E208 – Eletrônica Digital II **4º Período / 1º Semestre de 2021**

Professores: Bruno de Oliveira Monteiro bruno@inatel.br

Monitores: Felipe Pereira Silveira felipepereira@gea.inatel.br

Carlos Daniel Borges Vilela Marques carlos.marques@gea.inatel.br

Gualter Machado Mesquita machadomgualter@gmail.com

Maíra Alves Chagas mairaalves@gec.inatel.br

Pedro Henrique Praxedes dos Reis pedro.reis@gea.inatel.br

Thalita Fortes Domingos thalita.fortes@gec.inatel.br

Maria Luiza Rosestolato Araújo maria.luiza@gec.inatel.br

Marcos Henrique Rodrigues Lopes marcos.lopes@gea.inatel.br

Thiago da Rocha Miguel thiago.miguel@gec.inatel.br

Aluno: _____ **Matrícula:** _____ **Período:** _____ **Data:** ___ / ___ / ___

Assunto da semana: Flip Flop RS e JK

Relatório 2

Teoria

O Latch SR

A Figura 1 mostra o diagrama lógico de uma Latch S-R (Set e Reset) com enable, implementado com portas NAND:

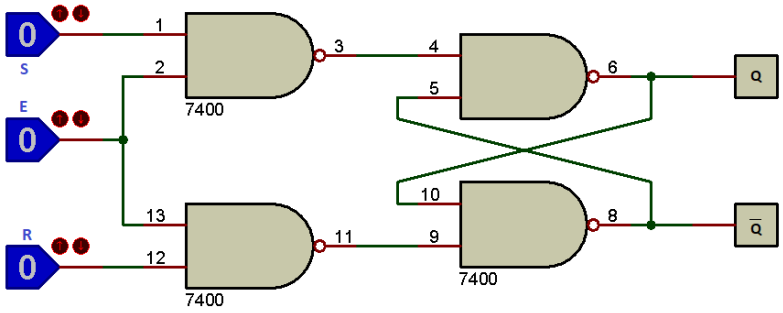


Figura 1

A Latch possui três entradas (SET, RESET e Enable) e duas saídas (Q e $\overline{Q}$). O valor lógico de Q e $\overline{Q}$ definem o estado da Latch. Note que a ação da entrada ENABLE é habilitar (E = 1) ou desabilitar (E = 0) a mudança do estado da Latch, mudança que está ao encargo das entradas SET e RESET.

O Flip-Flop SR - Mestre-Escravo

No flip-flop SR as entradas 'S & R' são habilitadas somente quando na entrada CLK ocorre uma transição do nível logico alto para o baixo (transição de descida - ou vice-versa depende do flip-flop), enquanto na Latch SR 'S & R' são habilitadas por nível. A Figura 2 mostra o diagrama lógico de um flip-flop S-R implementado com duas Latch's e uma porta inversora:

