

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
Trabalho Final de Curso II

Nercílio Antônio Urias Cavalcante Júnior

**IMPLEMENTAÇÃO DE UM PROTÓTIPO DE UMA INTELIGÊNCIA ARTIFICIAL EM UM
FPGA PARA AGRICULTURA**

Trabalho Final de Curso II como parte dos requisitos para
obtenção do título de bacharel em Engenharia da
Computação apresentado à Pontifícia Universidade
Católica de Goiás.

BANCA EXAMINADORA:

**Prof. Dr. Antônio Marcos Melo Medeiros – Orientador. Pontifícia
Universidade Católica.**

Prof. Dr. Bruno Quirino de Oliveira – Pontifícia Universidade Católica.

Prof. Me. Luis Fernando Pagotti – Pontifícia Universidade Católica.

Goiânia, 15 de dezembro de 2023.

IMPLEMENTAÇÃO DE UM PROTOTIPO DE UMA INTELIGENCIA ARTIFICIAL EM UM FPGA PARA AGRICULTURA

Nercílio A. U. C. Júnior¹, Antônio M. M. Medeiros¹, Bruno Q. Oliveira¹, Luis F. Pagotti
Escola politécnica e de artes, PUC-Goiás-Pontifícia Universidade Católica de Goiás¹

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 lógicas 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 OF AN ARTIFICIAL INTELLIGENCE 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 lógica 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 autômato 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:

$$\frac{dx_1(t)}{dt} = x_2(t) \quad \frac{dx_2(t)}{dt} = -\beta[x_2(t)]^3 + u(t) \quad (1)$$

Onde $\beta = 1$ 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, $y_r(t) = 1, \forall t \in [0, \infty)$, ou seja, $SP = y_r(t)$.

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

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

Onde a função $\text{sat}[\dots]$ dada por:

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

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 $V = 0,18$:

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

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

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

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

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

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

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

Então:

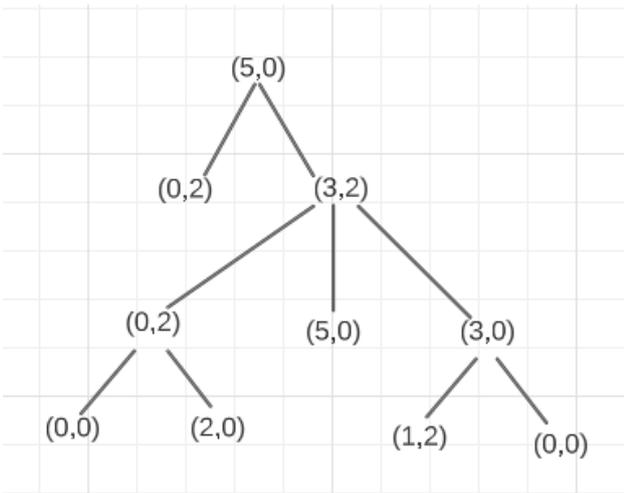
$$(aux \leftarrow aux + 0,05) \wedge (V \leftarrow V/aux)$$

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 que 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 água na primeira jarra e y na segunda, os passos para a solução do problema podem ser apresentados na seguinte forma gráfica, Figura 1.

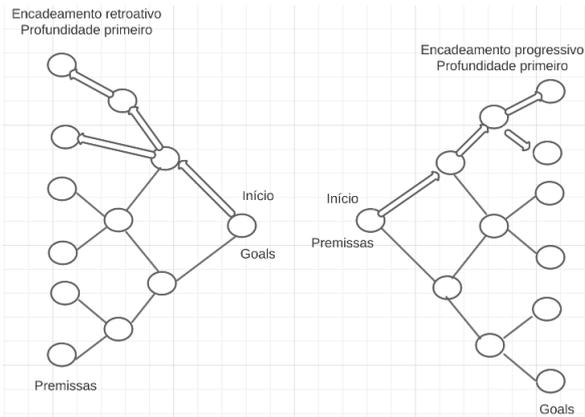
Figura 1: Árvore correspondente ao problema dos jarros d'água.



O problema das jarras pode, portanto ser resolvido mediante o uso de um algoritmo de busca em árvores. 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.

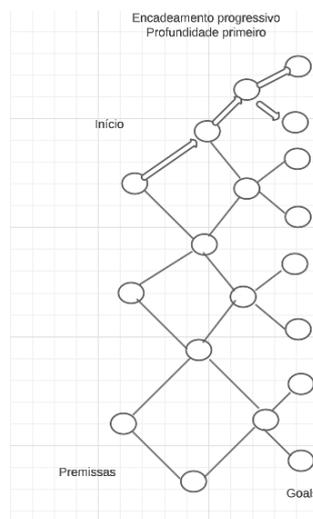
Como explicado anteriormente, um encadeamento progressivo trata-se de um agrupamento de ideias iniciais que irão provar uma intenção.

