


### 1º Projeto:

Objetivo: Projetar um circuito digital sequencial para um contador síncrono crescente de 4 bits que conte de 0 até 15 (módulo 16). Utilize uma variável RESET e PRESET para o usuário inicializar os Flip-Flops.

#### Código para o circuito contador de módulo 16:

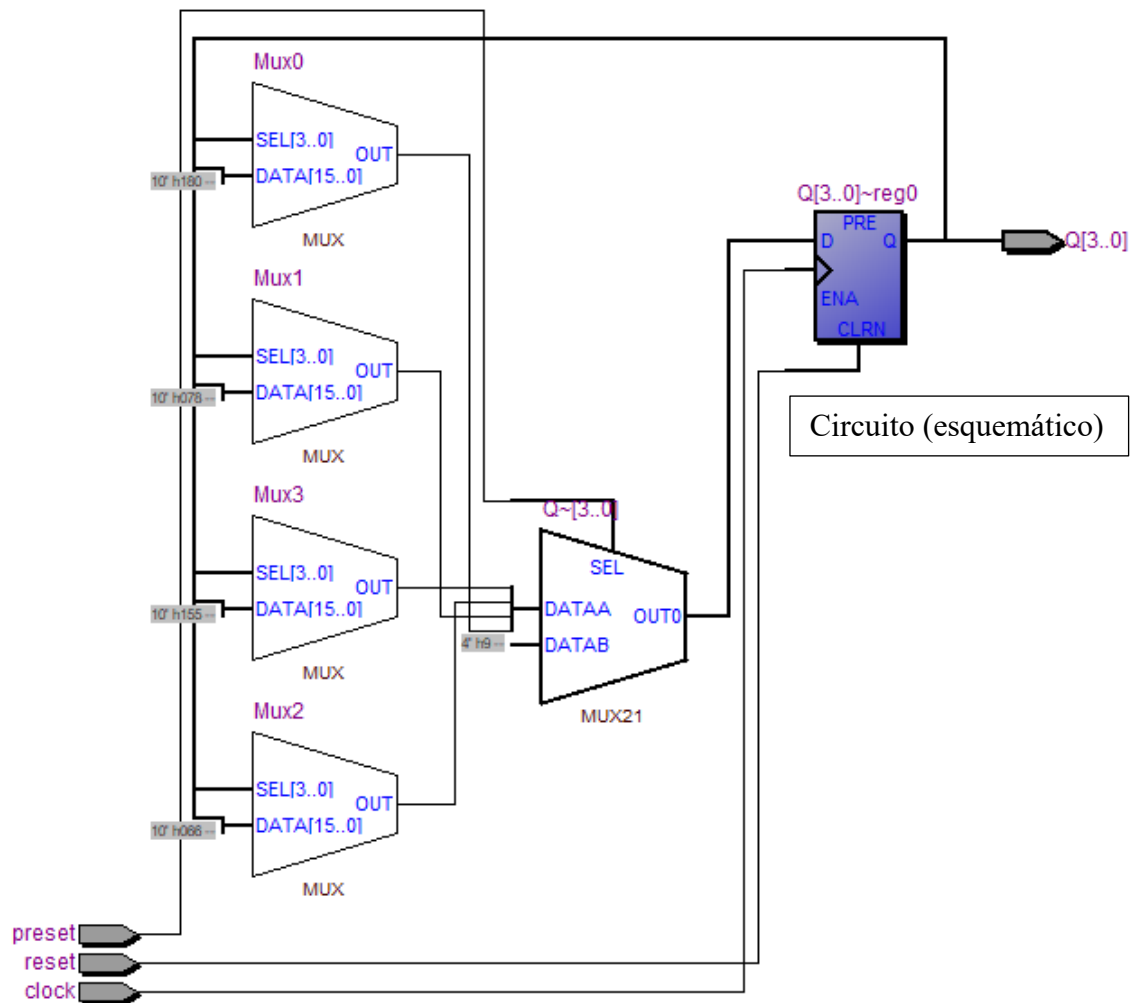
```
module contador(clock, reset, preset, Q);
  input clock, reset, preset; output reg [3:0] Q;
  always@(posedge clock or posedge reset)
  begin
    if(reset)
    begin
      Q <= 4'b0000;
    end //if
    else if(preset)
    begin
      Q <= 4'b1111;
    end //if
    else
    begin
      case (Q)
        4'b0000: Q <= 4'b0001;
        4'b0001: Q <= 4'b0010;
        4'b0010: Q <= 4'b0011;
        4'b0011: Q <= 4'b0100;
        4'b0100: Q <= 4'b0101;
        4'b0101: Q <= 4'b0110;
        4'b0110: Q <= 4'b0111;
        4'b0111: Q <= 4'b1000;
        4'b1000: Q <= 4'b1001;
        4'b1001: Q <= 4'b1010;
        4'b1010: Q <= 4'b1011;
        4'b1011: Q <= 4'b1100;
        4'b1100: Q <= 4'b1101;
        4'b1101: Q <= 4'b1110;
        4'b1110: Q <= 4'b1111;
        4'b1111: Q <= 4'b0000;
      endcase
    end //else
  end //always
endmodule
```