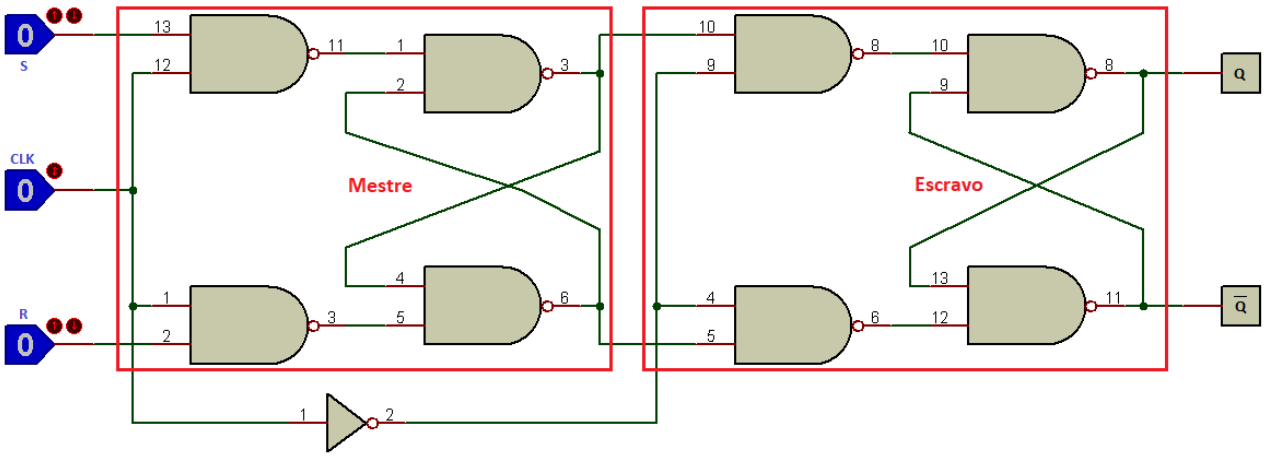


Figura 2

Analisando o circuito:

Suponha que o CLOCK está inicialmente no nível zero. Nessa condição, o bloco mestre está inativo e variações nas entradas S e R não produzem mudanças na saída.

Quando o CLOCK passa para 1, o circuito escravo é bloqueado, mantendo a saída Q anterior. Variações nas entradas produzem variações em Q e Q' da Latch Mestre, mas não afetam a saída porque a porta inversora se encarrega de deixar o Latch Escravo inativa.

Quando o CLOCK passa para zero, o mestre é bloqueado e o escravo, liberado. Assim, ele assume a saída correspondente ao estado anterior à transição. E a Tabela de Verdade é a mesma da Latch SR, considerando que as mudanças só ocorrem nas transições de 1 para 0 do CLOCK.

O flip-flop possui três entradas (SET, RESET e CLOCK) e duas saídas (Q e $\overline{Q}$). O valor lógico de Q e $\overline{Q}$ definem o estado do flip-flop. Note que a ação da entrada CLOCK é habilitar apenas na transição de decida (↓ - Demonstrado na figura 3), a mudança do estado do flip-flop, mudança que está ao encargo das entradas S e R. Isso acontece de modo em que a Latch 'Mestre' envia para a Latch 'Escravo' a ultima informação vista antes de ser desabilitada, assim logo depois do tempo de propagação da porta inversora o escravo pega essa informação e envia para a saída.

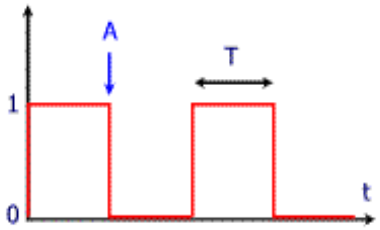


Figura 3

A Tabela abaixo mostra a 'Tabela da Verdade' do flip-flop S-R.

CLK	S	R	Q	$\overline{Q}$
0	*	*	Qa	$\overline{Q}$a
↑	*	*	Qa	$\overline{Q}$a
1	*	*	Qa	$\overline{Q}$a
↓	0	0	Qa	$\overline{Q}$a
↓	0	1	0	1
↓	1	0	1	0
↓	1	1	Inválida	

Tabela 1

Flip-Flop JK - Mestre-Escravo

O flip-flop JK foi desenvolvido para resolver certo problema que o ocorre com o flip-flop SR, quando as entradas 'S & R' são iguais a 1 o flip-flop entra num estado de indeterminação, variando constantemente suas saídas, sendo impossível determinar o nível logico de cada uma.

Na configuração JK utiliza-se duas portas E(&) para analisar a saída, assim quando 'S & R' encontra-se em 1 no instante de transição da entrada CLK, ele inverte o nível logico da saída anterior, como de 0 para 1 ou de 1 para 0.

A Figura 4 mostra como é realizada essa configuração.

