UFRN - UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Departamento de Engenharia de Computação e Automação

Controlador de Nível de Transferência entre Registradores para Placa Solar Girassol com Filtro Digital de Média Móvel

Sistemas Digitais (DCA 3301.0)

Discentes:

Célio Felipe Bezerra Santiago

Gabriel André Amrim Soares

To Ba Thanh Tung

Lucas Henrique Alvez de Queiroz

NATAL-RN

2025

Resumo

Este projeto apresenta a implementação e o desenvolvimento de um sistema de controle digital para um rastreamento de um eixo solar único (Girassol), planejando potencializar a eficiência na geração de energia. O propósito principal é decifrar a instabilidade e o desgaste mecânico causados por ruídos e variações abruptas na luminosidade em sensores LDR. A disposição do sistema foi planejada em nível de transferência entre registradores (RTL) que se dividem em dois blocos principais: Um caminho de Dados (Datapath) e um Bloco de Controle. O caminho de Dados aplica filtros digitais de média móvel de 4 pontos, melhorados através de deslocamento de bits (shift right) para concluir a divisão sem custos elevados de hardware. O Bloco de Controle utiliza uma Máquina de Estados Finitos (FSM) com histerese (liminar de 10 unidades) para assegurar decisões de movimentos estáveis. O código foi escrito em linguagem VHDL e sintetizado para FPGA no Software Cyclone II, na placa Altera DE2. Os dados obtidos demonstram a eficácia do filtro digital e da lógica de histerese na prevenção de acionamentos erráticos do motor.

Palavras chaves: energia solar; filtro digital; girassol; rastreador solar; RTL; FPGA.

I/Introdução

O sistema proposto visa o desenvolvimento de um sistema de controle digital para um rastreador solar de eixo único (Girassol), conforme mostrado na Figura 1. O objetivo é manter o painel perpendicular à incidência de luz para maximizar a geração de energia.

Os sistemas de rastreador solar usam sensores de luz que convertem a luz em valores de resistência; esses sensores são conhecidos como luminosidade de resistência (LDRs). Um LDR pode ser feito de sulfeto de cádmio (CdS) ou seleneto de cádmio (CdSe) e sua resistência diminui quando a luz é alta, e quando a luz é baixa, a resistência no LDR aumenta. Quando instalamos esses sistemas de Girassol em ambientes externos, os LDRs sofrem com ruídos e variações bruscas, como nuvens passageiras ou sombras, o que pode causar acionamentos erráticos do motor e desgaste mecânico.

Para resolver esses problemas associados com os sistemas de Girassol, implementamos uma arquitetura em hardware para projetar um controlador de nível de transferência entre registradores (RTL). Esse controlador integra um Filtro de Média Móvel no caminho de dados para visualizar as leituras e uma Máquina de Estados Finitos (FSM) com histerese para controlar a decisão de movimento, garantindo estabilidade e eficiência.

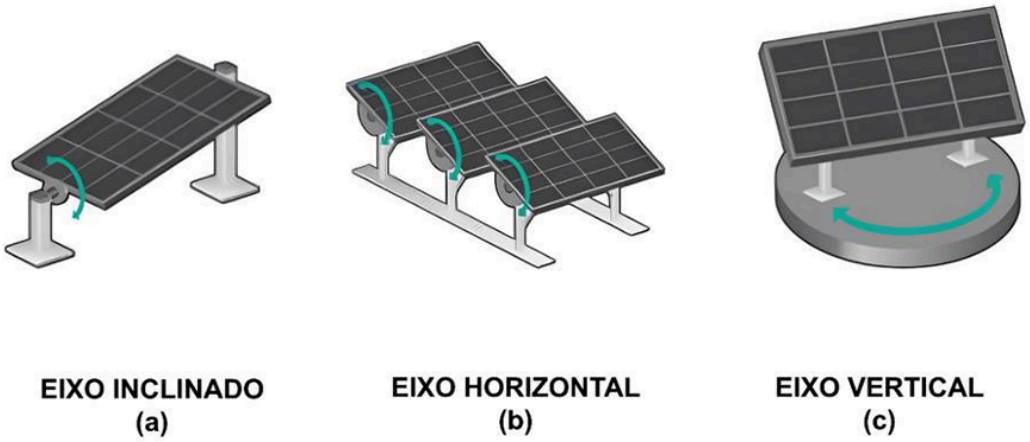

Figura 1: Exemplos de rastreador solar de eixo único. Fonte: Godoy (2019).