Atribuição não-bloqueante  
( <= )

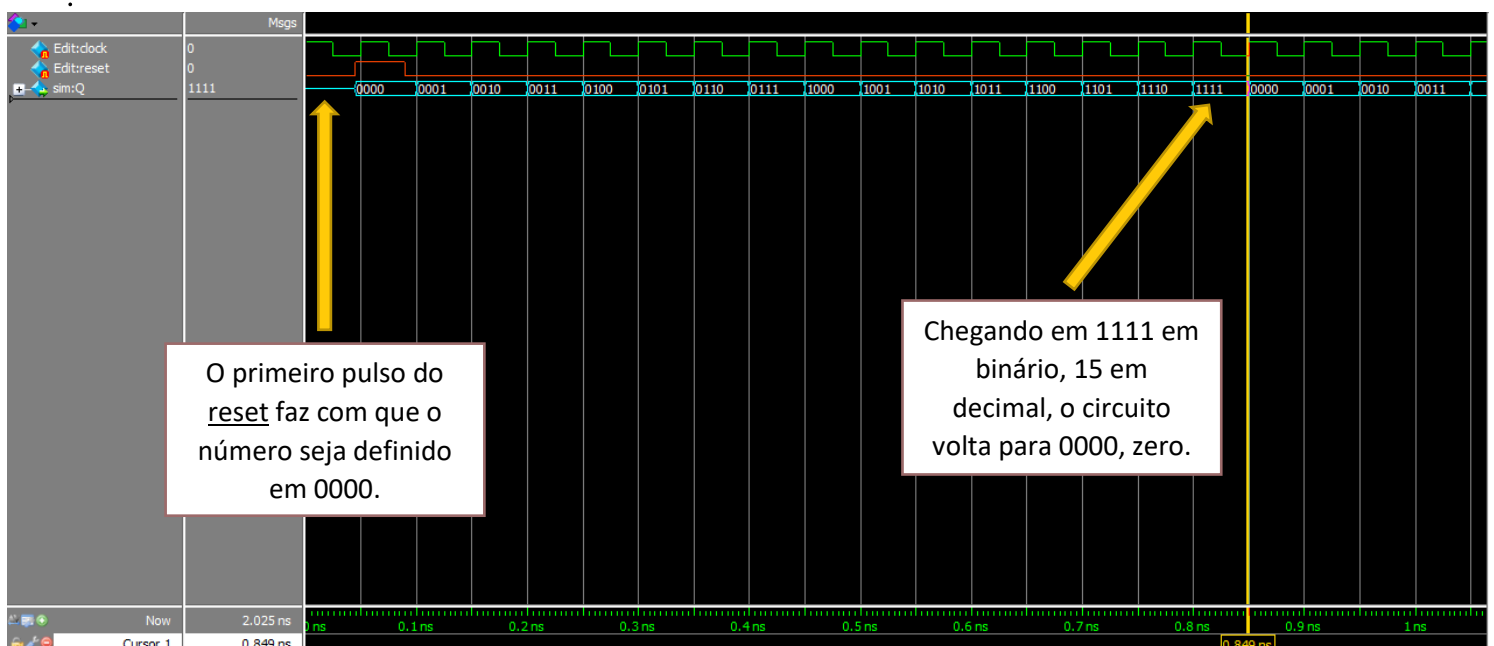


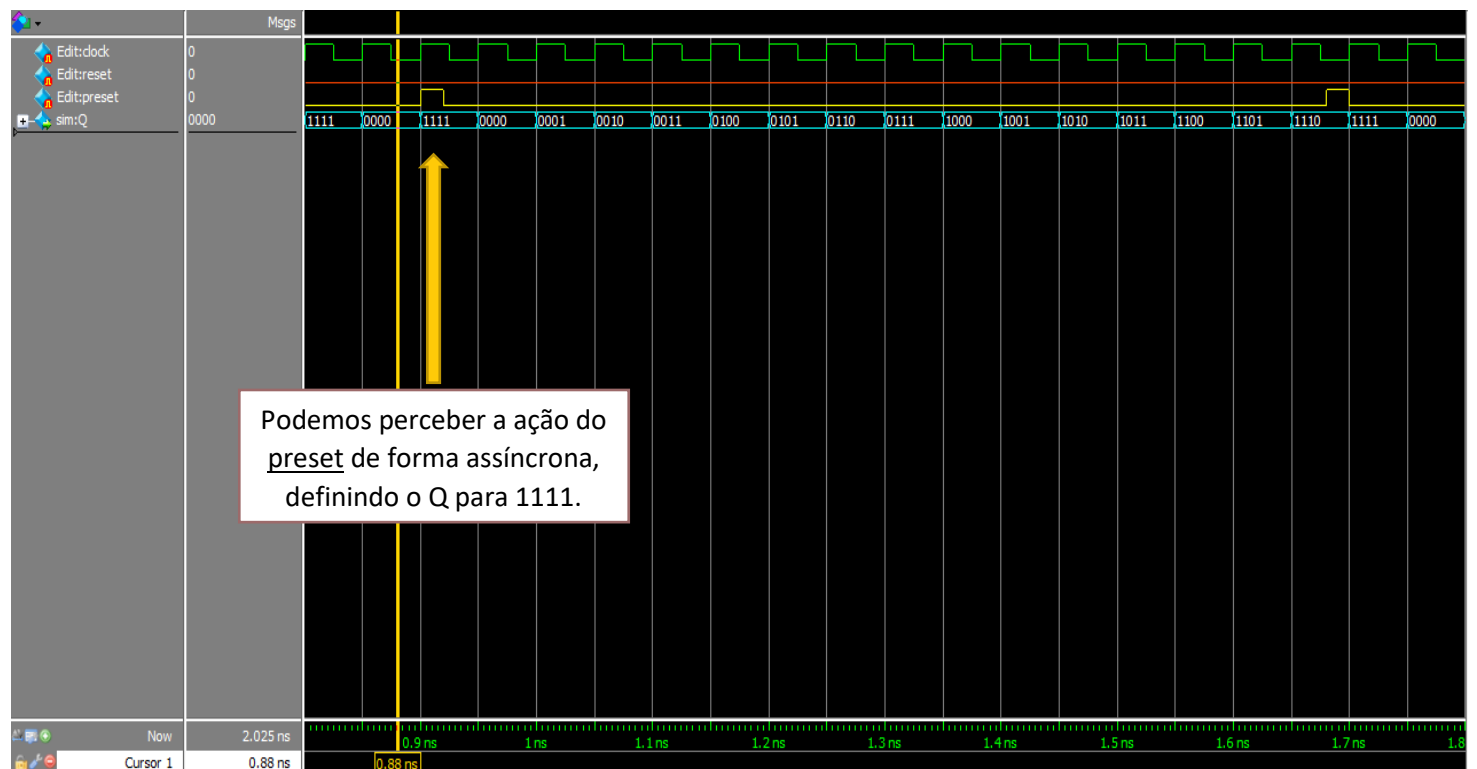