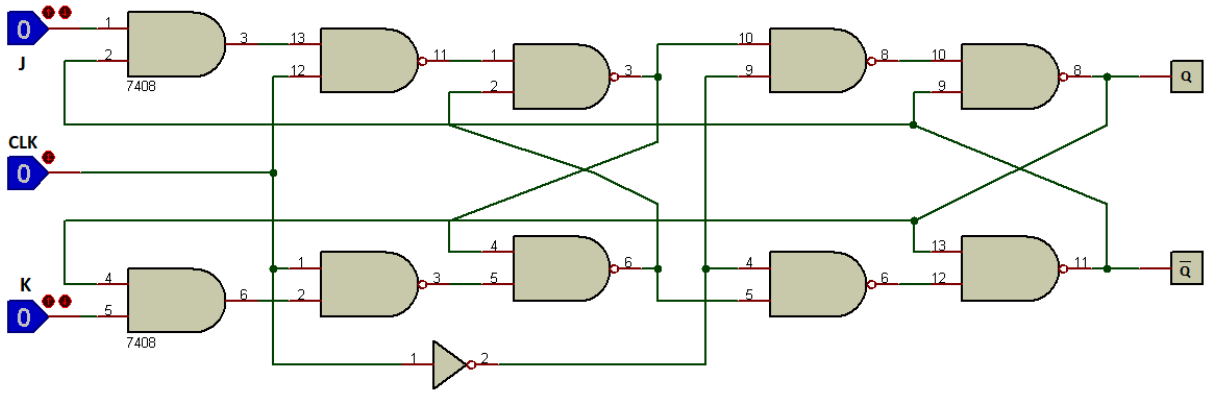


Figura 4

Tabela da verdade do flip flop JK:

CLK	J	K	Q	$\overline{Q}$a
0	*	*	Qa	$\overline{Q}$a
↑	*	*	Qa	$\overline{Q}$a
1	*	*	Qa	$\overline{Q}$a
↓	0	0	Qa	$\overline{Q}$a
↓	0	1	0	1
↓	1	0	1	0
↓	1	1	$\overline{Q}$a	Qa

Tabela 2

Uma vez solucionado todos os problemas da tabela do flip-flop, foi questionado que o circuito, ao ser iniciado (ou seja, ligado/energizado) não havia uma definição inicial em suas saídas. Foi então em que os pinos de Preset (ou Set) e Clear (ou Reset) vieram ser inseridos no circuito, de forma que, quando acionadas, suas definições de saídas prevaleçam.

Modelo de diagrama em bloco de um flip flop JK na Figura 5

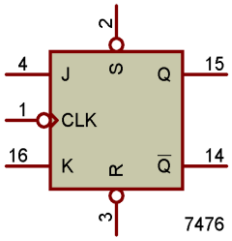


Figura 5

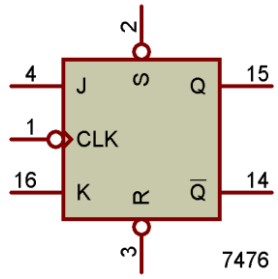
Tabela da verdade de um flip flop JK com o Preset e Clear:

