

## 8o Projeto - Registrador deslocador para direita

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

**Objetivos:** Implementar, usando Verilog HDL, um circuito digital sequencial registrador para direita que utilize o operador de atribuição bloqueante (=) e um que utilize o operador de atribuição não bloqueante (<=). A partir disso, vamos verificar a diferença entre os operadores.

**Primeira parte:** registrador de deslocamento de 4 bits usando atribuição bloqueante (=).

Inicialmente, montamos uma tabela com os números de 4 bits (A,B,C,D, com A sendo o mais significativo), a entrada, o clock e o reset para mostrar o funcionamento de um registrador de 4 bits com deslocamento para a direita.

Entrada	A	B	C	D	clock	reset
x	0	0	0	0	1	1
1	1	0	0	0	1	0
0	0	1	0	0	1	0
0	0	0	1	0	1	0
x	0	0	1	0	0	0
0	0	0	0	1	1	0
x	0	0	0	0	0	0

A partir disso, criamos o código em Verilog HDL no Quartus II:

```

module registrador_bloqueante(clock, reset, entrada, A, B, C, D);
input clock, reset, entrada;
output reg A, B, C, D;

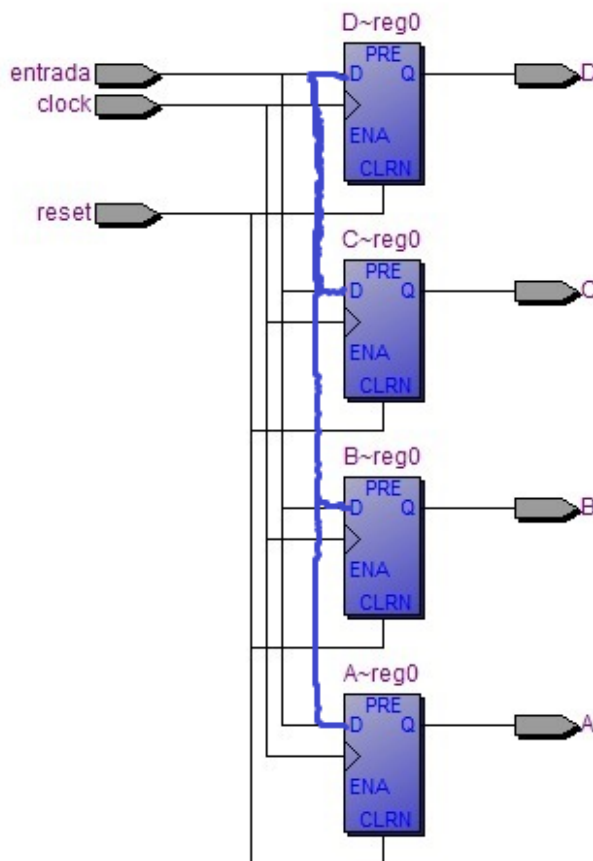
always @ (posedge clock or posedge reset)
begin
    if(reset) //quando o reset estiver em uma borda de subida
    begin
        A = 1'b0; //todas as variáveis vão para zero (são reinicializadas)
        B = 1'b0;
        C = 1'b0;
        D = 1'b0;
    end

    else //quando o clock estiver em borda de subida
    begin
        A = entrada; //A recebe a entrada
        B = A; //os demais dados são deslocados um bit para a direita
        C = B; //observe que foi utilizado o operador de atribuição bloqueante
        D = C; //verificaremos se o resultado estará de acordo com o esperado
    end
end
endmodule

```

O código foi feito com base no que foi passado em aula. Na parte inicial, detecta-se se o reset está em borda de subida, reinicializando as variáveis. Na segunda parte do código, quando se detecta o clock em borda de subida, a variável A recebe o valor de entrada, seguindo com o deslocamento para a direita.

