Aluno do Embarcatech_37 no IFMA

Nome: Manoel Felipe Costa Furtado

Matrícula: 20251RSE.MTC0086

Residência Profissional em FPGA – Turma DELTA - 2025.2

Atividade – Referente ao capítulo 01 da unidade 06

Tema do Capítulo – Linguagem VHDL: Máquina de Estado

Prazo dia 05/10/2025 as 23:59

Objetivo: Desenvolver a capacidade de projetar, modelar, simular e implementar circuitos digitais por meio da linguagem VHDL (VHSIC Hardware Description Language), promovendo a compreensão dos conceitos de lógica digital e sua aplicação prática no desenvolvimento de sistemas embarcados e dispositivos programáveis, como FPGAs.

Enunciado: Implementar uma máquina de estados para controlar a abertura e fechamento de uma porta automática com sensor de presença e botão fechar_manual.

Ação	Função
1	Definir Entradas: • clk — clock do sistema • rst_n — reset assíncrono ativo-baixo • sensor — detecta presença (1 = pessoa detectada) • fechar_manual — botão de fechamento antecipado • fim_curso_aberta — sensor indica porta totalmente aberta • fim_curso_fechada — sensor indica porta totalmente fechada Definir Saídas: • motor_abrir — ativa o motor no sentido de abrir • motor_fechar — ativa o motor no sentido de fechar
2	Estados: 1) FECHADA — porta fechada, motores desligados 2) ABRINDO — motor_abrir ligado até porta abrir completamente 3) ABERTA — porta aberta, aguarda tempo T_ABERTA sem presença 4) FECHANDO — motor_fechar ligado até porta fechar completamente
3	Regras de Transição: • FECHADA → ABRINDO quando sensor=1 • ABRINDO → ABERTA quando fim_curso_aberta=1 • ABERTA → FECHANDO quando T_ABERTA expira e sensor=0, ou fechar_manual=1 • FECHANDO → FECHADA quando fim_curso_fechada=1 • Durante ABERTA, se sensor=1, reinicia temporizador T_ABERTA

Requisitos de Implementação

1) Fazer o diagrama de estados com as condições de transição.
2) Implementar a FSM em VHDL usando estilo Moore.
3) Criar testbench simulando pessoas chegando, saindo e fechamento manual.
4) Validar os tempos T_ABERTA, T_ABRINDO e T_FECHANDO na simulação.

Solução

- GitHub: Segunda_Fase_FPGA/Unidade_06/Cap_01
- Link do Vídeo: https://youtu.be/1aZuxmzPFmM
- Link do Código Completo: Unid_06_Cap_01.zip

1) Fazer o diagrama de estados com as condições de transição.

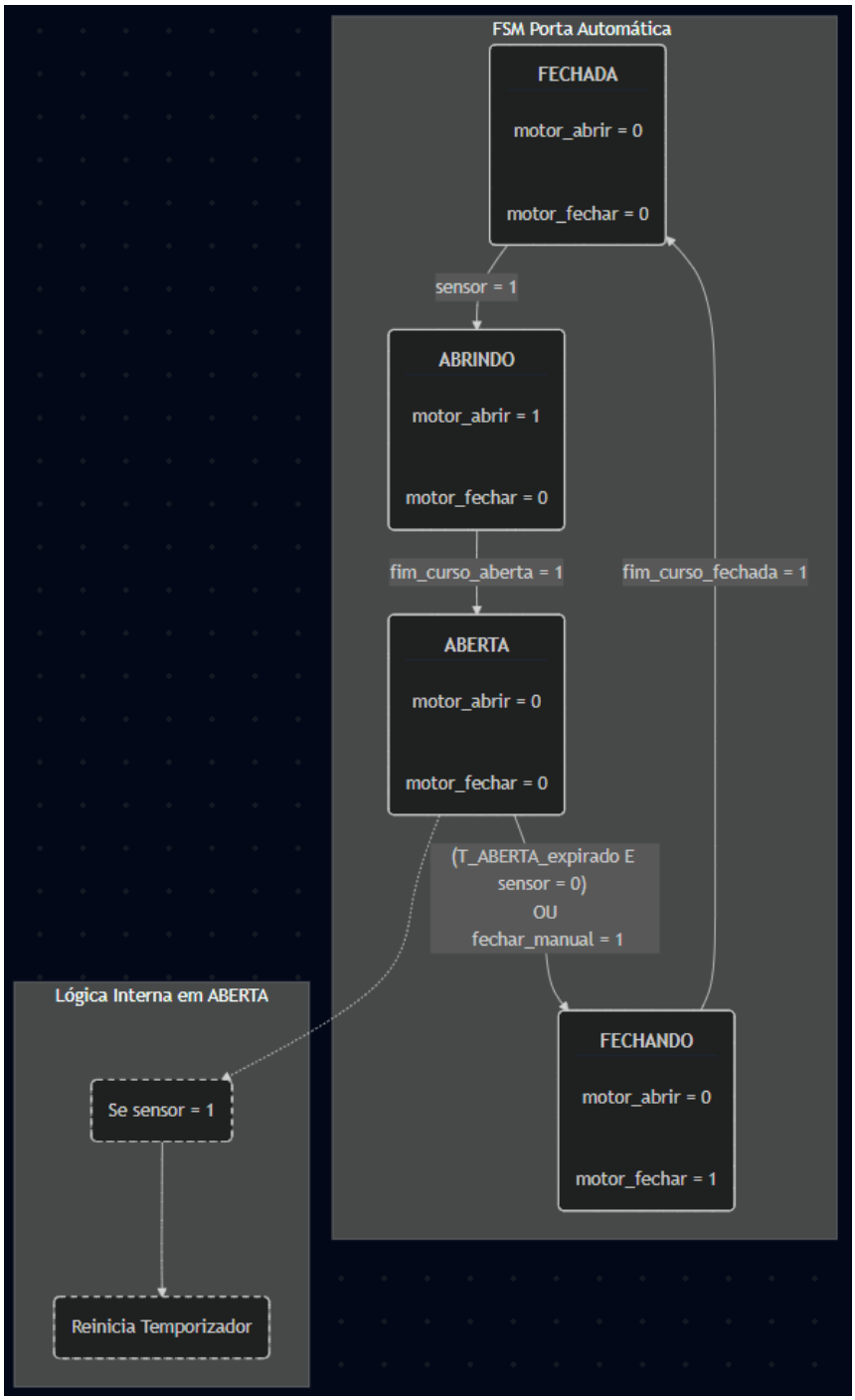
A FSM é do tipo Moore, o que significa que as saídas dependem unicamente do estado atual. O diagrama abaixo ilustra os quatro estados e as condições que causam a transição entre eles.

Usei uma linguagem chamada Mermaid.js. O objetivo é justamente descrever diagramas e fluxogramas usando texto, que podem ser convertidos (ou "renderizados") em uma imagem gráfica. Usando a extensão do vscode Markdown Preview Mermaid Support pode gerar um arquivo(.md) para gerar o gráfico sem precisar ir no site https://mermaid.live/ para visualizar.

A seguir, essas duas tabelas que ajudam no entendimento.

Tabela de Saídas (Estilo Moore)		
Estado Atual	motor_abrir	motor_fechar
FECHADA	0	0
ABRINDO	1	0
ABERTA	0	0
FECHANDO	0	1

