

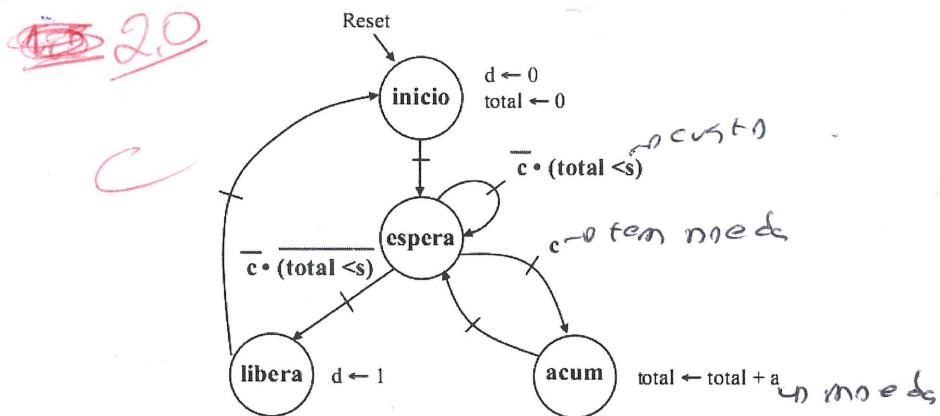
- (1,0) Explique o funcionamento do modelo Bloco de Controle e Bloco Operativa. Diferencie os dois e apresente um exemplo de aplicação.
- (1,0) Analise a figura a seguir.



Que tipo de máquina de estados finitos está representado na figura? Justifique de maneira breve sua resposta.

- (a) Mealy assíncrona
- (b) Mealy síncrona
- (c) Moore
- (d) MacGyver
- (e) Turing

- (3) Com base no diagrama de estados FSM abaixo implemente o código VHDL. As transições dos estados é feita através da borda de subida do clock. Adicionalmente faça um Reset que reinicie a máquina a colocando no estado inicial 'inicio'.



- 4.0 (5) Em uma usina nuclear, o sistema monitora condições críticas do reator com base em duas entradas: X (1 indica condição crítica, como aumento de temperatura ou pressão) e V (1 indica que o sistema de ventilação está operacional; 0 indica falha no sistema de ventilação). O sistema possui duas saídas: T, que ativa o sistema de resfriamento (ex.: inserção de barras de controle), e R, que ativa o sistema de emergência secundário (ex.: liberação de absorventes químicos). Após duas leituras consecutivas de X = 1, T é ativado. Caso X retorne a 0 e V = 1, o sistema volta ao estado normal sem ativar R. Se X permanecer em 1, o sistema entra em modo de "emergência", onde T e R permanecem ativados enquanto X = 1, independentemente de V. Após entrar no modo de emergência, o sistema só retorna ao estado normal após três leituras consecutivas de X = 0 e com V = 1. Durante esse período de transição, se X = 0, R é desativado, mas T continua ativado preventivamente até que o sistema de ventilação esteja operacional (V = 1). Caso V = 0 em qualquer momento, o sistema mantém T ativado como precaução, mesmo na ausência de condições críticas (X = 0). Para o sistema apresente o diagrama de estados, a tabela de saídas e a tabela de transições.

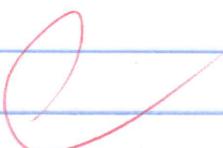
1) O bloco operativo, ou datapath, realiza as operações sobre os dados, através de um sistema alimentado por unidades funcionais (multiplicadores, somadores, ULA), elementos de armazenamento (registradores) e redes de interconexão.

O bloco de controle, ou control, gera os sinais de controle que dão movimento ao comportamento do datapath. Através de uma máquina de estados finita, utiliza informações sobre o estado atual para tomar decisões acerca das próximas ações ou sequências de ações que serão enviadas para o bloco operativo.

Um bom exemplo de aplicação é um sistema de controle de um elevador. O bloco operativo desse sistema vai apenas ativar os parafusos que fazem esse elevador subir ou descer, ligar ou desligar as luzes, abrir ou fechar a porta...

Ademais, o bloco de controle é responsável por decidir quando essas operações devem ser realizadas, quando o elevador está em estado de subida e não vai parar no meio do caminho. Pois o input de parar é para descer. Todas essas decisões são tomadas pelo bloco controlador por meio de lógica sequencial.

2) O modelo de máquina de estados da figura é o modelo de Moore. Pois a lógica final de saída depende apenas do estado atual, sem a intervenção de inputs primários.