Com isso, compilamos o código e obtivemos o esquemático no RTL viewer:



Utilizamos 4 tipos de flip-flops tipo D, como evidenciado no esquemático. Entretanto, observa-se que a entrada está ligada a todos os flip-flops, o que não é esperado. Isso é um problema no código, que é solucionado na segunda parte. Assim, utilizamos o seguinte clock e obtivemos a simulação do projeto:

Clock Attributes

Initial Value

St0

Clock Period

100

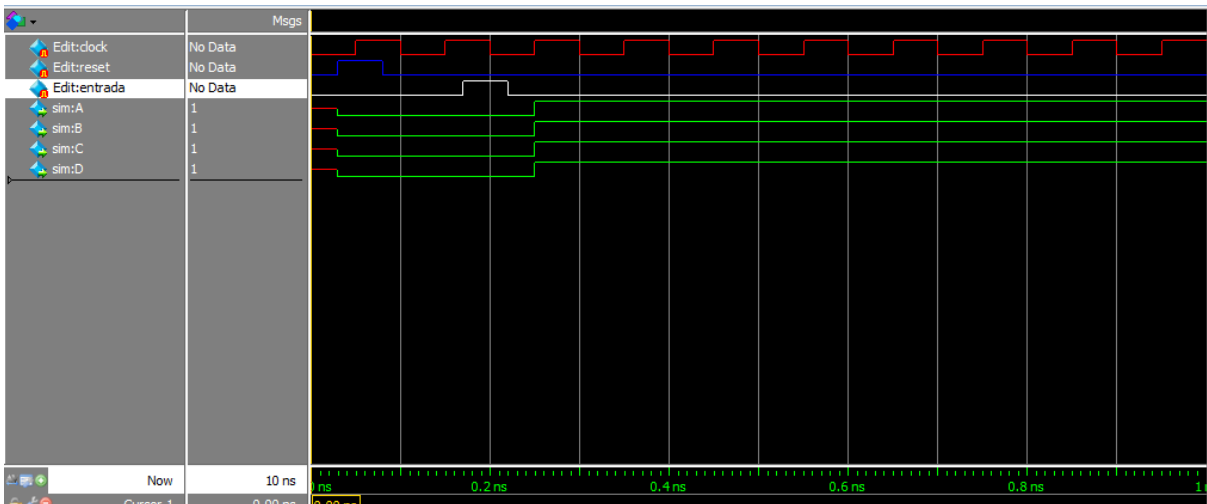
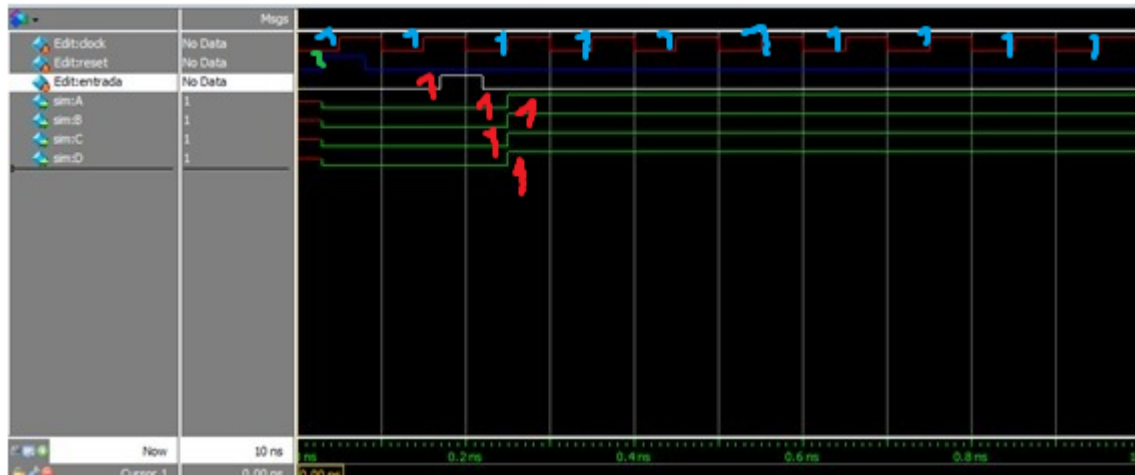
Time Unit

