9o Projeto - Contador Síncrono

**Integrantes do grupo:** Lara Gama Santos, Mateus Ribeiro Ferraz, Sulamita Ester Costa.

## 1o parte:

Contador síncrono crescente de 4 bits:

- **Tabela de transição de estados:**

| **Estados atuais** | | | | **Estados futuros** | | | | **Flip-flop** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Q3a | Q2a | Q1a | Q0a | Q3f | Q2f | Q1f | Q0f | D3 | D2 | D1 | D0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Expressões booleanas:**

Não foi necessário utilizar expressões booleanas, porque utilizamos descrição comportamental para criar o projeto contador síncrono crescente de 4 bits em verilog HDL.

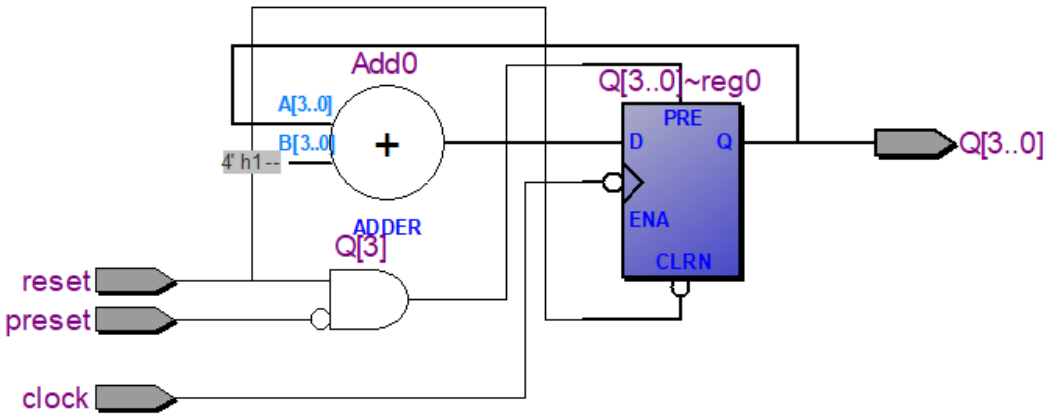
- **Lógica do código:**

Colocamos como entrada o clock, o reset e preset. O clock está em borda de descida, ou seja, a cada pulso soma-se 1 bit à variável Q. O reset coloca a variável Q em 0000 e o preset coloca Q em 1111. Além disso, criamos a saída Q como uma variável do tipo reg para que a saída possa ser modificada constantemente pelo procedimento, visto que initial e always só podem modificar valores do tipo registrador.

- **A descrição do circuito em Verilog HDL no Quartus II:**

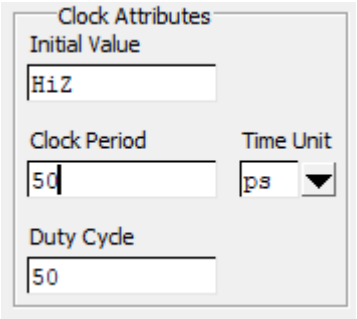
```
module contador_4bits (clock, reset, preset, Q);
    input clock, reset, preset;
    output reg [3:0] Q;

    initial
      begin
        Q= 4'b0000;
      end

    always @(negedge clock or negedge reset or negedge preset)
      begin
        if (reset == 1'b0)
        begin
            Q= 4'b0000;
        end

        else if (preset == 1'b0)
        begin
            Q= 4'b1111;
        end

         else
         begin

         Q = Q + 1;

         end
     end
endmodule
```

- **Esquemático do circuito:**

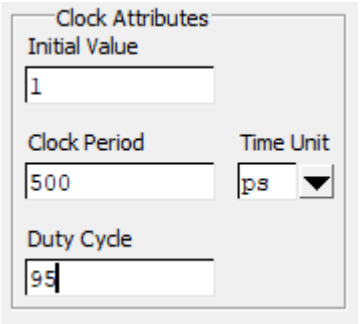



- **Períodos do Clock, preset e reset:**

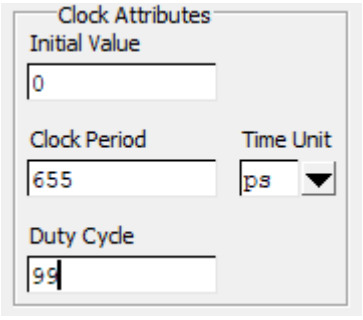
Clock:



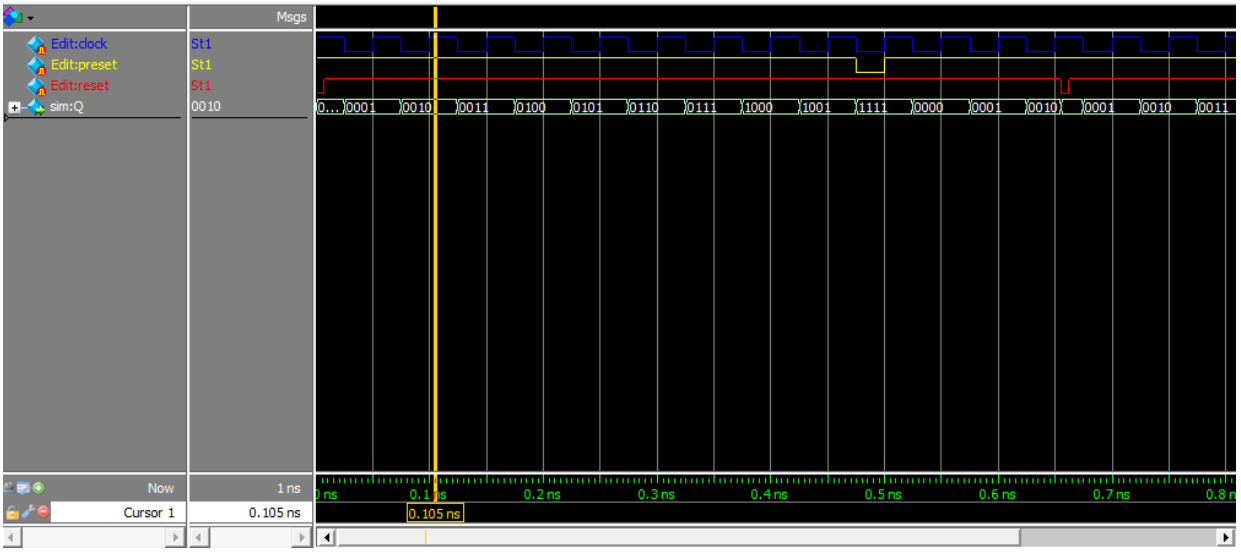

Preset:

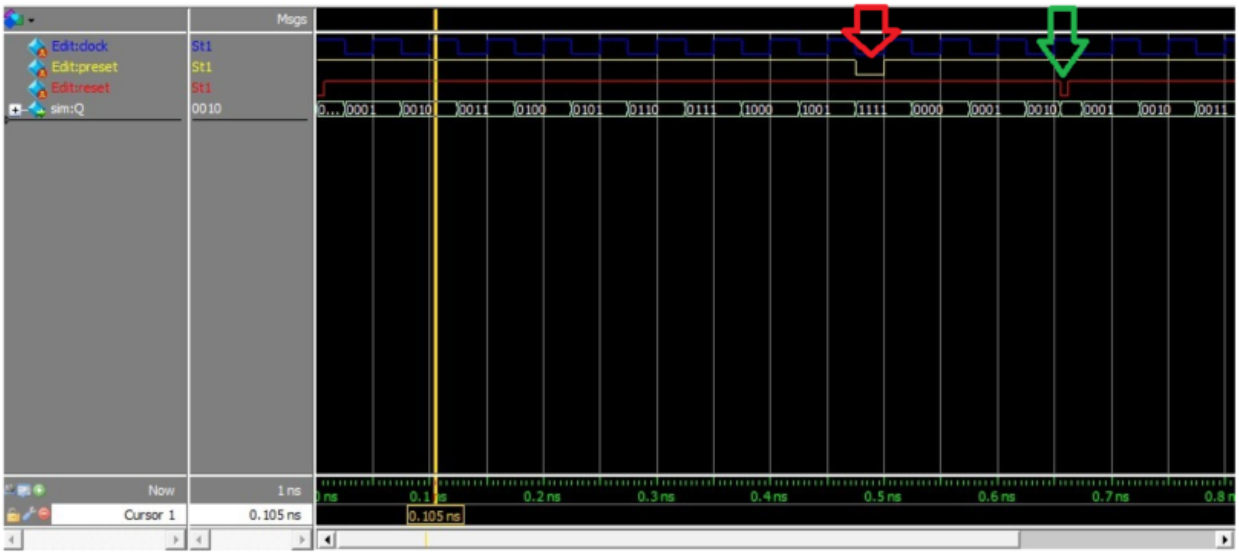



Reset:



- **Simulação do circuito no ModelSIM:**




LEGENDA:

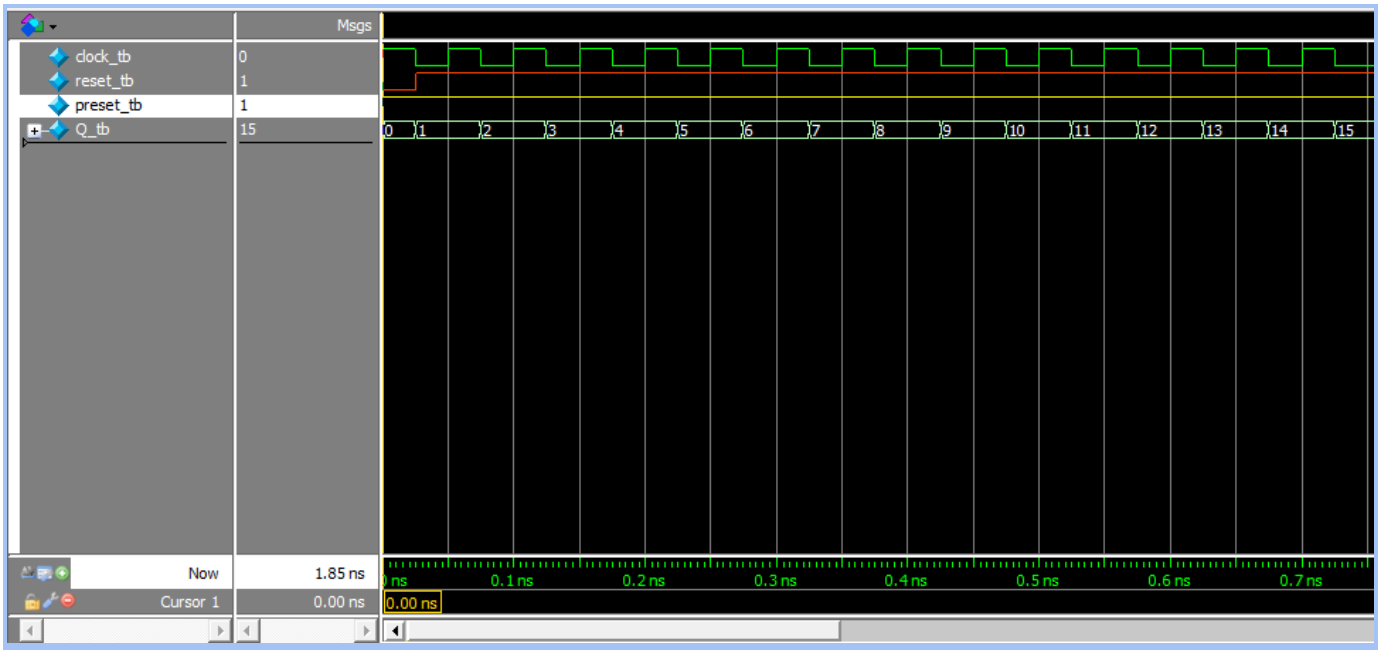
- A seta em vermelho representa o preset
- A seta em verde representa o reset

