Relatório 6

Nome: Gabriel Cruz Vaz Santos
Matrícula: 200049038
Turma: C

- Questão 1

Introdução

Foi solicitado a implementação de um flipflop JK gatilhado na borda de subida, utilizando a estrutura process, seguindo a seguinte tabela:

entradas					saída
PR	*CLR*	*CLK*	*J*	*K*	*Q*
1	x	x	x	x	1
0	1	x	x	x	0
0	0	↓	0	0	mantém
0	0	↓	0	1	0
0	0	↓	1	0	1
0	0	↓	1	1	inverte
0	0	outros	x	x	mantém

Teoria

Um flipflop é um sistema com memória, com terminais síncronos (só tem efeito se determinada condição do clock for cumprida) e assíncronos (tem efeito independente do clock). Note que no caso flipflop JK, os terminais síncronos (J e K) só terão efeito na borda de subida do clock e o preset e o clear serão nossos terminais assíncronos.

Código

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. A imagem 1 e 2 é da implementação da entidade do flipflop jk e a imagem 3 do testbench.

Relatorio6 > flipflopjk.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity flipflopJK is
    port(
    clk, clr, pr, J, K: in std_logic;
    Q: out std_logic
    );
end flipflopJK;

architecture flipflopJK_arch of flipflopJK is

    signal Qbuf: std_logic := '0';
    signal JK: std_logic_vector (1 downto 0);

    begin
        JK <= J & K;
        process(clk, clr, pr)
        begin
            if pr = '1' then Qbuf <= '1';
            elsif clr = '1' then Qbuf <= '0';
            elsif rising_edge(clk) then
                case JK is
                    when "01" => Qbuf <= '0';
                    when "10" => Qbuf <= '1';
                    when "11" => Qbuf <= not(Qbuf);
                    when others => Qbuf <= Qbuf;
                end case;
            end if;
        end process;

        Q <= Qbuf;
end flipflopJK_arch;
```

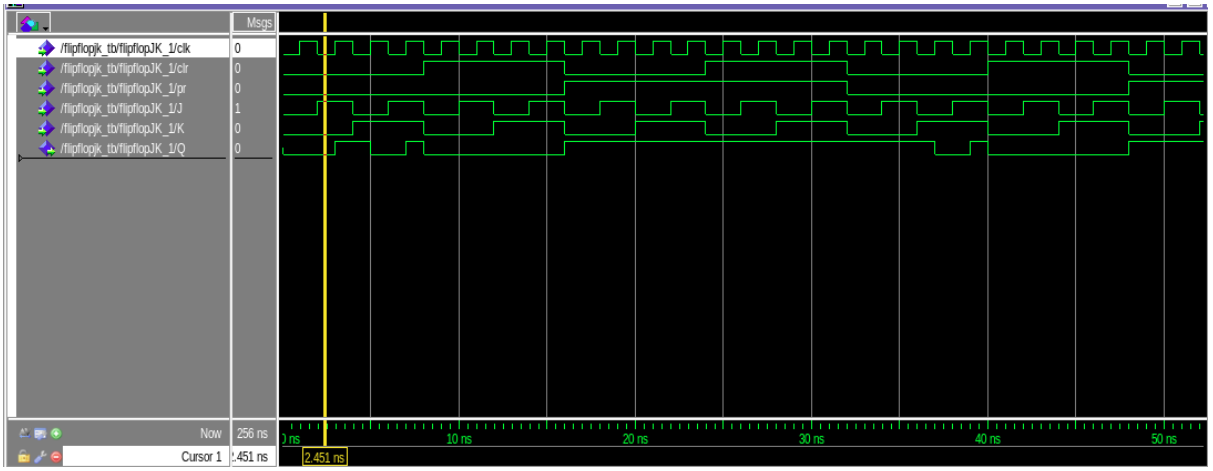
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity flipflopJK_tb is end;

architecture flipflopJK_arch of flipflopJK_tb is
    component flipflopJK is
        port(
            clk, clr, pr, J, K: in std_logic;
            Q: out std_logic
        );
    end component;

    signal entrance: std_logic_vector (4 downto 0) := "00000";

    constant time_0: time := 2 ns;
    constant time_1: time := 4 ns;
    constant time_2: time := 8 ns;
    constant time_3: time := 16 ns;
    constant time_4: time := 32 ns;

    begin
        flipflopJK_1: flipflopJK port map (clk =>  entrance(0), J => entrance(1), K => entrance(2), clr => entrance(3), pr => entran

        entrance(0) <= not entrance(0) after time_0/2;
        entrance(1) <= not entrance(1) after time_1/2;
        entrance(2) <= not entrance(2) after time_2/2;
        entrance(3) <= not entrance(3) after time_3/2;
        entrance(4) <= not entrance(4) after time_4/2;

end flipflopJK_arch;
```

