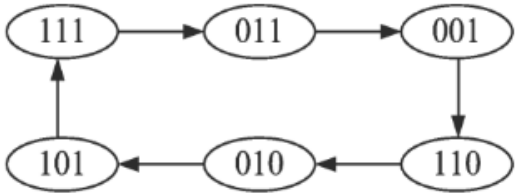
Automat de Tranziție
Bistabil JK
Multiplexor MUX4:1

Melinte Cosmin

Cuprins

Tema de Proiect

Automat de tranziție:



Bistabil JK :

r	clk	Action
1	x	Reset
0	↓	$Q^+ = JK$
otherwise		Wait

Multiplexor:

III. MUX 4:1 și porți logice

Rezolvarea temei de proiect:

Implementarea Bistabilului JK la nivel de cod VHDL

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

entity JK is
    Port ( CK : in STD_LOGIC;
           R : in STD_LOGIC;
           J : in STD_LOGIC;
           K : in STD_LOGIC;
           Q : out STD_LOGIC;
           Qn : out STD_LOGIC);
end JK;

architecture Behavioral of JK is

signal input : std_logic_vector(1 downto 0);
signal stare : std_logic;
begin
input <= J&K;
process(CK,R)
begin
if R='1' then stare <= '0';
else
    if falling_edge(CK) then case input is when "00" => stare <= stare;
                                           when "01" => stare <='0';
                                           when "10" => stare <='1';
                                           when "11" => stare <= not stare;
                                           when others => stare <= 'X';
                        end case;
    end if;
end if;
end process;
Q <= stare;
Qn <= not stare;
end Behavioral;
```

Parametri intrări și ieșiri

Front descendent

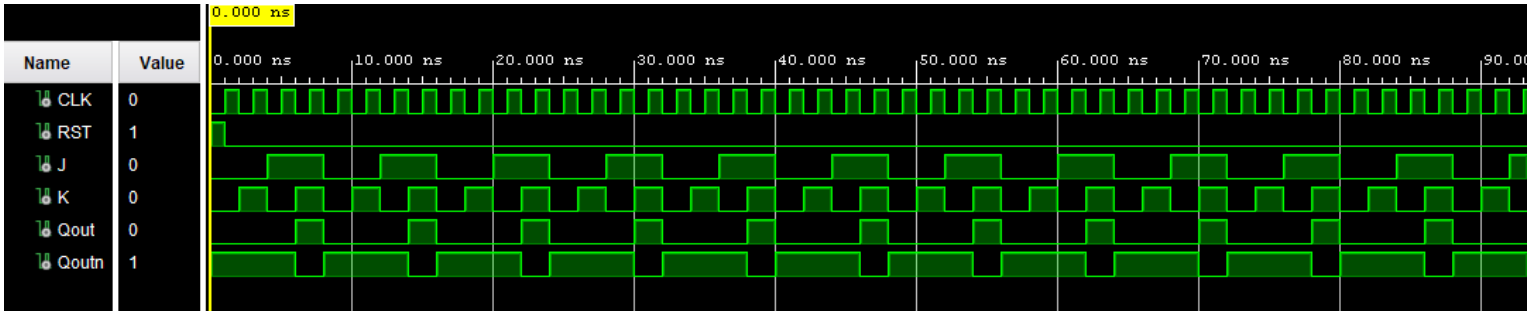
Tabel de adevăr

Q si Q_{negat}

Implementarea Sursei pentru Bistabilului JK

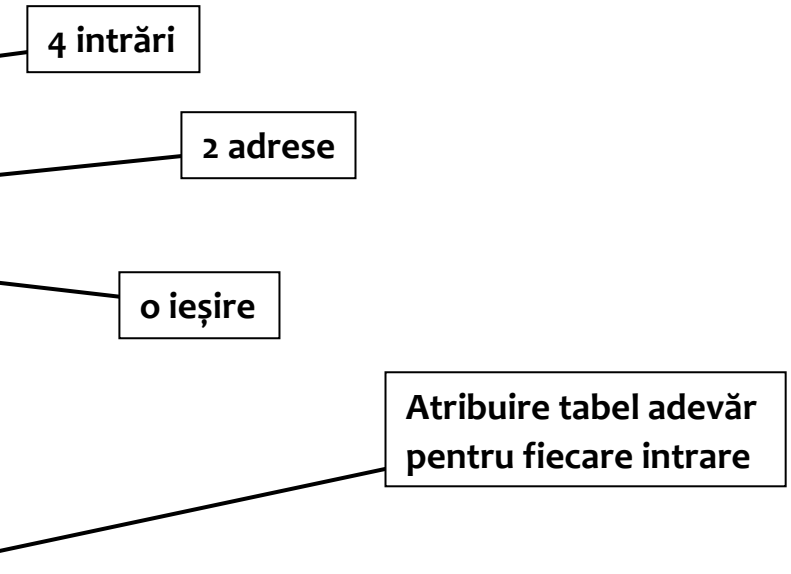
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JKTest is
end JKTest;

architecture Behavioral of JKTest is
component JK is
    Port ( CK : in STD_LOGIC;
           R : in STD_LOGIC;
           J : in STD_LOGIC;
           K : in STD_LOGIC;
           Q : out STD_LOGIC;
           Qn : out STD_LOGIC);
end component JK;

signal CLK, RST, J, K: std_logic;
signal Qout, Qoutn: std_logic;

begin
UUT: JK port map ( CK=>CLK, R=>RST, J=>J, K=>K, Q=>Qout, Qn=>Qoutn);

generate_CLK: process
begin
CLK<='0'; wait for 1ns;
CLK<='1'; wait for 1ns;
end process;

generate_rst: process
begin
RST<='1'; wait for 1ns;
RST<='0'; wait;
end process;
```

```vhdl

generate_J: process
begin
J<='0'; wait for 4ns;
J<='1'; wait for 4ns;
end process;

generate_K: process
begin
K<='0'; wait for 2ns;
K<='1'; wait for 2ns;
end process;

end Behavioral;
```