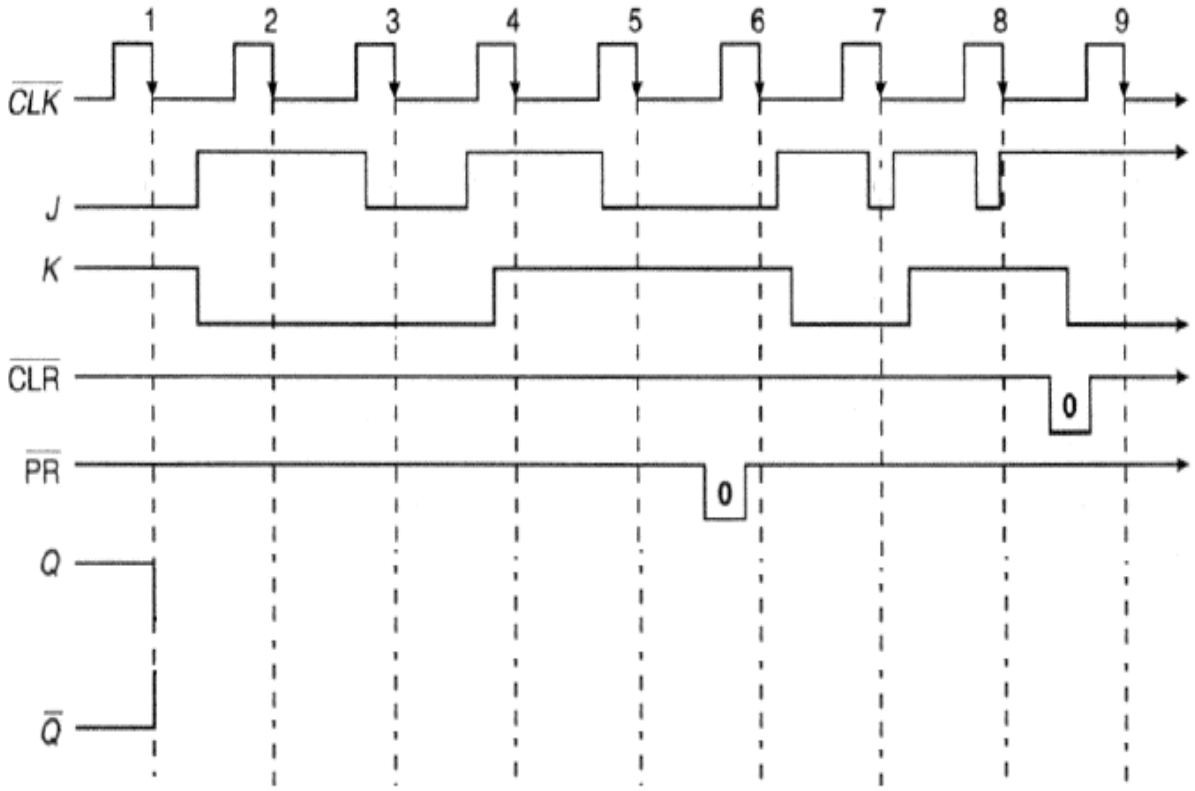
PR	Clear	CLK	J	K	Q	$\overline{Q}a$
0	0	*	*	*	*	*
0	1	*	*	*	1	0
1	0	*	*	*	0	1
1	1	0	*	*	Qa	$\overline{Q}a$
1	1	↑	*	*	Qa	$\overline{Q}a$
1	1	1	*	*	Qa	$\overline{Q}a$
1	1	↓	0	0	Qa	$\overline{Q}a$
1	1	↓	0	1	0	1
1	1	↓	1	0	1	0
1	1	↓	1	1	$\overline{Q}a$	Qa

Exercício Teórico

Questão 1. Determine as formas de onda nas saídas Q e $\overline{Q}$ do flip-flop JK quando as formas de onda de entrada são conforme a figura a seguir:

Dica: lembre-se que as entradas assíncronas CLEAR (R) e PRESET (S) tem prioridade sobre todas as outras entradas.





Questão 2. Baseado no seu conhecimento sobre VHDL procure entender o código a seguir, o qual simula um flip-flop, comente o código explicando a função de cada linha de código.