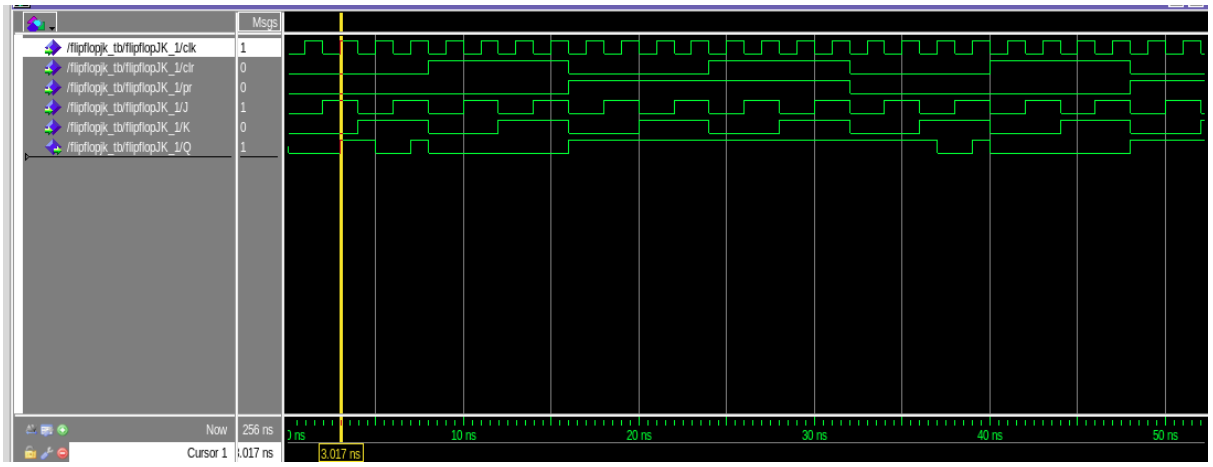
Compilador

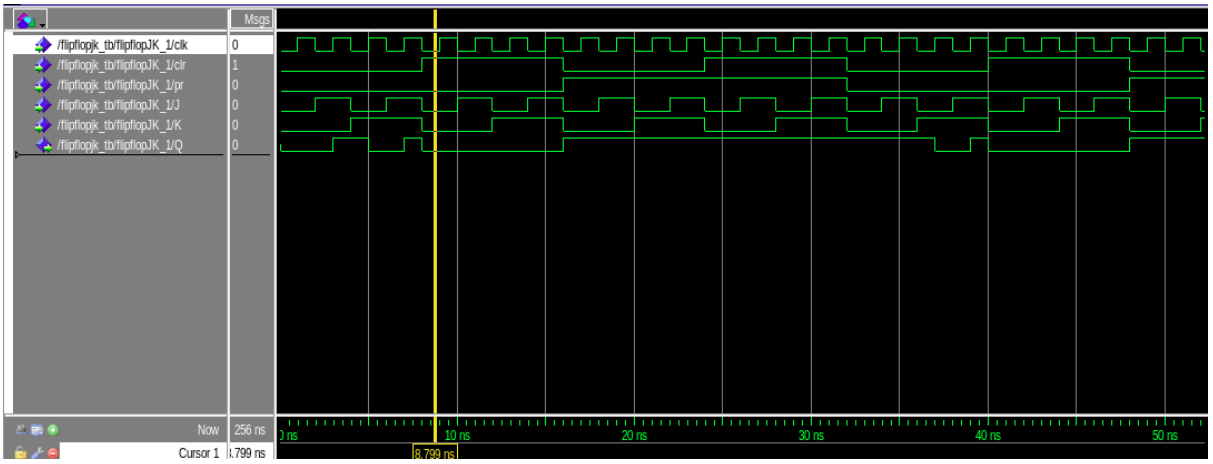
Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

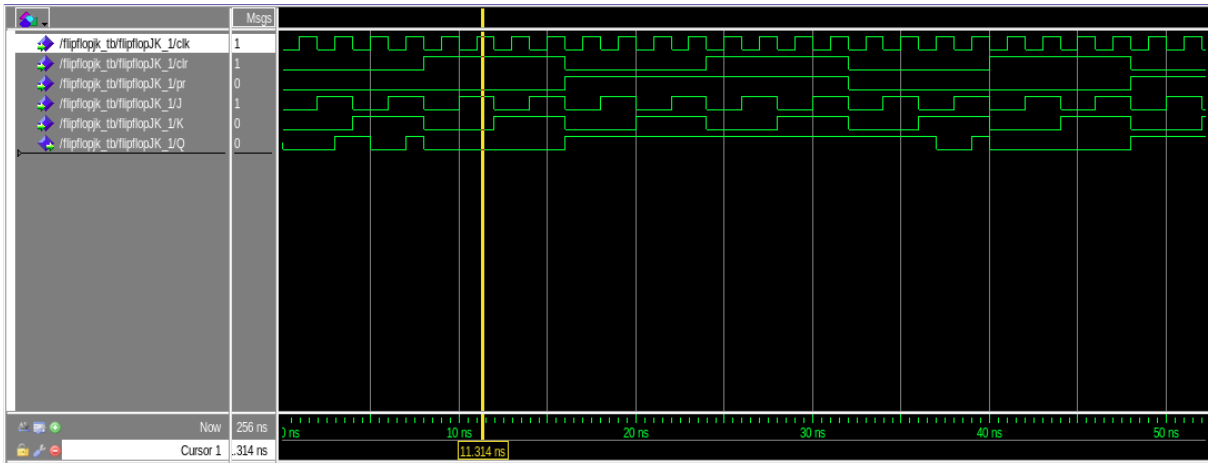
```
# Compile of flipflopjk.vhd was successful.
# Compile of flipflopjk_tb.vhd was successful.
# 2 compiles, 0 failed with no errors.
```

