UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Facultatea de Electronică,
Telecomunicații și
Tehnologia Informației

Proiect CID 2022-2023

Student: Roșca David-Sorin

Grupa: E.2121

Semigrupa: 2.

Alocare Proiect: 23-G-III

Data: 17.12.2022

23-G-III-Bistabil JK

23.

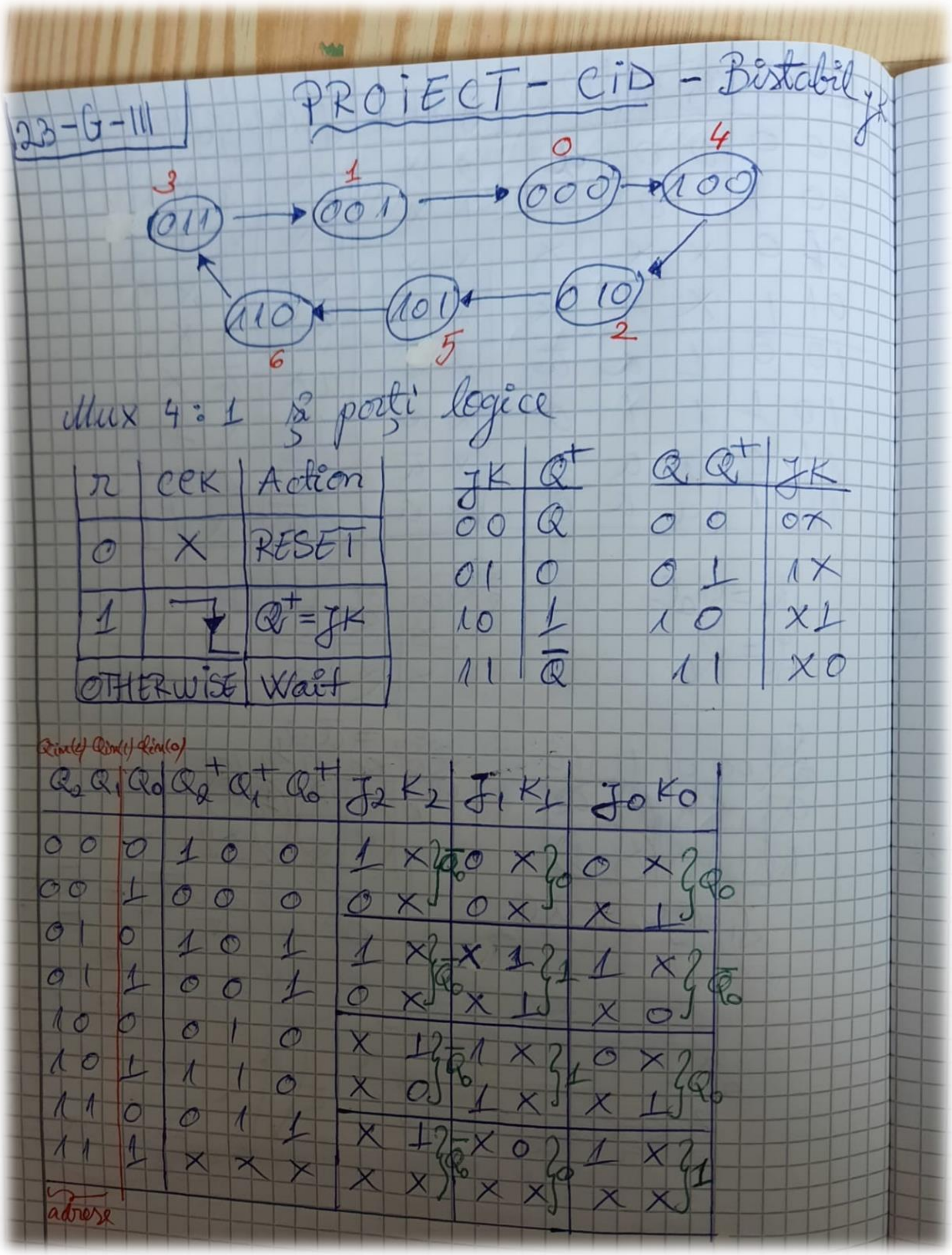
$$\begin{tikzcd}
011 \arrow[r] & 001 \arrow[r] & 000 \arrow[r] & 100 \arrow[d] \\
110 \arrow[u] & 101 \arrow[l] & 010 \arrow[l] &
\end{tikzcd}$$

G.

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

III. MUX 4:1 și porți logice

✓ *Pasul 1:* Am rezolvat pe foaie tema de proiect, m-am folosit de schema de tranziție si de mux 4:1, porți logice.



123-G-III

PROIECT - CiD - Bistabil JK

$$\begin{tikzcd}
\overset{3}{011} \arrow[r] & \overset{1}{001} \arrow[r] & \overset{0}{000} \arrow[r] & \overset{4}{100} \arrow[d] \\
\underset{6}{110} \arrow[u] & \underset{5}{101} \arrow[l] & & \underset{2}{010} \arrow[ll]
\end{tikzcd}$$