II/Arquitetura do Sistema

2.1)Máquina de Estados de Alto Nível

O sistema pode ser modelado como uma Máquina de estados de alto nível (HLSM) com três estados principais, conforme a Figura 2.

- PARADO: Estado de repouso (economia de energia).
- GIRA_DIR: Ativa o motor no sentido horário.
- GIRA_ESQ: Ativa o motor no sentido anti-horário.

A Tabela 1 lista as condições de transição entre os estados na Figura 2 e as correspondentes saídas de estado atual. Para simplificar as equações, denotamos R e L o valor filtrado pelo filtro direito e o valor filtrado pelo filtro esquerdo, respectivamente. Além disso, o sistema é sujeito a um limiar de histerese (Threshold=10). Temos duas variáveis na saída de cada estado para controlar o motor: motor_cw e motor_ccw. No estado PARADO, ambos motor_cw e motor_ccw são 0 pois o Girassol está parado. No estado GIRA_DIR, motor_cw=1 e motor_ccw=0 para mandar o Girassol girar para a direita. Finalmente, no estado GIRA_ESQ, motor_cw=0 e motor_ccw=1 para mandar o Girassol girar para a esquerda. O diagrama de controle do sistema é mostrado na Figura 3.

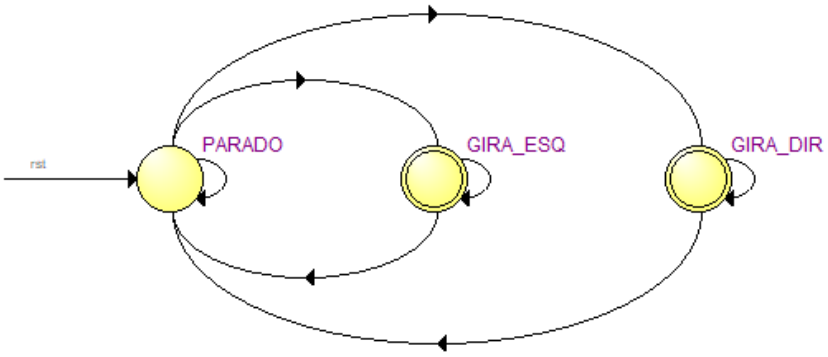

Figura 2: Máquina de estados de alto nível (HLSM) do sistema para um rastreador solar de eixo único (Girassol) com filtro digital. As condições de transição entre os estados e as correspondentes saídas do estado atual são mostradas na Tabela 1.

Estado Atual	Estado Final	Condição	Saídas
GIRA_DIR	PARADO	R <= L + Threshold	motor_cw=1; motor_ccw=0
GIRA_DIR	GIRA_DIR	[R <= L + Threshold]´	motor_cw=1; motor_ccw=0
GIRA_EQR	PARADO	L <= R + Threshold	motor_cw=0; motor_ccw=1
GIRA_EQR	GIRA_EQR	[L <= R + Threshold]´	motor_cw=0; motor_ccw=1
PARADO	PARADO	[R > L + Threshold]´ * [L > R + Threshold]´	motor_cw=0; motor_ccw=0
PARADO	GIRA_EQR	L > R+ Threshold	motor_cw=0; motor_ccw=0
PARADO	GIRA_DIR	[R > L + Threshold] * [L > R + Threshold]´	motor_cw=0; motor_ccw=0

Tabela 1: Condições de transição entre os estados do HLSM na Figura 2 e as correspondentes saídas de estado atual.