ps

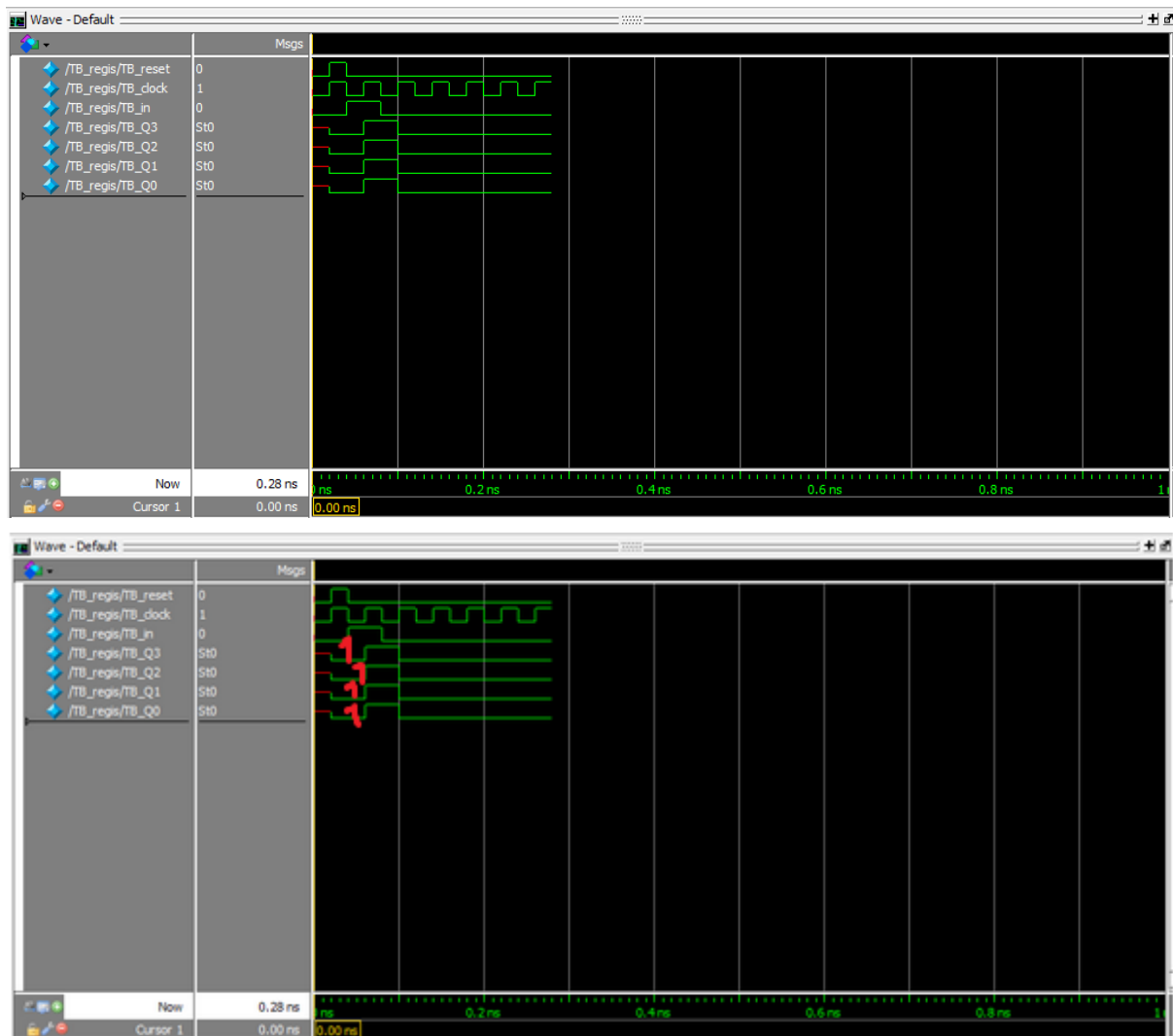
Duty Cycle

50

(Período do clock: 100 ps e duty cycle: 50 ps).