Mux 4:1 și porți logice

R	CLK	Action
0	X	RESET
1	↓	$Q^+ = JK$
OTHERWISE		Wait

JK	Q^+
00	Q
01	0
10	1
11	$\overline{Q}$

Q	Q^+	JK
0	0	0X
0	1	1X
1	0	X1
1	1	X0

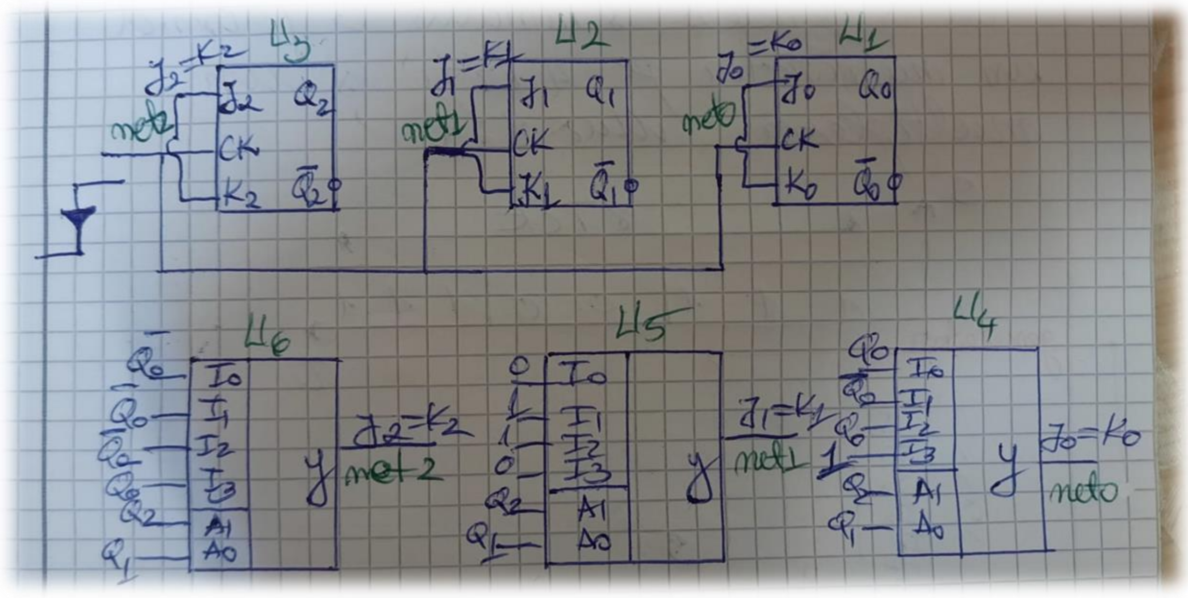
Qin(2) Qin(1) Qin(0)

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	J_2	K_2		J_1	K_1		J_0	K_0	
0	0	0	1	0	0	1	X	$\overline{Q_0}$	0	X	0	0	X	Q_0
0	0	1	0	0	0	0	X		0	X		X	1	
0	1	0	1	0	1	1	X	$\overline{Q_0}$	X	1	1	1	X	$\overline{Q_0}$
0	1	1	0	0	1	0	X		X	1		X	0	
1	0	0	0	1	0	X	1	$\overline{Q_0}$	1	X	1	0	X	Q_0
1	0	1	1	1	0	X	0		1	X		X	1	
1	1	0	0	1	1	X	1	$\overline{Q_0}$	X	0	0	1	X	1
1	1	1	X	X	X	X	X		X	X		X	X	

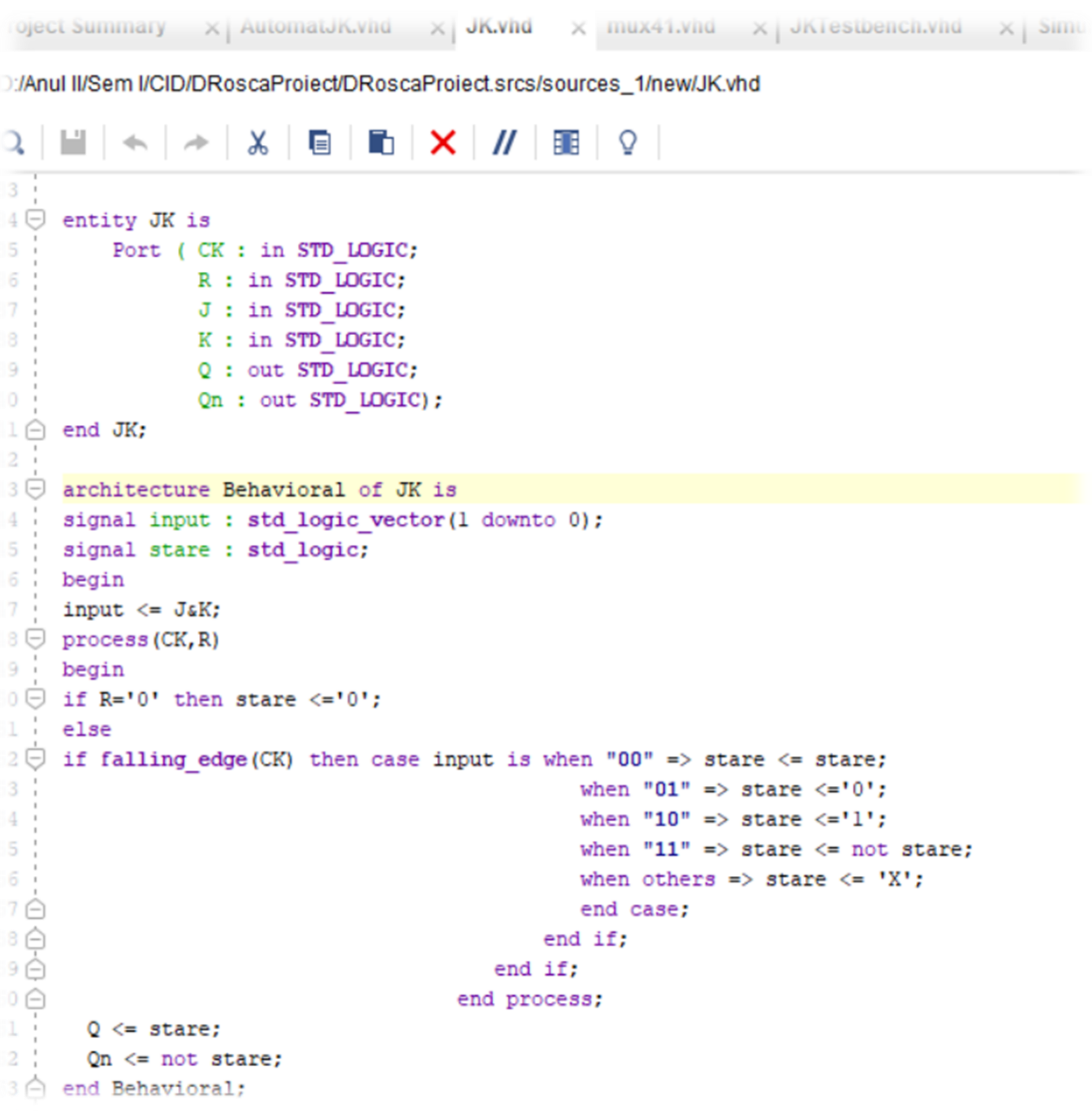
adrese

✓

✓ *Pasul 2:* Cu ajutorul tabelului de adevăr de mai sus am implementat un automat compus din: trei bistabile JK și trei multiplexoare 4:1. Deoarece bistabilul JK este foarte mic, nu am obținut nicio poartă logică. Totodată, putem observa, că primul multiplexor 4:1 se poate scrie mai simplu: qnegat=J2=K2.