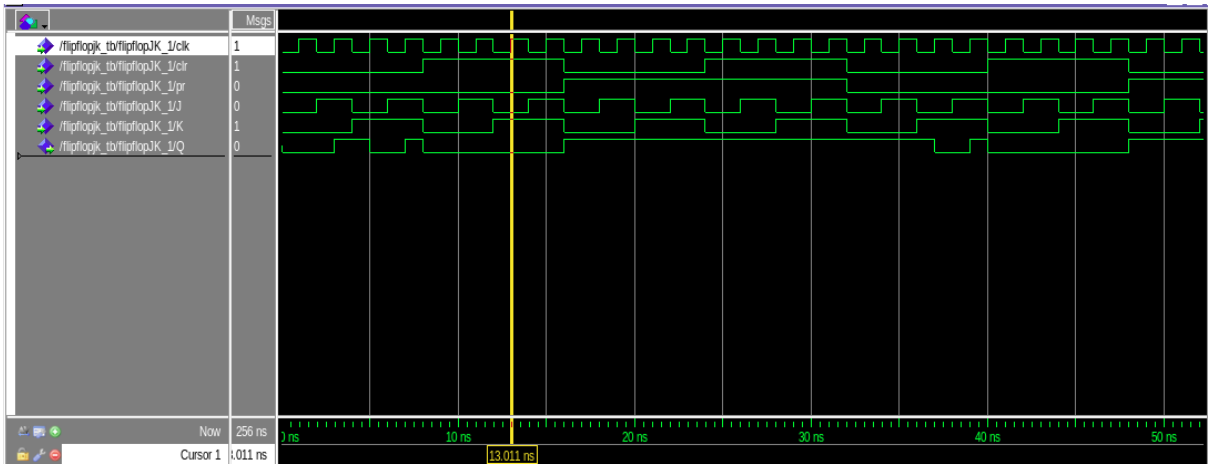
Simulação

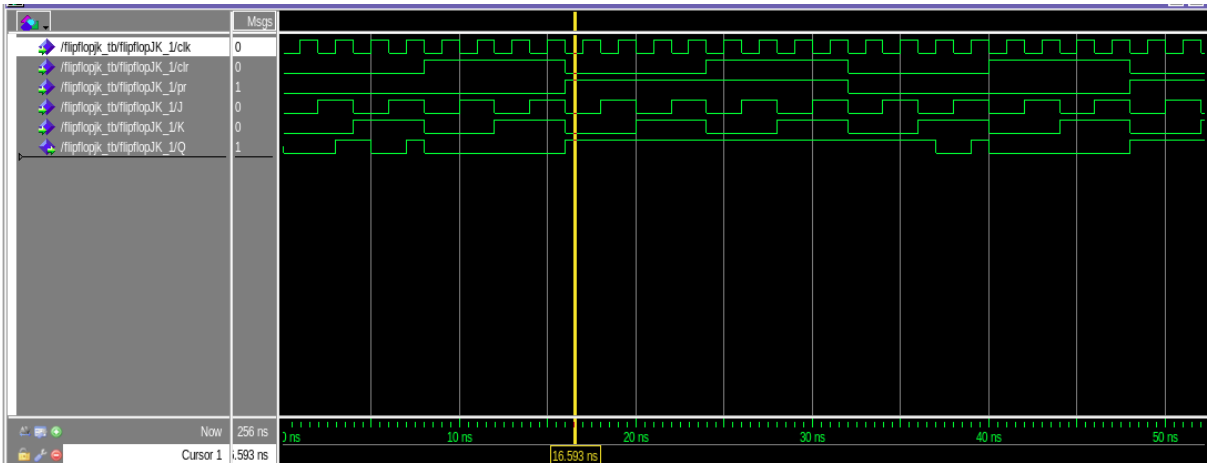


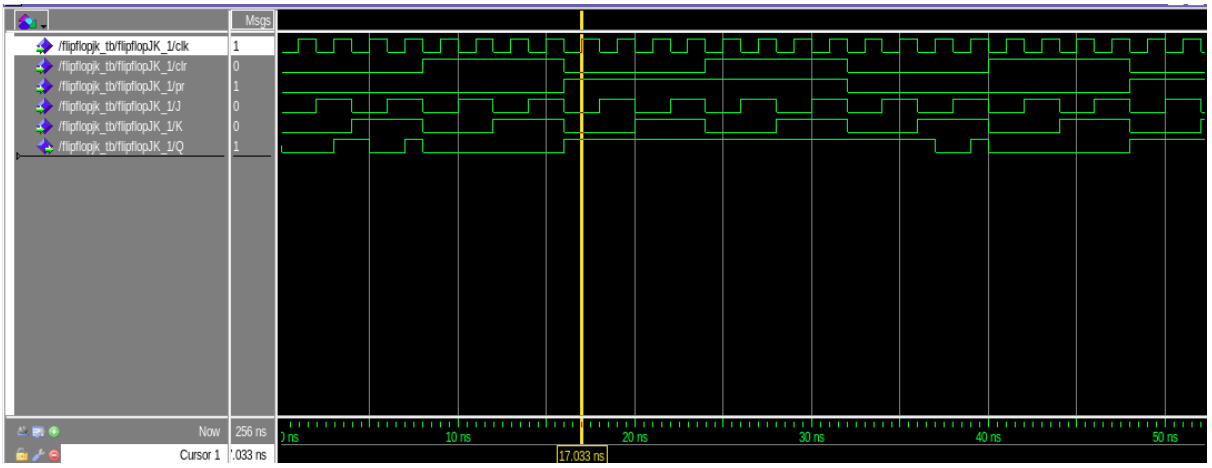


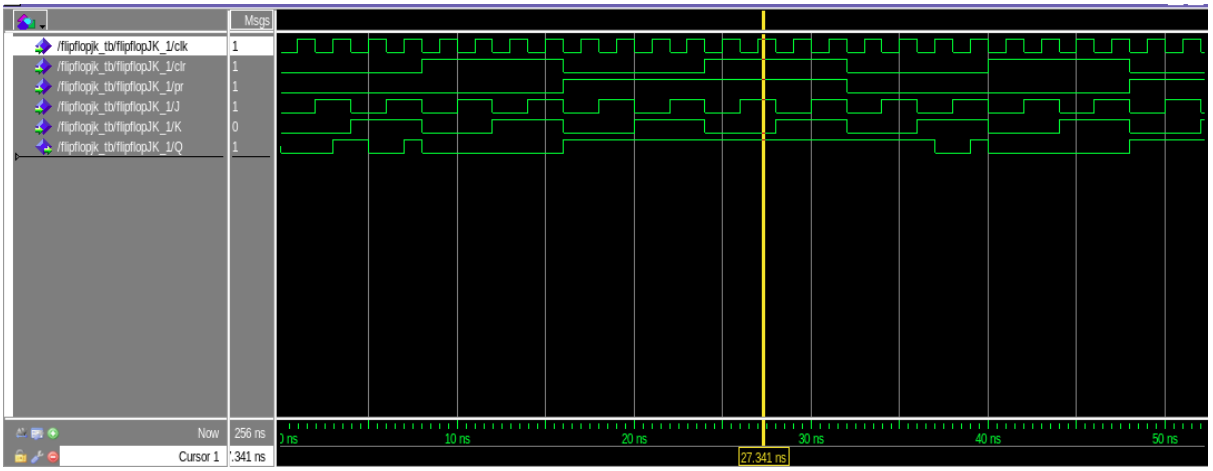

Msgs
/flipflopjk_tb/flipflopJK_1/clk
/flipflopjk_tb/flipflopJK_1/clr
/flipflopjk_tb/flipflopJK_1/pr
/flipflopjk_tb/flipflopJK_1/J
/flipflopjk_tb/flipflopJK_1/K
/flipflopjk_tb/flipflopJK_1/Q
0
1