Simulação test bench:



Código test bench: foi preciso mudar o nome e a ordem dos parâmetros para associar o test bench ao projeto.

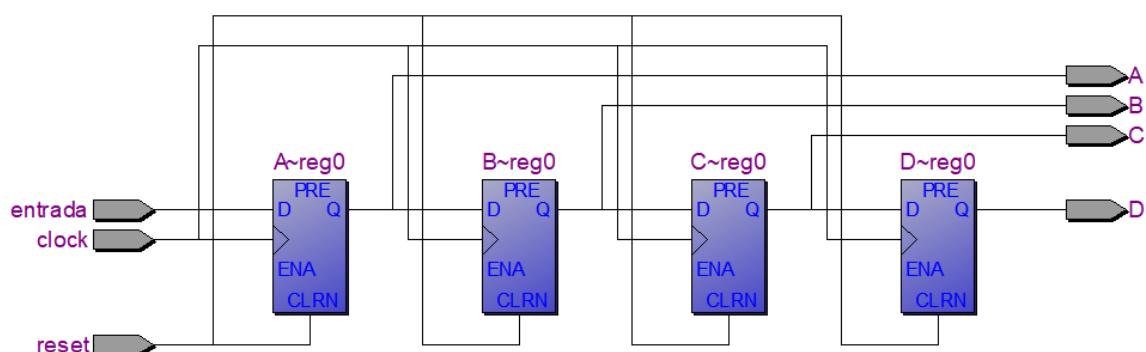
```
//module registrador_bloqueante(clock, reset, entrada, A, B, C, D,)|
    registrador_bloqueante dut(TB_clock, TB_reset, TB_in, TB_Q3, TB_Q2, TB_Q1, TB_Q0);

initial
begin
    TB_reset = 1'b0;    TB_clock = 1'b0;    TB_in = 1'b0;    #20
    TB_reset = 1'b1;    TB_clock = 1'b1;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b0;    TB_in = 1'b1;    #20
    TB_reset = 1'b0;    TB_clock = 1'b1;    TB_in = 1'b1;    #20
    TB_reset = 1'b0;    TB_clock = 1'b0;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b1;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b0;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b1;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b0;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b1;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b0;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b1;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b0;    TB_in = 1'b0;    #20
    TB_reset = 1'b0;    TB_clock = 1'b1;    TB_in = 1'b0;    #20
end
```

**Segunda parte: registrador de deslocamento de 4 bits usando atribuição não bloqueante (<=).**

Esse segundo código é baseado no primeiro feito, com algumas alterações. Ao invés de utilizar o operador de atribuição bloqueante, foi utilizado o operador de atribuição não bloqueante.

```
module registrador_naobloqueante(clock, reset, entrada, A, B, C, D);  
  
input clock, reset, entrada;  
output reg A, B, C, D;  
  
always @ (posedge clock or posedge reset)  
begin  
    if(reset) //quando o reset estiver em uma borda de subida  
    begin  
        A <= 1'b0; //todas as variáveis vão para zero (são reinicializadas)  
        B <= 1'b0;  
        C <= 1'b0;  
        D <= 1'b0;  
    end  
  
    else //quando o clock estiver em borda de subida  
    begin  
        A <= entrada; //A recebe a entrada  
        B <= A; //os demais dados são deslocados um bit para a direita  
        C <= B; //observe que foi utilizado o operador de atribuição não bloqueante  
        D <= C; //o resultado estará de acordo com o esperado  
    end  
end  
endmodule
```