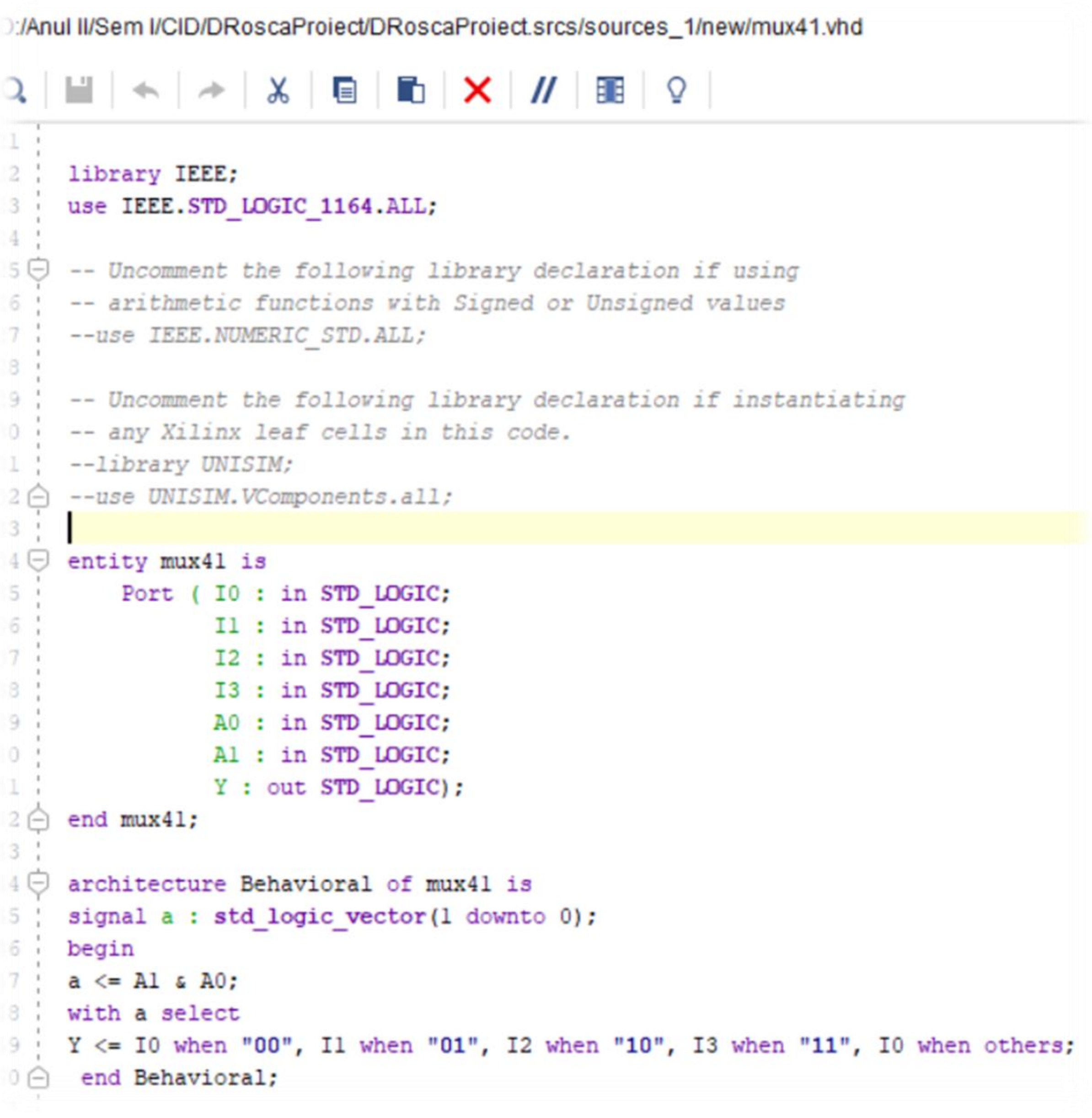
✓ *Pasul 3:* După ce am rezolvat pe foaie cerința de proiect am trecut să implementez în progamul vivado bistabilul JK. Aici am folosit tabelul dat, observăm că bistabilul este pe un front descendent(falling_edge), iar resetul este activ pe zero în tabelul dat.



:/Anul II/Sem I/CID/DRoscaProiect/DRoscaProiect.srcs/sources_1/new/JK.vhd

```
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='0' 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;
```

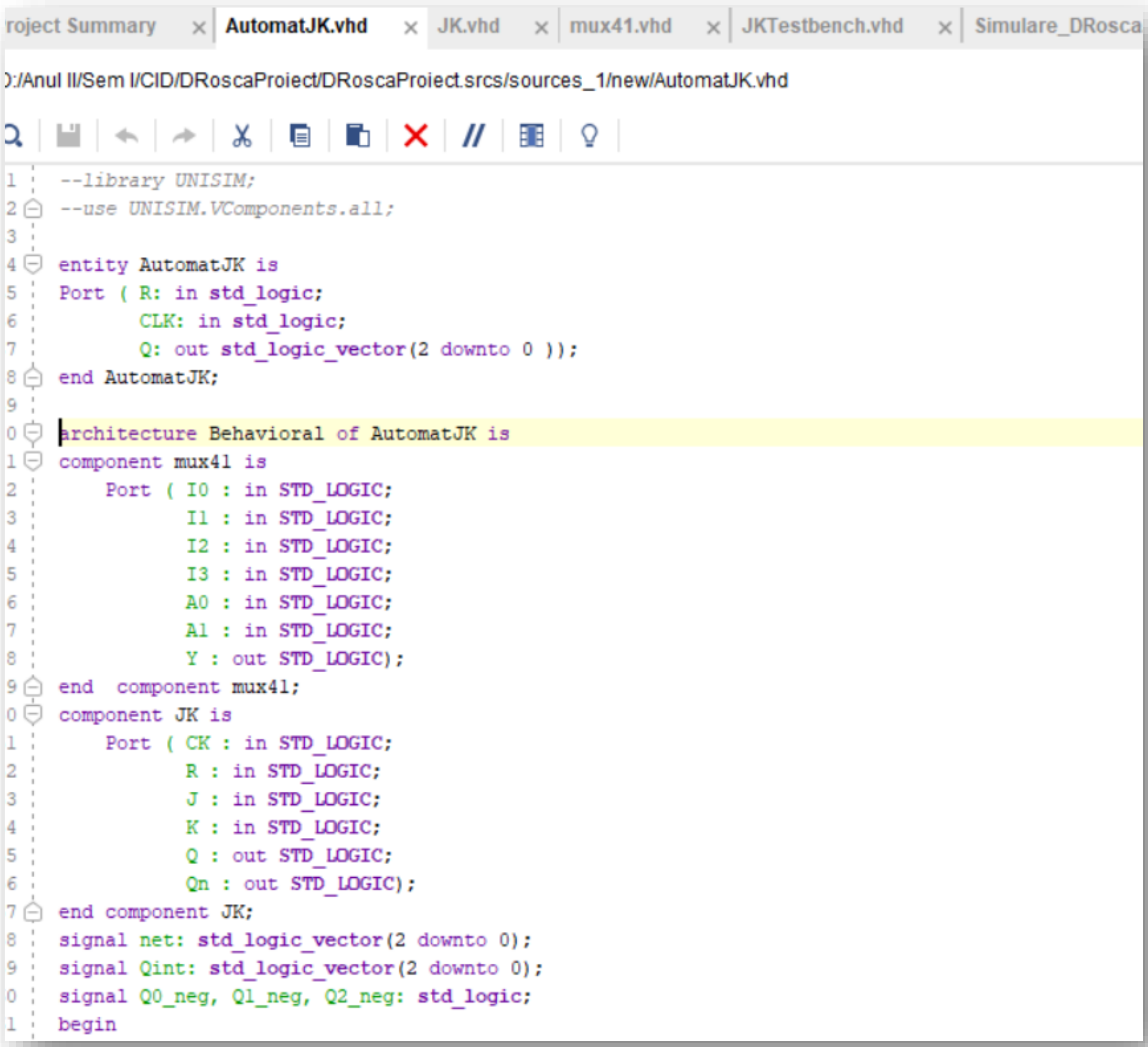
✓ *Pasul 6:* Am implementat codul pentru MUX 4:1.



