Instituto Tecnológico de Aeronáutica

Divisão de Ciência da Computação - IEC

Departamento de Sistemas de Computação - IEC-SC

EEA-25 - Sistemas Digitais Programáveis

Prova Nº 2

Prof. André da Fontoura Ponchet

São José dos Campos - SP

22 de novembro de 2023

Sumário

INSTRUÇÕES:

- A prova deve ser resolvida individualmente.
- É permitida consulta **APENAS** ao material disponibilizado no site da disciplina.
- Não é permitido o uso de ferramentas baseadas em inteligência artificial.
- Apresente a solução das questões de forma clara, justificando todos os passos.
- Crédito parcial será concedido caso uma solução incompleta seja apresentada.
- O arquivo .pdf contendo as soluções das questões deve estar **LEGÍVEL.**
- Boa prova!

1 QUESTÃO 1 (2,5)

Design a 4-bit serial bit sequencer detector by hand. The input to your state detector is called ***DIN*** and the output is called ***FOUND***. Your detector will assert FOUND anytime there is a 4-bit sequence of 0101. For all other input sequences, the output is not asserted.

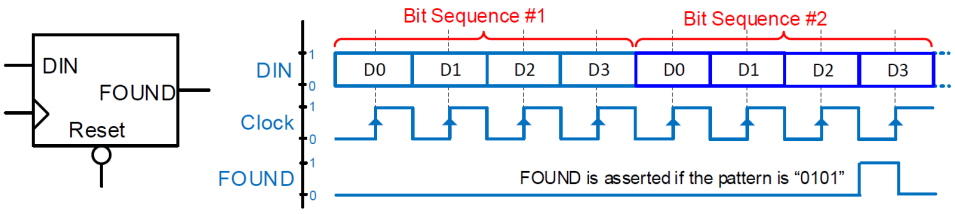

Figura 1: Block Diagram and signals of question 1

In your design, provide the following:

A. The state diagram for this FSM.

B. The number of flip-flops necessary to implement the state memory for this FSM.

C. The state transition table for this FSM.

D. The combinational logic expressions for the ***next state logic***.

E. The combinational logic expressions for the ***output logic***.

F. The logic diagram for this FSM.

2 Questão 2 (2,5)

Design a 20 cent vending machine controller by hand. Your controller will take in nickels and dimes and dispense a product anytime the customer has entered at least 20 cents. Your FSM has two inputs, ***Nin*** and ***Din***. Nin is asserted whenever the customer enters a nickel while Din is asserted anytime the customer enters a dime. Your FSM has two outputs, ***Dispense*** and ***Change***. Dispenser is asserted anytime the customer has entered more than 20 cents and needs a nickel in change. In your design, provide the following:

A. The state diagram for this FSM.

B. The number of flip-flops necessary to implement the state memory for this FSM.

C. The state transition table for this FSM.

D. The combinational logic expressions for the ***next state logic***.

E. The combinational logic expressions for the ***output logic***.

F. The logic diagram for this FSM.

3 Questão 3 (0,5)

What allows a finite state machine to make more intelligent decisions about the system outputs compared to combinational logic alone?

A. A finite machine has knowledge about the past inputs.

B. The D-flip-flops allow the output to be generated more rapidly.

C. The next stage and output logic allows the finite state machine to be more complex and implement larger truth tables.

D. A synchronous system is always more intelligent.

4 Questão 4 (0,5)

When designing a finite state machine, many of the details of the implementation can be abstracted. At what design step do the details of the implementation start being considered?

A. The state diagram step.

B. The state transition diagram step.

C. The state memory synthesis step.

D The word description.

5 QUESTÃO 5 (0,5)

Which of the following statements about the next state logic is FALSE?

A It is always a combinational logic.

B It always uses the current state as one of its inputs.

C Its outputs are connected to the D inputs of the D-flip-flops in the state memory.

D It uses the results of the output logic as part of its inputs.

6 QUESTÃO 6 (3,5)

In this exercise, you code a moderately difficult Verilog design for synthesis. Read the specification first and then follow the instructions in the lab section. Please make sure to use the same variable name as the ones shown in block diagrams.

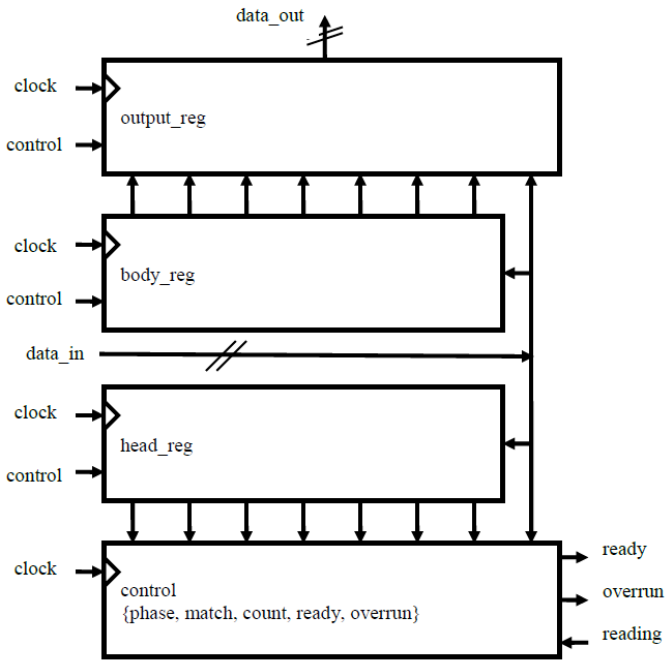

Figura 2: Block Diagrams of question 6

SPECIFICATIONS:

- data_in, reading, clk, and reset are single-bit input signals.
- data_out is an 8-bit output signal, whereas ready and overrun are 1-bit output.
- The design is clocked on the rising edge of clk.

- Every serial bit transmission has a preamble that needs to be sent before sending the character. For this design, the preamble value is fixed to 8'hA5.
- After the preamble, the ready signal will be high.

DESIGNING A SERIAL-TO-PARALLEL INTERFACE:

- Examine the following files provided by the professor: rcvr.v (incomplete receiver model), rcvr_test.v (receiver testbench).
- The receiver searches the serial input stream for a match character, and when found, loads the next serial input character to the output buffer, asserts a "ready" flag, and immediately searches for the next match. Modeling the individual parts of the receiver is like modeling individual components.
- In your favorite editor, modify the rcvr.v file to complete the receiver definition.

Check your design using the following commands on the terminal:

$$\begin{cases} iverilog\ rcvr\_test.v \\ vvp\ a.out \end{cases}$$

You should see the following messages on the terminal:

```
Message sending: I Love Verilog
At time 35: Put character "I"
At time 39: Got character "I"
At time 77: Put character ""
At time 95: Got character ""
At time 117: Put character "L"
At time 127: Got character "L"
At time 153: Put character "o"
At time 175: Got character "o"
At time 193: Put character "v"
At time 207: Got character "v"
At time 231: Put character "e"
At time 257: Got character "e"
At time 275: Put character ""
At time 297: Got character ""
At time 309: Put character "V"
At time 313: Got character "V"
At time 349: Put character "e"
At time 365: Got character "e"
At time 391: Put character "r"
At time 423: Got character "r"
```

```
At time 425: Put character "i"
At time 433: Got character "i"
At time 459: Put character "l"
At time 487: Got character "l"
At time 495: Put character "o"
At time 507: Got character "o"
At time 537: Put character "g"
At time 563: Got character "g"
Message received: I Love Verilog
TEST DONE
```