Implementarea Multiplexorului MUX4:1 la nivel de cod VHDL

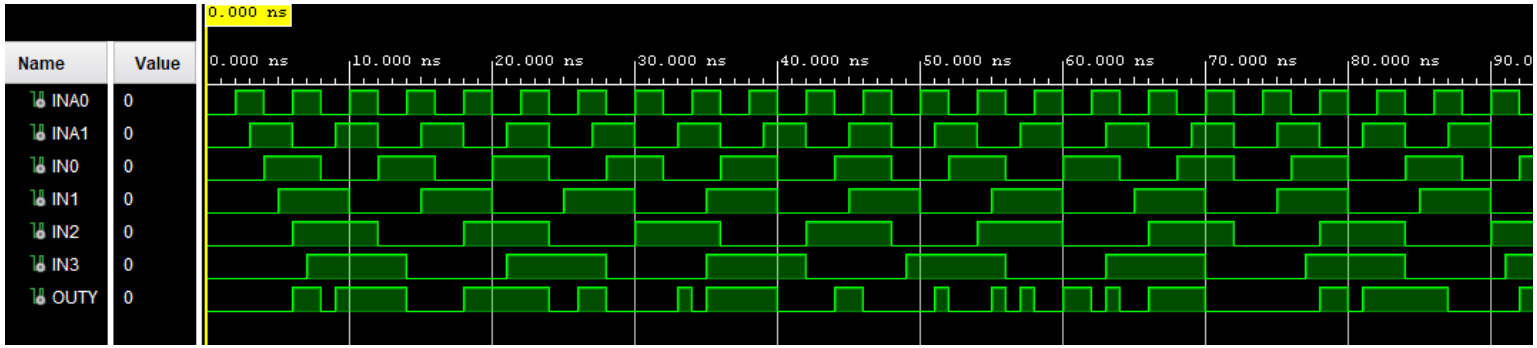
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity MUX4_1 is
5      Port ( I0 : in STD_LOGIC;
6             I1 : in STD_LOGIC;
7             I2 : in STD_LOGIC;
8             I3 : in STD_LOGIC;
9             A0 : in STD_LOGIC;
10            A1 : in STD_LOGIC;
11            Y : out STD_LOGIC);
12 end MUX4_1;
13
14 architecture Behavioral of MUX4_1 is
15
16 signal a: std_logic_vector(1 downto 0);
17
18 begin
19 a <= A1 & A0;
20 with a select
21 Y <= I0 when "00", I1 when "01", I2 when "10", I3 when "11", I0 when others;
22
23 end Behavioral;
```



Implementarea Sursei pentru Multiplexorului MUX4:1

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity MUX4_1Test is
5  end MUX4_1Test;
6
7  architecture Behavioral of MUX4_1Test is
8  component MUX4_1 is
9      Port ( I0 : in STD_LOGIC;
10            I1 : in STD_LOGIC;
11            I2 : in STD_LOGIC;
12            I3 : in STD_LOGIC;
13            A0 : in STD_LOGIC;
14            A1 : in STD_LOGIC;
15            Y : out STD_LOGIC);
16 end component MUX4_1;
17
18 signal INA0,INA1,IN0,IN1,IN2,IN3,OUTY: std_logic;
19 begin
20 UUT: MUX4_1 port map (I0 => IN0, I1 =>IN1, I2 =>IN2, I3 => IN3, A0 =>INA0, A1 =>INA1, Y =>OUTY);
21
22 generate_INA0: process
23 begin
24 INA0<='0'; wait for 2ns;
25 INA0<='1'; wait for 2ns;
26 end process;
27
28 generate_INA1: process
29 begin
30 INA1<='0'; wait for 3ns;
31 INA1<='1'; wait for 3ns;
32 end process;
```

```
33
34 generate_IN0: process
35 begin
36 IN0<='0'; wait for 4ns;
37 IN0<='1'; wait for 4ns;
38 end process;
39
40 generate_IN1: process
41 begin
42 IN1<='0'; wait for 5ns;
43 IN1<='1'; wait for 5ns;
44 end process;
45
46 generate_IN2: process
47 begin
48 IN2<='0'; wait for 6ns;
49 IN2<='1'; wait for 6ns;
50 end process;
51
52 generate_IN3: process
53 begin
54 IN3<='0'; wait for 7ns;
55 IN3<='1'; wait for 7ns;
56 end process;
57
58 end Behavioral;
```



