Relatório: Análise dos Códigos VHDL

Parte I - Detecção de Sequências em uma FSM

Este código implementa uma **Máquina de Estados Finitos (FSM)** que reconhece sequências de quatro "1"s ou quatro "0"s consecutivos na entrada `W`. A saída `Z` é ativada (1) após a quarta ocorrência consecutiva. O projeto é baseado em uma codificação *one-hot* para os estados, conforme descrito no **Tabela 1 do PDF**.

Funcionamento

1. **Definição dos Estados:**
    - Cada estado é representado por um único bit ativo (ex.: "000000001" para o estado A).
    - A transição entre estados ocorre com base na entrada `W`.
    - Quando `W` é "1" ou "0" por quatro pulsos consecutivos de clock, a FSM ativa `Z`.
2. **Reset Síncrono:**
    - A FSM é reiniciada ao estado A com `Reset` ativo (`SW0` = 0).
    - Um estado "modificado" é usado no reset para assegurar que todas as saídas de flip-flop sejam "0".
3. **Implementação de Estados:**
    - O código utiliza 9 flip-flops D instanciados diretamente para representar cada estado.

As condições de transição são especificadas utilizando *case statements*, como no exemplo abaixo:

```
 case saida is
  when "000000001" => estado <= "000000010"; -- Transição de A para B
  ...
end case;
```

    -
4. **Saída Z:**
    - A saída Z é ativada somente em estados específicos (ex.: "000010001" ou "100000001").

Pontos Importantes do Código

- **Sinal ze:** Controla diretamente a saída Z.

Flip-Flops D Instanciados:
Cada estado é atualizado utilizando flip-flops D conectados ao clock principal.
ff1: entity work.d_flip_flop
port map (
D => estado(0),
Clk => Clk,
Q => saida(0),
nQ => open
);

- **Simplicidade com One-Hot:**
  O uso de codificação one-hot reduz a complexidade das expressões lógicas.

VHDL:

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

ENTITY aula8part1 IS
port(
Clk, Reset, W : in std_logic;
Z : out std_logic;
ff : out std_logic_vector(8 downto 0)
);
END aula8part1;
ARCHITECTURE Behavioral OF aula8part1 IS
COMPONENT d_flip_flop IS
port(
D, Clk : in std_logic;
Q, nQ : out std_logic
);
END COMPONENT;
signal estado : std_logic_vector(8 downto 0);

begin

process(Clk) is
begin
if rising_edge(Clk) then
if (Reset = '0') then
estado <= "000000001";
z <= '0';
else
if (W = '1') then
case estado is
when "000000001" => estado <= "000100000"; -- A
when "000000010" => estado <= "000100000"; -- B
when "000000100" => estado <= "000100000"; -- C
when "000001000" => estado <= "000100000"; -- D
when "000010000" => estado <= "000100000"; -- E
when "000100000" => estado <= "001000000"; -- F
when "001000000" => estado <= "010000000"; -- G
when "010000000" => estado <= "100000000"; -- H
when others => estado <= "100000000"; -- I
end case;
elsif (W = '0') then
case estado is
when "000000001" => estado <= "000000010"; -- A
when "000000010" => estado <= "000000100"; -- B
when "000000100" => estado <= "000001000"; -- C
when "000001000" => estado <= "000010000"; -- D
when "000010000" => estado <= "000010000"; -- E
when "000100000" => estado <= "000000010"; -- F
when "001000000" => estado <= "000000010"; -- G
when "010000000" => estado <= "000000010"; -- H
when others => estado <= "000000010"; -- I
end case;
end if;

end if;
end if;
if (estado = "000010000" OR estado = "100000000") then
z <= '1';
else
z <= '0';
end if;
end process;


ff <= estado;

END Behavioral;
```

Parte IV - Codificador de Código Morse

Este código implementa um **codificador Morse** para as primeiras oito letras do alfabeto. A entrada (`Num`) determina a letra a ser codificada, e a saída (`led`) emite o padrão Morse correspondente através de pulsos curtos (pontos) e longos (traços).

Funcionamento

1. **Tabela Morse:**
    - Cada letra é representada por um vetor de 4 bits. Por exemplo, `"01--"` representa a letra "A".
    - Um vetor auxiliar `tamanhos` especifica o comprimento dos padrões de cada letra.
2. **Estados da FSM:**
    - **Estado "00":** Desativado, aguardando ativação (`Enable = '1'`).
    - **Estado "01":** Intervalo entre bits do código.
    - **Estado "10":** Representa um ponto (0,5 segundos).
    - **Estado "11":** Representa um traço (1,5 segundos).
3. **Controle de Tempo:**
    - Um contador de 26 bits (`Tempo1`) gerencia os pulsos para pontos e traços.
    - A transição de estados ocorre com base no valor do contador e na posição atual (`indice`) no padrão Morse.

Pontos Importantes do Código

Constantes `alfabeto` e `tamanhos`:
Representam os padrões Morse e seus comprimentos:

