Instituto Tecnológico de Aeronáutica

Divisão de Ciência da Computação - IEC

Departamento de Sistemas de Computação - IEC-SC

EEA-25 - Sistemas Digitais Programáveis

Prova Nº 1

Prof. André da Fontoura Ponchet

São José dos Campos - SP

19 de setembro de 2023

Sumário

Instruções:

- A prova deve ser resolvida individualmente.
- É permitida consulta **APENAS** ao material disponibilizado no site da disciplina.
- 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 (1,5)

Um museu possui 3 salas, cada uma com um sensor de movimento (S0, S1 e S2) que emite um sinal lógico 1 quando um movimento é detectado. À noite, a única pessoa no museu é um segurança que anda de sala em sala para verificar a presença de intrusos. Crie um circuito que soe um alarme (saída A em nível lógico 1) se algum movimento for detectado em mais de uma sala ao mesmo tempo. Apresente:

A. A tabela verdade e a equação lógica do sistema.
B. Um circuito lógico que implemente a equação apresentada no item a.
C. Um código Verilog que implemente o circuito apresentado no item b.

2 Questão 2 (1,5)

Crie um circuito para o museu da questão 1 que detecte se o guarda está patrulhando adequadamente o museu, detectado por apenas um sensor de movimento em nível lógico 1.

A. A tabela verdade e a equação lógica do sistema.
B. Um circuito lógico que implemente a equação apresentada no item a.
C. Um código Verilog que implemente o circuito apresentado no item b.

3 Questão 3 (2,0)

Dada a equação lógica abaixo,

$$S = \overline{X_1} X_2 \overline{X_3} + \overline{X_1} X_3 \overline{X_4} + X_1 \overline{X_2 X_3} X_4 + X_1 \overline{X_2} X_3 \overline{X_4} + \overline{X_1 X_2 X_3 X_4}$$

encontre:

A. A equação lógica simplificada.

B. O circuito lógico que implementa a equação simplificada no item A.

C. Um código Verilog que implemente (modelamento estrutural) o circuito no item B. Apresente o código comentado passo-a-passo.

D. O testbench para verificar a funcionalidade do circuito. Apresente os sinais no Gtkwave.

4 QUESTÃO 4 (2,5)

A tabela verdade de um somador *full adder* é apresentada conforme a Tabela 1:

Tabela 1: Tabela da Questão 4.

a_i	b_i	cin_{i-1}	$cout_i$	sum_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

O diagrama de bloco do somador é apresentado na Figura 1:

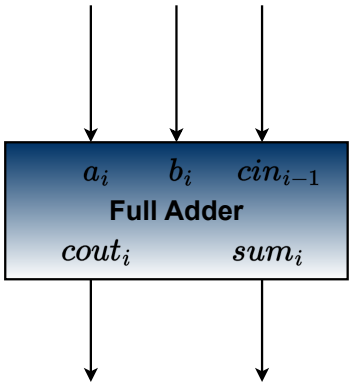

Figura 1: Full adder questão 4.

Com base na tabela e no diagrama apresentados, obtenha:

A. A equação lógica que implementa as saídas $cout_i$ e sum_i.

B. Um código Verilog (estrutural) que implemente o samador. Apresente o código comentado passo-a-passo.

C. O testbench para verificar a funcionalidade do somador. Apresente os sinais no Gtkwave.

5 QUESTÃO 5 (2,5)

Uma aplicação usual do *ripple counter* é a implementação de um divisor de *clock* programável. Em um *ripple counter*, o bit(0) tem uma frequência que é exatamente 1/2 do *clock* de entrada, o bit (1) tem um frequência que é exatamente 1/4 da entrada *clock*, o bit (2) tem uma frequência que é exatamente 1/8 do *clock* de entrada, etc. Este comportamento pode ser explorado para criar um *clock* que é dividido por múltiplos de 2n, selecionando-se um bit específico do contador. A configuração típica deste divisor de clock programável é rotear cada bit do contador para uma entrada de um multiplexador. Esta arquitetura é mostrada na Figura 2:

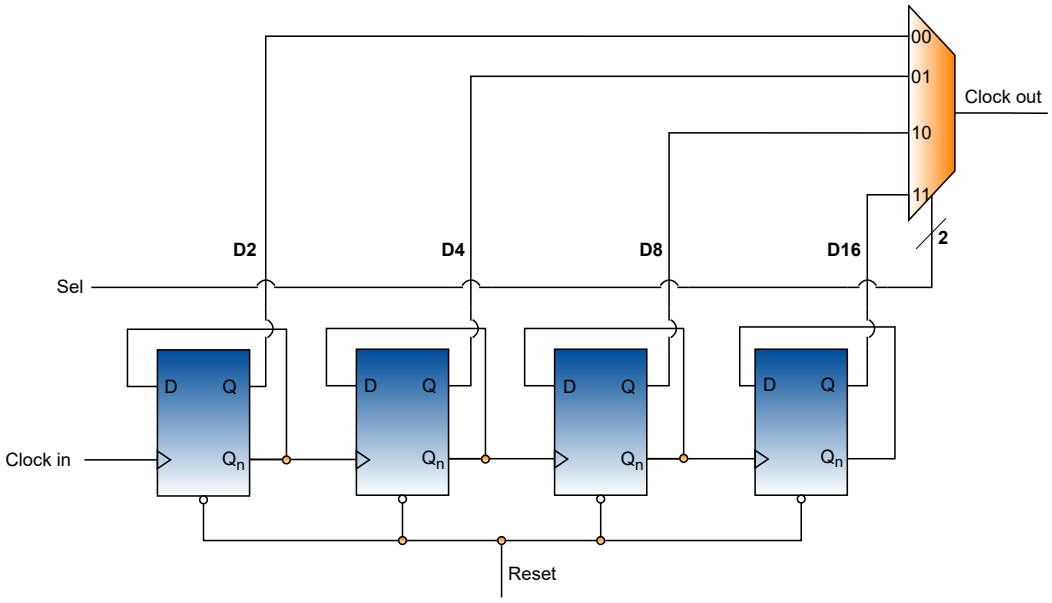

Figura 2: Dividor de clock da Questão 5.

Escreva um código em Verilog para implementar o divisor de *clock* programável apresentado na Figura 2, usando atribuição contínua (*continuous assignment*). Apresente os sinais no Gtkwave. Use um *ripple counter* de 4 bits para produzir quatro versões divididas do *clock* (1/2, 1/4, 1/8 e 1/16). Seu sistema deverá ser capaz de selecionar qual versão do *clock* será apresentada na saída (sinal *sel* de 2 bits). Instancie no seu código o modelo do *D-flip-flop* fornecido abaixo para implementar o *ripple counter*:

```
module dflipflop( output reg Q, Qn,
input wire Clock, Reset, D);

always @ (posedge Clock or negedge Reset)
begin
if (!Reset)
begin
Q <= 1'b0;
Qn <= 1'b0;
end
```

```
else
begin
Q <= D;
Qn <= ~D;
end
end

endmodule
```