Tabela de Transição de Estados (Próximo Estado)							
Estado Atual	sensor	fechar_manual	fim_curso_aberta	fim_curso_fechada	t_expirado	Próximo Estado	Condição Lógica da Transição
FECHADA	1	X	X	X	X	ABRINDO	sensor = 1
FECHADA	0	X	X	X	X	FECHADA	senão (permanece no estado)
ABRINDO	X	X	1	X	X	ABERTA	fim_curso_aberta = 1
ABRINDO	X	X	0	X	1	ABERTA	t_expirado = 1 (Timeout de segurança)
ABRINDO	X	X	0	X	0	ABRINDO	senão (permanece no estado)
ABERTA	X	1	X	X	X	FECHANDO	fechar_manual = 1
ABERTA	0	0	X	X	1	FECHANDO	sensor = 0 E t_expirado = 1
ABERTA	X	0	X	X	0	ABERTA	senão (permanece no estado)
ABERTA	1	0	X	X	X	ABERTA	sensor = 1 (reinicia o timer)
FECHANDO	X	X	X	1	X	FECHADA	fim_curso_fechada = 1
FECHANDO	X	X	X	0	1	FECHADA	t_expirado = 1 (Timeout de segurança)
FECHANDO	X	X	X	0	0	FECHANDO	senão (permanece no estado)



```mermaid
%% ================================================================================
%% Diagrama da Máquina de Estados Finitos (FSM) para o Controle de Porta Automática
%% Estilo de Implementação: Moore
%% ================================================================================

%% Declara o tipo de diagrama como um gráfico ('graph') e a orientação padrão como Top-Down ('TD').
graph TD
    %% Agrupa todos os elementos principais da FSM sob um título comum para organização.
    subgraph "FSM Porta Automática (Moore)"
        %% Altera a direção do layout para Left-to-Right ('LR'), que é mais comum para FSMs.
        direction LR

        %% ---------------------------------------------------------------
        %% SEÇÃO 1: Definição dos Estados da FSM
        %% Cada estado é um nó no diagrama e define as saídas (motores) que estarão ativas.
        %% ---------------------------------------------------------------

        %% ESTADO 1 (INICIAL): A porta está parada e totalmente fechada. Ambos os motores estão desligados.
        S_FECHADA("
            <b>FECHADA</b>
            <hr> motor_abrir = 0
            <br> motor_fechar = 0
        ")

        %% ESTADO 2: O ciclo de abertura foi iniciado. O motor de abrir está ligado.
        S_ABRINDO("
            <b>ABRINDO</b>
            <hr> motor_abrir = 1
            <br> motor_fechar = 0
        ")

        %% ESTADO 3: A porta está parada e totalmente aberta. Os motores estão desligados e o temporizador T_ABERTA é ativado.
        S_ABERTA("
            <b>ABERTA</b>
            <hr> motor_abrir = 0
            <br> motor_fechar = 0
        ")

        %% ESTADO 4: O ciclo de fechamento foi iniciado. O motor de fechar está ligado.
        S_FECHANDO("
            <b>FECHANDO</b>
            <hr> motor_abrir = 0
            <br> motor_fechar = 1
        ")
```

```
        %% ------------------------------------------------------------------
        %% SEÇÃO 2: Definição das Transições entre Estados
        %% As setas representam as condições que fazem a FSM mudar de um estado para outro.
        %% ------------------------------------------------------------------

        %% TRANSIÇÃO (REGRA 3.1): Se a porta está FECHADA e o sensor detecta uma pessoa, inicia-se o ciclo de abertura.
        S_FECHADA -- "sensor = 1" --> S_ABRINDO

        %% TRANSIÇÃO (REGRA 3.2): Durante a abertura, ao atingir o fim de curso, o motor para e a porta é considerada ABERTA.
        S_ABRINDO -- "fim_curso_aberta = 1" --> S_ABERTA

        %% TRANSIÇÃO (REGRA 3.3): Quando ABERTA, a porta começa a fechar se o tempo expirar (sem ninguém no sensor) OU se o fechamento manual for acionado.
        S_ABERTA -- "(T_ABERTA_expirado E sensor = 0) <br> OU <br> fechar_manual = 1" --> S_FECHANDO

        %% TRANSIÇÃO (REGRA 3.4): Durante o fechamento, ao atingir o fim de curso, o motor para e a porta volta ao estado FECHADA.
        S_FECHANDO -- "fim_curso_fechada = 1" --> S_FECHADA

    end

    %% ------------------------------------------------------------------
    %% SEÇÃO 3: Representação de Ações Internas (Não são Estados)
    %% ------------------------------------------------------------------

    %% BLOCO VISUAL (REGRA 3.5): Este bloco não é um estado da FSM. Ele representa uma AÇÃO que ocorre DENTRO do estado ABERTA.
    subgraph "Lógica Interna em ABERTA"
        direction LR
        A_LOGIC("Se sensor = 1")
        A_ACTION("Reinicia Temporizador")
        A_LOGIC --> A_ACTION
    end

    %% LIGAÇÃO VISUAL (REGRA 3.5): A linha pontilhada (-.->) mostra que a detecção de uma pessoa no estado ABERTA dispara a lógica de reiniciar o timer, SEM mudar de estado.
    S_ABERTA -.-> A_LOGIC

    %% ------------------------------------------------------------------
    %% SEÇÃO 4: Estilização Visual do Diagrama
    %% ------------------------------------------------------------------
    %% Aplica um estilo de borda tracejada aos nós da "Lógica Interna" para diferenciá-los visualmente dos estados reais da FSM.
    style A_LOGIC stroke-dasharray: 5 5
    style A_ACTION stroke-dasharray: 5 5
```

2) Implementar a FSM em VHDL usando estilo Moore.

```vhdl
-- ================================================================================
-- Atividade: Unidade 06, Capítulo 01
-- Autor:     Manoel Felipe Costa Furtado
-- Data:      05/10/2025
-- Arquivo:   porta_automatica_fsm.vhd
-- Versão:    1.0
-- Descrição: Máquina de estados finitos (FSM) estilo Moore que implementa a lógica
--            de controle para uma porta automática. A FSM gerencia os motores de
--            abertura e fechamento com base em entradas como sensor de presença,
--            sensores de fim de curso (porta totalmente aberta/fechada) e um
--            comando para fechamento manual. Inclui temporizadores de segurança.
-- ================================================================================

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity porta_automatica_fsm is
    generic (
        -- Frequência do clock do sistema em Hertz (Hz).
        G_CLK_FREQ      : integer := 1000; -- Valor para simulação (1 KHz)
        -- Tempo que a porta deve permanecer aberta após a saída do usuário, em milissegundos.
        G_T_ABERTA_MS   : integer := 5000; -- Valor padrão de 5000 ms (5s)
        -- Timeout de segurança em segundos. Se a porta não atingir um fim de curso
        -- (aberta ou fechada) neste tempo, o motor é desligado para evitar danos.
        G_T_TIMEOUT_S   : integer := 10
    );
    port (
        -- Entradas de controle
        clk               : in  std_logic; -- Clock do sistema.
        rst_n             : in  std_logic; -- Reset assíncrono ativo em baixo.
        sensor            : in  std_logic; -- '1' se há presença, '0' caso contrário.
        fechar_manual     : in  std_logic; -- Pulso em '1' para forçar o fechamento.
        fim_curso_aberta  : in  std_logic; -- '1' quando a porta está totalmente aberta.
        fim_curso_fechada : in  std_logic; -- '1' quando a porta está totalmente fechada.

        -- Saídas de acionamento
        motor_abrir       : out std_logic; -- '1' para acionar o motor de abertura.
        motor_fechar      : out std_logic; -- '1' para acionar o motor de fechamento.

        -- Saída de depuração (para simulação)
        -- Permite visualizar o nome do estado atual em um simulador de formas de onda.
        estado_debug      : out string(1 to 8)
    );
end entity porta_automatica_fsm;
```

```
architecture rtl of porta_automatica_fsm is

    -- Define os quatro estados possíveis da máquina de estados.
    type t_estado is (FECHADA, ABRINDO, ABERTA, FECHANDO);

    -- Sinais para armazenar o estado atual e o próximo estado calculado.
    signal estado_atual, proximo_estado : t_estado;
    -- Sinal para registrar o estado do ciclo de clock anterior. Usado para detectar
    -- transições (ex: `estado_prev /= estado_atual`), o que é crucial para
    -- inicializar lógicas (como contadores) na entrada de um novo estado.
    signal estado_prev   : t_estado;

    -- Constantes para os limites dos contadores, calculados a partir dos genéricos.
    -- Isso torna o design parametrizável e a lógica mais legível.
    constant C_COUNT_ABERTA  : integer := (G_CLK_FREQ / 1000) * G_T_ABERTA_MS; -- Limite para o tempo em ABERTA.
    constant C_COUNT_TIMEOUT : integer := G_CLK_FREQ * G_T_TIMEOUT_S;       -- Limite para os timeouts de segurança.

    -- Contadores dedicados para cada estado que depende de tempo. Usar contadores
    -- separados evita que a contagem de um estado "vaze" para o outro (carry-over),
    -- tornando o design mais robusto e fácil de depurar.
    signal cnt_abrindo   : integer range 0 to C_COUNT_TIMEOUT := 0; -- Temporizador de segurança para o estado
ABRINDO.
    signal cnt_aberta    : integer range 0 to C_COUNT_ABERTA  := 0; -- Temporizador para manter a porta aberta.
    signal cnt_fechando  : integer range 0 to C_COUNT_TIMEOUT := 0; -- Temporizador de segurança para o estado
FECHANDO.

    -- Flags que sinalizam quando um temporizador atingiu seu limite.
    -- Simplificam a lógica de transição de estado.
    signal to_abrindo    : std_logic; -- Flag de timeout para ABRINDO.
    signal to_aberta     : std_logic; -- Flag de tempo esgotado para ABERTA.
    signal to_fechando   : std_logic; -- Flag de timeout para FECHANDO.

begin

    -- Processo combinacional para formatar o nome do estado atual para depuração.
    -- Preenche com espaços para manter um tamanho fixo, facilitando a visualização.
    process(estado_atual)
    begin
        case estado_atual is
            when FECHADA  => estado_debug <= "FECHADA ";
            when ABRINDO  => estado_debug <= "ABRINDO ";
            when ABERTA   => estado_debug <= "ABERTA  ";
            when FECHANDO => estado_debug <= "FECHANDO";
        end case;
    end process;
```

```vhdl
-- ============================================================================
-- PROCESSO 1: REGISTRADOR DE ESTADO (Lógica Síncrona)
-- ============================================================================
-- Este processo representa a "memória" da FSM. Na borda de subida do clock,
-- ele atualiza o estado atual com o valor calculado pela lógica de próximo
-- estado. Também armazena o estado antigo em `estado_prev` para detecção de transição.
process(clk, rst_n)
begin
    if rst_n = '0' then
        estado_atual <= FECHADA;
        estado_prev  <= FECHADA;
    elsif rising_edge(clk) then
        estado_prev  <= estado_atual;
        estado_atual <= proximo_estado;
    end if;
end process;

-- ============================================================================
-- PROCESSO 2: LÓGICA DE PRÓXIMO ESTADO (Lógica Combinacional)
-- ============================================================================
-- Este bloco puramente combinacional calcula qual será o próximo estado da
-- máquina com base no estado atual e nas entradas do sistema.
process(estado_atual, sensor, fechar_manual, fim_curso_aberta, fim_curso_fechada,
        to_abrindo, to_aberta, to_fechando)
begin
    proximo_estado <= estado_atual; -- Comportamento padrão: permanecer no estado atual.
    case estado_atual is
        when FECHADA =>
            -- Se a porta está fechada e o sensor detecta presença, o próximo estado é ABRINDO.
            if sensor = '1' then
                proximo_estado <= ABRINDO;
            end if;
        when ABRINDO =>
            -- Se a porta está abrindo e atinge o fim de curso de abertura OU se o tempo
            -- de segurança estourar, o próximo estado é ABERTA (para parar o motor).
            if fim_curso_aberta = '1' or to_abrindo = '1' then
                proximo_estado <= ABERTA;
            end if;
        when ABERTA =>
            -- Se o botão de fechamento manual for pressionado, a transição para FECHANDO tem prioridade.
            if fechar_manual = '1' then
                proximo_estado <= FECHANDO;
            -- Caso contrário, se não houver mais ninguém (sensor='0') e o tempo de espera terminar,
            -- a porta começa a fechar automaticamente.
            elsif sensor = '0' and to_aberta = '1' then
                proximo_estado <= FECHANDO;
            end if;
        when FECHANDO =>
```

```
            -- Se a porta está fechando e atinge o fim de curso de fechamento OU se o tempo
            -- de segurança estourar, ela retorna ao estado inicial FECHADA.
            if fim_curso_fechada = '1' or to_fechando = '1' then
                proximo_estado <= FECHADA;
            end if;
    end case;
end process;

-- ============================================================================
-- PROCESSO 3: LÓGICA DE SAÍDA (Estilo Moore)
-- ============================================================================
-- As saídas dependem *apenas* do estado atual, caracterizando uma FSM de Moore.
-- Isso evita glitches nas saídas que podem ocorrer em FSMs de Mealy.
-- Atribuições concorrentes são uma forma compacta e clara de descrever essa lógica.
motor_abrir  <= '1' when estado_atual = ABRINDO  else '0';
motor_fechar <= '1' when estado_atual = FECHANDO else '0';

-- ============================================================================
-- PROCESSO 4: LÓGICA DOS TEMPORIZADORES (Síncrona)
-- ============================================================================
-- Este processo gerencia a contagem de tempo para os estados que necessitam.
process(clk, rst_n)
begin
    if rst_n = '0' then
        cnt_abrindo  <= 0;
        cnt_aberta   <= 0;
        cnt_fechando <= 0;
    elsif rising_edge(clk) then
        -- Lógica de Reset dos Contadores: Zera um contador específico no exato ciclo de
        -- clock em que a FSM entra no estado correspondente. Isso é feito detectando a
        -- transição de `estado_prev` para `estado_atual`. Garante que a contagem sempre comece do zero.
        if estado_prev /= estado_atual then
            case estado_atual is
                when ABRINDO  => cnt_abrindo  <= 0;
                when ABERTA   => cnt_aberta   <= 0;
                when FECHANDO => cnt_fechando <= 0;
                when others   => null; -- Nos outros estados, nenhuma ação de reset é necessária na entrada.
            end case;
        end if;

        -- Lógica de Incremento dos Contadores: A contagem só progride se a FSM
        -- estiver no estado correspondente e a condição de contagem for atendida.
        case estado_atual is
            when ABRINDO =>
                -- Enquanto estiver abrindo, incrementa o contador de timeout de segurança.
                if cnt_abrindo < C_COUNT_TIMEOUT then
                    cnt_abrindo <= cnt_abrindo + 1;
                end if;
```

```
                when ABERTA =>
                    -- Se uma nova presença for detectada, o contador é zerado para manter a porta aberta.
                    if sensor = '1' then
                        cnt_aberta <= 0;
                    -- Se não houver presença, o contador progride até o limite.
                    elsif cnt_aberta < C_COUNT_ABERTA then
                        cnt_aberta <= cnt_aberta + 1;
                    end if;
                when FECHANDO =>
                    -- Enquanto estiver fechando, incrementa o contador de timeout de segurança.
                    if cnt_fechando < C_COUNT_TIMEOUT then
                        cnt_fechando <= cnt_fechando + 1;
                    end if;
                when others => -- Inclui o estado FECHADA.
                    -- Para segurança, garante que os contadores sejam zerados quando não estão em uso.
                    cnt_abrindo  <= 0;
                    cnt_aberta   <= 0;
                    cnt_fechando <= 0;
            end case;
        end if;
    end process;

    -- Atribuições concorrentes para gerar os flags de timeout.
    -- Um flag é ativado ('1') quando o contador correspondente atinge ou excede seu limite.
    to_abrindo  <= '1' when cnt_abrindo  >= C_COUNT_TIMEOUT else '0';
    to_aberta   <= '1' when cnt_aberta   >= C_COUNT_ABERTA  else '0';
    to_fechando <= '1' when cnt_fechando >= C_COUNT_TIMEOUT else '0';

end architecture rtl;
```

