0';
```

Esta seção define a arquitetura `ROm_Arch` para a entidade `ROM_with_DateComparison`. Ela declara sinais internos que serão usados dentro desta arquitetura, incluindo `rom_d` e `data_reg` para armazenar dados da ROM, `addr_sel` para seleção de endereço, `current_date` e `stored_date` para datas, `difference` para armazenar a diferença entre datas e `six_days_passed` para indicar se seis dias se passaram

Figura 8:

```
component DateComparison is
    port (
        clk          : in std_logic;
        reset        : in std_logic;
        current_date  : out integer;
        stored_date   : out integer;
        date_compare  : out std_logic
    );
end component;
```

Este bloco contém o início da arquitetura `ROm_Arch`. Ele faz o seguinte:

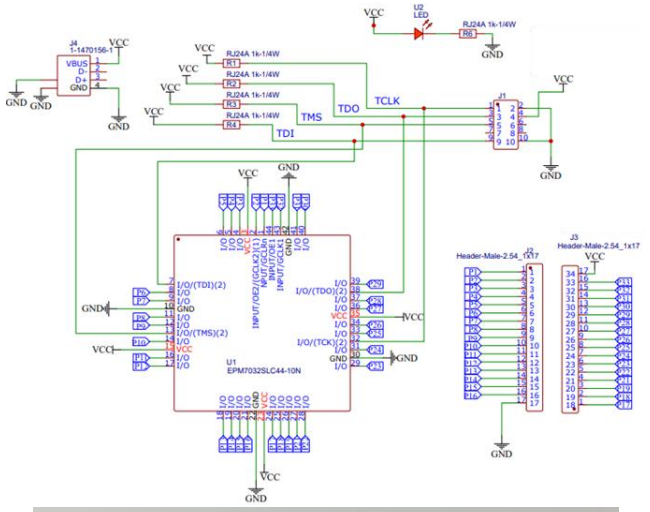
- `addr_sel` é atribuído a partir do sinal `addr`, que seleciona a linha da ROM a ser lida.
- `rom_proc` é um processo sensível ao clock que atualiza o registro `data_reg` na borda de subida do clock.
- `lookup_proc` é um processo sensível a `addr_sel` que implementa a lógica da ROM com base no endereço selecionado.
- `data` é atribuído ao valor de `data_reg`, que é a saída da ROM.
- `pump_signal` é acionado quando `six_days_passed` é verdadeiro, indicando que seis dias se passaram.

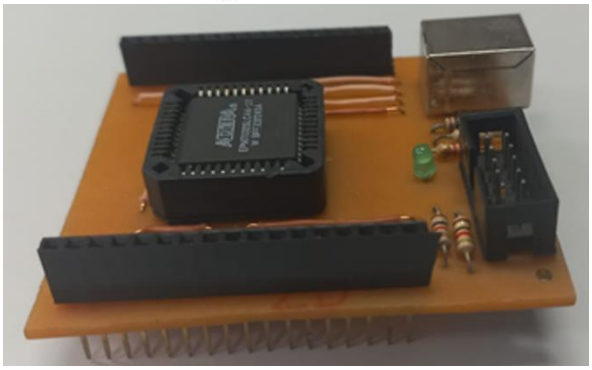
Finalmente, uma instância do componente `DateComparison` é criada e conectada às portas relevantes, com os sinais internos usados para armazenar as datas e a comparação de datas.

```
begin
    addr_sel <= addr;

    process (clk)
    begin
        if rising_edge(clk) then
            if reset = '1' then
                current_date <= 0;
            else
                if store_enable = '1' then
                    -- Escrever a nova data na RAM se o sinal store_enable estiver ativo
                    ram_data(to_integer(unsigned(addr_sel))) <= new_date;
                end if;

                current_date <= current_date + 1;
                if current_date - to_integer(unsigned(ram_data(to_integer(unsigned(addr_sel))))) = 6 then
                    six_days_passed <= '1';
                else
                    six_days_passed <= '0';
                end if;
            end if;
        end if;
    end process;