- **Análise sobre os resultados obtidos:**

Como podemos observar na simulação gerada pelo ModelSIM, temos que, a cada pulso do clock (em borda de descida) é somado 1 ao número binário de 4 bits (Q). A contagem inicia em 0 e continua até atingir o número 15. Em seguida, reinicia em 0. Além disso, quando a variável “preset” está em nível lógico baixo, todos os bits de Q são alterados para 1, enquanto que, quando a variável “reset” está em nível lógico baixo, todos os bits de Q são colocados em 0.

- **Testbench**

```
10  module contador_4bits_tb;
11
12     reg clock_tb, reset_tb, preset_tb;
13     wire [3:0] Q_tb;
14
15     contador_4bits dut(clock_tb, reset_tb, preset_tb, Q_tb);
16
17    initial
18
19    begin
20     reset_tb = 1'b0; preset_tb = 1'b1; clock_tb = 1'b1; #25
21     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
22     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
23     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
24     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
25     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
26     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
27     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
28     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
29     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
30     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
31     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
32     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
33     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
34     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
35     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
36     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
37     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
38     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
39     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
40     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
41     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
42     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
43     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
44     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
45     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
46     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
47     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
48     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
49     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25;
50  end
51
52  endmodule
```




## 2o parte:

Contador síncrono crescente de 4 bits que conte de 0 a 9:

- **Tabela de transição de estados:**

| **Estados atuais** | | | | **Estados futuros** | | | | **Flip-flop** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Q3a | Q2a | Q1a | Q0a | Q3f | Q2f | Q1f | Q0f | D3 | D2 | D1 | D0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

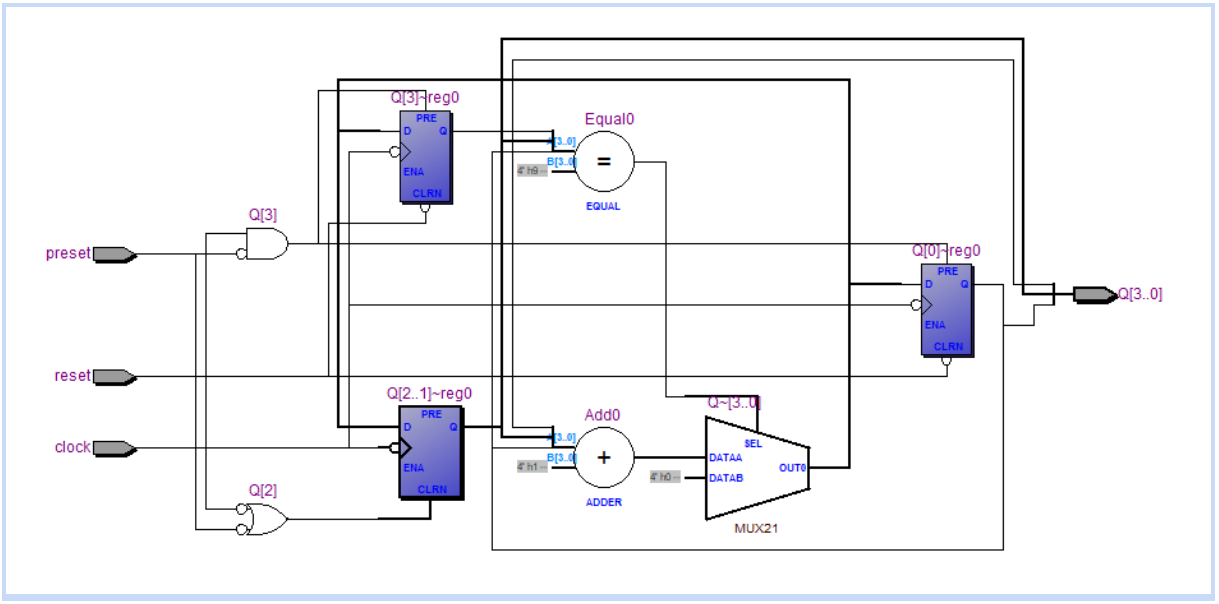
- **Expressões booleanas:**

Não foi necessário utilizar expressões booleanas, porque utilizamos descrição comportamental para criar o projeto contador síncrono crescente de 4 bits em verilog HDL.