0
0
0
0
Now
256 ns
Cursor 1
10 ns
20 ns
30 ns
40 ns
50 ns
8.799 ns

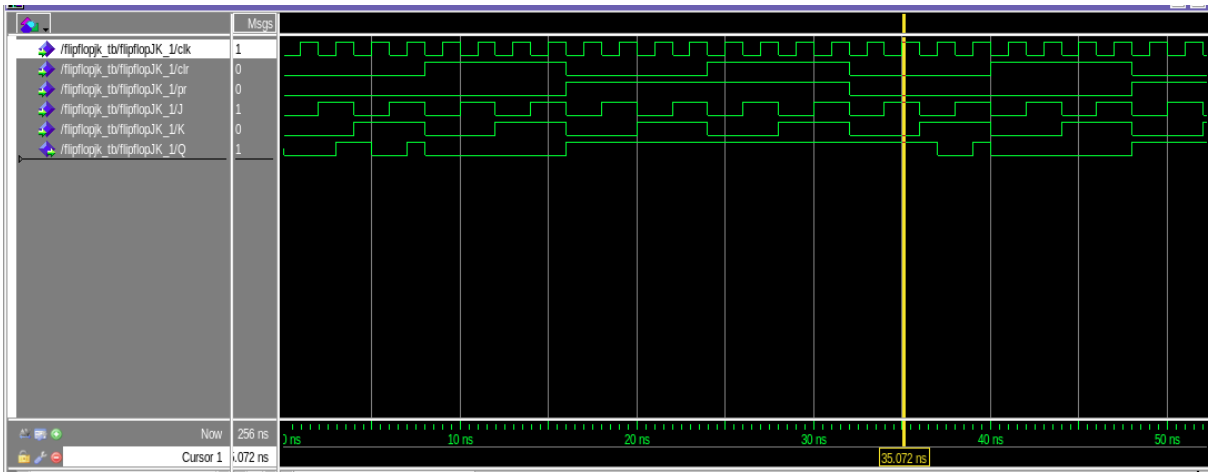

Msgs
/flipflopjk_tb/flipflopJK_1/clk
/flipflopjk_tb/flipflopJK_1/clr
/flipflopjk_tb/flipflopJK_1/pr
/flipflopjk_tb/flipflopJK_1/J
/flipflopjk_tb/flipflopJK_1/K
/flipflopjk_tb/flipflopJK_1/Q
1
1
0
1
0
0
Now
256 ns
Cursor 1
10 ns
20 ns
30 ns
40 ns
50 ns
11.314 ns


