

**Projeto para Excelência em Microeletrônica (PEM)**

**Resolução da segunda Lista de atividades abordando**

**SystemVerilog**

Professor: Marcos Morais

Coach: Antonio Agripino

Aluno: Rodrigo Farias Oliveira

# 1ª Questão

**Código**

module Quest1 (input logic swap, input logic enable,

input logic clock,

output logic[3:0] upcount,

utput logic[3:0] downcount);

always\_ff @(posedge clock)

if (enable)

if (swap) begin

upcount <= downcount;

downcount <= upcount;

end else begin

upcount <= upcount + 4'd1;

downcount <= downcount - 4'd1;

end

else begin

upcount <= upcount;

downcount <= downcount;

end

endmodule

# 2ª Questão

**Código**

module Quest2 (input logic clock,

output logic f);

logic[8:0] count;

always\_ff @(posedge clock) begin

if (count == 9'd499) count = 9'd0;

else count = count + 9'd1;

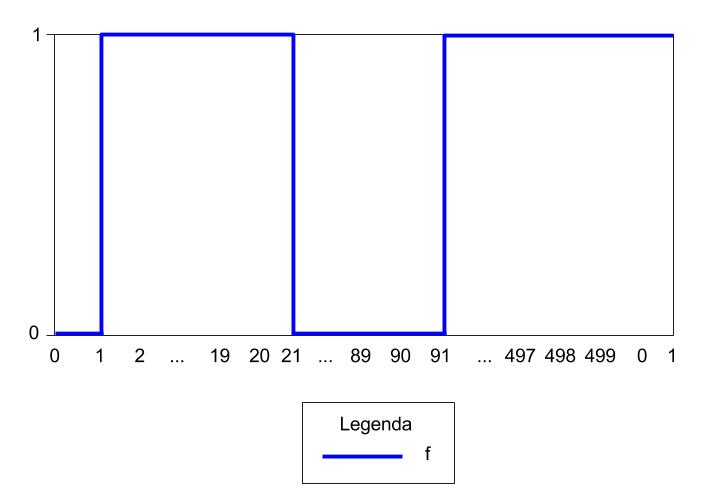
if (count > 9'd19 && count < 9'd90) f = 1'b0;

else f = 1'b1;