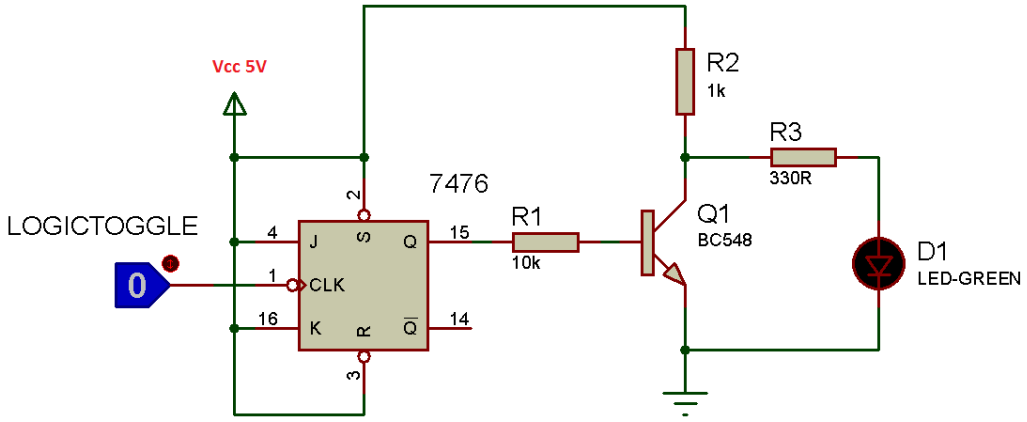
Código de VHDL

```
ENTITY FF_JK is
    PORT(   k : in  STD_LOGIC;
            j : in  STD_LOGIC;
            reset : in  STD_LOGIC;
            clk : in  STD_LOGIC;
            q : out  STD_LOGIC;
            q_bar : out  STD_LOGIC);
END FF_JK;

ARCHITECTURE Behavioral OF FF_JK IS

signal sinal_q: STD_LOGIC;

BEGIN
    process(clk, reset)
BEGIN

    IF (reset = '1') THEN
        sinal_q <= '0';
    ELSIF rising_edge(clk) THEN
        IF (j = '0' and k = '0') THEN
            sinal_q <= sinal_q;
        ELSIF (j = '0' and k = '1') THEN
            sinal_q <= '0';
        ELSIF (j = '1' and k = '0') THEN
            sinal_q <= '1';
        ELSIF (j = '1' and k = '1') THEN
            sinal_q <= not (sinal_q);
        END IF;
    ELSIF falling_edge(clk) THEN
        sinal_q <= sinal_q;
    END IF;

    END PROCESS;

    q <= sinal_q;
    q_bar <= not sinal_q;


END Behavioral;
```

Exercício Prático

Questão 3. Faça a montagem no módulo digital e no Proteus ISIS (ao mesmo tempo divida tarefas) demonstrada pela figura abaixo. No software Proteus:

Obs: Todas as atividades que solicitarem montagem no módulo digital devem ser realizadas no TinkerCad enquanto as aulas não retornarem de forma presencial.



- LOGICTOGGLE: categoria Debbugging Tools.

- Flip-flop JK: categoria TTL 74 series, sub-categoria Flip-Flop & Latch.

- Resistores: categoria Resistors, sub-categoria 0.6W Metal Film.

- Transistor: categoria Transistors, sub-categoria Bipolar.

- LED: categoria Optoelectronics, sub-categoria LEDs.

a. Descreva comentários sobre a configuração do flip-flop JK, exaltando o que acontece com a saída Q para cada pulso de decida do CLOCK.

b. O que acontece com a saída Q do flip-flop quando o CLOCK recebe um pulso de subida?

c. Observe o que acontece com o LED-GREEN comparado com a saída Q do flip-flop. Descreva comentários sobre sua observação e exalte comentários técnicos relacionando a importância do transistor.

Questão 4. Ainda no modulo verifique todas as condições possíveis do flip-flop, conecte as entras J, K e CLk nas chaves do modulo, as duas saídas em dois led's do modulo e gerando um pulso manualmente teste as 4 condições de J e K quando:

a) Set = 0 & Rest = 1;

b) Set = 1 & Rest = 1;

c) Set = 1 & Rest = 0;

d) Set = 0 & Rest = 0;

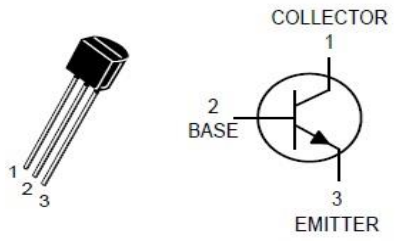
Questão 5. Pesquise na Internet o datasheet do CI 7476 e encontre o tempo de propagação do Flip-flop JK, ou seja, o tempo em que o flip-flop demora para alterar em suas saídas quando recebe um pulso de clock.

Questão 6. Realize a simulação de um flip-flop JK no software ISE(Questão 2).