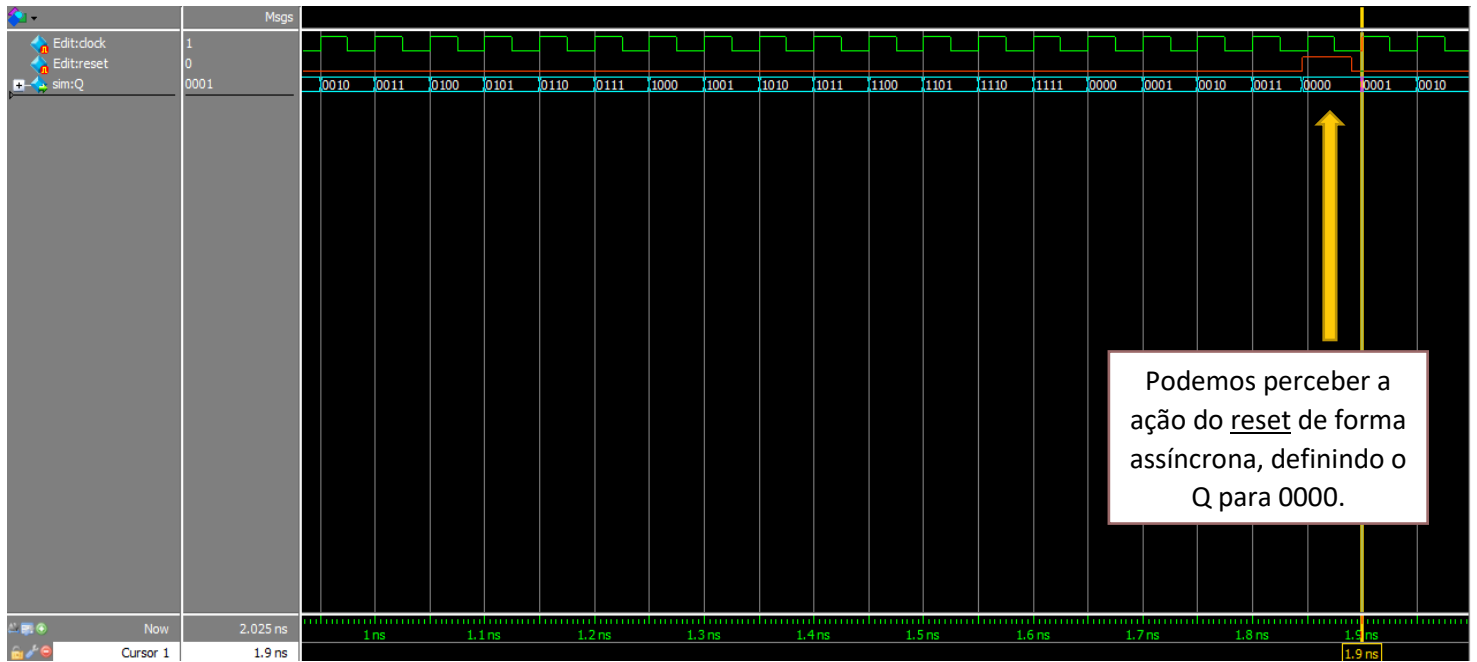
Para a confecção dessa prática, utilizamos o sistema comportamental para facilitar e evitar erros. Ademais, foi usado a instrução ( <= ), com atribuição não-bloqueante.