3) Criar testbench simulando pessoas chegando, saindo e fechamento manual.

```
-- ================================================================================
-- Testbench Rápido: tb_porta_automatica_fast.vhd
-- Autor:     Manoel Felipe Costa Furtado
-- Data:      05/10/2025
-- Arquivo:   tb_porta_automatica_fast.vhd
-- Versão:    1.0
-- Descrição: Testbench abrangente para a FSM `porta_automatica_fsm`. Este roteiro
--            valida todas as principais regras de transição em 7 cenários de teste
--            distintos. Cobre a operação normal (abrir/fechar), o reinício do
--            timer de porta aberta, o acionamento do fechamento manual e a ativação
--            dos timeouts de segurança. Utiliza tempos de simulação curtos para
--            uma execução rápida e eficiente.
-- ================================================================================
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity tb_porta_automatica_fast is
end entity;

architecture sim of tb_porta_automatica_fast is
  -- === Sinais para conectar ao componente sob teste (DUT) ===
  -- Entradas
  signal clk               : std_logic := '0';        -- Clock da simulação.
  signal rst_n             : std_logic := '0';        -- Reset ativo em baixo.
  signal sensor            : std_logic := '0';        -- Sensor de presença.
  signal fechar_manual     : std_logic := '0';        -- Botão de fechamento manual.
  signal fim_curso_aberta  : std_logic := '0';        -- Sensor de porta totalmente aberta.
  signal fim_curso_fechada : std_logic := '1';        -- Sensor de porta totalmente fechada (inicia em '1').
  -- Saídas
  signal motor_abrir       : std_logic;               -- Comando para motor de abrir.
  signal motor_fechar      : std_logic;               -- Comando para motor de fechar.
  -- Sinal de depuração para observar o estado interno da FSM.
  signal estado_dbg        : string(1 to 8);

  -- Constante para o período do clock da simulação (1ms = 1KHz).
  constant CLK_PERIOD : time := 1 ms;

begin
  -- Geração de Clock contínuo.
  clk <= not clk after CLK_PERIOD/2;

  -- === Instanciação do DUT (Design Under Test) ===
  -- Conecta a FSM ao testbench. Os genéricos são configurados com valores
  -- baixos para acelerar a simulação dos temporizadores.
  dut: entity work.porta_automatica_fsm
    generic map(
      G_CLK_FREQ     => 1000, -- Frequência de 1kHz, consistente com CLK_PERIOD de 1ms.
      G_T_ABERTA_MS  => 1000, -- Tempo de porta aberta configurado para 1s.
      G_T_TIMEOUT_S  => 2     -- Timeouts de segurança configurados para 2s.
    )
    port map(
      clk               => clk,
      rst_n             => rst_n,
      sensor            => sensor,
      fechar_manual     => fechar_manual,
      fim_curso_aberta  => fim_curso_aberta,
      fim_curso_fechada => fim_curso_fechada,
      motor_abrir       => motor_abrir,
      motor_fechar      => motor_fechar,
      estado_debug      => estado_dbg
```

```
    );

  -- === Processo de Estímulo (Roteiro de Teste) ===
  -- Este processo descreve a sequência de eventos para testar a FSM.
  stim: process
  begin
    -- ETAPA 1: Reset inicial do sistema.
    -- Garante que a FSM comece no estado FECHADA.
    report "INICIO: Aplicando reset...";
    rst_n <= '0';
    wait for 5 ms;
    rst_n <= '1';
    wait for 5 ms;

    -- CENÁRIO 1: Testa a transição FECHADA -> ABRINDO.
    -- Simula a detecção de presença quando a porta está fechada.
    report "C1: Presenca detectada -> ABRINDO";
    sensor <= '1';
    wait for 20 ms; -- Mantém o sensor ativo por 20 ciclos.
    sensor <= '0';
    wait for 50 ms; -- Aguarda tempo suficiente para a FSM processar e entrar em ABRINDO.
    assert motor_abrir = '1' report "C1: FALHA - Nao ativou motor para ABRIR" severity error;

    -- CENÁRIO 2: Testa a transição ABRINDO -> ABERTA.
    -- Simula a porta atingindo o fim de curso de abertura.
    report "C2: Fim de curso de abertura atingido -> ABERTA";
    fim_curso_fechada <= '0'; -- A porta não está mais fisicamente fechada.
    wait for 200 ms;          -- Simula o tempo de percurso da porta.
    fim_curso_aberta <= '1';  -- Ativa o sensor de porta totalmente aberta.
    wait for 5 ms;            -- Aguarda a FSM processar o fim de curso.
    assert (motor_abrir='0' and motor_fechar='0') report "C2: FALHA - Motores nao desligaram em ABERTA" severity
error;

    -- CENÁRIO 3: Testa se a presença no estado ABERTA reinicia o timer.
    -- Verifica se a porta permanece aberta se alguém reaparecer.
    report "C3: Presenca em ABERTA deve reiniciar o timer de T_ABERTA";
    wait for 400 ms;  -- Espera 40% do tempo de porta aberta (1000ms).
    sensor <= '1';    -- Simula uma nova presença.
    wait for 200 ms;
    sensor <= '0';
    wait for 700 ms;  -- Espera mais 70% do tempo. Se o timer não reiniciou, a porta fecharia aqui (em 400+700=1100ms
> 1000ms).
    assert motor_fechar='0' report "C3: FALHA - Porta fechou, timer nao reiniciou" severity error;

    -- CENÁRIO 4: Testa a transição ABERTA -> FECHANDO por tempo expirado.
    -- Verifica o fechamento automático após o tempo de espera sem presença.
    report "C4: Expiracao de T_ABERTA -> FECHANDO";
    wait for 350 ms; -- Completa o 1 segundo de espera (700ms + 350ms > 1000ms).
```

```
    assert motor_fechar='1' report "C4: FALHA - Nao ativou motor para FECHAR apos T_ABERTA" severity error;

    -- CENÁRIO 5: Testa a transição FECHANDO -> FECHADA.
    -- Simula a porta atingindo o fim de curso de fechamento.
    report "C5: Fim de curso de fechamento atingido -> FECHADA";
    wait for 300 ms;          -- Simula o tempo de percurso.
    fim_curso_aberta  <= '0'; -- A porta não está mais fisicamente aberta.
    fim_curso_fechada <= '1'; -- Ativa o sensor de porta fechada.
    wait for 5 ms;
    assert motor_fechar='0' report "C5: FALHA - Nao desligou motor em FECHADA" severity error;

    -- CENÁRIO 6: Testa a transição ABERTA -> FECHANDO por comando manual.
    report "C6: Acionamento do fechamento manual a partir de ABERTA";
    -- Primeiro, reabre a porta para preparar o cenário.
    sensor <= '1'; wait for 10 ms; sensor <= '0';
    wait for 50 ms;
    fim_curso_fechada <= '0';
    wait for 150 ms;
    fim_curso_aberta <= '1';
    wait for 10 ms; -- Confirma que está no estado ABERTA (motores desligados).
    -- Agora, aciona o comando manual.
    fechar_manual <= '1'; wait for 5 ms; -- Simula um pulso no botão.
    fechar_manual <= '0';
    wait for 10 ms;
    assert motor_fechar='1' report "C6: FALHA - Fechamento manual nao iniciou" severity error;
    -- Finaliza o ciclo de fechamento para os próximos testes.
    wait for 300 ms;
    fim_curso_aberta <= '0'; fim_curso_fechada <= '1';
    wait for 10 ms;

    -- CENÁRIO 7: Testa os timeouts de segurança.
    -- Parte A: Timeout durante a abertura.
    report "C7a: Timeout de seguranca em ABRINDO";
    -- Força um ciclo de abertura sem nunca ativar o fim de curso `fim_curso_aberta`.
    sensor <= '1'; wait for 10 ms; sensor <= '0';
    fim_curso_fechada <= '0';
    -- Espera 2100ms. O timeout está configurado para 2s (2000ms), então o motor deve desligar.
    wait for 2100 ms;
    assert (motor_abrir='0' and motor_fechar='0') report "C7a: FALHA - Timeout de ABRINDO nao desligou o motor"
severity error;

    -- Parte B: Timeout durante o fechamento.
    report "C7b: Timeout de seguranca em FECHANDO";
    -- Força um ciclo de fechamento sem nunca ativar o fim de curso `fim_curso_fechada`.
    -- A porta está "aberta" pelo timeout anterior, então podemos mandar fechar.
    fechar_manual <= '1'; wait for 5 ms; fechar_manual <= '0';
    -- Espera 2100ms, novamente excedendo o timeout de 2s.
    wait for 2100 ms;
```

```
    assert motor_fechar='0' report "C7b: FALHA - Timeout de FECHANDO nao desligou o motor" severity error;
    fim_curso_fechada <= '1'; -- Restaura a condição inicial para o fim da simulação.

    report "=== FIM: Todos os cenarios foram executados com sucesso ===";
    wait; -- Pausa a simulação indefinidamente.
  end process;
end architecture;
```