end

endmodule

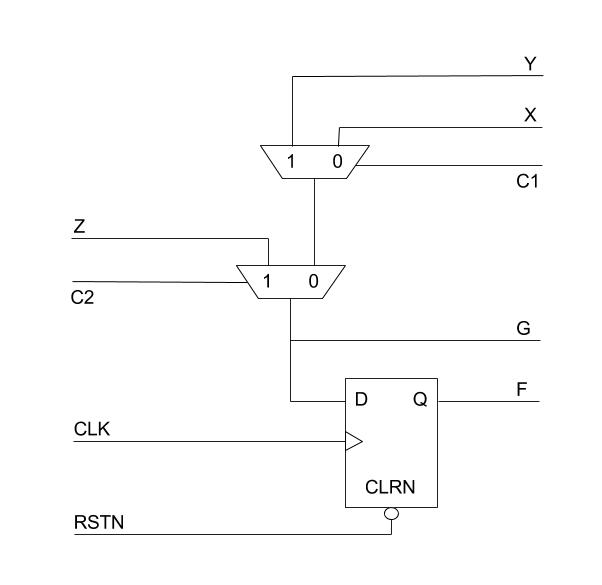
**Formas De Onda**



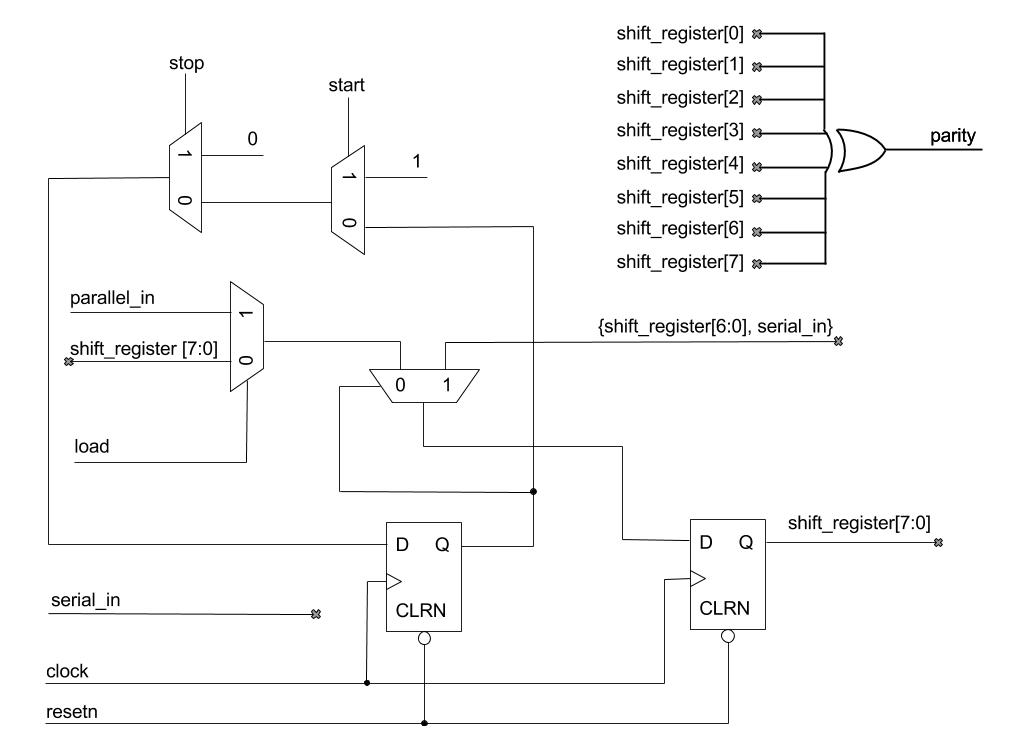
**Resposta**

A frequência do sinal de saída em f é de 0,2 Mhz.

