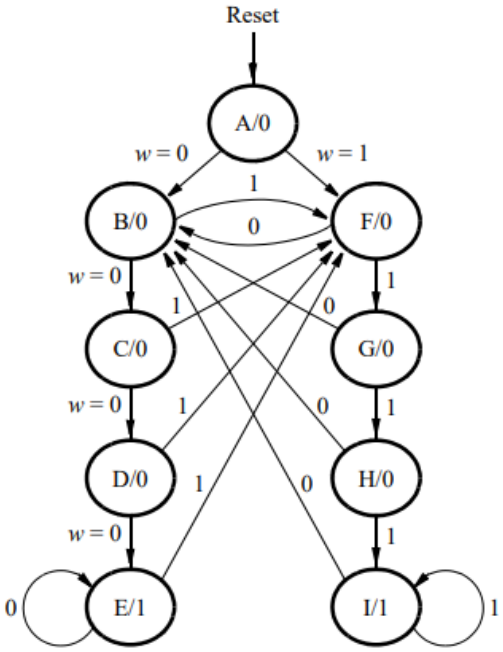
Aula 8 - Finite State Machines

Parte 1: fizemos uma máquina de estado finito que detecta 4 entradas iguais consecutivas. Caso o input w permaneça com o valor 0, ou 1, por 4 ciclos de clock consecutivos, o sinal z é atualizado para 1. Enquanto w permanecer nesse mesmo sinal, z permanecerá ligado. Assim que w mudar de sinal, z será desligado. O seguinte diagrama representa essa FSM.



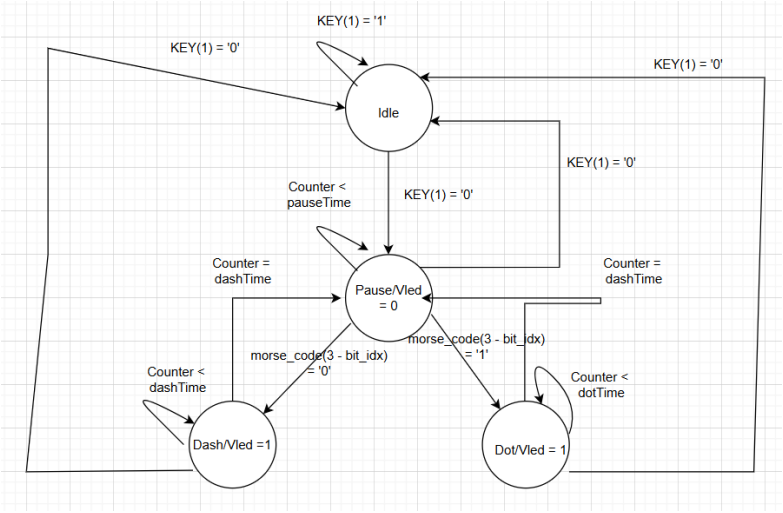
Quando w permanece ligado durante 4 clocks consecutivos, alcança o estado E, ou permance desligado durante 4 clocks consecutivos, alcança o estado I, o LED9, representando a saída z, é ligado.

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

entity parte1 is
    port (
        clk     : in std_logic;     -- KEY0
        reset_n : in std_logic;     -- SW0
        w       : in std_logic;     -- SW1
        z       : out std_logic;    -- LED 9
        leds    : out std_logic_vector(8 downto 0)  -- LEDs 8 a 0
    );
end entity parte1;

architecture behavioral of parte1 is
    signal state : std_logic_vector(8 downto 0);

begin

    process (clk)
    begin
        if rising_edge(clk) then
            if reset_n = '0' then
                state <= "000000000";  -- reset to state A
            else
                case state is
                    when "000000000" =>  -- estado A
                        if w = '1' then
                            state <= "000000011";  -- estado B
                        else
                            state <= "000100001";  -- estado F
                        end if;

                    when "000000011" =>  -- estado B
                        if w = '1' then
                            state <= "000000101";  -- Estado C
                        else
                            state <= "000100001";  -- Estado F
                        end if;

                    when "000000101" =>  -- estado C
                        if w = '1' then
                            state <= "000001001";  -- D
                        else
                            state <= "000100001";  -- F
                        end if;

                    when "000001001" =>  -- Estado D
                        if w = '1' then
                            state <= "000010001";  -- E
                        else
                            state <= "000100001";  -- F
                        end if;

                    when "000010001" =>  -- E
                        if w = '1' then
                            state <= "000010001";  -- E
                        else
                            state <= "000100001";  -- F
                        end if;

                    when "000100001" =>  -- F
                        if w = '0' then
                            state <= "001000001";  -- G
                        else
                            state <= "000000011";  -- B
                        end if;
                    when "001000001" =>  -- G
                        if w = '0' then
                            state <= "010000001";  -- H
                        else
                            state <= "000000011";  -- B
                        end if;

                    when "010000001" =>  -- H
                        if w = '0' then
                            state <= "100000001";  -- I
                        else
                            state <= "000000011";  -- B
                        end if;
                    when "100000001" =>  -- I
                        if w = '0' then
                            state <= "100000001";  -- I
                        else
                            state <= "000000011";  -- B
                        end if;

                    when others =>
                        state <= "000000000";  -- A
                end case;
            end if;
        end if;
    end process;