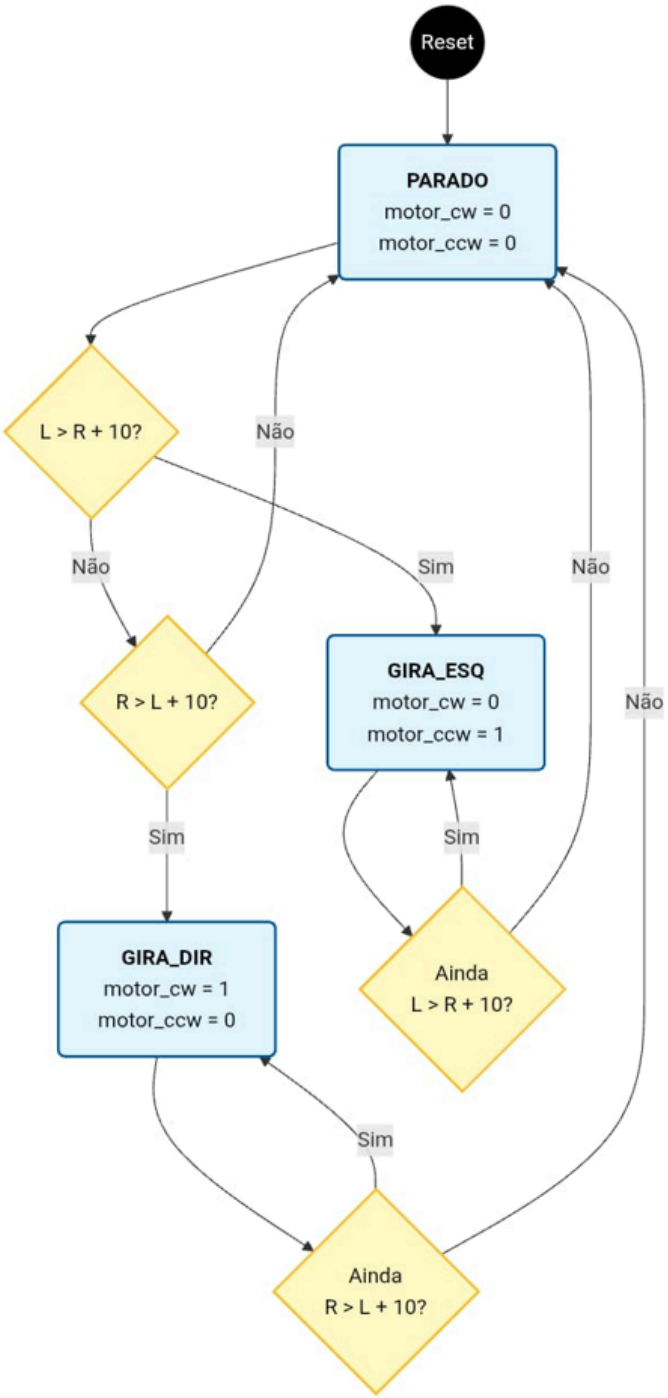

Figura 3: Diagrama de controle para a HLSM mostrada na Figura 2 do sistema de um rastreador solar de eixo único (Girassol) com filtro digital.

2.2)Máquina de Estados Finitos

O bloco de controle foi modelado como uma Máquina de Estados Finitos (FSM) com três estados principais conforme a Figura 4, que possui a mesma forma como a HLSM na Figura 2, mas com as condições de transição entre os estados finitos fornecidas pelo caminho de dados.

O bloco de controle é conectado com um caminho de dados que vamos discutir na Seção 2.3 e 2.4 que tem como saída as variáveis LessThan0, LessThan1, LessThan2 e LessThan3. O caminho de dados contém dois filtros digitais na direita e na esquerda que calculam os valores filtrados R e L, respectivamente. As variáveis LessThan0, LessThan1, LessThan2 e LessThan3 medem a relação entre L e R, sujeito a um limiar de histerese (Threshold). As transições ocorrem apenas quando a diferença entre os sensores filtrados supera Threshold = 10. Os valores lógicos dos variáveis de transição calculados pelo caminho de dados são:

LessThan0 = [L > R + Threshold]. (1)

LessThan1 = [R > L + Threshold]. (2)

LessThan2 = [L <= R + Threshold]. (3)

LessThan3 = [R <= L + Threshold]. (4)

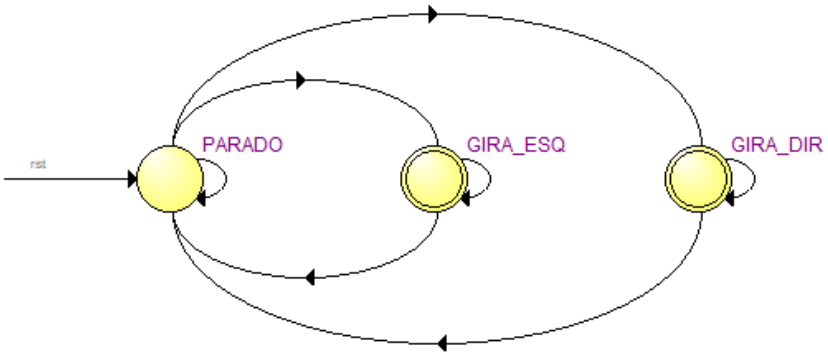

Figura 4: Máquina de estados finitos (FSM) do bloco de controle para um rastreador solar de eixo único (Girassol). As condições de transição entre os estados finitos são mostradas na Tabela 2. Esta FSM corresponde ao retângulo amarelo na direita da Figura 6.