# 3ª Questão

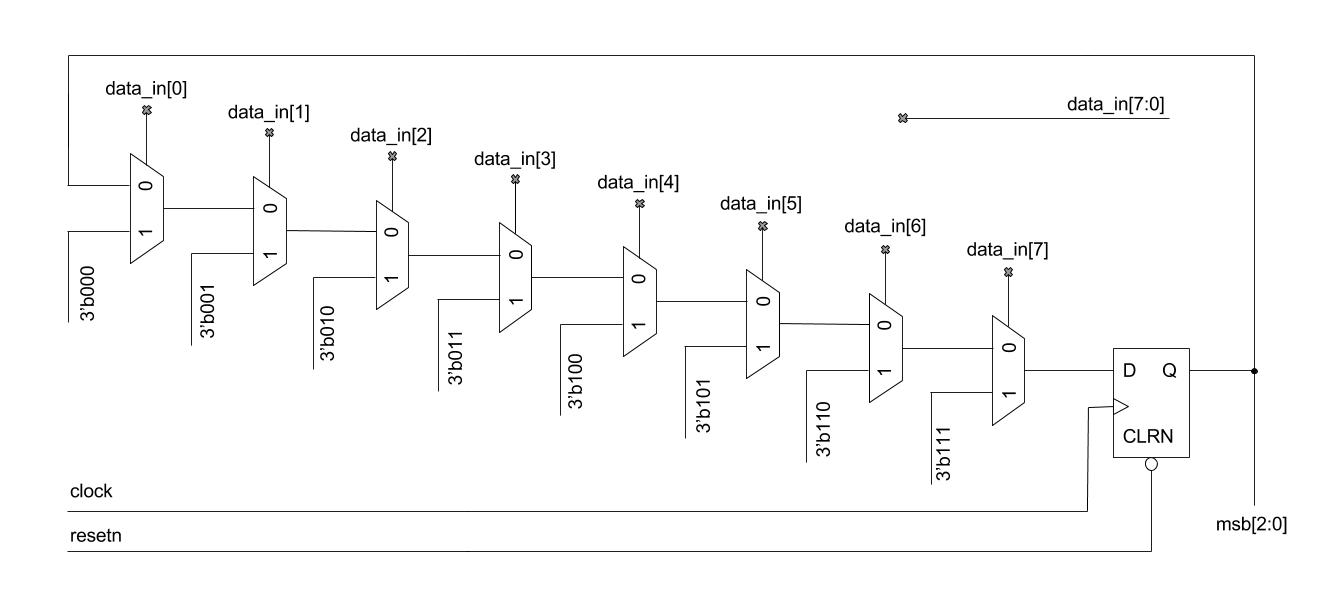


# 4ª Questão



# 5ª Questão

Sim este código é sintetizável, segue a imagem do circuito lógico:



Sim é possível substituir o for pelo *while*, o código continuará sintetizável e o *while* fará a mesma função do for. Todo o código ficaria da seguinte forma:

module Quest5 (input logic resetn, clock,

input logic [7:0] data\_in,

output logic [2:0] msb);

integer i;

always\_ff @(posedge clock or negedge resetn)

begin

if (!resetn) begin

msb <= 3'b000;

end

else begin

i = 0;

while (i < 8) begin

if (data\_in[i]) msb <= i;

i++;