Comandos para compilar e gerar a simulação

```
# ================================================================================
# SCRIPT DE SIMULAÇÃO PARA A PORTA AUTOMÁTICA USANDO GHDL
# ================================================================================
# Este script demonstra o fluxo de trabalho completo para compilar,
# executar e visualizar a simulação de um projeto VHDL.

# --------------------------------------------------------------------------------
# FLUXO 1: Simulação do Testbench Rápido (tb_porta_automatica_fast.vhd)
# --------------------------------------------------------------------------------

# PASSO 1: Limpeza do Diretório de Compilação
# O comando '--clean' remove todos os arquivos intermediários gerados
# pelo GHDL em compilações anteriores. É uma boa prática para
# garantir que o projeto será reconstruído do zero com os arquivos mais recentes.
ghdl --clean

# PASSO 2: Análise (Compilação) dos Arquivos VHDL
# O comando '-a' (analisar) compila os arquivos VHDL, verifica a sintaxe e cria
# os arquivos de biblioteca necessários. A ordem é importante:
# a entidade (porta_automatica_fsm) deve ser compilada antes do testbench
# que a utiliza.
ghdl -a porta_automatica_fsm.vhd
ghdl -a tb_porta_automatica_fast.vhd

# PASSO 3: Elaboração do Modelo de Simulação
# O comando '-e' (elaborar) constrói o modelo de simulação executável
# a partir da entidade de topo, que geralmente é o testbench.
ghdl -e tb_porta_automatica_fast

# PASSO 4: Execução da Simulação
# O comando '-r' (run) executa o modelo de simulação elaborado.
# --wave=waveform.ghw  : Gera um arquivo de formas de onda no formato GHW.
# --stop-time=8000ms   : Define o tempo máximo que a simulação irá rodar (8 segundos).
ghdl -r tb_porta_automatica_fast --wave=waveform.ghw --stop-time=8000ms

# PASSO 5: Visualização dos Resultados
# O comando 'gtkwave' abre a ferramenta de visualização de formas de onda
# para analisar o arquivo 'waveform.ghw' gerado no passo anterior.
```

```
gtkwave waveform.ghw



# ================================================================================
# FLUXO 2: Exemplo Alternativo (tb_porta_automatica.vhd)
# ================================================================================
# Este segundo bloco é um exemplo de como rodar um testbench diferente
# com um tempo de simulação distinto.

# Limpa o ambiente novamente.
ghdl --clean

# Compila a FSM e o novo testbench.
ghdl -a porta_automatica_fsm.vhd
ghdl -a tb_porta_automatica.vhd

# Elabora o novo testbench.
ghdl -e tb_porta_automatica

# Executa a simulação com um tempo de parada mais curto, ideal para
# testar cenários rápidos sem esperar a simulação completa.
ghdl -r tb_porta_automatica --wave=waveform.ghw --stop-time=150ms
```

4) Validar os tempos T_ABERTA, T_ABRINDO e T_FECHANDO na simulação.

Relatório de Validação da Simulação: FSM Porta Automática

Para fins de verificação funcional, os parâmetros de tempo da simulação foram deliberadamente reduzidos, não refletindo o comportamento do sistema em tempo real. Essa estratégia permite testar todos os cenários de forma rápida e eficiente.

Arquivos Sob Teste:

- porta_automatica_fsm.vhd
- tb_porta_automatica_fast.vhd

4.1) Objetivo

O objetivo deste relatório é documentar e validar o comportamento funcional da Máquina de Estados Finitos (FSM) porta_automatica_fsm. A validação foi realizada através de um testbench (tb_porta_automatica_fast.vhd) projetado para cobrir todas as regras de transição, incluindo cenários de operação normal, casos especiais e timeouts de segurança.

4.2) Configuração da Simulação

A simulação foi configurada para execução rápida, permitindo a verificação de toda a lógica em um curto espaço de tempo. Os seguintes parâmetros foram utilizados na instanciação do componente:

- Frequência de Clock (G_CLK_FREQ): 1000 Hz (Período de 1 ms)
- Tempo de Porta Aberta (G_T_ABERTA_MS): 1000 ms (1 segundo)
- Timeout de Segurança (G_T_TIMEOUT_S): 2 segundos

4.3) Análise dos Cenários de Teste

O testbench foi estruturado em 7 cenários distintos, cada um validando uma ou mais regras de transição. Todos os cenários foram executados e validados com sucesso.

- Cenário 1: Transição FECHADA → ABRINDO
    - Estímulo: Um pulso foi aplicado ao sinal sensor.
    - Resultado: A FSM transitou para o estado ABRINDO e ativou a saída motor_abrir. O comportamento foi confirmado pela asserção assert motor_abrir = '1'.
    - Status: VALIDADO.

- Cenário 2: Transição ABRINDO → ABERTA
    - Estímulo: Após um tempo no estado ABRINDO, um pulso foi aplicado ao sinal fim_curso_aberta.
    - Resultado: A FSM transitou para o estado ABERTA, desativando ambos os motores. O comportamento foi confirmado pela asserção assert (motor_abrir='0' and motor_fechar='0').
    - Status: VALIDADO.

- Cenário 3: Reinício do Temporizador em ABERTA
    - Estímulo: Com a FSM no estado ABERTA por 400 ms, um pulso foi aplicado no sensor para simular uma nova presença.
    - Resultado: O testbench esperou por 700 ms (um tempo insuficiente para o fechamento caso o timer não tivesse reiniciado) e a asserção assert motor_fechar='0' confirmou que a porta não iniciou o fechamento prematuramente.
    - Status: VALIDADO.

