Universidade de Brasília
Faculdade de Tecnologia
Laboratório de Sistemas Digitais

Relatório 06

Carla de Araujo Clementino Ribeiro Mat:180030736

Professor:
Guilherme de Sousa Torres

1 Objetivos

Implementar um flip-flop do tipo JK e um registrador deslocador bidirecional de 4 bits utilizando a estrutura process.

2 Questões Propostas

1. Usando a estrutura "process", implementar em VHDL e simular no ModelSim um flipflop JK gatilhado pela borda de subida.
2. Usando a estrutura "process", implementar em VHDL e simular no ModelSim um registrador de deslocamento bidirecional de 4 bits.

3 Desenvolvimento Teórico

3.1 Flip-Flop JK

Um flip-flop do tipo JK é um elemento de memória assíncrono, que armazena um único bit de informação. Ele possui duas entradas principais: J (Set) e K (Reset), além da entrada de clock (CLK) e uma possível entrada de reset (CLR).

A operação básica do flip-flop JK é determinada pela combinação das entradas J e K, juntamente com os flancos do sinal de clock. Aqui estão as principais características do flip-flop JK:

1. Set (J): Quando J é acionado (J=1), e K está em nível baixo (K=0), no momento em que ocorre uma transição de subida do clock (de 0 para 1), a saída Q é forçada a 1.
2. Reset (K): Quando K é acionado (K=1), e J está em nível baixo (J=0), no momento em que ocorre uma transição de subida do clock (de 0 para 1), a saída Q é forçada a 0.
3. Manutenção do Estado: Se tanto J quanto K estiverem em nível alto (J=K=1) ou ambos em nível baixo (J=K=0), a saída Q permanece no seu estado atual, independentemente do sinal de clock.
4. Toggle: Quando J e K estão ambos em nível alto (J=K=1), no flanco de subida do clock, a saída Q muda para o estado oposto do seu estado atual. Ou seja, se Q estava em 1, vai para 0 e vice-versa.
5. Entrada de Clock (CLK): O flip-flop JK é sensível ao flanco de subida ou descida do sinal de clock, dependendo da implementação específica.
6. Entrada de Reset (CLR): Algumas implementações de flip-flop JK incluem uma entrada de reset assíncrona (CLR), que, quando ativada, força a saída Q para um estado específico (normalmente 0).

3.2 Registrador Deslocador Bidirecional

Um registrador deslocador bidirecional de 4 bits é um componente digital que pode armazenar e deslocar quatro bits de dados em duas direções: esquerda e direita. As principais características são:

1. Armazenamento Inicial: No início, o registrador deslocador contém quatro bits de dados que podem ser inseridos através de uma entrada de dados.
2. Controle de Direção: O registrador deslocador tem um sinal de controle que determina a direção do deslocamento. Por exemplo, um sinal de controle "1"pode significar deslocamento para a esquerda, enquanto um sinal "0"pode significar deslocamento para a direita.
3. Deslocamento: Quando o sinal de controle de direção é ativado, os bits no registrador são deslocados em direção à direção especificada. Por exemplo, se o deslocamento for para a esquerda, os bits se movem para a esquerda e um novo bit pode ser inserido no lado direito do registrador. Se o deslocamento for para a direita, os bits se movem para a direita e um novo bit pode ser inserido no lado esquerdo do registrador.
4. Entrada de Dados: Além do deslocamento, o registrador deslocador também pode ter uma entrada de dados que permite a inserção de um novo bit de dados. Este bit pode ser inserido no lado oposto do registrador, dependendo da direção do deslocamento.
5. Saída de Dados: O registrador deslocador também tem uma saída que fornece o bit de dados mais à direita (ou mais à esquerda, dependendo da direção do deslocamento). Esta saída pode ser usada para conectar o registrador a outros componentes do circuito.

Em resumo, um registrador deslocador bidirecional de 4 bits pode armazenar quatro bits de dados e deslocá-los em duas direções, permitindo uma variedade de operações úteis em circuitos digitais, como deslocamento de bits, rotação de bits e implementação de operações lógicas.