Msgs
/flipflopjk_tb/flipflopJK_1/clk
/flipflopjk_tb/flipflopJK_1/clr
/flipflopjk_tb/flipflopJK_1/pr
/flipflopjk_tb/flipflopJK_1/J
/flipflopjk_tb/flipflopJK_1/K
/flipflopjk_tb/flipflopJK_1/Q
1
1
0
0
1
0
Now
256 ns
Cursor 1
10 ns
20 ns
30 ns
40 ns
50 ns
13.011 ns


Msgs
/flipflopjk_tb/flipflopJK_1/clk
/flipflopjk_tb/flipflopJK_1/clr
/flipflopjk_tb/flipflopJK_1/pr
/flipflopjk_tb/flipflopJK_1/J
/flipflopjk_tb/flipflopJK_1/K
/flipflopjk_tb/flipflopJK_1/Q
0
0
1
0
0
1
Now
256 ns
Cursor 1
.593 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
16.593 ns


Msgs
/flipflopjk_tb/flipflopJK_1/clk
/flipflopjk_tb/flipflopJK_1/clr
/flipflopjk_tb/flipflopJK_1/pr
/flipflopjk_tb/flipflopJK_1/J
/flipflopjk_tb/flipflopJK_1/K
/flipflopjk_tb/flipflopJK_1/Q
1
0
1
0
0
1
Now
256 ns
Cursor 1
.033 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
17.033 ns





Análise

O Flip Flop JK desenvolvido tem como terminais assíncronos as variáveis PR e CLR e síncronos as variáveis J e K, que só tem efeito na borda de subida do clock.

Na imagem 1 da análise, estamos na borda de descida do clock e os terminais assíncronos são iguais a ‘0’, logo o valor da saída não se altera ao anterior. Como nesse caso ‘Q’ foi inicializado como ‘0’, ‘Q‘ se mantém como ‘0’.

Na imagem 2, nossos terminais assíncronos continuam como zero, porém estamos na borda de subida do clock (clk). Os valores J e K são respectivamente 1 e 0, e, assim como previsto, percebemos que a saída Q será ‘1’.

Percebemos também que dos terminais assíncronos, o com mais precedência é o “pr”, que quando vale ‘1’, a saída ‘Q’ sempre será ‘1’ independente dos valores de ‘clr’, ‘J’ e ‘K’.

Percebemos agora que se “pr” = ‘0’ e se “clr” = ‘1’, a saída ‘Q’ sempre será ‘0’, independente dos valores dos terminais síncronos ‘J’ e ‘K’.

Conclusão

Nesse experimento conseguimos com êxito implementar em vhdl um flipflop JK gatilhado pela borda de subida utilizando a estrutura “process”. As simulações se comportaram como esperado e de acordo com a tabela disponibilizada no exercício.

- Questão 2

Introdução

Foi solicitado a implementação de um registrador de deslocamento bidirecional com 4 bits, com o funcionamento seguindo a tabela:

entradas							saída
CLK	*RST*	*LOAD*	*D*	*DIR*	*L*	*R*	*Q*
↓	1	x	xxxx	x	x	x	0000
↓	0	1	$D_3D_2D_1D_0$	x	x	x	$D_3D_2D_1D_0$
↓	0	0	xxxx	0	0	x	$Q_2Q_1Q_0$ 0
↓	0	0	xxxx	0	1	x	$Q_2Q_1Q_0$ 1
↓	0	0	xxxx	1	x	0	0 $Q_3Q_2Q_1$
↓	0	0	xxxx	1	x	1	1 $Q_3Q_2Q_1$
outros	x	x	xxxx	x	x	x	$Q_3Q_2Q_1Q_0$

Teoria

Um registrador de deslocamento é um circuito que implementa uma memória, à medida que novos bits são inseridos, os bits da palavra binária podem ser deslocados para direita ou para esquerda.

Código

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. A imagem 1 da implementação da entidade do registrador de deslocamento e as imagens 2 e 3 do testbench.

Relatorio6 > regis.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity regis is
    port(
        D: in std_logic_vector (3 downto 0);
        clock, load, reset, dir, L, R : in std_logic;
        Q: out std_logic_vector (3 downto 0)
    );
end regis;

architecture regis_arch of regis is

    signal Qbuf: std_logic_vector (3 downto 0);

begin
    process(clock)
    begin
        if rising_edge(clock) then
            if reset = '1' then Qbuf <= "0000";
            elsif load = '1' then Qbuf <= D;
            elsif dir = '0' then
                Qbuf <= Qbuf(2) & Qbuf(1) & Qbuf(0) & L;
            elsif dir = '1' then
                Qbuf <= R & Qbuf(3) & Qbuf(2) & Qbuf(1);
            end if;
        end if;
    end process;
        Q <= Qbuf;