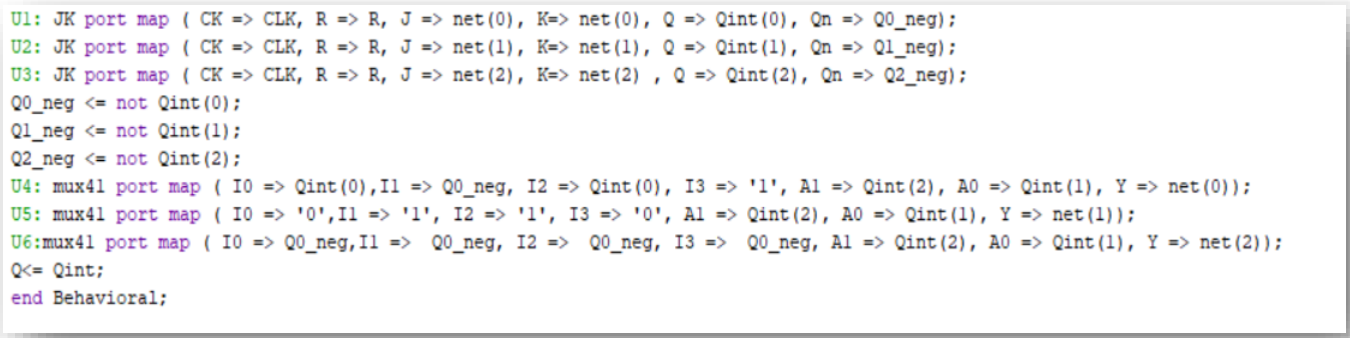
:/Anul II/Sem I/CID/DRoscaProiect/DRoscaProiect.srcs/sources_1/new/mux41.vhd

```vhdl

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mux41 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 mux41;

architecture Behavioral of mux41 is
signal a : std_logic_vector(1 downto 0);
begin
a <= A1 & A0;
with a select
Y <= I0 when "00", I1 when "01", I2 when "10", I3 when "11", I0 when others;
 end Behavioral;
```

✓ *Pasul 7:* Am construit automatul JK cu cele 3 multiplexoare așa cum reiese din schema făcută mai sus pe hârtie.


roject Summary
AutomatJK.vhd
JK.vhd
mux41.vhd
JKTestbench.vhd
Simulare_DRosca
):/Anul II/Sem I/CID/DRoscaProiect/DRoscaProiect.srcs/sources_1/new/AutomatJK.vhd

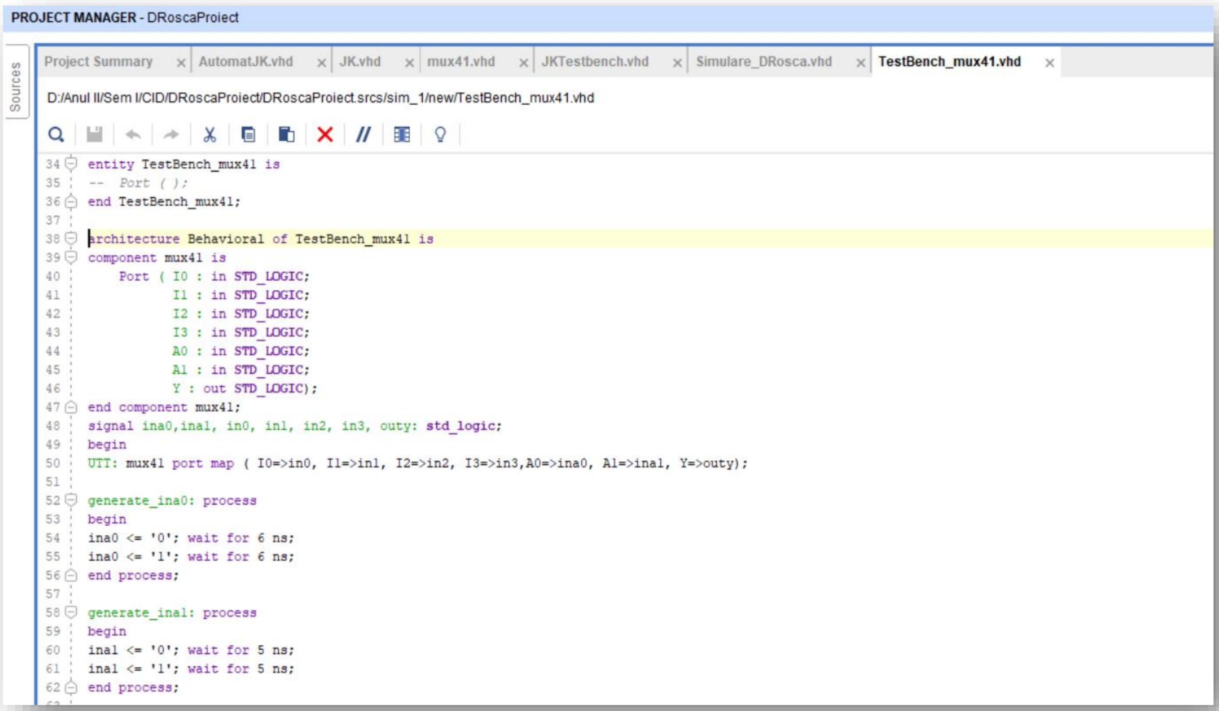
```vhdl
--library UNISIM;
--use UNISIM.VComponents.all;

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 mux41 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 mux41;
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
```



```vhdl
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: mux41 port map ( I0 => Qint(0),I1 => Q0_neg, I2 => Qint(0), I3 => '1', A1 => Qint(2), A0 => Qint(1), Y => net(0));
U5: mux41 port map ( I0 => '0',I1 => '1', I2 => '1', I3 => '0', A1 => Qint(2), A0 => Qint(1), Y => net(1));
U6:mux41 port map ( I0 => Q0_neg,I1 =>  Q0_neg, I2 =>  Q0_neg, I3 =>  Q0_neg, A1 => Qint(2), A0 => Qint(1), Y => net(2));
Q<= Qint;
end Behavioral;
```