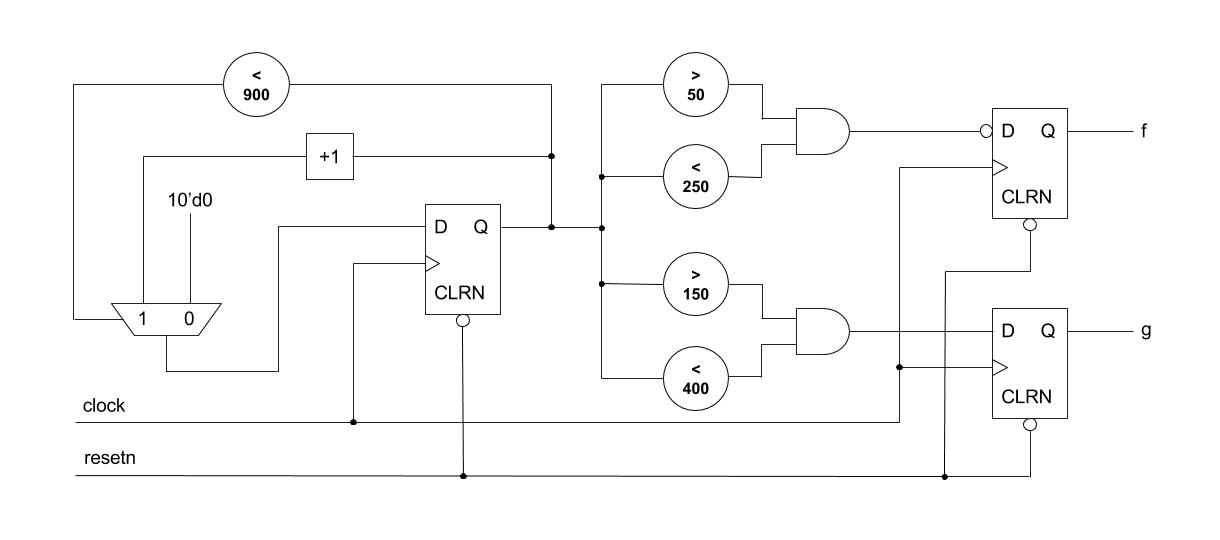
end

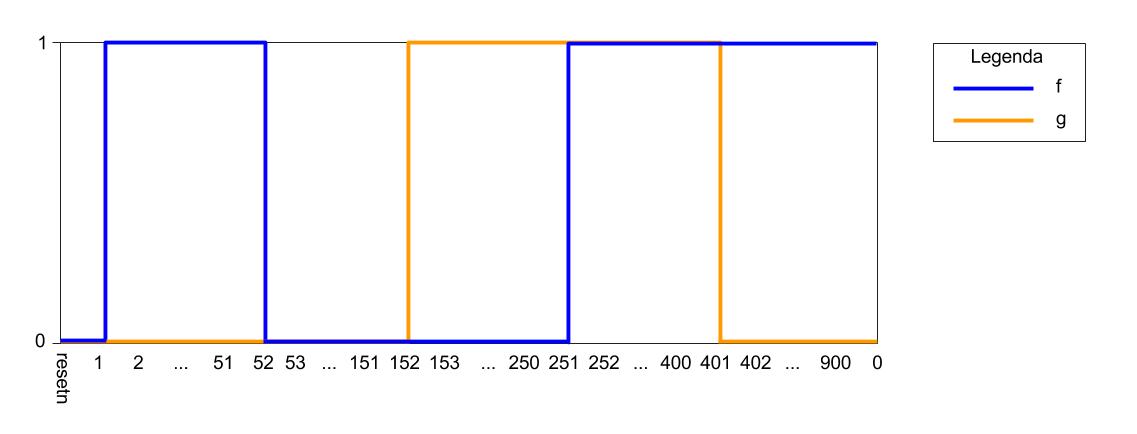
end

end

endmodule

# 6ª Questão

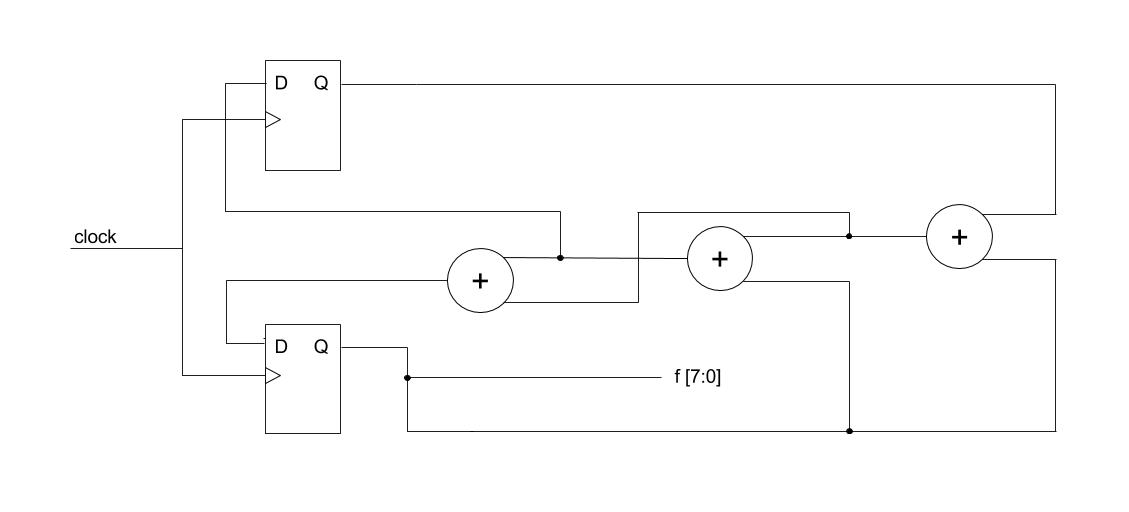




**Resposta**

Caso o sinal de *clock* seja de 200 Mhz teremos uma frequência de sinal de saída em f e g de 0,22 Mhz.

# 7ª Questão



**Resposta**

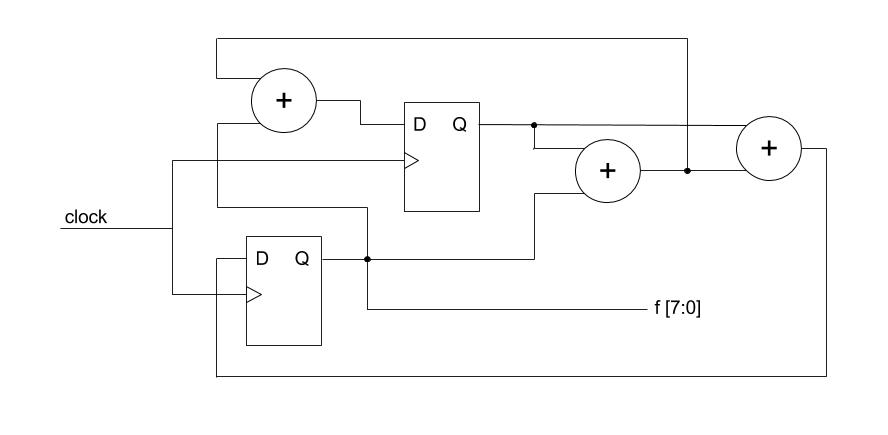
Neste código ocorre o seguinte:

“a” recebe o valor de “b” adicionado do valor de “c” (que foi atualizado após o pulso do *clock*) naquele mesmo momento;

“b” recebe o valor de “c” (que foi atualizado após o pulso do *clock*) adicionado do valor de “a” (o qual acabou de ser alterado anteriormente) naquele mesmo momento;

“c” receberá a soma de “a” e “b” (os quais acabaram de ser alterados anteriormente) apenas no próximo pulso de *clock*.

# 8ª Questão



**Resposta**

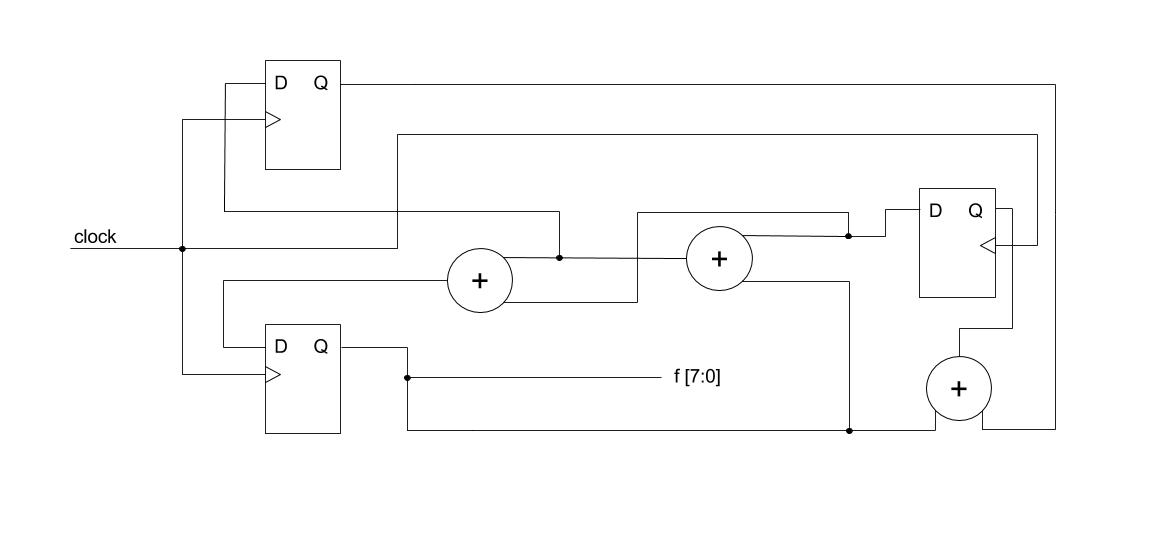
Neste código ocorre o seguinte:

“a” recebe o valor de “b” adicionado do valor de “c” naquele mesmo momento;

“c” recebe o valor de “a” (o qual acabou de ser alterado anteriormente) adicionado do valor de “b” (que foi atualizado após o pulso do *clock*) naquele mesmo momento;

“b” receberá o valor de “c” (o qual acabou de ser alterado anteriormente) adicionado do valor de “a” (o qual acabou de ser alterado anteriormente) naquele mesmo momento.

# 9ª Questão



**Resposta**

“b” recebe o valor de “c” (não alterado ainda) adicionado do valor de “a” (que foi atualizado após o pulso do *clock*) naquele mesmo momento;

“c” recebe o valor de “a” (que foi atualizado após o pulso do *clock*) adicionado do valor de “b” (o qual acabou de ser alterado anteriormente) naquele mesmo momento;

“a” receberá a soma de “b” e “c” (os quais acabaram de ser alterados anteriormente) apenas no próximo pulso de *clock*.

# 10ª Questão