3-)

85) Implementar una

entity maquina is port(

clock, c, reset; in STD-logic; in avivado 0 a 1
s, a: in STD-logic-vector (7 down to 0); en orden
d: out STD-logic);

end entity maquina;

architecture behave of maquina is

component b1OP is port(

clock, cTotal, rTotal; in STD-logic; salidas sh
s, a: in STD-logic-vector (7 down to 0);
~~d menor~~: out STD-logic);

end component;

component b1CON is port(

reset, clock, c, menor; in STD-logic;

d, cTotal, rTotal; out STD-logic); sh menor

end component;

signal cTotal, rTotal, menor: STD-logic;

Begin

b-operativo: b1OP port map (b, clock, c, s, a, menor);

b-control: b1CON port map (reset, clock, c, menor);

sh d, cTotal, rTotal, menor);

end behave;

entity b1OP is port(

clock, cTotal, rTotal; in STD-logic;

s, a: in STD-logic-vector (7 down to 0);

menor; out STD-logic);

end entity b1OP;

3-)

Architecture behave of b1DP is

```
signal soma, total: STD-BASIC-VECTOR(7 DOWNTO 0);  
begin  
    res : register port map (clock, cTotal, rTotal, soma, total);  
    soma: somador port map (total, a, soma);  
    menor: menor-que port map (total, s, menor);  
end behave;
```

entity registrador is port(sup_rmem: in std_logic;
 clock, carga, reset: in STD-BASIC-VECTOR(7 DOWNTO 0);
 d: in STD-BASIC-VECTOR(7 DOWNTO 0); -- entrada bits
 q: out STD-BASIC-VECTOR(7 DOWNTO 0)); -- saída bits

```
end entity registrador;
```

Architecture behave of registrador is

```
begin  
    process (clock, reset) begin  
        if (reset = '1') then  
            q <= "00000000";  
        elsif (risings-edge and carga = '1') then  
            q <= d;  
        end if;  
    end process;
```

```
end behave;
```

```

entity somadr is port(
    a, b: in STD-LOGIC-VECTOR(7 DOWNTO 0);
    g: out STD-LOGIC-VECTOR(7 DOWNTO 0)
);
end entity somadr;

```

Architecture behavior of somadr is begin

```

    g <= a + b;
end behave;

```

```

entity memr que is port(
    a, b: in STD-LOGIC-VECTOR(7 DOWNTO 0);
    menor: out STD-LOGIC
);

```

menor = {1:a, 0:b}, worked bns

```

end entity memr que;

```

Architecture behavior of memr que is begin

menor = '1' when A < B else '0'; end behave;

```

end behave;

```

```

entity blCON is port(
    reset, clock, c, menor: in STD-LOGIC;
    d, rTotal, lTotal: out STD-LOGIC
);

```

```

end entity blCON;

```

Architecture behavior of blCON is begin

```

    type state_type is (ini, perA, acum, lib);
    signal state: state_type;

```

```

begin process(clock, reset) begin
    state := ini;
    if (reset = '1') then
        state := lib;
    elsif (clock'event and clock = '1') then
        case state is
            when ini => state := perA;
            when perA => state := acum;
            when acum => state := lib;
        end case;
    end if;
end process;

```

Proxime Folgs

$b \Rightarrow A$

$a \Rightarrow A$, $c \Rightarrow A$, $d \Rightarrow A$; bns

$b \Rightarrow A$, $c \Rightarrow A$, $d \Rightarrow A$; bns

guards of bns

2023 0001128

68

3) Process do clock do RLCON

(state) 200101

if (reset = '1') then

state C = ini;

elsif (clock = rising_edge) = then

case state is

when ini => '1' => totalT₀

state C = pera; totalT₀

when pera =>

~~state~~ '1' => b

if (C = '1') then > totalT₀

state C = accum; T₀

elsif (menor = '0') then

state C = lib; b

else '1' => totalT₀

state C = pera; T₀

end if;

when accum => '1' => b

state C = pera; totalT₀

when lib => '1' => totalT₀

state C = ini;

end case;

end if;

end process;

- Proxima pgina

8511000 8000
3-)

process (state)

Begin

case state is

when ins => void at block of process

 d <= '0'; i = 0; rTotal = 0; cTotal = 0;

 rTotal <= '1'; i = m; rTotal = 0;

 cTotal <= '0'; i = 0;

when pers =>

 d <= '0'; i = 0; rTotal = 0;

 rTotal <= '0'; i = 0; rTotal = 0;

 cTotal <= '0'; i = 0;

when cum => = various states

 d <= '0'; i = 0; rTotal = 0;

 rTotal <= '0'; i = 0; rTotal = 0;

 cTotal <= '1'; i = 0;

when lib =>

 d <= '1'; i = 0; rTotal = 0;

 rTotal <= '0'; i = 0; rTotal = 0;

 cTotal <= '0'; i = 0; rTotal = 0;

end case; i = 0; rTotal = 0;

end process;

end behave

4-)

H-)

		Estado Actual		Estado deseado	
		X	V	Rising	Estado deseado
normal	normal	X	V	rising	normal
emergencia	emergencia	X	V	rising	normal
emergencia	normal	X	V	rising	normal
normal	normal	X	V	rising	normal
emergencia	normal	X	V	rising	normal
normal	emergencia	X	V	rising	normal
emergencia	emergencia	X	V	rising	normal
normal	emergencia	X	V	rising	normal
emergencia	emergencia	X	V	rising	normal
normal	emergencia	X	V	rising	normal
emergencia	normal	X	V	rising	normal
normal	normal	X	V	rising	normal
emergencia	emergencia	X	V	rising	normal