```
 constant alfabeto : array_morse := (
  0 => "01--", -- A
  1 => "1000", -- B
   ...
);
constant tamanhos : array_int := (
  0 => 2, -- A
  1 => 4, -- B
   ...
);
```

-
- **Gerenciamento de Estados:**
  As transições entre estados são controladas pela FSM, garantindo o envio correto dos pulsos Morse.
- **Reset Assíncrono:**
  O botão `KEY0` redefine os sinais `aceso`, `indice` e `estado` para seus valores iniciais.

VHDL:

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

use IEEE.NUMERIC_STD.ALL;

ENTITY aula8part4 is
    port(
        Num                     : in std_logic_vector(2 downto 0);
        Enable, Reset, Clk   : in std_logic;
        led                     : out std_logic
    );
END aula8part4;
ARCHITECTURE main of aula8part4 is

signal Tempo1 : std_logic_vector(26 downto 0);
signal aceso : std_logic := '0';
signal indice : INTEGER := 0;
signal numero : INTEGER;

signal asyn_reset : std_logic;
signal estado : std_logic_vector(0 to 1) := "00";

type array_morse is array (0 to 7) of std_logic_vector(0 to 3);
type array_int is array (0 to 7) of integer;
constant alfabeto : array_morse := (
    0 => "01--",
    1 => "1000",
    2 => "1010",
    3 => "100-",
    4 => "0---",
    5 => "0010",
    6 => "110-",
    7 => "0000"
);
constant tamanhos : array_int := (
    0 => 2,
    1 => 4,
    2 => 4,
    3 => 3,
    4 => 1,
    5 => 4,
    6 => 3,
    7 => 4
);
begin
    -- 00 = desativado
    -- 01 = intervalo entre bits
    -- 10 = ponto
    -- 11 = traço
```

```
50      process (Clk) is
51      begin
52      if(rising_edge(Clk)) then
53          if (estado = "00") then
54              if(Enable = '0') then
55                  numero <= to_integer(unsigned(Num));
56                  indice <= 0;
57                  Tempo1 <= (others => '0');
58                  estado <= "01";
59              end if;
60
61          elsif (estado = "01") then
62              if(to_integer(unsigned(Tempo1)) < 25000000 ) then
63                  Tempo1 <= std_logic_vector(unsigned(Tempo1) + 1);
64              else
65                  aceso <= '1';
66                  indice <= (indice + 1);
67                  estado <= '1' & alfabeto(numero)(indice);
68                  Tempo1 <= (others => '0');
69              end if;
70
71          elsif (estado = "10") then
72              if(to_integer(unsigned(Tempo1)) < 25000000 ) then
73                  Tempo1 <= std_logic_vector(unsigned(Tempo1) + 1);
74              else
75                  if(indice = tamanhos(numero)) then
76                      estado <= "00";
77                  else
78                      estado <= "01";
79                  end if;
80                  aceso <= '0';
81                  Tempo1 <= (others => '0');
82              end if;
83
84          else
85              if(to_integer(unsigned(Tempo1)) < 75000000 ) then
86                  Tempo1 <= std_logic_vector(unsigned(Tempo1) + 1);
87              else
88                  if(indice = tamanhos(numero)) then
89                      estado <= "00";
90                  else
91                      estado <= "01";
92                  end if;
93                  aceso <= '0';
94                  Tempo1 <= (others => '0');
95              end if;
96          end if;
97
98          if(asyn_reset = '0') then
99              aceso <= '0';
100             indice <= 0;
101             estado <= "00";
102         end if;
103     end if;
```

Conclusão

Ambos os códigos demonstram aplicações distintas de FSMs:

- **Parte I:** Foco em lógica digital para detecção de padrões binários.
- **Parte IV:** Codificação Morse, usando FSMs e lógica sequencial para uma aplicação prática e interativa.