end regis_arch;
```

Relatorio6 > regis_tb.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity regis_tb is end;

architecture regis_arch of regis_tb is
    component regis is
        port(
            D: in std_logic_vector (3 downto 0);
            clock, load, reset, dir, L, R : in std_logic;
            Q: out std_logic_vector (3 downto 0)
        );
    end component;

    signal entrance: std_logic_vector (9 downto 0) := "0000000000";

    constant time_0: time := 2 ns;
    constant time_1: time := 4 ns;
    constant time_2: time := 8 ns;
    constant time_3: time := 16 ns;
    constant time_4: time := 32 ns;
    constant time_5: time := 64 ns;
    constant time_6: time := 128 ns;
    constant time_7: time := 256 ns;
    constant time_8: time := 512 ns;
    constant time_9: time := 1024 ns;

begin
    regis_1: regis port map (clock => entrance(0), R => entrance(1), L => entrance(2), dir => entrance(3), D => entrance(7 downto 4)

    entrance(0) <= not entrance(0) after time_0/2;
```

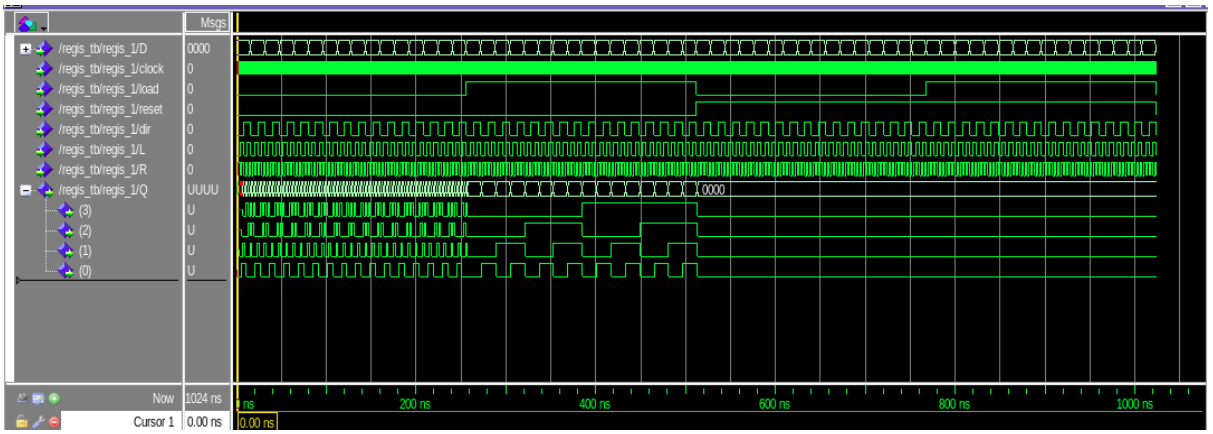
```
32      entrance(1) <= not entrance(1) after time_1/2;
33      entrance(2) <= not entrance(2) after time_2/2;
34      entrance(3) <= not entrance(3) after time_3/2;
35      entrance(4) <= not entrance(4) after time_4/2;
36      entrance(5) <= not entrance(5) after time_5/2;
37      entrance(6) <= not entrance(6) after time_6/2;
38      entrance(7) <= not entrance(7) after time_7/2;
39      entrance(8) <= not entrance(8) after time_8/2;
40      entrance(9) <= not entrance(9) after time_9/2;
41  end regis_arch ; -- regis_tb
```