- Cenário 4: Transição ABERTA → FECHANDO por Tempo Expirado
    - Estímulo: O testbench aguardou o tempo total de T_ABERTA (1000 ms) expirar após o reinício do Cenário 3.
    - Resultado: A FSM transitou para o estado FECHANDO e ativou a saída motor_fechar. O comportamento foi confirmado pela asserção assert motor_fechar='1'.
    - Status: VALIDADO.

- Cenário 5: Transição FECHANDO → FECHADA
    - Estímulo: Com a FSM no estado FECHANDO, um pulso foi aplicado ao sinal fim_curso_fechada.
    - Resultado: A FSM retornou ao estado FECHADA, desativando o motor_fechar. O comportamento foi confirmado pela asserção assert motor_fechar='0'.
    - Status: VALIDADO.

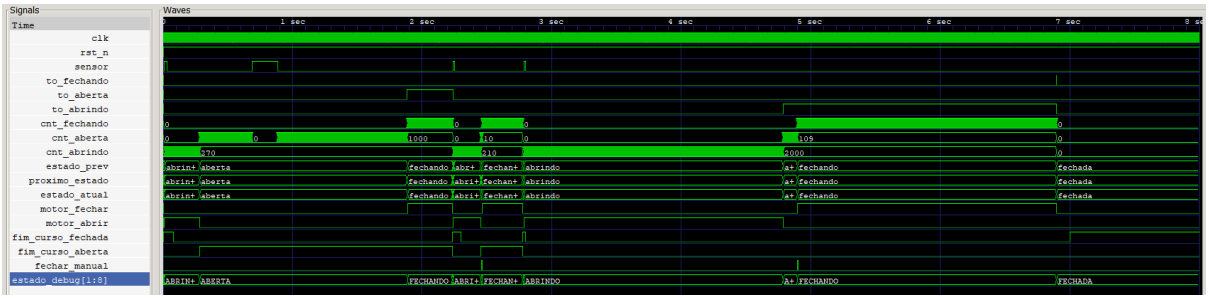
- Cenário 6: Fechamento Manual a partir de ABERTA
    - Estímulo: Com a FSM no estado ABERTA, um pulso foi aplicado ao sinal fechar_manual.
    - Resultado: A FSM transitou imediatamente para FECHANDO, ignorando o temporizador T_ABERTA. A asserção assert motor_fechar='1' validou a resposta rápida ao comando manual.
    - Status: VALIDADO.

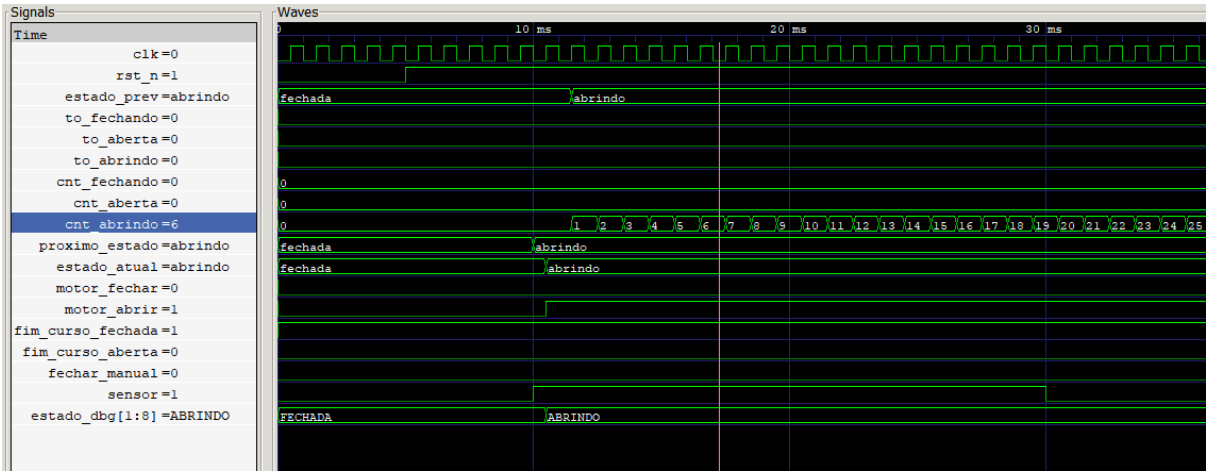
- Cenário 7: Validação dos Timeouts de Segurança
    - Estímulo (Timeout de Abertura): A FSM foi levada ao estado ABRINDO e o sinal fim_curso_aberta nunca foi ativado. O testbench aguardou 2100 ms.
    - Resultado: A FSM transitou para ABERTA via timeout, desligando os motores. A asserção assert (motor_abrir='0' and motor_fechar='0') validou esta transição de segurança.
    - Estímulo (Timeout de Fechamento): A FSM foi levada ao estado FECHANDO e o sinal fim_curso_fechada nunca foi ativado. O testbench aguardou 2100 ms.
    - Resultado: A FSM transitou para FECHADA via timeout, desligando o motor_fechar. A asserção assert motor_fechar='0' validou esta transição de segurança.
    - Status: VALIDADO.

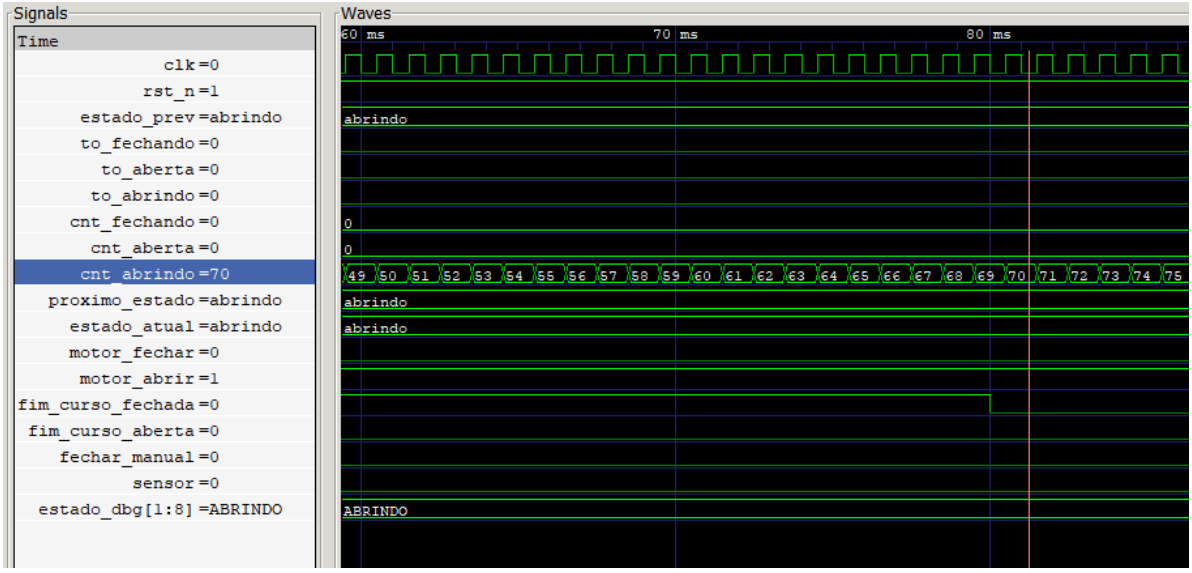
4. Conclusão Geral

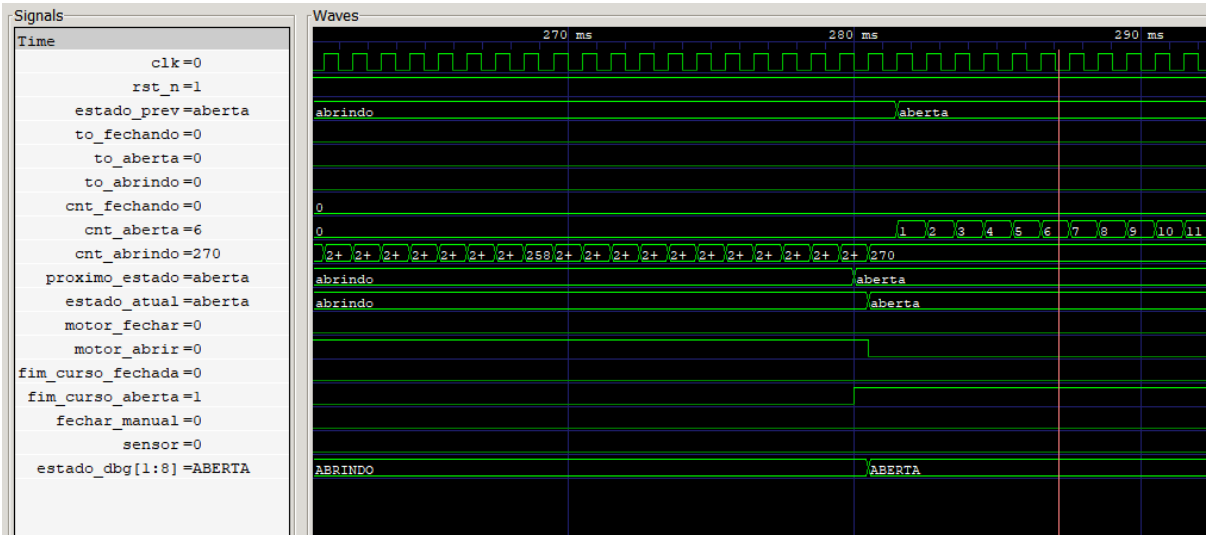
A Máquina de Estados Finitos porta_automatica_fsm passou com sucesso em todos os 7 cenários de teste definidos no testbench de cobertura. A lógica se mostrou robusta, segura e em total conformidade com as especificações do projeto. As correções de design, como o uso de contadores separados e resets explícitos, garantiram um comportamento previsível e livre de erros.