- **A descrição do circuito em Verilog HDL no Quartus II:**

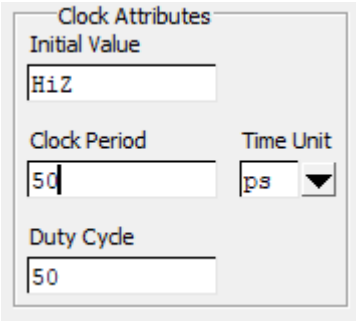
```
 9  module contador_modulo10 (clock, reset, preset, Q);
10      input clock, reset, preset;
11      output reg [3:0] Q;
12
13      initial
14        begin
15          Q= 4'b0000;
16        end
17
18      always @(negedge clock or negedge reset or negedge preset)
19        begin
20          if (reset == 1'b0)
21          begin
22              Q= 4'b0000;
23          end
24
25          else if (preset == 1'b0)
26          begin
27              Q= 4'b1001;
28          end
29
30          else if (Q == 4'b1001) //quando o contador atinge o valor 9 em decimal, deve ser reiniciado para zero
31          begin
32              Q= 4'b0000;
33          end
34
35          else
36          begin
37
38          Q = Q + 1;
39
40          end
41      end
42  endmodule
```

- **Esquemático do circuito:**

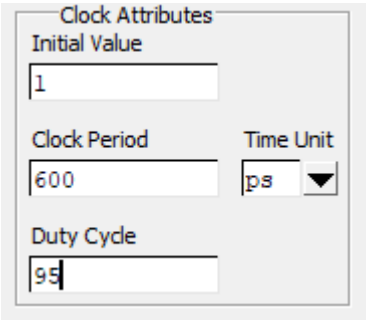



- **Períodos do clock, preset e reset:**

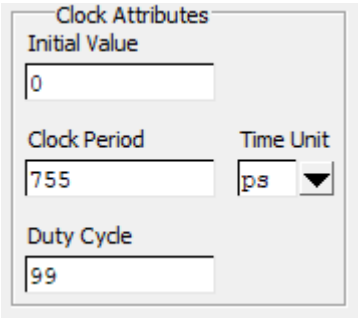
Clock:



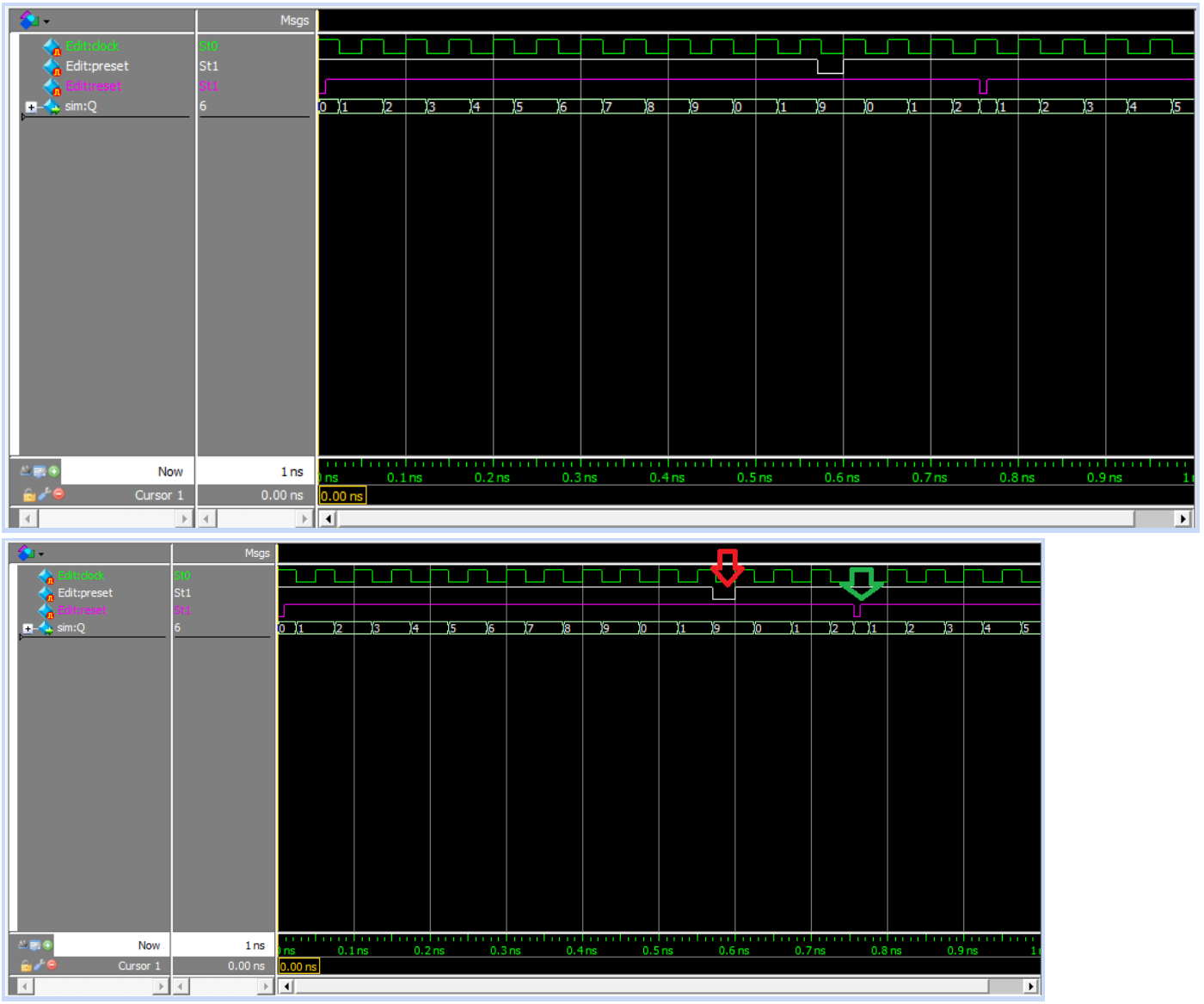

Preset:




Reset:




- **Simulação do circuito no ModelSIM:**




LEGENDA:

- A seta em vermelho representa o preset
- A seta em verde representa o reset

- **Análise sobre os resultados obtidos:**

Este circuito funciona de maneira semelhante ao circuito realizado na parte 1. A variável Q é acrescida de 1 bit a cada borda de descida do clock. Entretanto, a contagem inicia em zero e vai até nove (o contador é de módulo 10). Para tal, foi adicionada uma nova condição ao código responsável por reiniciar a contagem quando Q atinge o valor 9. Além disso, quando o preset atinge nível lógico baixo, Q assume o valor de 1001.

- **Testbench**

```
10  module contador_4bits_tb;
11
12     reg clock_tb, reset_tb, preset_tb;
13     wire [3:0] Q_tb;
14
15     contador_modulo10 dut(clock_tb, reset_tb, preset_tb, Q_tb);
16
17    initial
18
19    begin
20     reset_tb = 1'b0; preset_tb = 1'b1; clock_tb = 1'b1; #25
21     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
22     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
23     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
24     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
25     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
26     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
27     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
28     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
29     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
30     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
31     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
32     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
33     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
34     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
35     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
36     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
37     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
38     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
39     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
40     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
41     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
42     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
43     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
44     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
45     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
46     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
47     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25
48     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b1; #25
49     reset_tb = 1'b1; preset_tb = 1'b1; clock_tb = 1'b0; #25;
50  end
51
52  endmodule
```