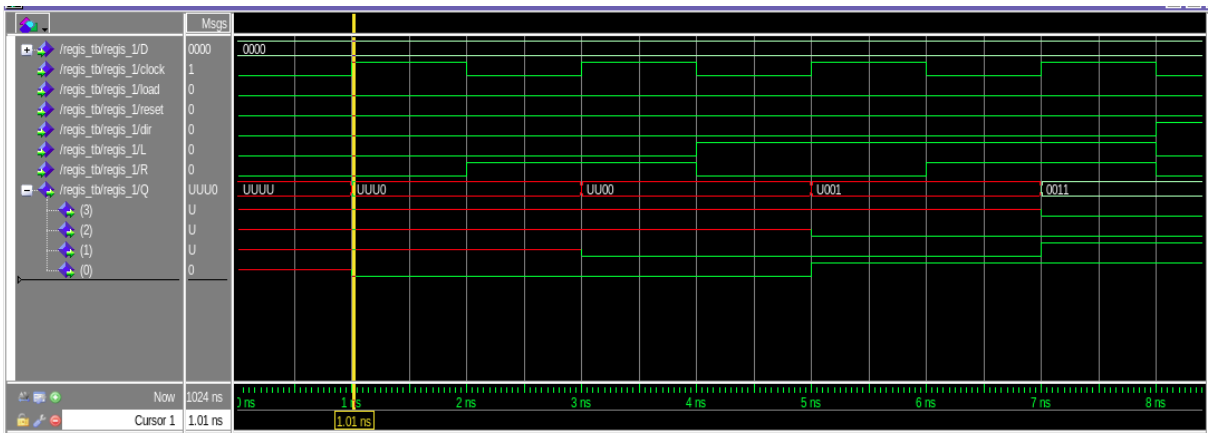
Compilador

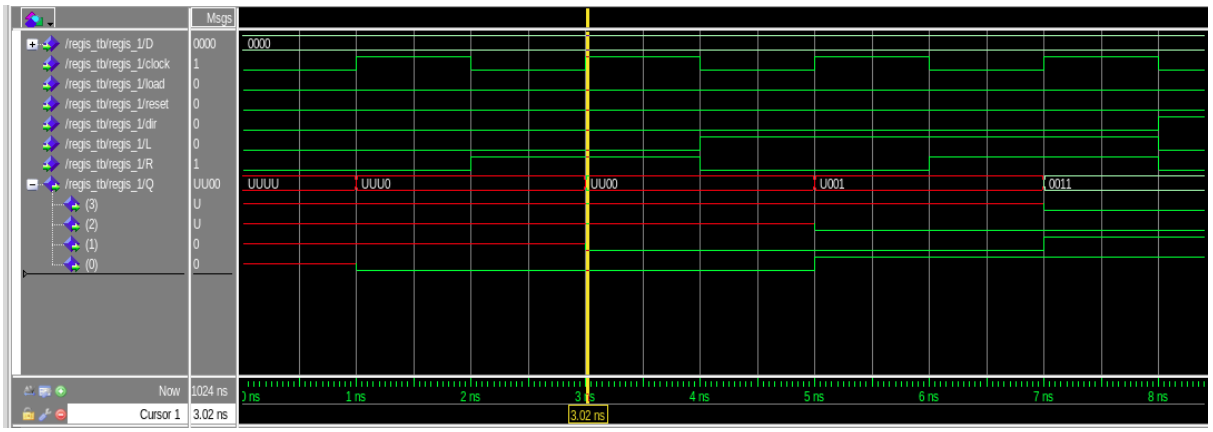
Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

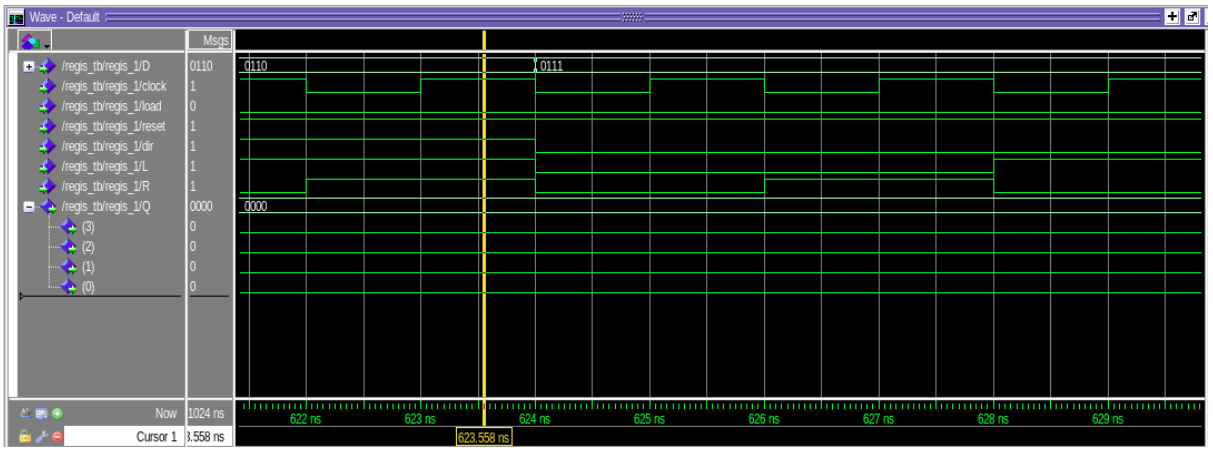
```
# Compile of regis.vhd was successful.
# Compile of regis_tb.vhd was successful.
```