Estado Atual	Estado Final	Condição	Saídas
GIRA_DIR	PARADO	LessThan3	motor_cw=1; motor_ccw=0
GIRA_DIR	GIRA_DIR	LessThan3´	motor_cw=1; motor_ccw=0
GIRA_EQR	PARADO	LessThan2	motor_cw=0; motor_ccw=1
GIRA_EQR	GIRA_EQR	LessThan2´	motor_cw=0; motor_ccw=1
PARADO	PARADO	LessThan1´*LessThan0´	motor_cw=0; motor_ccw=0
PARADO	GIRA_EQR	LessThan0	motor_cw=0; motor_ccw=0
PARADO	GIRA_DIR	LessThan1*LessThan0´	motor_cw=0; motor_ccw=0

Tabela 2: Condições de transição entre os estados finitos na Figura 4 e as correspondentes saídas de estado atual. As variáveis LessThan0, LessThan1, LessThan2 e LessThan3 são calculadas pelo caminho de dados usando as equações (1), (2), (3) e (4), respectivamente.

2.3)Caminho de Dados (Datapath) do Filtro Digital

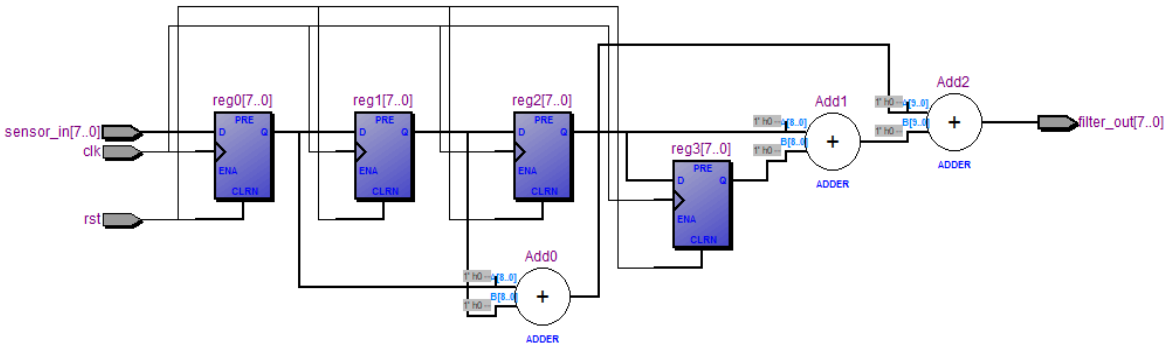

Figura 5: Caminho de Dados para um filtro digital de média móvel. Esta arquitetura corresponde a cada retângulo verde na esquerda da Figura 6. Note que a saída do somador Add2 é sujeita a divisão por 4 que foi implementada através de um deslocamento de bits para a direita (shift right) de 2 posições. Esse componente de deslocamento não é mostrado nesta arquitetura.

O Caminho de Dados implementa dois filtros digitais de média móvel de 4 pontos, a arquitetura de cada filtro é mostrado na Figura 5 que recebe os valores de um sensor LDR. No armazenamento de dados, utilizamos uma cadeia de registradores (pipeline) para armazenar as últimas 4 amostras do sensor. Na parte aritmética, a soma é realizada por uma árvore de somadores combinacionais. Finalmente, para otimizar o processamento de dados, a divisão por 4 foi implementada através de um deslocamento de bits para a direita (shift right) de 2 posições, eliminando a necessidade de um circuito divisor complexo e economizando áreas na FPGA. Note que este Caminho de Dados em si é um sistema RTL com predomínio de dados e não necessita um bloco de controle próprio; isto é parecido com o filtro de resposta finita ao impulso (FIR) (Vahid, 2008).