4 Simulação

4.1 Primeira Questão

Primeiramente, foi implementada a entidade do flip-flop tipo K de acordo com a tabela verdade conhecida:

```
Ln#
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3
 4  entity jk_flipflop is
 5    port(
 6      J, K, CLK, CLR, PR : in  std_logic;
 7      Q, Qbar      : out std_logic
 8    );
 9  end jk_flipflop;
10
11  architecture behavior of jk_flipflop is
12    signal temp_q : std_logic;
13    signal aux: std_logic_vector(1 downto 0);
14
15  begin
16    aux <= J & K;
17
18    process(CLK, CLR, PR)
19    begin
20      if PR = '1' then
21            temp_q <= '1';
22      elsif CLR = '1' then
23            temp_q <= '0';
24      elsif PR = '1' then
25            temp_q <= '1';
26      elsif rising_edge(CLK) then
27        if aux = "00" then
28          temp_q <= temp_q;
29        elsif aux = "01" then
30          temp_q <= '0';
31        elsif aux = "10" then
32          temp_q <= '1';
33        elsif aux = "11" then
34          temp_q <= not temp_q;
35        else
36          temp_q <= temp_q;
37        end if;
38      end if;
39    end process;
40
41    Q <= temp_q;
42    Qbar <= not temp_q;
43  end behavior;
44
```

Figura 1: Flip-Flop JK. Fonte: Autor.

Por fim, foi criado um arquivo de teste para simular todas as possíveis entradas do circuito:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity test_bench is
end test_bench;

architecture main of test_bench is

component jk_flipflop is
    port(
        J, K, CLK, CLR, PR : in std_logic;
        Q, QBar          : out std_logic
    );
end component;

signal u_j, u_k, u_clk, u_clr, u_pr, u_q, u_qbar : std_logic := '0';

begin

  test_bench_ff_JK : jk_flipflop port map(
    CLK => u_clk,
    CLR => u_clr,
    J => u_j,
    K => u_k,
    PR => u_pr,
    Q => u_q,
    QBar => u_qbar
  );