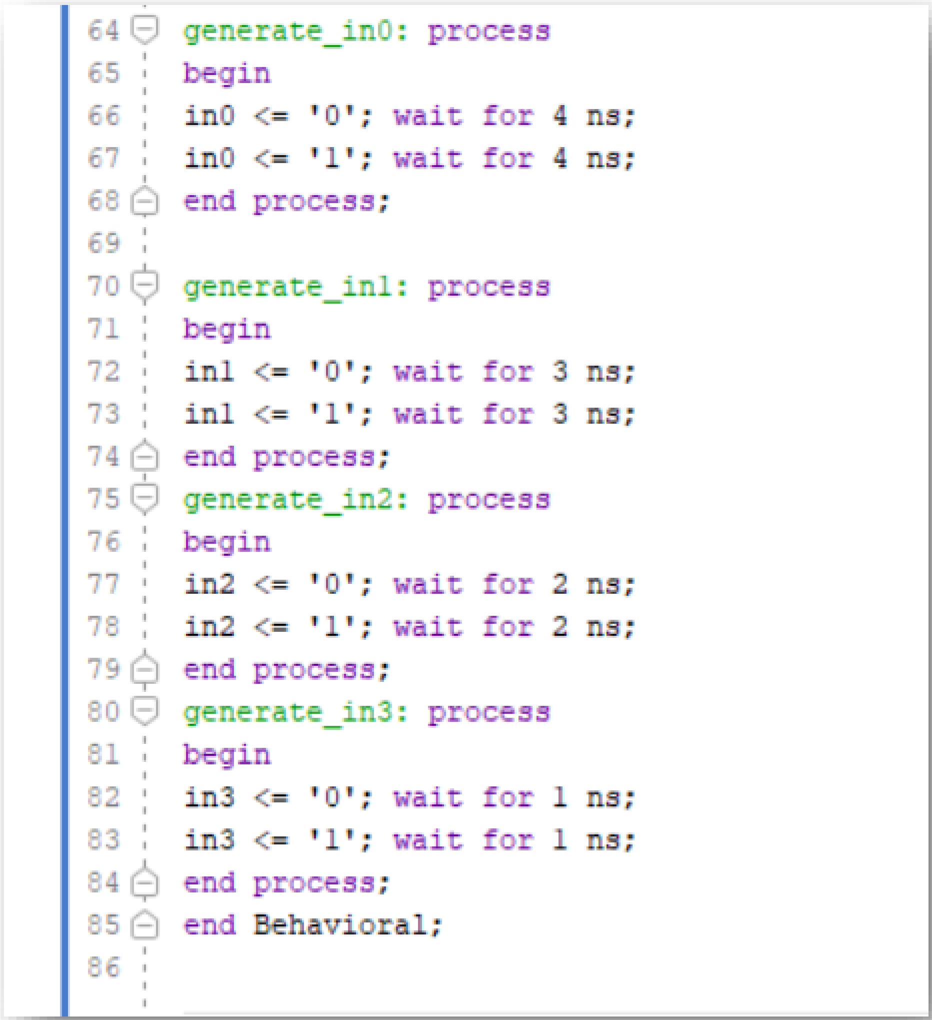
✓ *Pasul 8:* Am făcut o simulare de test pentru MUX 4:1.

Cod Simulare Test MUX:



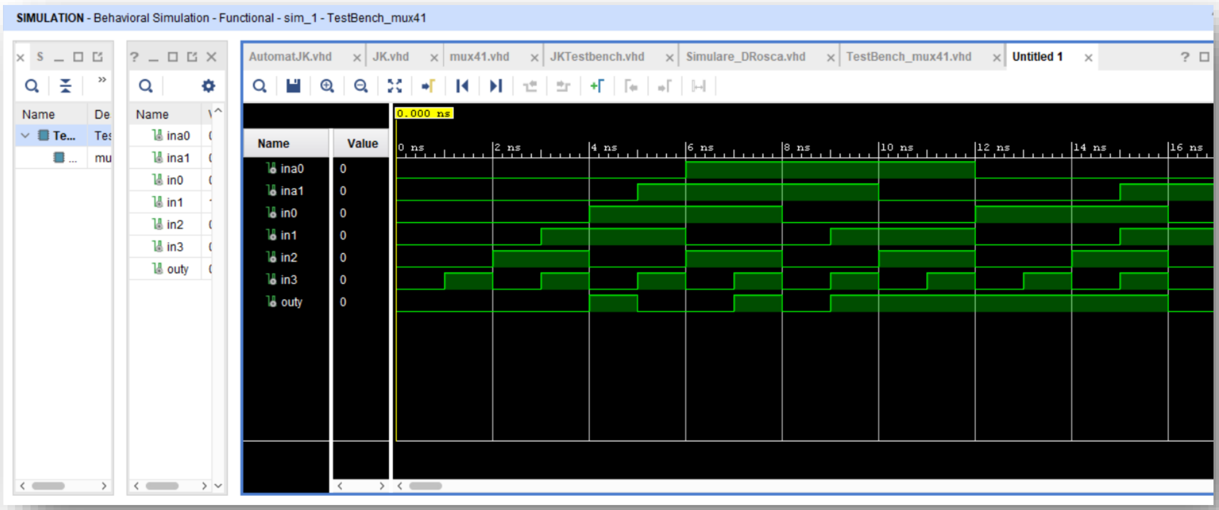
D:/Anul II/Sem I/CID/DRoscaProiect/DRoscaProiect.srcs/sim_1/new/TestBench_mux41.vhd

```
entity TestBench_mux41 is
--  Port ( );
end TestBench_mux41;

architecture Behavioral of TestBench_mux41 is
component mux41 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 mux41;
signal ina0,ina1, in0, in1, in2, in3, outy: std_logic;
begin
UTT: mux41 port map ( I0=>in0, I1=>in1, I2=>in2, I3=>in3,A0=>ina0, A1=>ina1, Y=>outy);

generate_ina0: process
begin
ina0 <= '0'; wait for 6 ns;
ina0 <= '1'; wait for 6 ns;
end process;

generate_ina1: process
begin
ina1 <= '0'; wait for 5 ns;
ina1 <= '1'; wait for 5 ns;
end process;
```