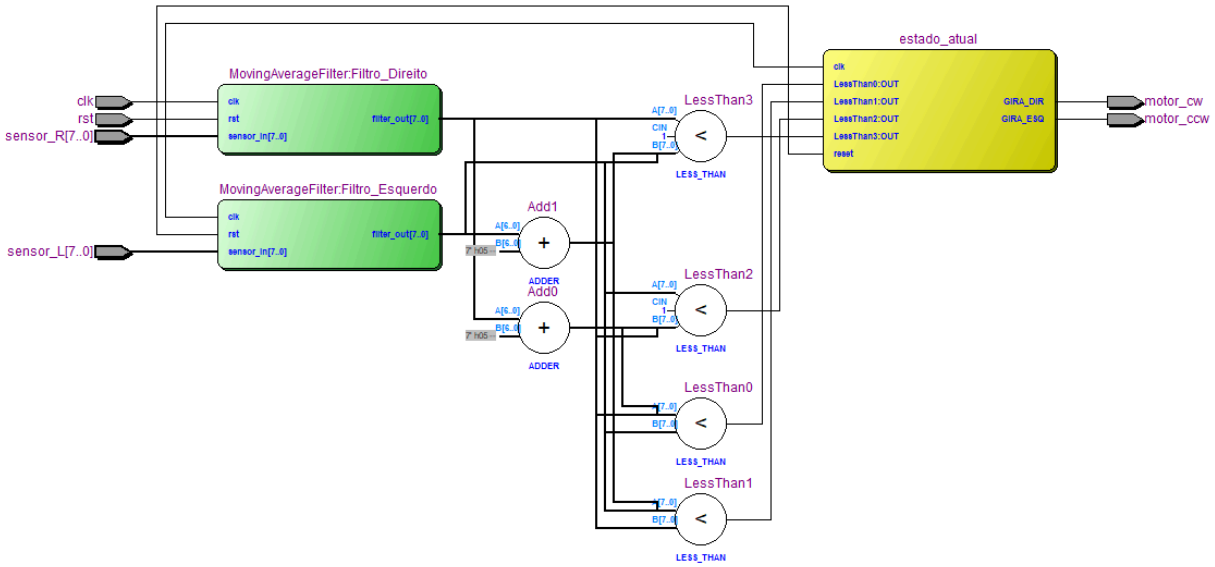

Figura 6: Sistema Completo em nível de Transferência entre Registradores que corresponde com a HLSM na Figura 2 e Tabela 1. O retângulo amarelo à direita da figura corresponde com a FSM na Figura 4 e Tabela 2. Cada retângulo verde corresponde com a arquitetura na Figura 5.

2.4)Sistema Completo em nível de Transferência entre Registradores

Figura 6 apresenta a síntese RTL de topo (Top Level). O bloco de controle é mostrado à direita em amarelo. Relembre da Seção 2.2 que ele recebe quatro entradas como saídas do caminho de dados: LessThan0, LessThan1, LessThan2, LessThan3. Essas variáveis controlam a transição entre os estados e as saídas do bloco de controle.

O caminho de dados é mostrado no lado esquerdo do bloco de controle. Ele possui dois filtros digitais de média móvel (mostrados em verde). O filtro direito recebe dados do sensor LDR a direita, enquanto o filtro esquerdo recebe dados do sensor LDR a esquerda. A arquitetura de cada filtro é mostrada na Figura 5. Note que os dados do sensor tem oito bits porque o valor de resistência do sensor LDR é entre 0 e 220 Ohms, logo precisamos de 8 bits (que tem valores entre 0 e 255) para representar essa resistência.

Os filtros digitais calculam os valores filtrados dos sensores que chamamos R (sensor a direita) e L (sensor a esquerda). Depois disso, esses valores filtrados são passados por um outro caminho de dado que contém somadores e comparadores para calcular os valores das variáveis LessThan0, LessThan1, LessThan2 e LessThan3 e retorna-as para o bloco de controle conforme discutimos na Seção 2.2.