## RTL Viewer do circuito contador do circuito contador de módulo 16:



## Simulação no Model Sim do circuito contador de módulo 16:





## Especificações:

Entradas: **Clock** , **RESET** , **PRESET** ;

Saída: **Q** ;

Tempo máximo: 2000 ps;


## 2º Projeto:

Objetivo: Projete um circuito contador síncrono crescente de 4 bits que conte a sequência de 0 até 9 (0,1,2 ,3,4,5,6,7,8,9,0,1,2...). Ou seja, módulo 10.

### Código para o circuito contador de módulo 10:

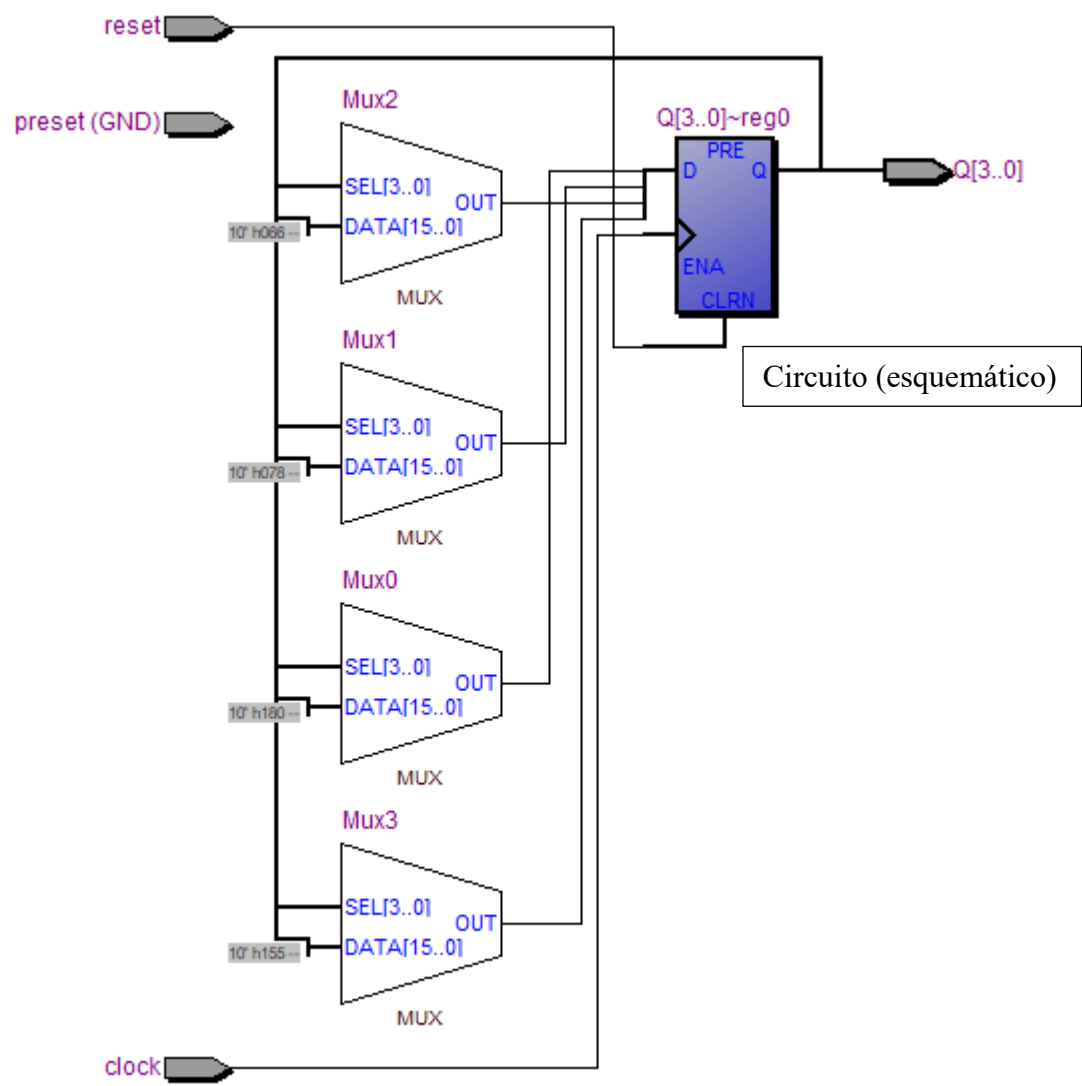
```
module contadorMod10(clock, reset, preset, Q);
    input clock, reset, preset;
    // Entradas: clock, reset e preset
    output reg [3:0] Q;
    // Saídas: [3:0] Q ; Número de 4 bits a ser contado.
    always@(posedge clock or posedge reset)
    begin
        if(reset)
        begin
            Q <= 4'b0000;
        end //if(reset)
        else if(preset)
        begin
            Q <= 4'b1001;
        end//else if(preset)
        else
        begin
            case(Q) //Definição comportamental com atribuição não-bloqueante para o circuito.
                4'b0000: Q <= 4'b0001;
                4'b0001: Q <= 4'b0010;
                4'b0010: Q <= 4'b0011;
                4'b0011: Q <= 4'b0100;
                4'b0100: Q <= 4'b0101;
                4'b0101: Q <= 4'b0110;
                4'b0110: Q <= 4'b0111;
                4'b0111: Q <= 4'b1000;
                4'b1000: Q <= 4'b1001;
                4'b1001: Q <= 4'b0000;
            endcase
        end //else
    end //always@
endmodule
```