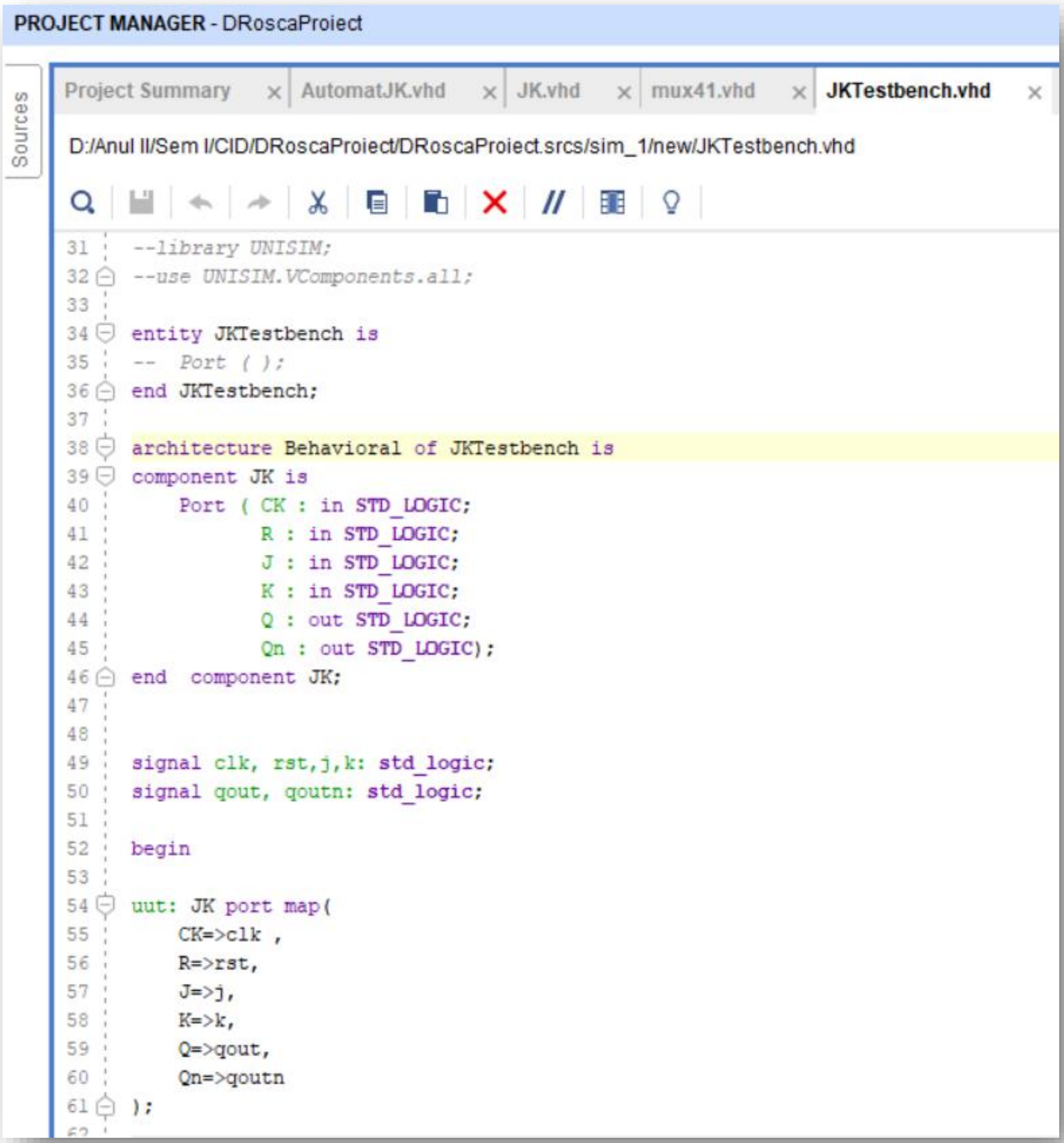
```
generate_in0: process
begin
in0 <= '0'; wait for 4 ns;
in0 <= '1'; wait for 4 ns;
end process;

generate_in1: process
begin
in1 <= '0'; wait for 3 ns;
in1 <= '1'; wait for 3 ns;
end process;
generate_in2: process
begin
in2 <= '0'; wait for 2 ns;
in2 <= '1'; wait for 2 ns;
end process;
generate_in3: process
begin
in3 <= '0'; wait for 1 ns;
in3 <= '1'; wait for 1 ns;
end process;
end Behavioral;
```

Simulare Test MUX 4:1



✓ *Pasul 9:* Am făcut o simulare de test pentru codul JK.