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 9 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.
6. Regra de sazonalidade por planta:
 - a. Planta 1: aumentar a frequência de irrigação durante o verão.
 - b. Planta 2: aumentar a frequência de irrigação durante a primavera.
 - c. Planta 3: aumentar a frequência de irrigação durante o outono.
 - d. Ajustar a frequência de irrigação com base na estação do ano específica para cada planta.

7. Regra de luz solar direta:
 - a. Irrigar todas as plantas quando a luz solar direta exceder um certo limite durante o dia.
 - b. Medir a intensidade da luz solar direta por meio de sensores e acionar a irrigação quando necessário.
8. Regra de chuva recente:
 - a. Não irrigar as plantas se tiver chovido nas últimas 24 horas.
 - b. Verificar registros de chuva recentes ou previsões meteorológicas para evitar irrigação desnecessária.
9. Regra de umidade foliar:
 - a. Irrigar todas as plantas quando a umidade foliar estiver abaixo de um determinado limite.
 - b. Medir a umidade foliar por meio de sensores e ativar a irrigação quando necessário.

Agora, tais regras serão convertidas em lógica digital usando portas lógicas disponíveis em VHDL, como portas AND, OR, NOT, entre outras [8]. Cada regra pode ser representada como uma combinação de portas lógicas para tomar decisões com base nas entradas [9].

Como exemplo, usaremos uma abordagem simplificada, onde apenas uma regra de controle é implementada: a planta precisa de água se pelo menos uma das duas primeiras posições do vetor *current_plant* estiverem definidas como '1'. O intervalo de irrigação é definido como 2 dias.

Os Field-Programmable Gate Arrays (FPGAs) são dispositivos eletrônicos versáteis e poderosos que oferecem flexibilidade excepcional na implementação de circuitos digitais. Ao contrário dos circuitos integrados fixos, os FPGAs são reconfiguráveis e permitem que os desenvolvedores programem a funcionalidade do hardware depois que o dispositivo é fabricado. Eles consistem em uma matriz de blocos lógicos configuráveis e interconexões programáveis que podem ser personalizados para atender a uma ampla gama de aplicações, desde processamento de sinais até sistemas embarcados complexos. A capacidade de reprogramar esses dispositivos torna-os ideais para prototipagem rápida, desenvolvimento de produtos e até mesmo para aplicações em evolução, permitindo adaptações rápidas e eficientes.

Além disso, os FPGAs oferecem vantagens significativas em termos de desempenho e eficiência energética. Sua arquitetura paralela permite a execução simultânea de várias tarefas, resultando em alta velocidade e capacidade de processamento. Esses dispositivos são altamente valorizados em áreas como inteligência artificial, redes neurais, telecomunicações e computação de alto desempenho, onde a capacidade de processamento paralelo e a capacidade de adaptação são essenciais. Com a capacidade de personalização e otimização para tarefas específicas, os FPGAs continuam a desempenhar um papel fundamental na

inovação tecnológica e no desenvolvimento de sistemas computacionais avançados

IV. RESULTADOS

Foi desenvolvido a programação no software Quartus II em VHDL própria para portas lógicas programáveis [10]. Na Figura 4 tem o início da programação, essa implementação pode ser expandida para incorporar mais regras e condições usando portas lógicas adicionais. No entanto, à medida que a complexidade das regras aumenta [11].

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

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity IrrigationControl is
5  port (
6    clk : in std_logic;
7    current_date : in std_logic_vector(7 downto 0);
8    current_time : in std_logic_vector(7 downto 0);
9    current_plant : in std_logic_vector(3 downto 0);
10   irrigation_signal : out std_logic;
11 );
12 end entity IrrigationControl;
13
14 architecture behavioral of IrrigationControl is
15   constant WATERING_INTERVAL : integer := 2; -- Intervalo de dias entre as irrigações
16   signal last_watered_date : std_logic_vector(7 downto 0);
17   signal irrigate_plant : std_logic;
18   signal plant_needs_water : std_logic;
19
20 begin
21   process (clk)
22   begin
23     if rising_edge(clk) then
24       -- Verifica se é hora de irrigar a planta
25       if current_date /= last_watered_date and current_date(current_date'high - 1 downto 0) = "000000" then
26         irrigate_plant <= '1';
27       else
28         irrigate_plant <= '0';
29       end if;
30
31       -- Determina se a planta precisa de água com base em regras específicas
32       plant_needs_water <= current_plant(0) or current_plant(1); -- Exemplo de regra simples
33
34       -- Atualiza a data da última irrigação
35       if irrigate_plant = '1' then
36         last_watered_date <= current_date;
37       end if;
38     end if;
39   end process;
40
41   -- Sinal de irrigação ativado se for hora de irrigar e a planta precisa de água
42   irrigation_signal <= '1' when irrigate_plant = '1' and plant_needs_water = '1' else '0';
43
44 end architecture behavioral;
45
46
47

```

A Figura 5 mostra 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;