    -- LED 9 acende nos estados E e I
    z <= '1' when (state = "000010001" or state = "100000001") else '0';

    -- Assign state to LEDs
    leds <= state;

end architecture behavioral;
```

Parte 4: implementamos o programa de código morse com uma máquina de estados.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity parte4 is
    Port (
        SW : in STD_LOGIC_VECTOR (2 downto 0);  -- Entradas SW2-0 (Seleção de letras de A até H)
        KEY : in STD_LOGIC_VECTOR (1 downto 0); -- KEY1 (iniciar), KEY0 (reset)
        clk : in STD_LOGIC;                     -- Clock
        LEDR : out STD_LOGIC                    -- Saída para o LED (Morse)
    );
end parte4;

architecture Behavioral of parte4 is

    type state_type is (idle, dot, dash, pause);  -- Definição dos estados: idle, ponto, traço e pausa
    signal state : state_type := idle;
    signal morse_code : STD_LOGIC_VECTOR(3 downto 0); -- Vetor de 8 bits para armazenar o código Morse
    signal counter : INTEGER := 0;                    -- Contador para definir duração dos pontos e traços
    signal tam_cod : INTEGER := 0;
    signal bit_idx : INTEGER := 0;                    -- Índice para o bit atual no código Morse
    signal vled : STD_LOGIC := '0';
    --signal active : STD_LOGIC := '0';               -- Indica se o encoder está ativo ou não

    --signal debug_state : STD_LOGIC_VECTOR(3 downto 0);

    -- Constantes para definir a duração dos pontos e traços
    constant dot_time : INTEGER := 25000000;  -- Meio segundo para ponto (assumindo 50 MHz clock)
    constant dash_time : INTEGER := 75000000;-- 1.5 segundos para traço
    constant pause_time : INTEGER := 25000000;-- Meio segundo de pausa entre cada símbolo

begin
    process(clk)
    begin
    --variable vled : std_logic := 0;
      if KEY(0) = '0' then  -- Reset assíncrono
          state <= idle;
          counter <= 0;
          vled <= '0';

      elsif rising_edge(clk) then
        case state is
            when idle =>
                if KEY(1) = '0' then
                    -- Selecionar o código Morse correspondente à letra escolhida
                    case SW is
                        when "000" =>
                            morse_code <= "1000"; -- A: .-
                            tam_cod <= 2;
                        when "001" =>
                            morse_code <= "0111"; -- B: -...
                            tam_cod <= 4;
                        when "010" =>
                            morse_code <= "0101"; -- C: -.-.
                            tam_cod <= 4;
                        when "011" =>
                            morse_code <= "0110"; -- D: -..
                            tam_cod <= 3;
                        when "100" =>
                            morse_code <= "1000"; -- E: .
                            tam_cod <= 1;
                        when "101" =>
                           morse_code <= "1101"; -- F: ..-.
                           tam_cod <= 4;
                        when "110" =>
                            morse_code <= "0010"; -- G: --.
                            tam_cod <= 3;
                        when "111" =>
                            morse_code <= "1111"; -- H: ....
                            tam_cod <= 4;
                        when others => morse_code <= "0000";
                    end case;
                    --active <= '1';
                    bit_idx <= 0;
                    state <= pause;
                    counter <= 0;

                else
                    state <= idle;
                end if;
                vled <= '0';

            when dot =>
                if counter < dot_time then
                    counter <= counter + 1;
                    vled <= '1';
                    state <= dot;
                else
                    counter <= 0;
                    bit_idx <= bit_idx + 1;
                    state <= pause;
                end if;

            when dash =>
                if counter < dash_time then
                    counter <= counter + 1;
                    vled <= '1';
                    state <= dash;
                else
                    counter <= 0;
                    bit_idx <= bit_idx + 1;
                    state <= pause;
                end if;

            when pause =>

                if counter < pause_time then
                    counter <= counter + 1;
                    vled <= '0';
                    state <= pause;
                else
                    counter <= 0;
                    if bit_idx < tam_cod then
                        if morse_code(3 - bit_idx) = '1' then
                            state <= dot;
                        elsif morse_code(3 - bit_idx) = '0' then
                            state <= dash;
                        end if;
                    else
                        state <= idle;
                        --active <= '0';
                    end if;
                end if;
        end case;
      end if;
    end process;
   LEDR <= vled;
end Behavioral;
```

O seguinte diagrama representa essa máquina de estados:



O estado idle representa o momento em que o programa espera que o usuário confirme sua entrada em um dos botões da FPGA e o estado de reset. O estado pause representa o intervalo entre a representação de um sinal. Enquanto o contador for menor que pausetime, a máquina permanecerá nesse estado e o LED ficará desligado. Quando o contador alcançar esse valor, após meio segundo, um bit do vetor morse_code é lido, e se for igual a 1, o estado mudará para Dot, mas se for 0, mudará para Dash. Caso o primeiro estado seja selecionado, o LED ficará ligado até que o contador esteja nesse estado, meio segundo, e depois disso o estado volta ao estado de pausa. Se o estado Dash for selecionado, o LED ficará ligado pelo tempo de duração do estado, 1,5 segundo, até que o estado volte para Pause. Em qualquer estado a qualquer momento, apertar o botão de reset muda o estado para idle.