

Código para o registrador, usando a atribuição bloqueadora:

```

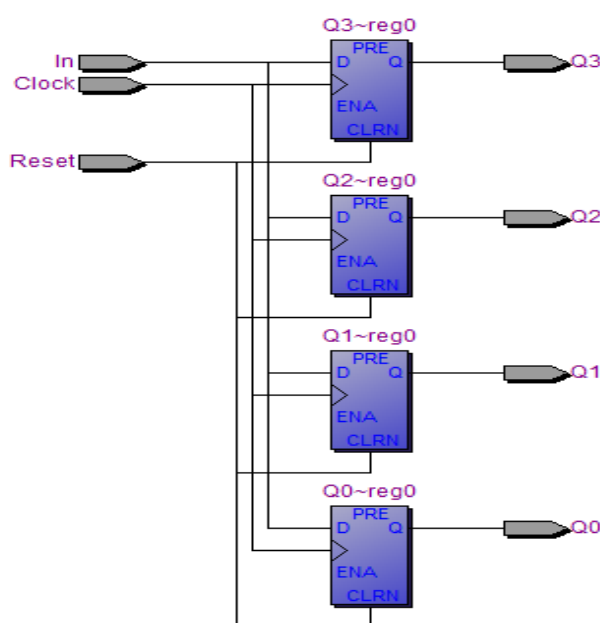
8  module registrador(Clock, Reset, In, Q0,Q1,Q2,Q3);
9
10     input Clock, Reset, In;
11     output reg Q0,Q1,Q2,Q3;
12
13     always@(posedge Reset or posedge Clock)
14     begin
15         if(Reset)
16         begin
17             Q3 = 1'b0;
18             Q2 = 1'b0;
19             Q1 = 1'b0;
20             Q0 = 1'b0;
21         end //if
22     else
23     begin
24         Q3 = In;
25         Q2 = Q3;
26         Q1 = Q2;
27         Q0 = Q1;
28     end //else
29     end //always
30 endmodule

```

Atribuição bloqueante

(=)

RTL Viewer do registrador do circuito acima:



Podemos perceber que o circuito está com a mesma entrada em todos os flips flops, o que não é o que queremos.

O desejável é que a entrada de um seja a saída do próximo.

Código do testbench usado para simulação:

```
module testbench_registrador;
    //variaveis intermediarias
    reg Clock_T, Reset_T, In_T;

    wire Q0_T,Q1_T,Q2_T,Q3_T;
    //registrador(Clock, Reset, In, Q0,Q1,Q2,Q3);

    registrador dut (Clock_T, Reset_T, In_T, Q0_T,Q1_T,Q2_T,Q3_T);

    initial
    begin
        Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
        Reset_T = 1'b1;      Clock_T = 1'b1;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b1;      #20
        Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b1;      #20
        Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
        Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20;
    end
endmodule
```

Simulação no ModelSim-Altera, usando a atribuição bloqueadora



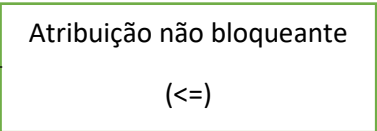
Parte 2 - registrador atribuição NÃO bloqueante (<=):

Código para o registrador, usando a atribuição não bloqueante:

```
module registradorNaoBloqueante(Clock, Reset, In, Q0,Q1,Q2,Q3);

    input Clock, Reset, In;
    output reg Q0,Q1,Q2,Q3;

    always@(posedge Reset or posedge Clock)
    begin
        if(Reset)
            begin
                Q3 = 1'b0;
                Q2 = 1'b0;
                Q1 = 1'b0;
                Q0 = 1'b0;
            end //if
        else
            begin
                Q3 <= In;
                Q2 <= Q3;
                Q1 <= Q2;
                Q0 <= Q1;
            end //else
        end //always
    endmodule
```



Atribuição não bloqueante
(<=)