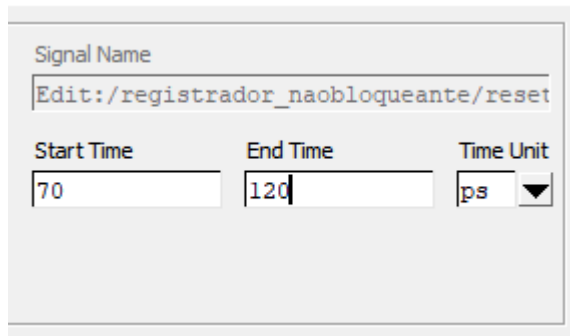


Diferente do primeiro esquemático, esse se encontra no esperado, com entrada apenas ligada no flip-flop A, que representa o bit mais significativo. O flip-flop D representa o bit menos significativo. Ressalta-se que os flip-flops, também, são do tipo D.

Dessa maneira, foi representado o esquemático de um circuito sequencial, em que a saída não depende, apenas, das entradas iniciais.

### Simulação no model-sim:

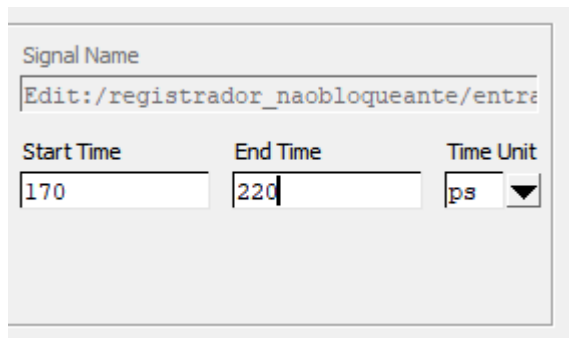
**reset:** (colocado em nível lógico alto nesse intervalo de tempo)



Signal Name	Start Time	End Time	Time Unit
Edit:/registrador_naobloqueante/reset	70	120	ps

(Tempo: 70-120ps)

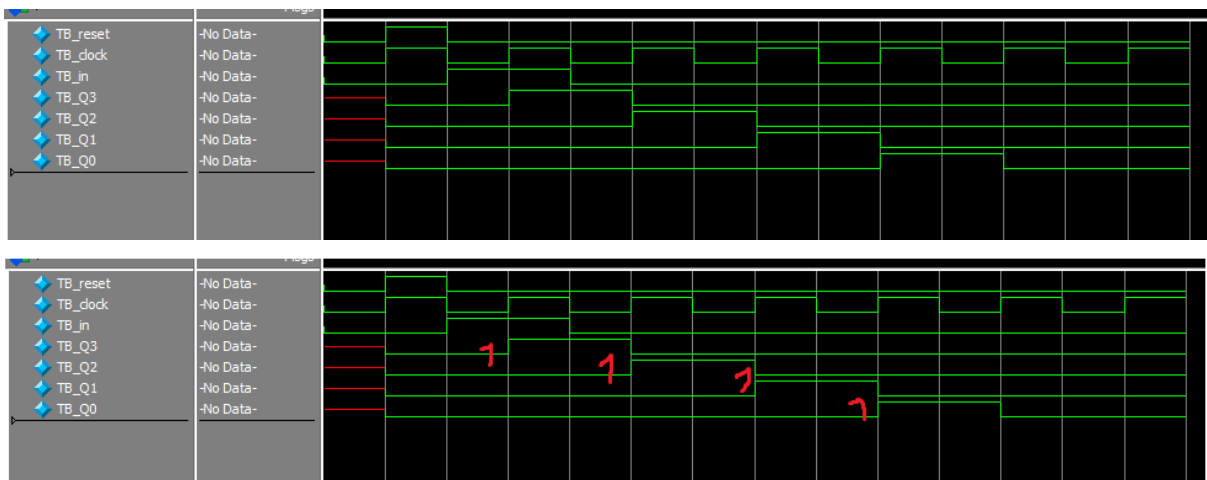
**entrada:**



Signal Name	Start Time	End Time	Time Unit
Edit:/registrador_naobloqueante/entra	170	220	ps

(Tempo: 170-220ps)

### Simulação test bench:



Código test bench: foi preciso associar o código do test bench ao registrador.