Atividade:

Construir os circuito sequenciais dos Flip-flops apresentados em aula teórica:

- FF JK com Clock, Preset e Clear, chamado carinhosamente de FFJK
- FF TIPO-D
- FF TIPO-T

Implementar em VHDL e simular.

Entrega:

- Arquivos .vhdl, ghw e gtkw utilizados
- via TAREFAS na Equipe Teams COM ANEXO (pacote .zip)

Importante:

- O objetivo da prática de hoje é implementar os FFs JK, D e T que serão usados em aulas posteriores. Sendo assim, esta aula é fundamental para implementação das práticas seguintes.

Circuito Flip-Flop JK:

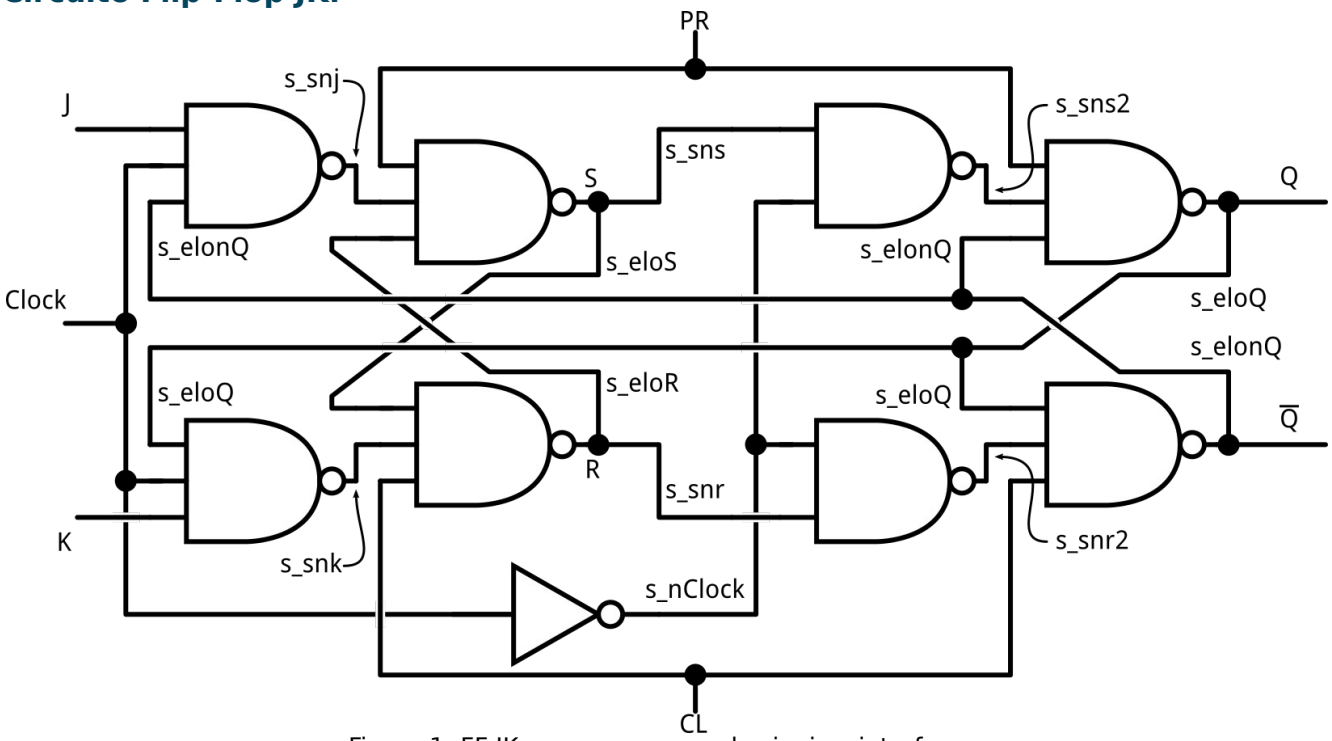

Figura 1: FF JK como os nomes de sinais e interfaces.

Os arquivos base para o Flip-Flop JK e Testbench encontram-se na sequência.

- Não esquecer de produzir os FF TIPO-D e TIPO-T

VHDL base para Flip-Flop JK:

```
01. library ieee;
02. use ieee.std_logic_1164.all;
03.
04. entity ffjk is
05.       port(
06.             j, k   : in  std_logic;
07.             clock  : in  std_logic;
08.             pr, cl : in  std_logic;
09.             q, nq  : out std_logic
10.       );
11. end ffjk;
12.
13. architecture ff of ffjk is
14.     signal s_snj , s_snk  : std_logic;
15.     signal s_sns , s_snr  : std_logic;
16.     signal s_sns2, s_snr2 : std_logic;
17.     signal s_eloS, s_eloR : std_logic;
18.     signal s_eloQ, s_elonQ: std_logic;
19.     signal s_nClock       : std_logic;
20. begin
21.
22.     s_nClock <= not(clock);
23.     -- envio de saídas de NAND para Q e NQ
24.
25.     --  s_snj
26.     -- NAND de 3 entradas? Faça not( X and Y and Z)
27.
28.     --  s_snk
29.
30.     --  s_sns
31.
32.     --  s_snr
33.
34.     --  s_sns2
35.
36.     --  s_snr2
37.
38.     --  s_eloS
39.
40.     --  s_eloR
41.
42.     --  s_eloQ
43.
44.     --  s_elonQ
45.
46. end architecture ff;
```

VHDL base para Testbench do Flip-Flop JK:

```
01. library ieee;
02. use ieee.std_logic_1164.all;
03.
04. entity tb_ffjk is
05.     -- entidade vazia
06. end tb_ffjk;
07.
08. architecture test of tb_ffjk is
09.     constant CLK_PERIOD : time := 20 ns;
10.
11.     component ffjk is
12.     port(
13.         j, k   : in  std_logic;
14.         clock  : in  std_logic;
15.         pr, cl : in  std_logic;
16.         q, nq  : out std_logic
17.     );
18.     end component;
19.
20.     signal sj, sk, spr, scl, sq, snq : std_logic;
21.     signal sclk : std_logic := '1';
22.
23. begin
24.     -- instancia de JK e port map
25.
26.     -- process
27.     tbp : process
28.     begin
29.         spr <= '1';
30.         scl <= '0';
31.         sj  <= '0';
32.         sk  <= '0';
33.
34.         -- desativação de clear
35.
36.         wait for CLK_PERIOD;
37.
38.         -- alteração de J e K
39.
40.     end process;
41.
42.     -- process para Clock
43.     p_clock : process
44.     begin
45.         sclk <= not(sclk);
46.         wait for CLK_PERIOD/2;
47.     end process;
48.
49. end architecture test;
```

Última observação:

Nas futuras práticas, em caso de problema com o FF JK do aluno, sendo o culpado o GHDL, será possível utilizar outra implementação destes FF que utiliza **process** para determinar as saídas Q e $\overline{Q}$, e será disponibilizado.