Código para o testbench:

```
module testebench_registrador;
    //variaveis intermediarias
    reg Clock_T, Reset_T, In_T;

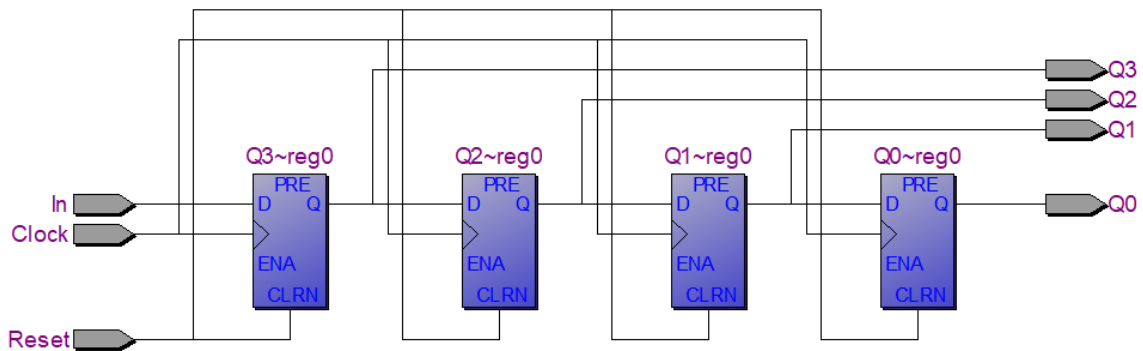
    wire Q0_T,Q1_T,Q2_T,Q3_T;
    //registradorNaoBloqueante(Clock, Reset, In, Q0,Q1,Q2,Q3);

    registradorNaoBloqueante dut (Clock_T, Reset_T, In_T, Q0_T,Q1_T,Q2_T,Q3_T);

    initial
        begin
            Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
            Reset_T = 1'b1;      Clock_T = 1'b1;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b1;      #20
            Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b1;      #20
            Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b0;      In_T = 1'b0;      #20
            Reset_T = 1'b0;      Clock_T = 1'b1;      In_T = 1'b0;      #20;
        end
    endmodule
```

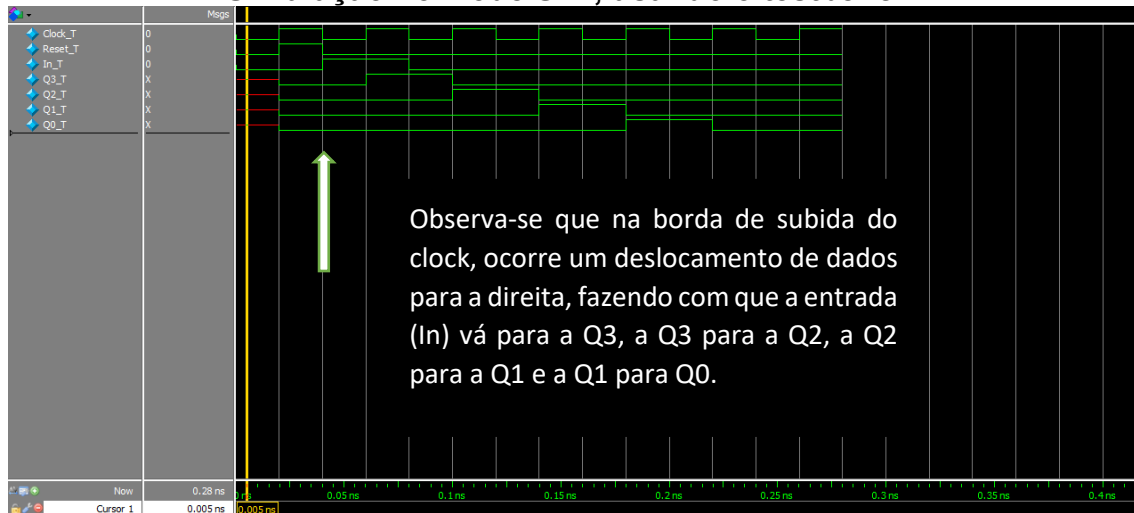
Semelhante ao do bloqueante, somente mudando a chamada do dut para o respectivo .v

RTL Viewer para o registrador, usando a atribuição não bloqueante:



Como esperado, usando a atribuição não bloqueante, o circuito funciona como planejamos, tendo como entrada de um flip flop, a saída do outro, conforme demonstrado acima.

Simulação no ModelSim, usando o testbench:



Com quatro clocks de clock conseguimos registrar e deslocar a informação dada.