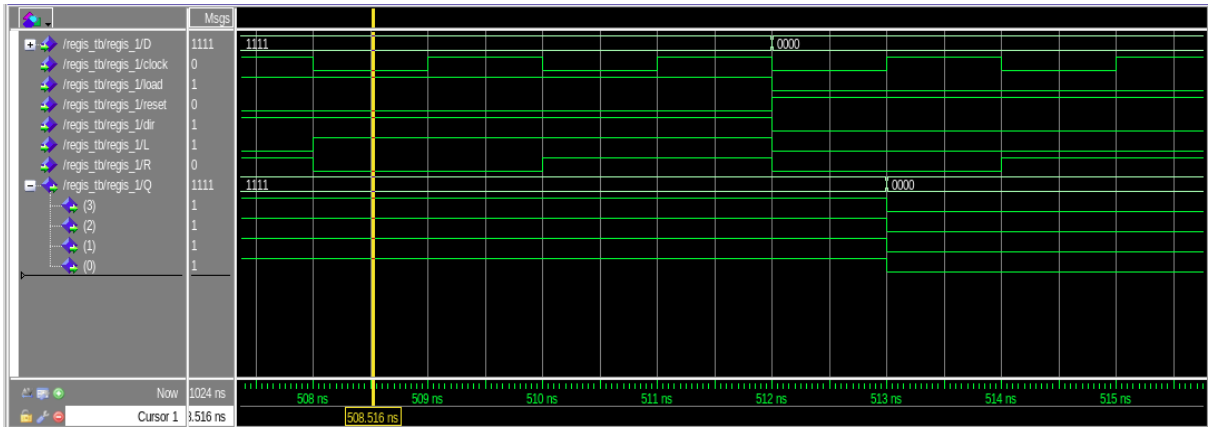
Simulação

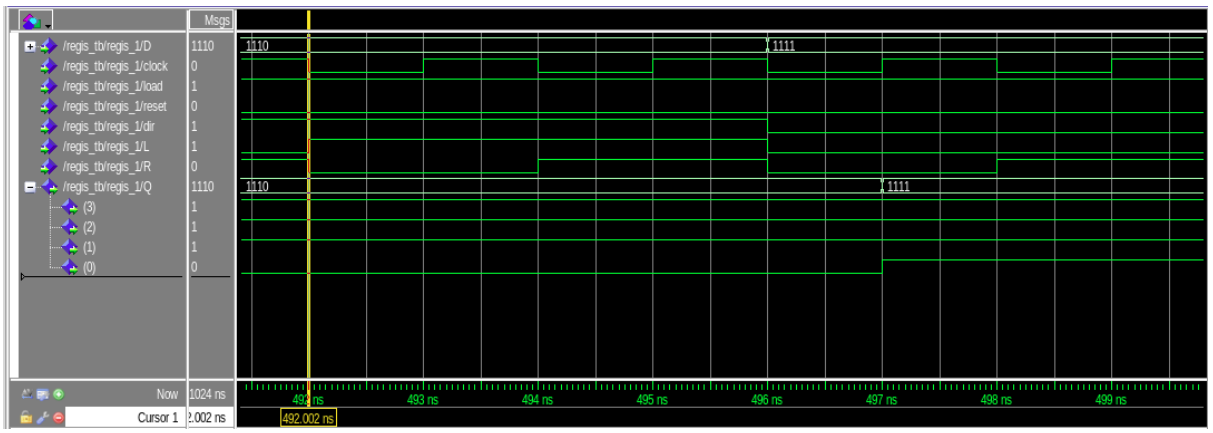



Msgs
/regis_tb/regis_1/D
/regis_tb/regis_1/clock
/regis_tb/regis_1/load
/regis_tb/regis_1/reset
/regis_tb/regis_1/dir
/regis_tb/regis_1/L
/regis_tb/regis_1/R
/regis_tb/regis_1/Q
(3)
(2)
(1)
(0)
0000
1
0
0
0
0
0
UUU0
U
U
U
0
UUUU
UUU0
UU00
U001
0011
Now
1024 ns
Cursor 1
1.01 ns
1 ns
2 ns
3 ns
4 ns
5 ns
6 ns
7 ns
8 ns


Msgs
/regis_tb/regis_1/D
/regis_tb/regis_1/clock
/regis_tb/regis_1/load
/regis_tb/regis_1/reset
/regis_tb/regis_1/dir
/regis_tb/regis_1/L
/regis_tb/regis_1/R
/regis_tb/regis_1/Q
(3)
(2)
(1)
(0)
0000
1
0
0
0
0
1
UU00
U
U
0
0
UUUU
UUU0
UU00
U001
0011
Now
1024 ns
Cursor 1
3.02 ns
1 ns
2 ns
3 ns
4 ns
5 ns
6 ns
7 ns
8 ns


Wave - Default
Msgs
/regis_tb/regis_1/D
/regis_tb/regis_1/clock
/regis_tb/regis_1/load
/regis_tb/regis_1/reset
/regis_tb/regis_1/dir
/regis_tb/regis_1/L
/regis_tb/regis_1/R
/regis_tb/regis_1/Q
(3)
(2)
(1)
(0)
0110
1
0
1
1
1
1
0000
0
0
0
0
0110
0111
0000
Now
1024 ns
Cursor 1
623.558 ns
622 ns
623 ns
624 ns
625 ns
626 ns
627 ns
628 ns
629 ns





Análise

O registrador desenvolvido começa com suas variáveis de saída como “U”, por não estarem pré-definidas no arquivo .vhd referido a entidade desenvolvida.

A figura 2 da simulação nos mostra que na borda de subido do clock, quando dir = reset = load = 0, que a saída Q(0) terá como valor o de ‘L’ enquanto Q(1) receberá o antigo Q(0), Q(2) receberá o antigo Q(1) e Q(3) receberá o antigo Q(2). Gerando um deslocamento da palavra segundo o valor de ‘L’. Na figura 3, esse processo se repete e por isso nossa saída Q é agora “UU00”.

Quando reset e load = 0, porém dir = 1, o deslocamento da palavra ocorre em outro sentido e com influência do valor de ‘R’. Nesse caso, Q(3) receberá o valor de ‘R’, Q(2) o antigo Q(3), Q(1) o antigo Q(2) e Q(0) o antigo Q(1).

A figura 4 da simulação nos mostra que quando reset vale 1, nossa saída Q sempre será “0000”, independe do valor de quaisquer outras entradas.

As figuras 5 e 6 nos mostram que, caso reset não seja 1 e o load seja 1, então a saída Q será igual a entrada D;

Conclusão

Nesse experimento conseguimos com êxito implementar em vhdl um registrador de deslocamento de 4 bits utilizando a estrutura “process”. As simulações se comportaram como esperado e de acordo com a tabela disponibilizada no exercício.