Implementarea Automatului JK MUX4:1 la nivel de cod VHDL

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

entity AutomatJK is
    Port ( R : in std_logic;
           CLK : in std_logic;
           Q : out std_logic_vector(2 downto 0));
end AutomatJK;

architecture Behavioral of AutomatJK is

component MUX4_1 is
    Port ( I0 : in STD_LOGIC;
           I1 : in STD_LOGIC;
           I2 : in STD_LOGIC;
           I3 : in STD_LOGIC;
           A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           Y : out STD_LOGIC);
end component MUX4_1;

component JK is
    Port ( CK : in STD_LOGIC;
           R : in STD_LOGIC;
           J : in STD_LOGIC;
           K : in STD_LOGIC;
           Q : out STD_LOGIC;
           Qn : out STD_LOGIC);
end component JK;

signal net: std_logic_vector(2 downto 0);
signal Qint: std_logic_vector(2 downto 0);
signal Q0_neg, Q1_neg, Q2_neg: std_logic;

begin
U1: JK port map ( CK=>CLK, R=>R, J=>net(0), K=>net(0), Q=>Qint(0), Qn=>Q0_neg );
U2: JK port map ( CK=>CLK, R=>R, J=>net(1), K=>net(1), Q=>Qint(1), Qn=>Q1_neg );
U3: JK port map ( CK=>CLK, R=>R, J=>net(2), K=>net(2), Q=>Qint(2), Qn=>Q2_neg );
Q0_neg <= not Qint(0);
Q1_neg <= not Qint(1);
Q2_neg <= not Qint(2);
U4: MUX4_1 port map ( I0=>'1', I1=>Q0_neg, I2=>Q0_neg, I3=>'0', A1=>Qint(2), A0=>Qint(1), Y=>net(0) );
U5: MUX4_1 port map ( I0=>'1', I1=>'1', I2=>'1', I3=>'0', A1=>Qint(2), A0=>Qint(1), Y=>net(1) );
U6: MUX4_1 port map ( I0=>'1', I1=>Q0_neg, I2=>Q0_neg, I3=>'1', A1=>Qint(2), A0=>Qint(1), Y=>net(2) );

Q <= Qint;

end Behavioral;
```

Valori Automat de Tranziție

Legăm ieșirile la J și K

Implementarea Sursei pentru Automatul JK MUX4:1

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity SimulareAutomatJK is
5  end SimulareAutomatJK;
6
7  architecture Behavioral of SimulareAutomatJK is
8
9  component AutomatJK is
10     Port ( R : in std_logic;
11            CLK : in std_logic;
12            Q : out std_logic_vector(2 downto 0));
13 end component AutomatJK;
14
15 signal R,CLK: std_logic;
16 signal Q: std_logic_vector(2 downto 0);
17
18 begin
19 UTT: AutomatJK port map ( R=>R, CLK=>CLK, Q=>Q );
20 R<='1' after 0ns, '0' after 20ns;
21 process
22 begin
23 CLK<='0'; wait for 5ns;
24 CLK<='1'; wait for 5ns;
25 end process;
26
27 end Behavioral;
```

Perioada de funcționare a Reset-ului

Perioada de funcționare Clock