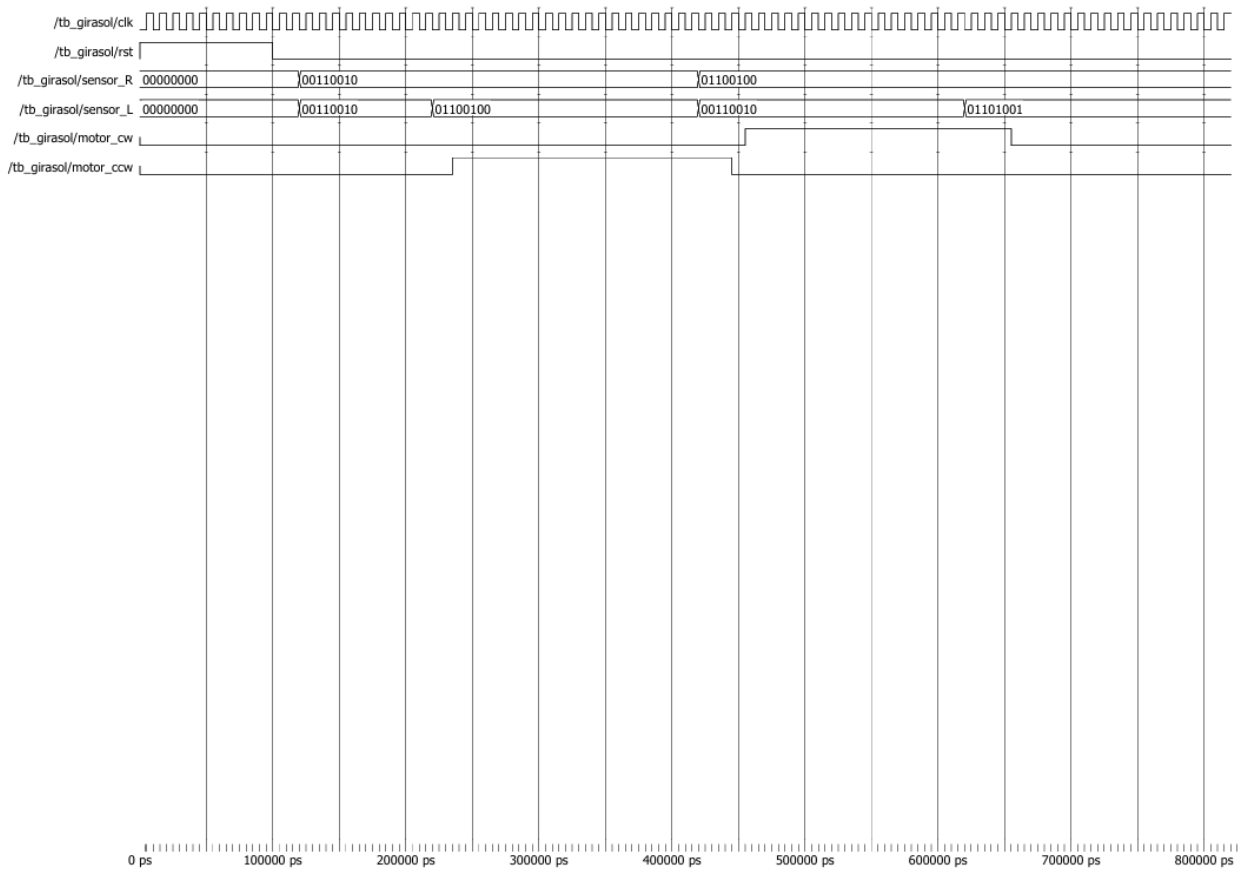

Figura 7: Resultados de simulação no ModelSim.

III/Resultados de Simulação

A Figura 7 mostra os resultados de Simulação no ModelSim. Note que sensor_R e sensor_L são valores filtrados do sensor à direita R e do sensor à esquerda L, respectivamente. Observe-se no gráfico cinco regimes de forma de onda.

(i) No intervalo entre 0 e 100ns: O sistema inicia em reset.

(ii) No intervalo entre 100 e 240ns: Os valores dos dois sensores são iguais a 50.

(iii) No intervalo entre 240ns e 450ns: Realizamos o teste de Detecção à Esquerda. Ao elevar o sensor_L para 100 (mantendo sensor_R em 50), a saída motor_ccw é ativada corretamente.

(iv) No intervalo entre 450ns e 650ns: Realizamos o teste de Detecção à Direita. Ao inverter a condição, o sistema aciona motor_cw.

(v) No intervalo entre 650ns e 800ns: Realizamos o teste de Histerese. Simulou-se uma diferença de apenas 5 unidades (sensorR=100 e sensorL=105). Como é menor que o limiar (Threshold = 10), o motor motor_ccw permaneceu desligado, comprovando a imunidade a ruídos pequenos.

IV/Implementação em FPGA

O projeto foi sintetizado para o dispositivo Cyclone II (EP2C35F672) da placa Altera DE2. O mapeamento foi feito conforme as regras seguintes. Primeiramente, os bits mais significativos (MSB) dos sensores foram mapeados nas chaves SW17 e SW0. Segundamente, os sinais de controle do motor foram conectados aos LEDs Verdes (LEDG1 e LEDG0). O resultado esperado é que os testes em hardware confirmaram o comportamento observado na simulação.

V/Código VHDL

O código fonte completo (arquivos .vhd) encontra-se no Apêndice no final do documento.

VI/Dificuldades e Soluções

6.1)Restrição de Codificação RTL Estrita

Dificuldade: A maior dificuldade foi adaptar o pensamento lógico para não utilizar processos comportamentais (process) na lógica aritmética e de decisão. Estávamos acostumados a descrever o comportamento (o que o circuito faz), e tivemos que mudar para descrever a estrutura (como o circuito é ligado).