  u_clk <= not u_clk after 1 ns;
  u_clr <= not u_clr after 2 ns;
  u_j <= not u_j after 4 ns;
  u_k <= not u_k after 8 ns;
  u_pr <= not u_pr after 16 ns;
end main;
```

Figura 2: Test Bench - Flip-Flop JK. Fonte: Autor.

Realizando a simulação e comparando com a tabela verdade fornecida obtemos os seguintes resultados:

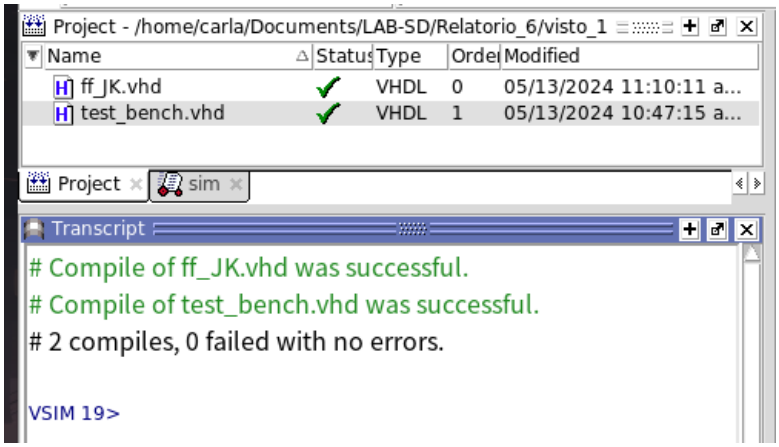

Figura 3: Compilação - Flip-Flop JK. Fonte: Autor.

Legenda:

- Sinal u_j: Entrada J;
- Sinal u_k: Entrada K;
- Sinal u_pr: Entrada PR;
- Sinal u_clk: Sinal de clock;

- Sinal u_clr: Sinal de clear;
- Sinal u_q: Saída Q (destacado em branco);
- Sinal u_qbar: Saída Q barrada;

1. Para PR = 1

Figura 4: Simulação - PR habilitado. Fonte: Autor.

2. Para CLR = 1

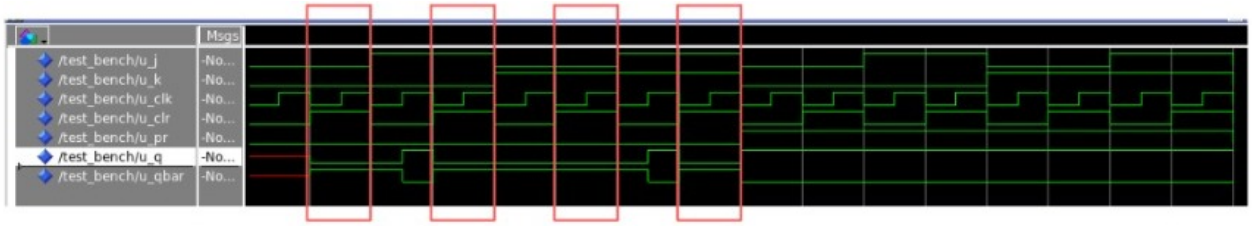

Figura 5: Simulação - CLR habilitado. Fonte: Autor.

3. Todas as entradas desabilitadas

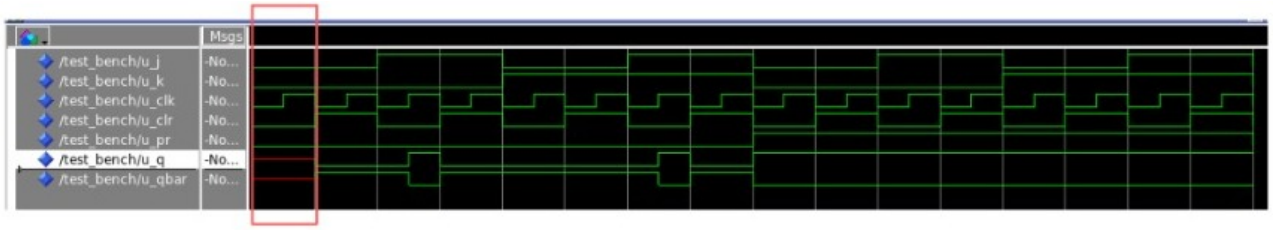

Figura 6: Simulação - Entradas desabilitadas. Fonte: Autor.

4. K habilitado

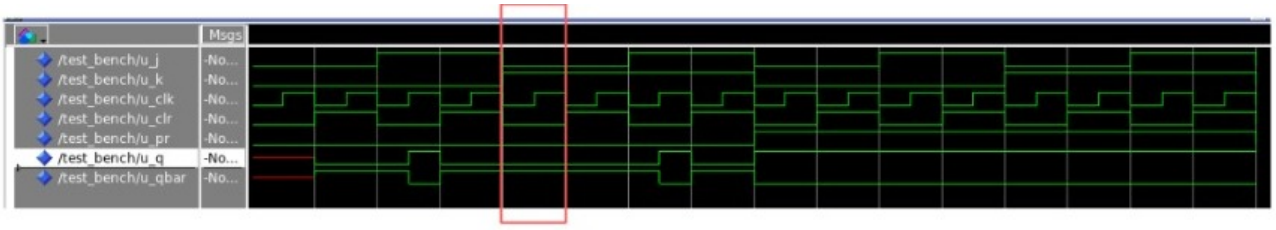

Figura 7: Simulação - K habilitado. Fonte: Autor.

5. J habilitado

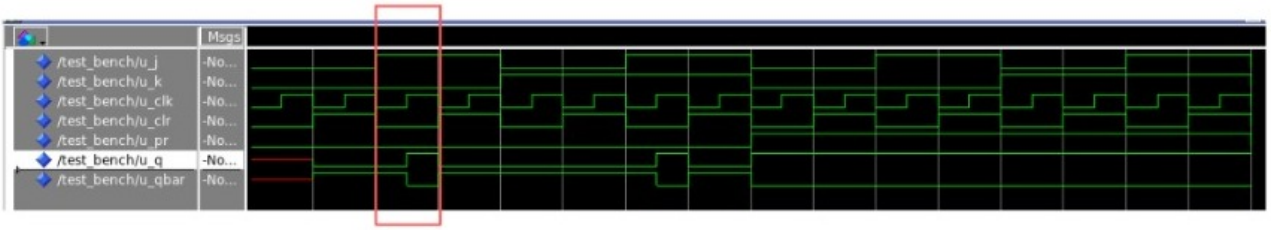