Atribuição não-bloqueante  
( <= )

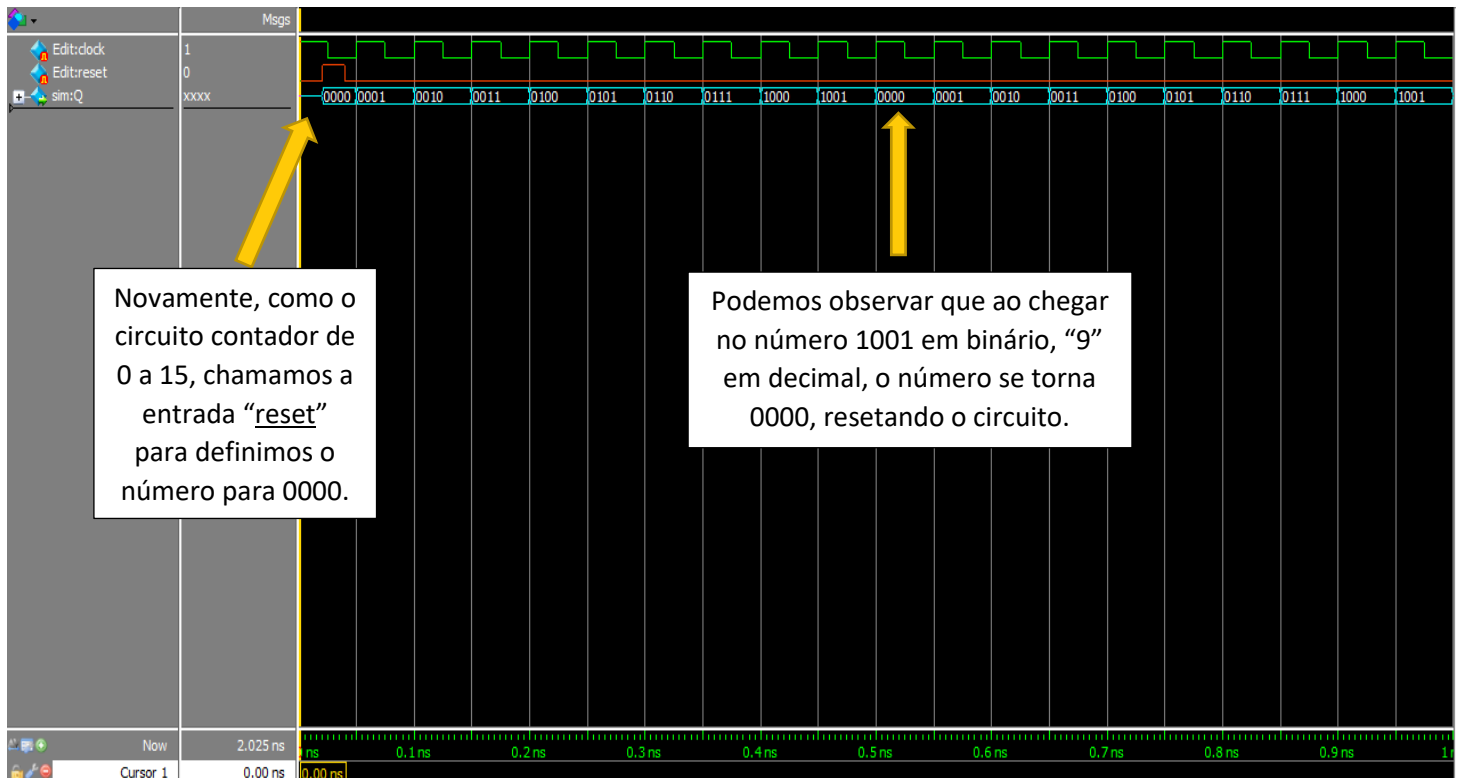


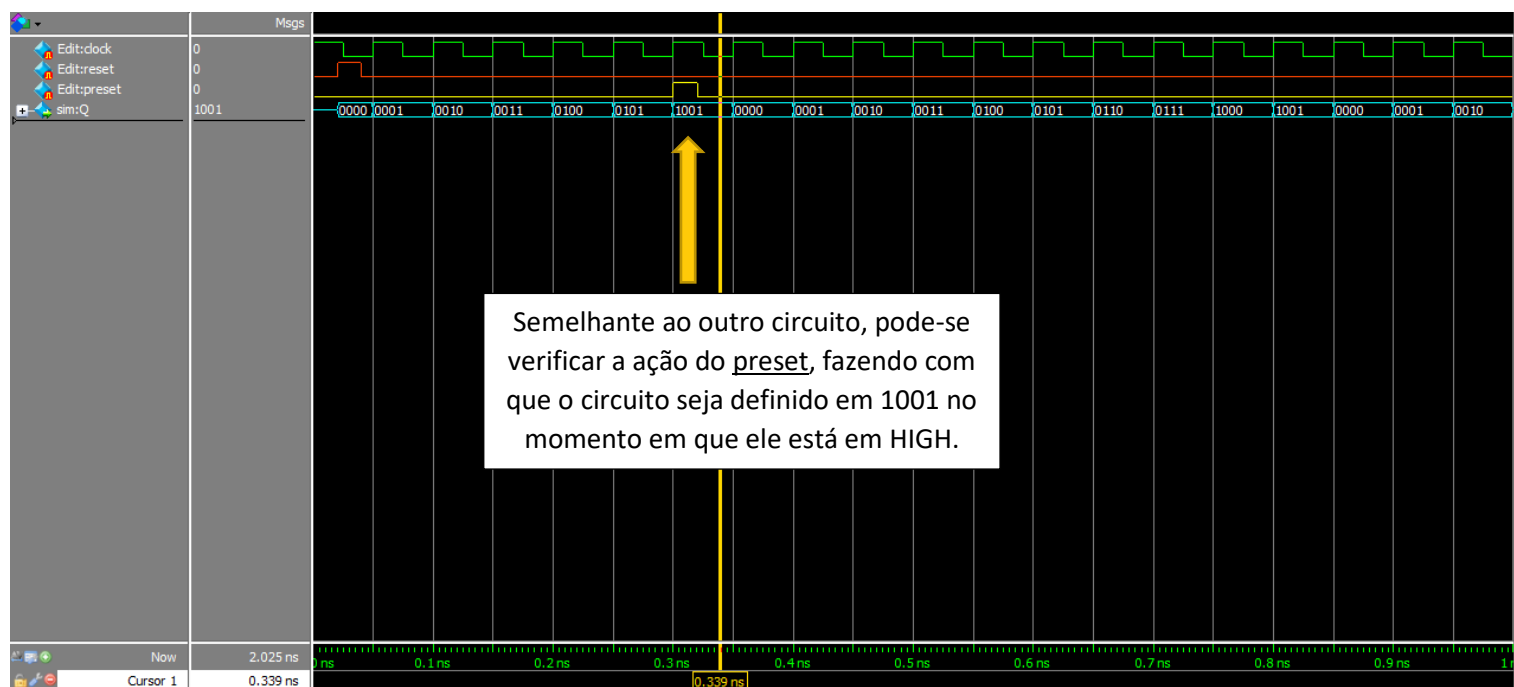
Semelhante ao circuito contador síncrono crescente de 0 a 15, usamos o sistema comportamental para facilitar e evitar erros. Além disso, foi utilizada a instrução ( <= ), com atribuição não-bloqueante.

RTL Viewer do circuito contador do circuito contador de módulo 10:



## Simulação no Model Sim do circuito contador de módulo 10:





### Especificações:

Entradas: Clock , RESET , PRESET ;

Saída: Q ;

Tempo máximo: 2000 ps;

### Conclusão:

Após a elaboração das simulações dos dois projetos solicitados, observa-se que o circuito funciona como um contador, usando a borda de subida como marco para a atualização.

Foi especificado que a ação do reset faz com que o número de 4 bits "Q" seja definido em 0000 e a ação do preset depende do projeto, no de módulo 16, o número se torna 1111 e no de módulo 10, 1001.

Além disso, pode-se perceber que em ambos projetos, quando Q chega no limite de sua contagem, ele é resetado, ou seja, definido em 0000.