Código de teste

```
clk <= '0';
reset <= '1';
wait for 50 ns;
reset <= '0';

j <= '0';
k <= '0';
wait for 50 ns;
clk <= '0';
wait for 50 ns;
clk <= '1';
wait for 50 ns;
clk <= '0';
wait for 50 ns;

j <= '0';
k <= '1';
wait for 50 ns;
clk <= '0';
wait for 50 ns;
clk <= '1';
wait for 50 ns;
clk <= '0';
wait for 50 ns;

j <= '1';
k <= '0';
wait for 50 ns;
clk <= '0';
wait for 50 ns;
clk <= '1';
wait for 50 ns;
clk <= '0';
wait for 50 ns;

j <= '1';
k <= '1';
wait for 50 ns;
clk <= '0';
wait for 50 ns;
clk <= '1';
wait for 50 ns;
clk <= '0';
wait for 50 ns;

reset <= '1';
```

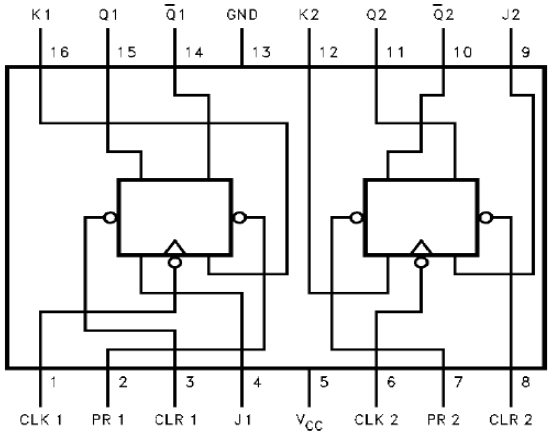
Transistor BC 548



BC 548 Transistor

CI 7476

Connection Diagram



Function Table

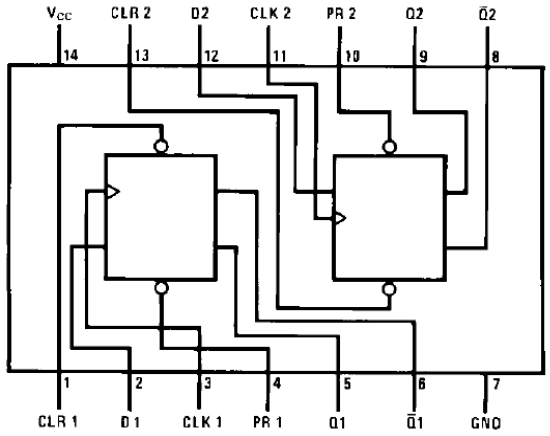
Inputs					Outputs	
PR	CLR	CLK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H (Note 1)	H (Note 1)
H	H	⎍	L	L	Q_0	$\overline{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	Toggle	

H = HIGH Logic Level
L = LOW Logic Level
X = Either LOW or HIGH Logic Level
⎍ = Positive pulse data. The J and K inputs must be held constant while the clock is HIGH. Data is transferred to the outputs on the falling edge of the clock pulse.
Q_0 = The output logic level before the indicated input conditions were established.
Toggle = Each output changes to the complement of its previous level on each complete active HIGH level clock pulse.

Note 1: This configuration is nonstable; that is, it will not persist when the preset and/or clear inputs return to their inactive (HIGH) level.

CI 7474

Connection Diagram



Function Table

Inputs				Outputs	
PR	CLR	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H (Note 1)	H (Note 1)
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

H = HIGH Logic Level
X = Either LOW or HIGH Logic Level
L = LOW Logic Level
↑ = Positive-going transition of the clock.
Q_0 = The output logic level of Q before the indicated input conditions were established.

Note 1: This configuration is nonstable; that is, it will not persist when either the preset and/or clear inputs return to their inactive (HIGH) level.

Itens que devem conter no quite:

- Um protoboard;
- Um CI 7476;
- Um transistor BC 548;
- Resistores de 1kR – 10kR – 330R;
- Um LED 5m.