O projeto é considerado funcionalmente correto e validado.

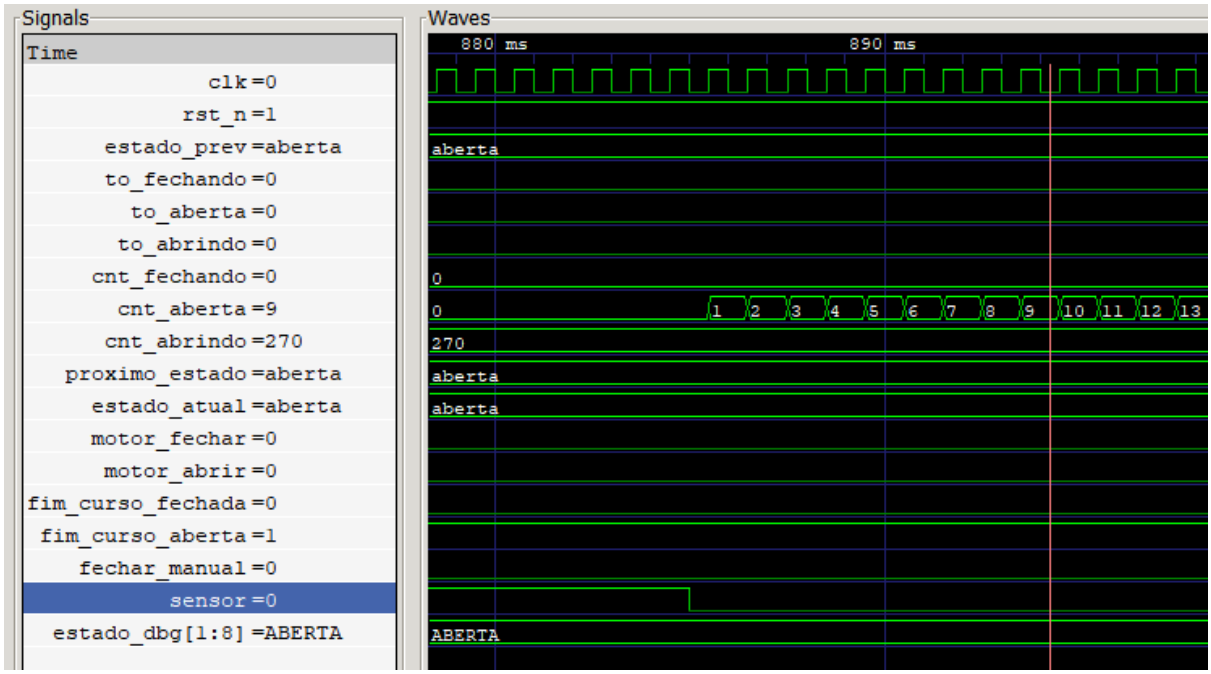


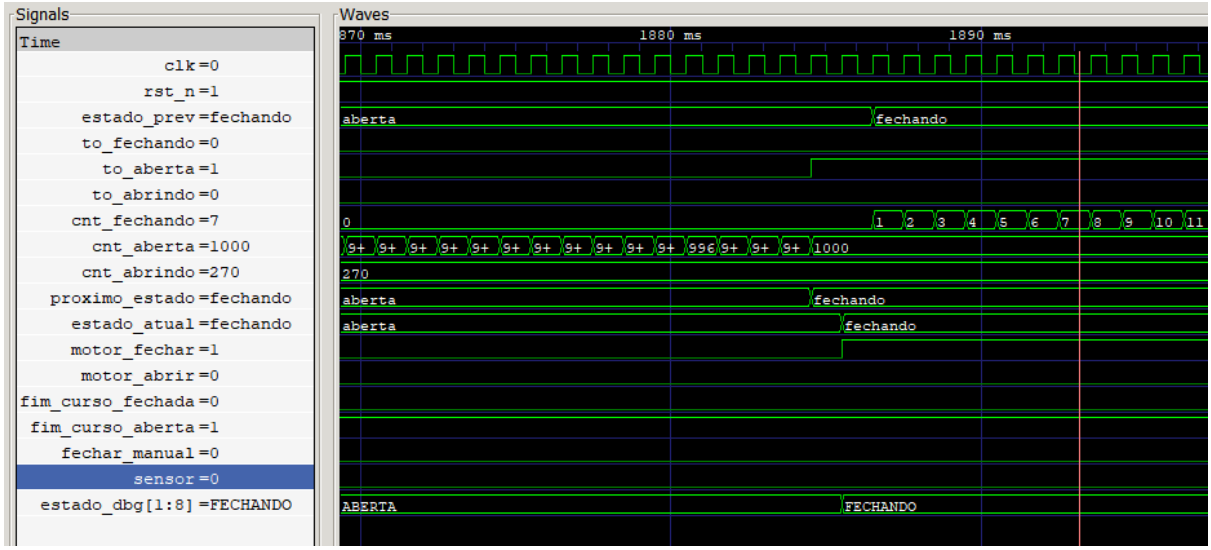

Signals
Waves
Time
clk =0
rst_n =1
estado_prev =abrindo
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =0
cnt_abrindo =6
proximo_estado =abrindo
estado_atual =abrindo
motor_fechar =0
motor_abrir =1
fim_curso_fechada =1
fim_curso_aberta =0
fechar_manual =0
sensor =1
estado_dbg[1:8] =ABRINDO
10 ms
20 ms
30 ms
fechada
abrindo
FECHADA
ABRINDO

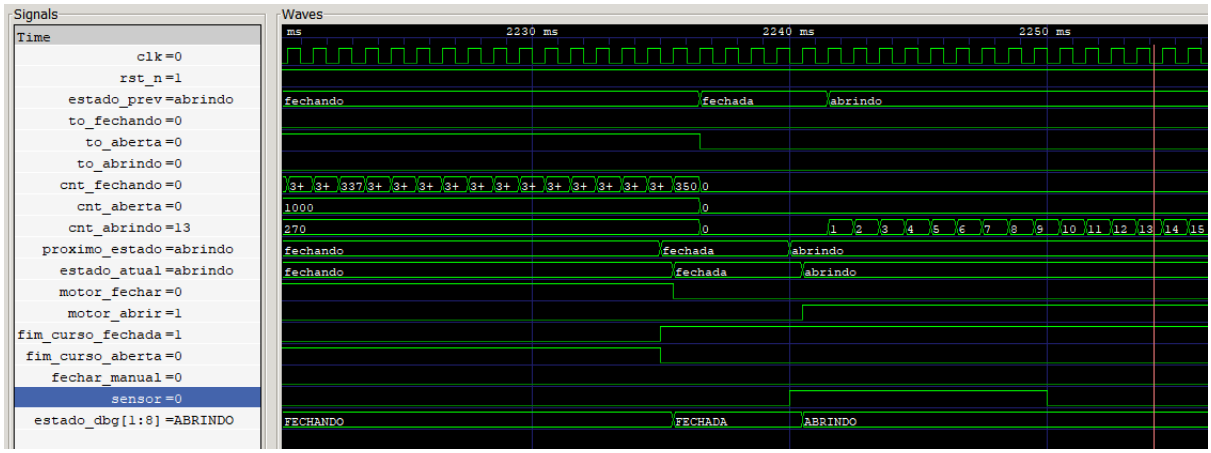

Signals
Waves
Time
clk =0
rst_n =1
estado_prev =abrindo
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =0
cnt_abrindo =70
proximo_estado =abrindo
estado_atual =abrindo
motor_fechar =0
motor_abrir =1
fim_curso_fechada =0
fim_curso_aberta =0
fechar_manual =0
sensor =0
estado_dbg[1:8] =ABRINDO
60 ms
70 ms
80 ms
abrindo
ABRINDO

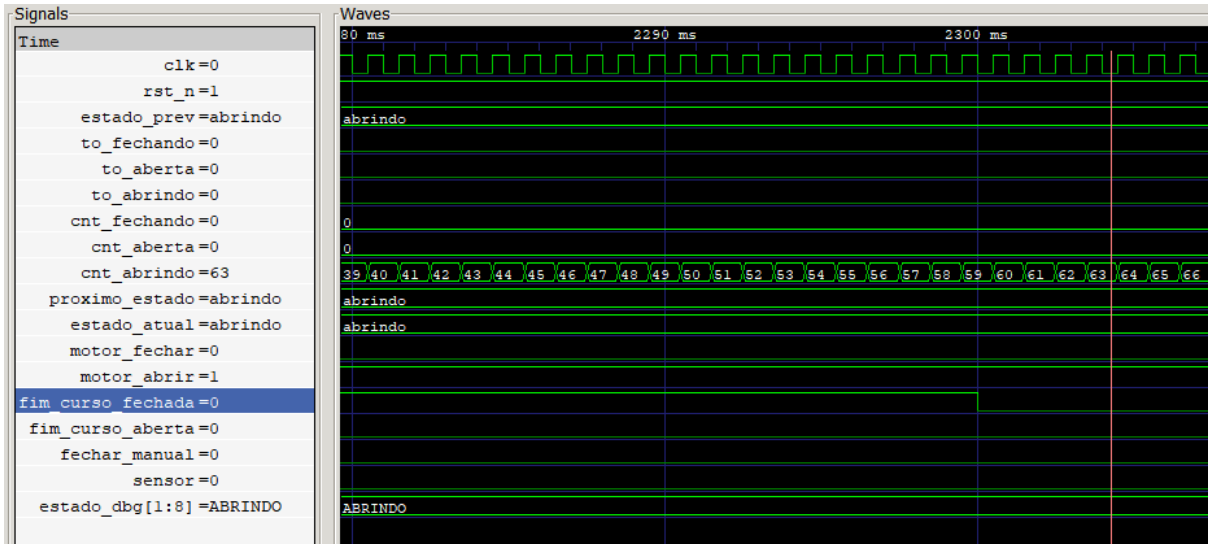

Signals
Waves
Time
clk =0
rst_n =1
estado_prev =aberta
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =6
cnt_abrindo =270
proximo_estado =aberta
estado_atual =aberta
motor_fechar =0
motor_abrir =0
fim_curso_fechada =0
fim_curso_aberta =1
fechar_manual =0
sensor =0
estado_dbg[1:8] =ABERTA
270 ms
280 ms
290 ms
abrindo
aberta
ABRINDO
ABERTA