    data <= ram_data(to_integer(unsigned(addr_sel)));
    pump_signal <= six_days_passed;
end architecture Behavioral;
```

Após isso, o programa foi gravado em um FPGA Altera, modelo EPM7032SLC44-10 [12] e conectado à uma base de dados que contém o nome das plantas, datas para pulverização e irrigação. Os FPGA, são dispositivos lógicos programáveis que possui uma arquitetura baseada em blocos lógicos configuráveis, constituídos por portas lógicas e flip-flops que visam implementar funções lógicas, também é estruturado por chamadas de blocos de entrada e saída (*IOB – In/Out Blc ks*), possuem uma arquitetura configurável de forma distinta dos microprocessadores ou microcontroladores usuais, pois não fazem processamento de funções com tarefas executadas de forma sequencial a o longo de um determina do período. Estes dispositivos executam processamento em paralelo, envolvendo diversas unidades funcionais, a fim de diminuir o tempo de resposta, aumentar o desempenho de execução dos conjuntos de instruções, permitindo a customização da capacidade computacional da máquina de acordo com a a plicação. Na Figura 9, o hardware desenvolvido durante a fase de implementação da IA no FPGA.

Figura 9: Circuito e placa desenvolvida para a implementação da IA no FPGA.





V. CONCLUSÕES

Este trabalho explorou a implementação e aplicação de uma rede neural em um FPGA para agricultura, demonstrando as vantagens dessa abordagem na otimização de processos agrícolas. Os resultados obtidos evidenciam que a utilização de redes neurais em um FPGA pode trazer melhorias significativas na precisão e velocidade do processamento de dados agrícolas, permitindo a tomada de decisões mais eficientes e aprimorando a produtividade do setor.

A implementação de uma rede neural em um FPGA oferece a vantagem de uma arquitetura altamente paralela e reconfigurável, permitindo a adaptação rápida a diferentes tarefas e demandas específicas da agricultura. Além disso, o uso de um FPGA oferece baixa latência e baixo consumo de energia, tornando-o uma solução promissora para aplicações em tempo real em ambientes agrícolas.

Através da aplicação da rede neural implementada em um FPGA, é possível obter detecção de doenças em plantas, no monitoramento e controle de irrigação, na classificação de culturas e na previsão de safras. Essas aplicações demonstram o potencial dessa tecnologia para melhorar a eficiência dos processos agrícolas, reduzir o uso de recursos e promover uma produção mais sustentável

Em suma, este estudo destaca o potencial da implementação de redes neurais em FPGA para melhorar a eficiência e a produtividade na agricultura. Com o contínuo avanço da tecnologia e aprimoramento das técnicas de inteligência artificial, espera-se que a utilização de redes neurais em FPGA desempenhe um papel cada vez mais relevante no desenvolvimento de soluções inovadoras para os desafios enfrentados pelo setor agrícola, contribuindo para um futuro mais sustentável e produtivo.

REFERÊNCIAS

[1] G. R. Correia, H. R. O. Rocha, S. D. Rissimo, Automação de sistemas de irrigação com monitoramento via aplicativo web, Revista Engenharia na Agricultura - REVENG Engenharia na agricultura, viçosa - MG, v.24 n.4, julho /agosto 2005, 314-325p.

[2] V. G. Guimarães, Automação e Monitoramento Remoto de Sistemas de Irrigação Visando Agricultura Familiar, [Distrito Federal]. xiii, 81p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2011). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia, 2011.

[3] Embrapa. Visão 2030: o futuro da agricultura brasileira. – Brasília, DF: Embrapa, 2018. 212 p.: il. color.; 18,5 cm x 25,5 cm. ISBN 978-85- 7035-799- 1

[4] F. M. A Lamas, A tecnologia na agricultura. Embrapa Agropecuária Oeste. EMBRAPA, Brasília, DF, 2017. Disponível em: < https://www.embrapa.br/busca-de-noticias/-/noticia/30015917/artigo-a-tecnologia-na-agricultura > Acesso em: 08 junho 2023.

[5] C. NASCIMENTO JR., T. YONEYAMA, Inteligência artificial em controle e automação. São Paulo: Blucher, 2000.

[6] D. J. Gunke, Comunicação e inteligência artificial: novos desafios e oportunidades para a pesquisa em comunicação Galaxia (São Paulo, online), ISSN 1982-2553, n. 34, jan-abr., 2017, p. 05-19. http://dx.doi.org/10.1590/1982-2554201730816

[7] Z. L. Kovacs, Redes neurais artificiais: fundamentos e aplicações - um texto básico. 4. ed. São Paulo: Livraria da Fisica, 2006.

[8] J. Cullen, A. Gerbeth, M. Dorojevets, FPGA-based Satisfiability Filters for Deep Packet Inspection. 2018, IEEE Long Island Systems, Applications and Technology Conference (LISAT). Farmingdale:. 2018. p. 1-4

[9] C. Cocollo, Implementação em FPGA de uma rede neural de Hopfield, 2015. 52 f. Trabalho de Conclusão de Curso – Departamento de Engenharia Elétrica e de Computação. Universidade de São Paulo – Escola de Engenharia de São Carlos.

[10] P. P. Chu, Embedded SoPC design with NIOSII processor and Verilog examples, John Wiley & Sons, Inc., Hoboken, New Jersey, 2012.

[11] A. R. Omondi, J. C. Rajapakse, FPGA Implementations of Neural Networks, 1a ed., Ed. Springer, New York City, 2006

[12] ALTERA CORPORATION. MAX 7000 Programmable Logic Device Family. Altera Corporation, ver. 6.7, 2005. Data sheet. Disponível em: https://cdrdv2-public.intel.com/654718/m7000.pdf. Acessado em: 31 nov. 2023.