Solução: Desenhamos primeiro o diagrama de blocos no papel para identificar exatamente onde precisávamos de somadores e comparadores. Utilizamos a construção WHEN...ELSE e WITH...SELECT para garantir que o sintetizador gerasse lógica puramente combinacional.

6.2)Gerenciamento de Tipos e Largura de Bits (Bit-width)

Dificuldade: Na implementação do filtro, percebemos que somar quatro números de 8 bits (máximo 255) poderia resultar em um número de 10 bits (1020), causando estouro (overflow). Além disso, o VHDL é fortemente tipado, gerando erros ao misturar std_logic_vector com unsigned.

Solução: Utilizamos a função resize() da biblioteca numeric_std para expandir os sinais para 9 e 10 bits antes das somas. Para a divisão por 4, aplicamos a técnica de descartar os 2 bits menos significativos (LSBs), o que economiza hardware comparado a um divisor real.

6.3)Configuração do Ambiente de Simulação

Dificuldade: Houve problemas na integração entre o Quartus II Web Edition e o ModelSim, pois o caminho do executável não estava configurado automaticamente, impedindo a abertura da simulação.

Solução: Configuramos manualmente o caminho de arquivo (path) da ferramenta EDA nas opções do Quartus para apontar para a pasta win32aloem correta, permitindo a execução do Testbench.

VII/Conclusão

O projeto cumpriu com êxito o objetivo de desenvolver um controlador digital eficiente para o rastreador solar “Girassol”. A implementação da arquitetura em nível RTL, junto com os filtros de média móvel e uma Máquina de Estados com histerese, foi determinante para o desempenho do sistema. Essa abordagem mostrou-se capaz de filtrar oscilações de leitura dos sensores LDR e garantir o acionamento do motor. Os resultados obtidos nos testes da simulação e na síntese na FPGA ratificam o funcionamento do sistema, assegurando que a solução sólida previne o desgaste mecânico prematuro e garante o posicionamento correto do painel.

REFERÊNCIAS

Godoy, L. G. K. Projeto de um rastreador solar digital de um eixo comparado a um rastreador analógico. **Escola Politécnica da Universidade Federal do Rio de Janeiro - UFRJ**, Trabalho de Conclusão de Curso de Graduação. Rio de Janeiro, p. 25. 2019.

VAHID, Frank. **Sistemas Digitais: projeto, otimização e HDLs**. Tradução Anatólio Laschuk. Porto Alegre: Artmed, pp. 264 - 269. 2008.

Link do repositório

https://github.com/DIGAOZX/sistemas_digitais_girassol

Apêndice

Código VHDL

A.1)Caminho de Dados (Datapath) do Filtro Digital

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;


entity MovingAverageFilter is

    Port (

        clk : in STD_LOGIC;

        rst : in STD_LOGIC;

        sensor_in : in STD_LOGIC_VECTOR(7 downto 0);

        filter_out : out STD_LOGIC_VECTOR(7 downto 0)

    );

end MovingAverageFilter;


architecture RTL of MovingAverageFilter is

    -- Sinais internos para guardar os valores (Registradores)
```

```
    signal reg0, reg1, reg2, reg3 : unsigned(7 downto 0);

    -- Sinais internos para fazer a soma (precisa ser maior para não estourar)
    signal sum_stage1_a : unsigned(8 downto 0);
    signal sum_stage1_b : unsigned(8 downto 0);
    signal total_sum    : unsigned(9 downto 0);

begin
    -- PARTE 1: Memória (Registradores)
    -- Aqui guardamos as últimas 4 leituras do sensor
    process(clk, rst)
    begin
        if rst = '1' then
            reg0 <= (others => '0');
            reg1 <= (others => '0');
            reg2 <= (others => '0');
            reg3 <= (others => '0');
        elsif rising_edge(clk) then
            reg0 <= unsigned(sensor_in); -- Entra valor novo
            reg1 <= reg0;              -- Passa pro lado ->
            reg2 <= reg1;              -- Passa pro lado ->
            reg3 <= reg2;              -- Passa pro lado ->
        end if;
    end process;

    -- PARTE 2: Cálculo (Lógica Combinacional)
```

```
    -- Soma tudo. Note que NÃO usamos "process" aqui.
    sum_stage1_a <= resize(reg0, 9) + resize(reg1, 9);
    sum_stage1_b <= resize(reg2, 9) + resize(reg3, 9);
    total_sum    <= resize(sum_stage1_a, 10) + resize(sum_stage1_b, 10);

