Laboratório 7

Máquinas de estados finitos

Este laboratório tem como objetivo o estudo e implementação de máquinas de estados finitos.

Parte I

Desejamos implementar uma máquina de estados finitos (FSM) capaz de reconhecer duas sequências de símbolos de entrada, em específico: quatro 1's consecutivos ou quatro 0's consecutivos. Para isso, considere uma entrada w e uma saída z, ambas de 1 bit. Sempre que $w = 1$ ou $w = 0$ por quatro pulsos de clock consecutivos, o valor de z deve ser 1; caso contrário, $z = 0$. Sequências que se sobrepõem são permitidas, ou seja, se $w = 1$ por cinco pulsos de clock consecutivos a saída z será igual a 1 após o quarto e quinto pulsos. A Figura 1 ilustra a relação pedida entre w e z.

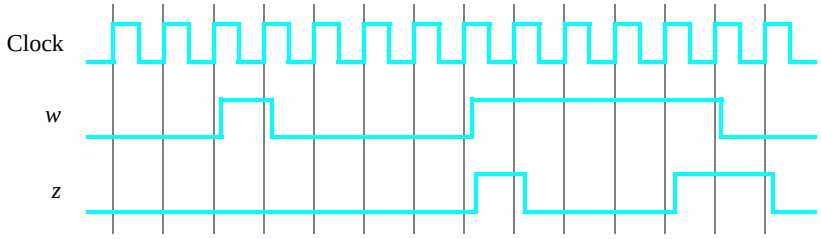

Figura 1: Diagrama de tempo para a saída z.

O diagrama de estados para esta FSM é mostrada na Figura 2. Nesta parte, você deve projetar manualmente um circuito que implemente este diagrama de estados, incluindo as expressões lógicas que definem cada estado dos flip-flops.

Uma opção (que pode facilitar o cálculo das entradas dos flip-flops), é implementar a FSM usando nove flip-flops para os estados, chamados de $y_8, \ldots, y_0$ e a tabela de descrição de estados fornecidos na Tabela 1.

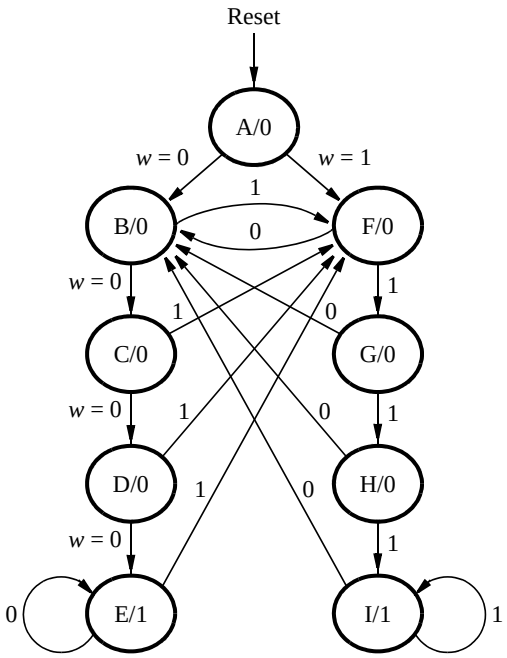

Figura 2: Diagrama de Estados para a FSM.

Nome	Código para o Estado $y_8y_7y_6y_5y_4y_3y_2y_1y_0$
A	000000000
B	000000011
C	000000101
D	000001001
E	000010001
F	000100001
G	001000001
H	010000001
I	100000001

Tabela 1: Códigos para a FSM.

Note que esta codificação não é única, sendo permitido o uso de outras.

Projete e implemente seu circuito na placa, usando um botão como reset, outro como entrada de clock e uma chave como entrada y. Use um LED verde como a saída z e LEDs vermelhos para mostrarem os estados dos flip-flops.

Parte II

Nesta parte você irá escrever outro estilo de código VHDL para a FSM da Figura 2. Nesta versão do código você não deverá definir as expressões lógicas necessárias para cada estado do flip-flop, mas deverá descrever a tabela de estados para a FSM usando uma expressão CASE em um bloco de PROCESS. Use um segundo bloco PROCESS para atribuir o estado dos flip-flops na borda sensível. A saída z pode ser obtida com atribuições lógicas simples

ou com um terceiro bloco PROCESS.

Um esqueleto para o código VHDL é dado na Figura 3. Observe que os vetores de estado presente e futuro da FSP são definidos como um tipo *enumerate* com os possíveis valores sendo os símbolos A até I. O compilador VHDL determina quantos flip-flops usar para o circuito e automaticamente escolhe os códigos para os estados

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part2 IS
    PORT (  . . . define input and output ports
            . . .);
END part2;

ARCHITECTURE Behavior OF part2 IS
    . . . declare signals
    TYPE State_type IS (A, B, C, D, E, F, G, H, I);

    SIGNAL y_Q, y_D : State_type; - - y_Q is present state, y_D is next state
BEGIN
    . . .
    PROCESS (w, y_Q)
    BEGIN
            case y_Q IS
                    WHEN A =>
                            IF (w = '0') THEN y_D <= B;
                            ELSE y_D <= F;
                            END IF;
                    . . . outro estados
            END CASE;
    END PROCESS; - - tabela de estados

    PROCESS (Clock)
    BEGIN
            . . .
    END PROCESS;

    . . . atribuições para a saída z e os LEDS
END Behavior;
```

Figura 3: Esqueleto do código VHDL para a FSM

Implemente e teste este circuito da mesma forma que foi feita na parte anterior.
Envie os dois códigos para o Ensino Aberto.