Figura 8: Simulação - J habilitado. Fonte: Autor.

6. J e K habilitado

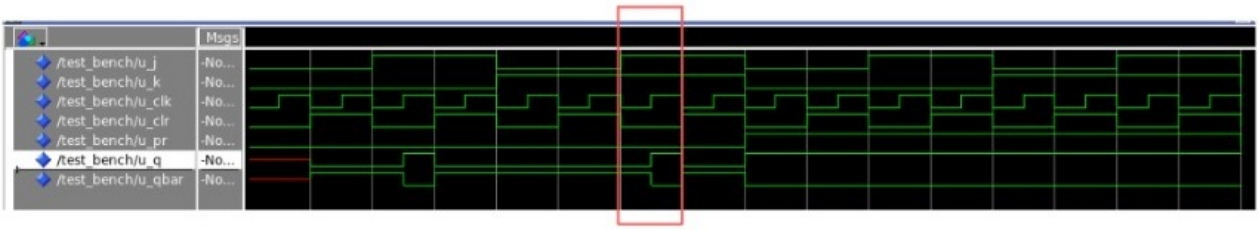

Figura 9: Simulação - J e K habilitado. Fonte: Autor.

4.2 Segunda Questão

Primeiramente, foi implementada a entidade do registrador deslocador bidirecional de 4 bits:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4
5  entity shift_register is
6      port(
7          DIR, L, R, RST, LOAD, CLK: in  std_logic;
8          Q: out std_logic_vector(3 downto 0));
9      end shift_register;
10
11 architecture main of shift_register is
12     signal temp_q : std_logic_vector(3 downto 0);
13 begin
14     process (CLK)
15         begin
16             if rising_edge(CLK) then
17                 if RST = '1' then temp_q <= "0000";
18                 elsif LOAD = '1' then temp_q <= temp_q;
19                 elsif DIR = '0' then temp_q <= temp_q(2) & temp_q(1) & temp_q(0) & L;
20                 elsif DIR = '1' then temp_q <= L & temp_q(3) & temp_q(2) & temp_q(1);
21                 end if;
22             else temp_q <= temp_q;
23             end if;
24     end process;
25     Q <= temp_q;
26 end main;
```