```

A declaração da entidade, define as portas de entrada e saída do módulo *IrrigationControl*. Inclui um sinal de relógio de entrada (*clk*), sinais de entrada para data atual (*current_date*), hora atual (*current_time*) e planta atual (*current_plant*), bem como um sinal de saída para o controle de irrigação (*irrigation_signal*), Figura 6.

Figura 6: Declaração da entidade.

```

4  entity IrrigationControl is
5  port (
6    clk : in std_logic;
7    current_date : in std_logic_vector(7 downto 0);
8    current_time : in std_logic_vector(7 downto 0);
9    current_plant : in std_logic_vector(3 downto 0);
10   irrigation_signal : out std_logic;
11 );
12 end entity IrrigationControl;

```

O bloco de arquitetura é denominado "*behavioral*", é define o comportamento do módulo *IrrigationControl*. Ele contém três sinais internos: *last_watered_date*, *irrigate_plant* e *plant_needs_water*, Figura 7.

Figura 7: Bloco behavioral.

```

21  begin
22  process (clk)
23  begin
24    if rising_edge(clk) then
25      -- Verifica se é hora de irrigar a planta
26      if current_date /= last_watered_date and current_date(current_date'high - 1 downto 0) = "000000" then
27        irrigate_plant <= '1';
28      else
29        irrigate_plant <= '0';
30      end if;
31
32      -- Determina se a planta precisa de água com base em regras específicas
33      plant_needs_water <= current_plant(0) or current_plant(1); -- Exemplo de regra simples
34
35      -- Atualiza a data da última irrigação
36      if irrigate_plant = '1' then
37        last_watered_date <= current_date;
38      end if;
39    end if;
40  end process;
41
42  -- Sinal de irrigação ativado se for hora de irrigar e a planta precisa de água
43  irrigation_signal <= '1' when irrigate_plant = '1' and plant_needs_water = '1' else '0';
44
45  end architecture behavioral;

```

Dentro do bloco de processo, que é sensível à borda de subida do relógio (*clk*), o código verifica se é hora de irrigar a planta. Ele compara a data atual (*current_date*) com a data da última irrigação (*last_watered_date*) e verifica se a hora atual (*current_time*) é igual a "000000". Se essas condições forem atendidas, o sinal *irrigate_plant* é definido como '1'; caso contrário, é definido como '0'.

Em seguida, o código determina se a planta precisa de água com base em regras específicas. Neste exemplo, ele verifica se o primeiro ou o segundo bit do sinal *current_plant*, na Figura 8 mostra a tela do simulador.

Figura 8: Bloco de variáveis no simulador.

Nome	Valor	Tipo	Modo
clk	U	Signal	In
current_date	UUUUUUUU	Signal	In
(7)	U	Signal	In
(6)	U	Signal	In
(5)	U	Signal	In
(4)	U	Signal	In
(3)	U	Signal	In
(2)	U	Signal	In
(1)	U	Signal	In
(0)	U	Signal	In
current_time	UUUUUUUU	Signal	In
current_plant	UUUU	Signal	In
irrigation_signal	U	Signal	Out
last_watered_date	UUUUUUUU	Signal	Internal
irrigate_plant	U	Signal	Internal
plant_needs_water	U	Signal	Internal
WATERING_INTERVAL	2	Con...	Internal

Tal bloco mostra todas as variáveis e bits dos *vectors* *current_date*, *current_time*, *current_plant* e *last_wastered_date*.

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 Blcks*), 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

RESOLUÇÃO nº 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O estudante NERCÍLIO ANTÔNIO VRIAS CAVALCANTE JÚNIOR do Curso de ENGENHARIA DA COMPUTAÇÃO, matrícula 2019.10033.0020 telefone: (62)98422-7793, e-mail nercilio.jr@gmail.com.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado IMPLEMENTAÇÃO E APLICAÇÃO DE UMA REDE NEURAL EM UM FPGA PARA AGRICULTURA, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 14 de DEZEMBRO de 2023.

Assinatura do autor: Nercílio Antônio Vrias Cavalcante Júnior

Nome completo do autor: NERCÍLIO ANTÔNIO VRIAS CAVALCANTE JÚNIOR

Assinatura do professor-orientador: Antônio Marcos Melo Medeiros

Nome completo do professor-orientador: ANTÔNIO MARCOS MELO MEDEIROS