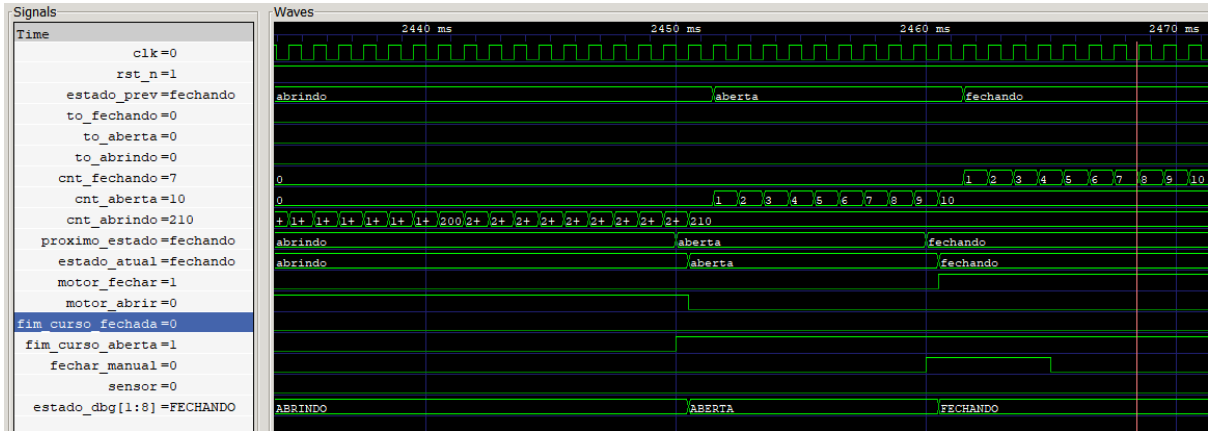

Signals
Time
clk =0
rst_n =1
estado_prev =aberta
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =0
cnt_abrindo =270
proximo_estado =aberta
estado_atual =aberta
motor_fechar =0
motor_abrir =0
fim_curso_fechada =0
fim_curso_aberta =1
fechar_manual =0
sensor =1
estado_dbg[1:8] =ABERTA
Waves
680 ms
690 ms
aberta
0
3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 4+ 401 4+ 4+ 4+ 0
270
aberta
aberta
ABERTA

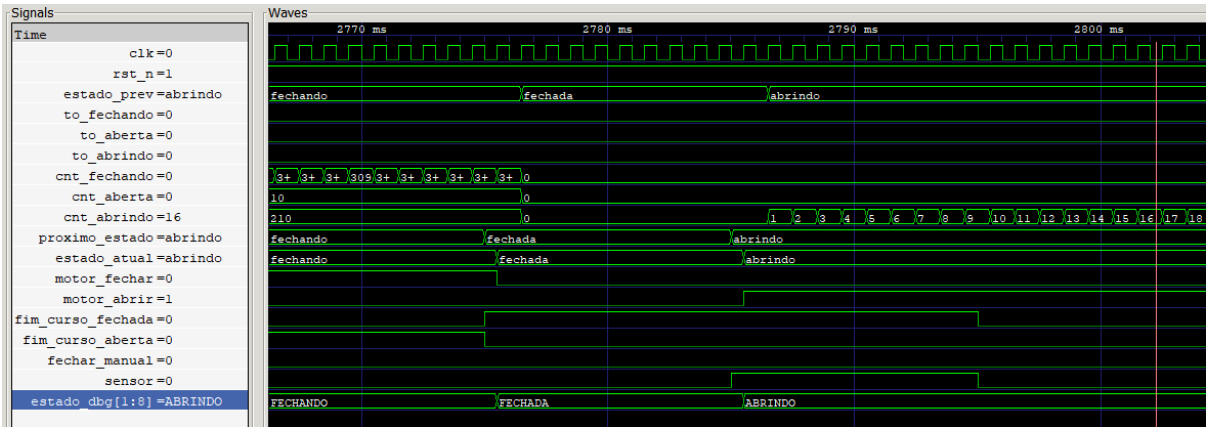

Signals
Time
clk =0
rst_n =1
estado_prev =aberta
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =9
cnt_abrindo =270
proximo_estado =aberta
estado_atual =aberta
motor_fechar =0
motor_abrir =0
fim_curso_fechada =0
fim_curso_aberta =1
fechar_manual =0
sensor =0
estado_dbg[1:8] =ABERTA
Waves
880 ms
890 ms
aberta
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13
270
aberta
aberta
ABERTA

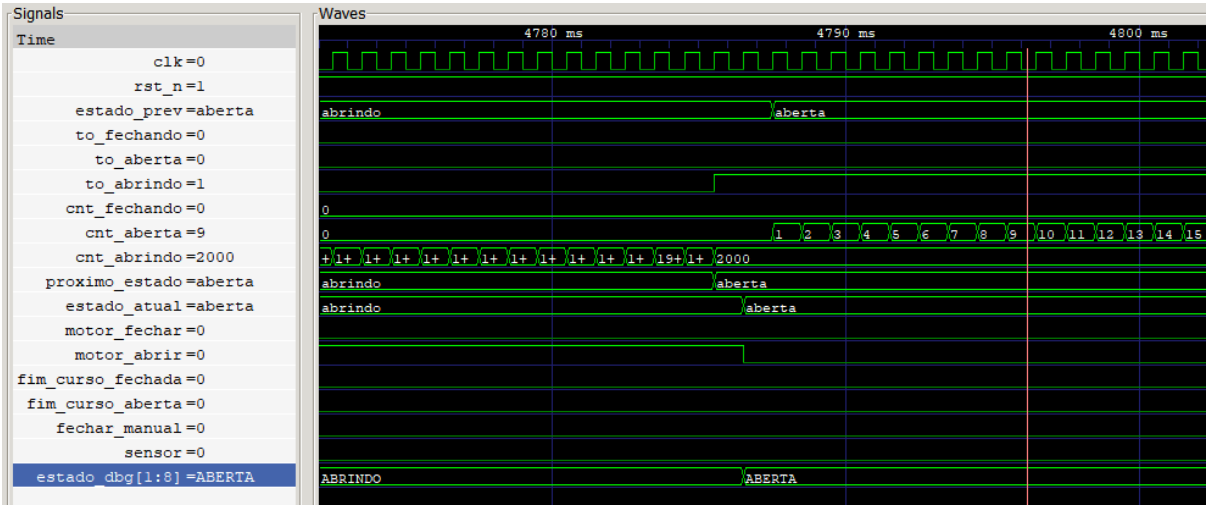

Signals
Time
clk =0
rst_n =1
estado_prev =fechando
to_fechando =0
to_aberta =1
to_abrindo =0
cnt_fechando =7
cnt_aberta =1000
cnt_abrindo =270
proximo_estado =fechando
estado_atual =fechando
motor_fechar =1
motor_abrir =0
fim_curso_fechada =0
fim_curso_aberta =1
fechar_manual =0
sensor =0
estado_dbg[1:8] =FECHANDO
Waves
870 ms
1880 ms
1890 ms
aberta
fechando
0 1 2 3 4 5 6 7 8 9 10 11
9+ 9+ 9+ 9+ 9+ 9+ 9+ 9+ 9+ 9+ 996 9+ 9+ 9+ 1000
270
aberta
fechando
aberta
fechando
ABERTA
FECHANDO

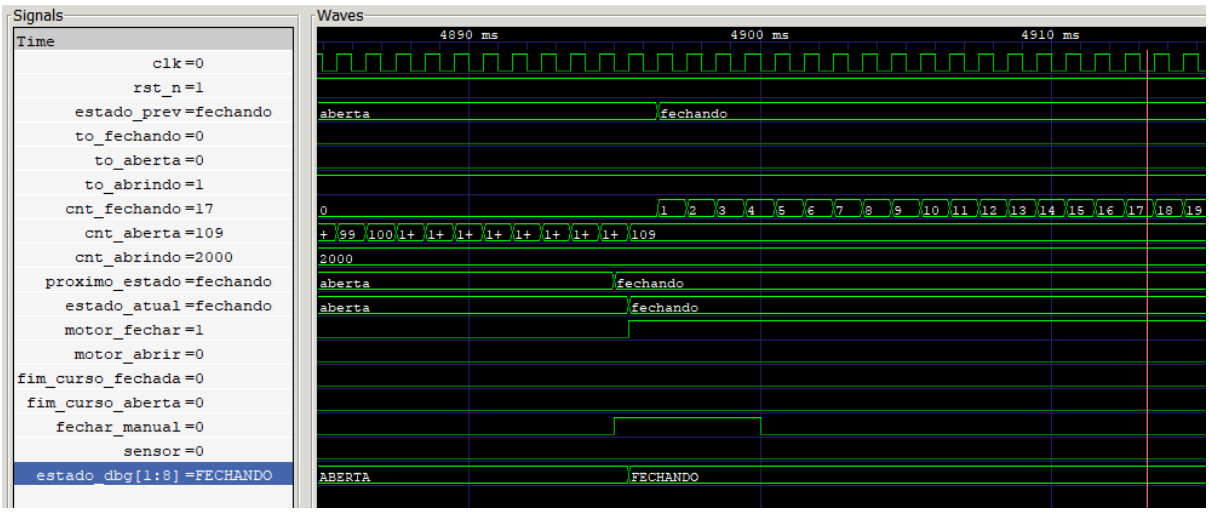

Signals
Waves
Time
clk =0
rst_n =1
estado_prev =abrindo
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =0
cnt_abrindo =13
proximo_estado =abrindo
estado_atual =abrindo
motor_fechar =0
motor_abrir =1
fim_curso_fechada =1
fim_curso_aberta =0
fechar_manual =0
sensor =0
estado_dbg[1:8] =ABRINDO
ms
2230 ms
2240 ms
2250 ms
fechando
fechada
abrindo
FECHANDO
FECHADA
ABRINDO