Figura 10: Entidade - Registrador Deslocador. Fonte: Autor.

Em seguida, foi criado um arquivo de teste para simular todas as possíveis entradas do circuito:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity test_bench is
end;

architecture main of test_bench is
  component shift_register is
    port(
      DIR, L, R, RST, LOAD, CLK: in  std_logic;
      Q: out std_logic_vector(3 downto 0)
    );
  end component;
  signal u_dir : std_logic := '1';
  signal u_l, u_r, u_rst, u_load, u_clk: std_logic := '0';
  signal u_q : std_logic_vector(3 downto 0);

begin
  tb_shift_register : shift_register port map(
    DIR => u_dir,
    L => u_l,
    R => u_r,
    RST => u_rst,
    LOAD => u_load,
    CLK => u_clk,
    Q => u_q
  );

  u_clk <= not u_clk after 1 ns;
  u_l <= not u_l after 2 ns;
  u_r <= not u_r after  2 ns;
  u_dir <= not u_dir after 64 ns;
  u_load <= not u_load after 32 ns;
  u_rst <= not u_rst after 128 ns;

end main;
```

Figura 11: Test Bench - Registrador Deslocador. Fonte: Autor.

Realizando a simulação e comparando com a tabela verdade fornecida obtemos os seguintes resultados:

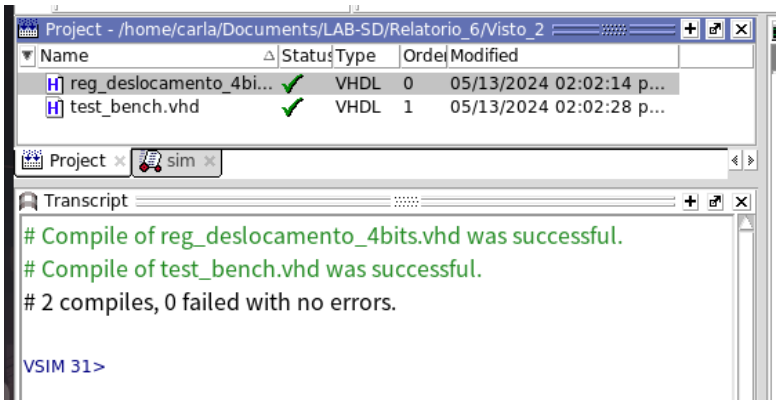

Figura 12: Compilação - Registrador Deslocador. Fonte: Autor.

Legenda:

- Sinal u_dir: Entrada DIR;
- Sinal u_l: Entrada L;

- Sinal u_r: Entrada R;
- Sinal u_rst: Sinal de RST;
- Sinal u_clk: Sinal de CLK;
- Sinal u_q: Saída Q

1. DIR = 1

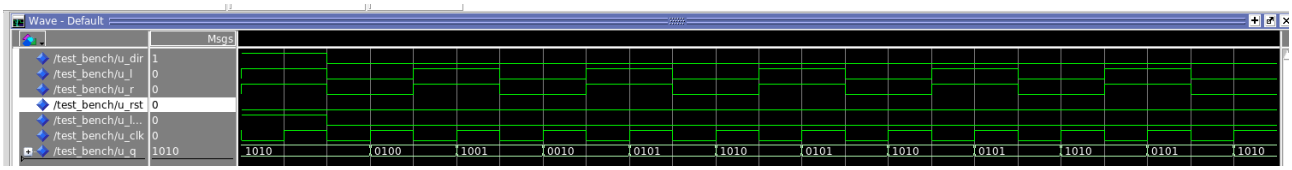

Figura 13: Simulação - DIR habilitado. Fonte: Autor.

2. DIR = 0

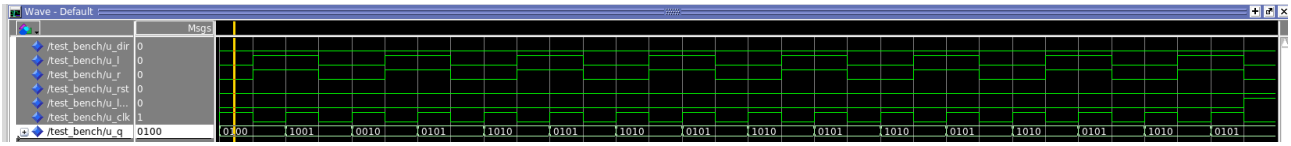

Figura 14: Simulação - DIR desabilitado. Fonte: Autor.

3. LOAD = 1

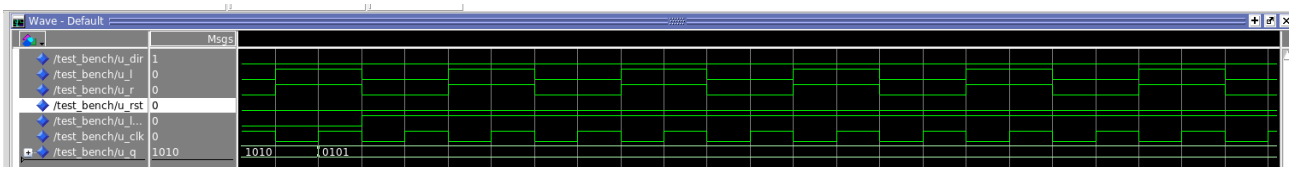

Figura 15: Simulação - Load habilitado. Fonte: Autor.

5 Conclusão

A estrutura de processos do VHDL nos permitiu modelar o comportamento assíncrono e síncrono do flip-flop do tipo JK e do registrador deslocador bidirecional, incluindo a sensibilidade aos sinais de clock, reset e outras entradas relevantes. A simulação no ModelSim nos deu a oportunidade de verificar o funcionamento correto dos circuitos, observar as transições de estados e identificar possíveis erros ou problemas de projeto.