    -- PARTE 3: Divisão por 4
    -- Pegamos os bits mais significativos (descartamos os 2 últimos)
    filter_out <= std_logic_vector(total_sum(9 downto 2));

end RTL;
```

A.2)Sistema Completo em nível de Transferência entre Registradores

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Girasol_RTL is
    Port (
        clk       : in STD_LOGIC;
        rst       : in STD_LOGIC;
        sensor_L  : in STD_LOGIC_VECTOR(7 downto 0); -- Entrada Sensor Esquerdo
        sensor_R  : in STD_LOGIC_VECTOR(7 downto 0); -- Entrada Sensor Direito
        motor_cw  : out STD_LOGIC; -- Girar Horário (Clockwise)
        motor_ccw : out STD_LOGIC  -- Girar Anti-Horário
    );
end Girasol_RTL;
```

```
architecture RTL of Girasol_RTL is

    -- 1. Declaramos o COMPONENTE que criamos antes (o Filtro)
    component MovingAverageFilter
        Port (
            clk : in STD_LOGIC;
            rst : in STD_LOGIC;
            sensor_in : in STD_LOGIC_VECTOR(7 downto 0);
            filter_out : out STD_LOGIC_VECTOR(7 downto 0)
        );
    end component;

    -- Sinais para conectar as saídas dos filtros
    signal L_filtrado, R_filtrado : std_logic_vector(7 downto 0);
    signal L_unsigned, R_unsigned : unsigned(7 downto 0);

    -- Definição da FSM (Máquina de Estados)
    type state_type is (PARADO, GIRA_ESQ, GIRA_DIR);
    signal estado_atual, proximo_estado : state_type;

    -- Constante de Histerese (Zona Morta)
    -- O motor só mexe se a diferença for maior que 10 (para evitar tremedeira)
    constant THRESHOLD : unsigned(7 downto 0) := to_unsigned(10, 8);

begin
```

```
-- ========================================================
-- BLOCO 1: INSTANCIAÇÃO DO DATAPATH (Os Filtros)
-- ========================================================
-- Filtro do Sensor Esquerdo
Filtro_Esquerdo: MovingAverageFilter
port map (
    clk => clk,
    rst => rst,
    sensor_in  => sensor_L,
    filter_out => L_filtrado
);

-- Filtro do Sensor Direito
Filtro_Direito: MovingAverageFilter
port map (
    clk => clk,
    rst => rst,
    sensor_in  => sensor_R,
    filter_out => R_filtrado
);

-- Conversão para unsigned para fazer contas
L_unsigned <= unsigned(L_filtrado);
R_unsigned <= unsigned(R_filtrado);
```

```
-- ========================================================
-- BLOCO 2: MÁQUINA DE ESTADOS (FSM)
-- ========================================================
-- Processo A: Memória de Estado
process(clk, rst)
begin
    if rst = '1' then
        estado_atual <= PARADO;
    elsif rising_edge(clk) then
        estado_atual <= proximo_estado;
    end if;
end process;

-- Processo B: Lógica de Próximo Estado
-- Aqui decidimos o que fazer baseados nos sensores
process(estado_atual, L_unsigned, R_unsigned)
begin
    -- Valor padrão: fica como está
    proximo_estado <= estado_atual;

    case estado_atual is
        when PARADO =>
            if L_unsigned > (R_unsigned + THRESHOLD) then
                proximo_estado <= GIRA_ESQ; -- Sol ta na esquerda
            elsif R_unsigned > (L_unsigned + THRESHOLD) then
                proximo_estado <= GIRA_DIR; -- Sol ta na direita
```

```
            else
                proximo_estado <= PARADO;
            end if;

        when GIRA_ESQ =>
            -- Se já equilibrou (diferença pequena), para.
            if L_unsigned <= (R_unsigned + THRESHOLD) then
                proximo_estado <= PARADO;
            else
                proximo_estado <= GIRA_ESQ;
            end if;

        when GIRA_DIR =>
            -- Se já equilibrou, para.
            if R_unsigned <= (L_unsigned + THRESHOLD) then
                proximo_estado <= PARADO;
            else
                proximo_estado <= GIRA_DIR;
            end if;
    end case;
end process;

-- ============================================================
-- BLOCO 3: SAÍDAS
-- ============================================================
with estado_atual select
```

```
        motor_cw <= '1' when GIRA_DIR,
                    '0' when others;


    with estado_atual select
        motor_ccw <= '1' when GIRA_ESQ,
                     '0' when others;


end RTL;
```