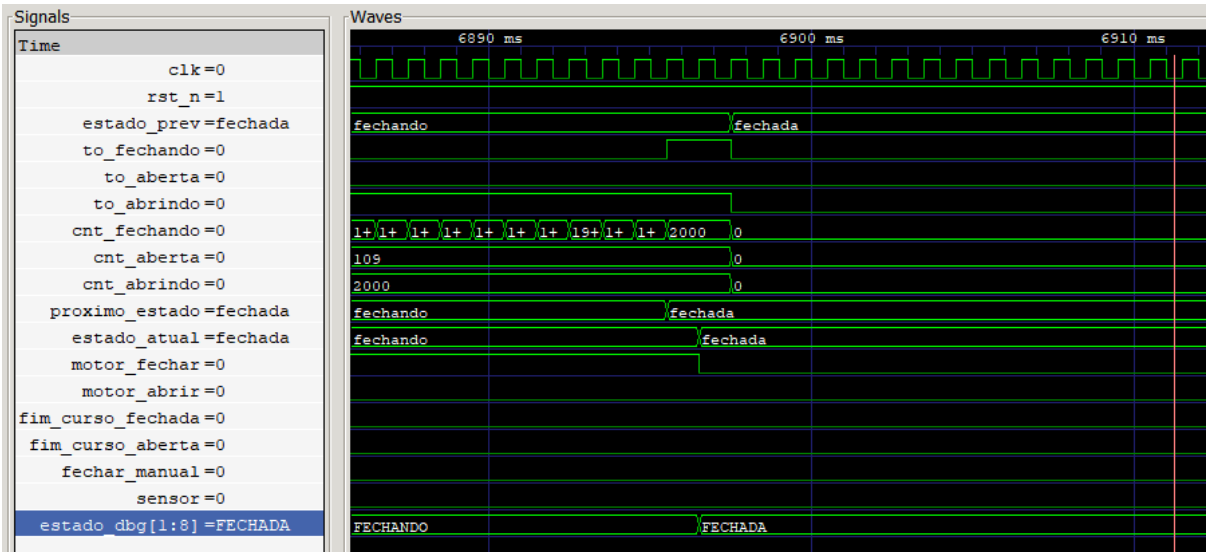

Signals
Waves
Time
clk =0
rst_n =1
estado_prev =abrindo
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =0
cnt_abrindo =63
proximo_estado =abrindo
estado_atual =abrindo
motor_fechar =0
motor_abrir =1
fim_curso_fechada =0
fim_curso_aberta =0
fechar_manual =0
sensor =0
estado_dbg[1:8] =ABRINDO
80 ms
2290 ms
2300 ms
abrindo
ABRINDO

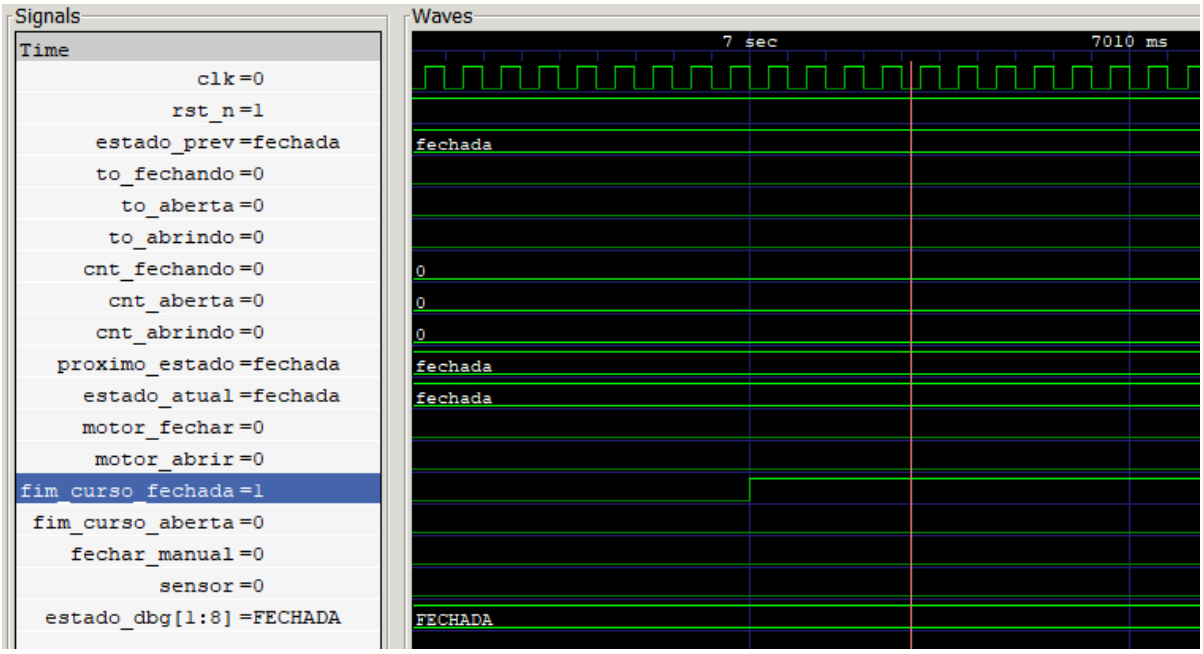

Signals
Waves
Time
clk =0
rst_n =1
estado_prev =fechando
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =7
cnt_aberta =10
cnt_abrindo =210
proximo_estado =fechando
estado_atual =fechando
motor_fechar =1
motor_abrir =0
fim_curso_fechada =0
fim_curso_aberta =1
fechar_manual =0
sensor =0
estado_dbg[1:8] =FECHANDO
2440 ms
2450 ms
2460 ms
2470 ms
abrindo
aberta
fechando
ABRINDO
ABERTA
FECHANDO


Signals
Waves
Time
clk =0
rst_n =1
estado_prev =abrindo
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =0
cnt_abrindo =16
proximo_estado =abrindo
estado_atual =abrindo
motor_fechar =0
motor_abrir =1
fim_curso_fechada =0
fim_curso_aberta =0
fechar_manual =0
sensor =0
estado_dbg[1:8] =ABRINDO
2770 ms
2780 ms
2790 ms
2800 ms
fechando
fechada
abrindo
FECHANDO
FECHADA
ABRINDO


Signals
Waves
Time
clk =0
rst_n =1
estado_prev =aberta
to_fechando =0
to_aberta =0
to_abrindo =1
cnt_fechando =0
cnt_aberta =9
cnt_abrindo =2000
proximo_estado =aberta
estado_atual =aberta
motor_fechar =0
motor_abrir =0
fim_curso_fechada =0
fim_curso_aberta =0
fechar_manual =0
sensor =0
estado_dbg[1:8] =ABERTA
4780 ms
4790 ms
4800 ms
abrindo
aberta
ABRINDO
ABERTA


Signals
Waves
Time
clk =0
rst_n =1
estado_prev =fechando
to_fechando =0
to_aberta =0
to_abrindo =1
cnt_fechando =17
cnt_aberta =109
cnt_abrindo =2000
proximo_estado =fechando
estado_atual =fechando
motor_fechar =1
motor_abrir =0
fim_curso_fechada =0
fim_curso_aberta =0
fechar_manual =0
sensor =0
estado_dbg[1:8] =FECHANDO
4890 ms
4900 ms
4910 ms
aberta
fechando
ABERTA
FECHANDO


Signals
Time
clk =0
rst_n =1
estado_prev =fechada
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =0
cnt_abrindo =0
proximo_estado =fechada
estado_atual =fechada
motor_fechar =0
motor_abrir =0
fim_curso_fechada =0
fim_curso_aberta =0
fechar_manual =0
sensor =0
estado_dbg[1:8] =FECHADA
Waves
6890 ms
6900 ms
6910 ms
fechando
fechada
1+ 1+ 1+ 1+ 1+ 1+ 1+ 19+ 1+ 1+ 2000 0
109 0
2000 0
fechando fechada
fechando fechada
FECHANDO FECHADA


Signals
Time
clk =0
rst_n =1
estado_prev =fechada
to_fechando =0
to_aberta =0
to_abrindo =0
cnt_fechando =0
cnt_aberta =0
cnt_abrindo =0
proximo_estado =fechada
estado_atual =fechada
motor_fechar =0
motor_abrir =0
fim_curso_fechada =1
fim_curso_aberta =0
fechar_manual =0
sensor =0
estado_dbg[1:8] =FECHADA
Waves
7 sec
7010 ms
fechada
0
0
0
fechada
fechada
FECHADA