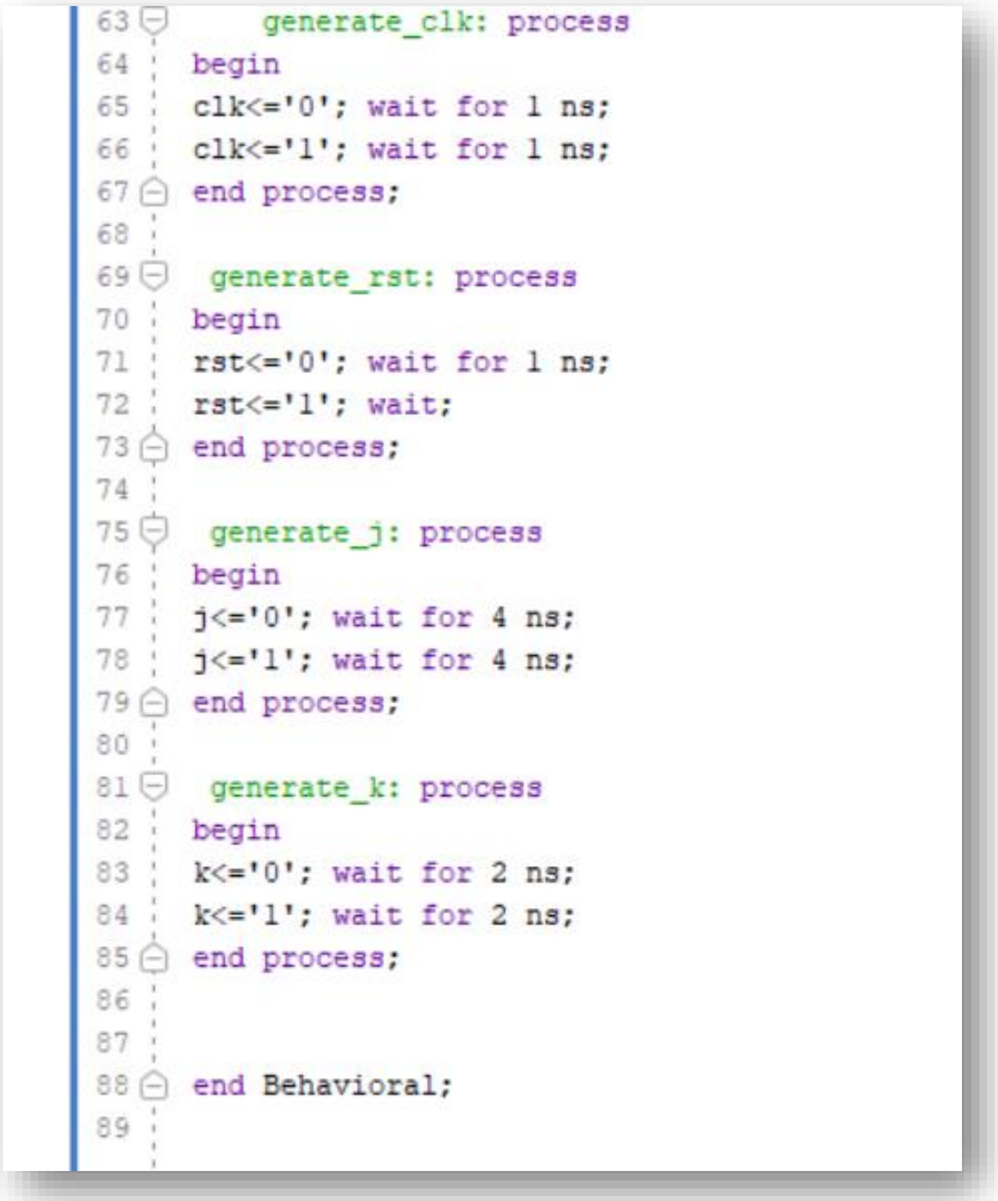
Cod Simulare test JK:



PROJECT MANAGER - DRoscaProiect

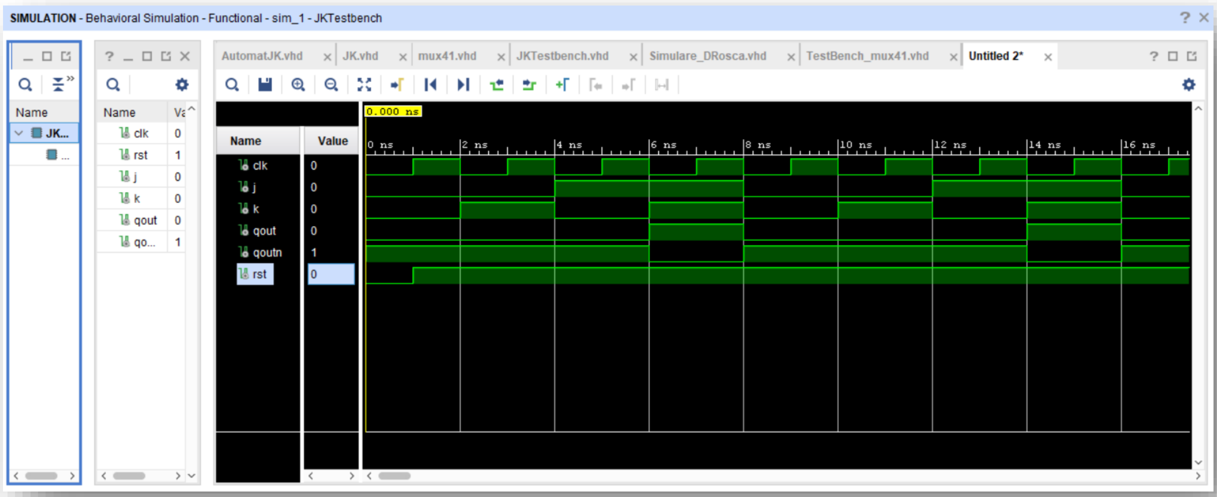
D:/Anul II/Sem I/CID/DRoscaProiect/DRoscaProiect.srcs/sim_1/new/JKTestbench.vhd

```
--library UNISIM;
--use UNISIM.VComponents.all;

entity JKTestbench is
--  Port ( );
end JKTestbench;

architecture Behavioral of JKTestbench 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 1 ns;
clk<='1'; wait for 1 ns;
end process;

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

 generate_j: process
begin
j<='0'; wait for 4 ns;
j<='1'; wait for 4 ns;
end process;

 generate_k: process
begin
k<='0'; wait for 2 ns;
k<='1'; wait for 2 ns;
end process;


end Behavioral;

```

Simulare test JK:


SIMULATION - Behavioral Simulation - Functional - sim_1 - JKTestbench

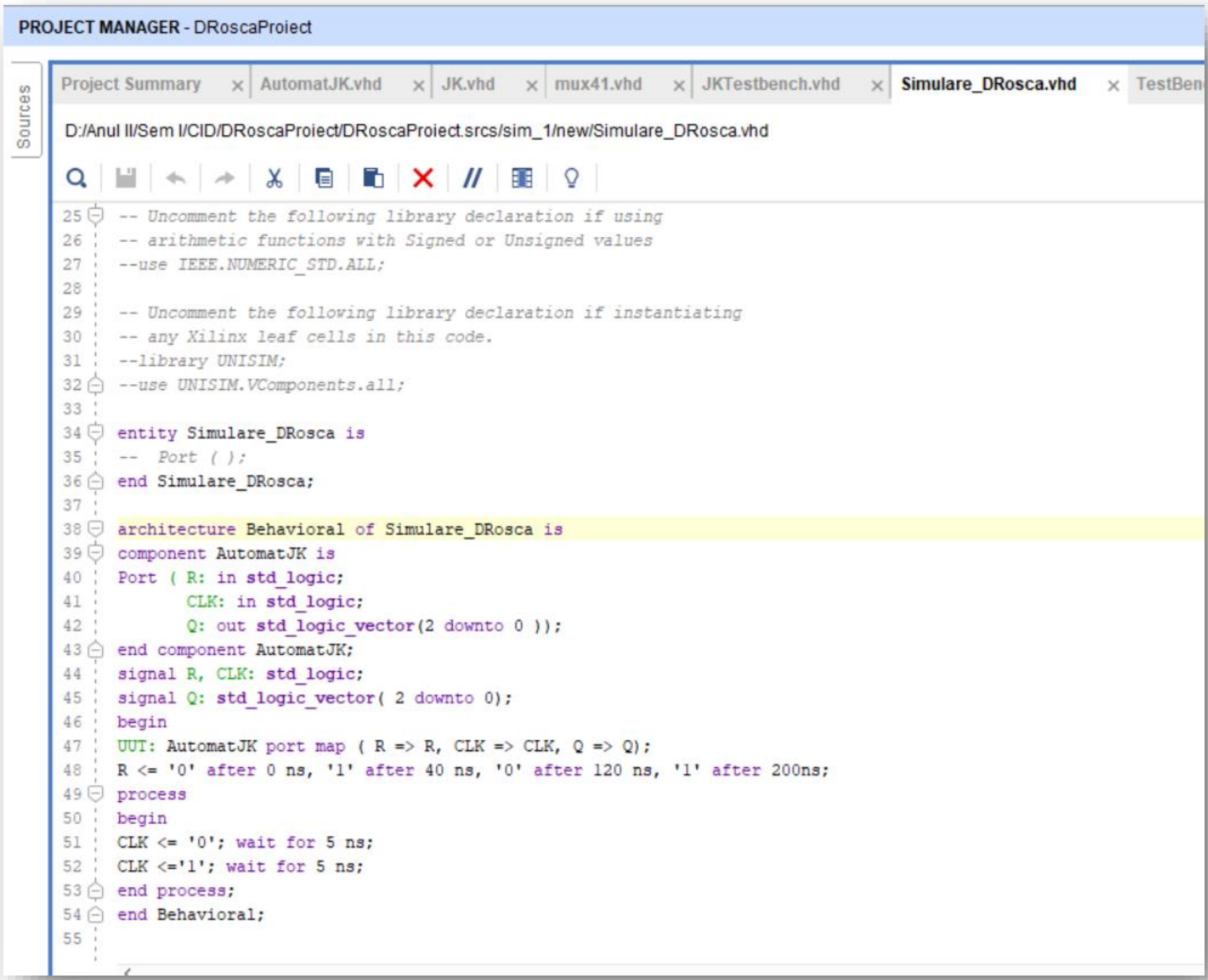
AutomatJK.vhd | JK.vhd | mux41.vhd | JKTestbench.vhd | Simulare_DRosca.vhd | TestBench_mux41.vhd | Untitled 2*

Name	Value
clk	0
rst	1
j	0
k	0
qout	0
qo...	1

Name	Value
clk	0
j	0
k	0
qout	0
qoutn	1
rst	0

✓ *Pasul 10:* Am făcut simularea mare pentru automatul JK(cel din desenul de pe hârtie).

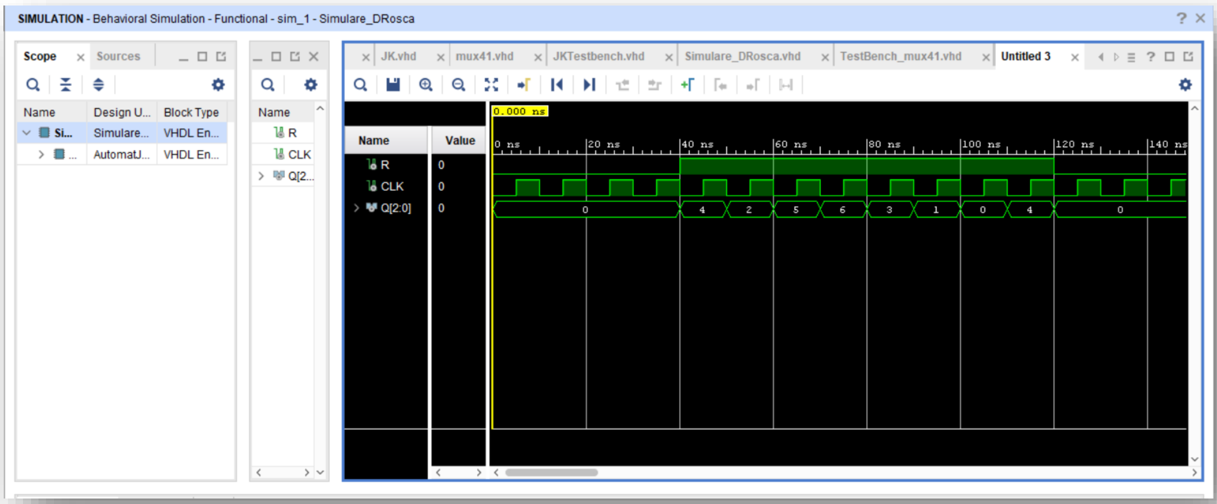
Cod Simulare automat mare:



D:/Anul II/Sem I/CID/DRoscaProiect/DRoscaProiect.srcs/sim_1/new/Simulare_DRosca.vhd

```
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Simulare_DRosca is
--  Port ( );
end Simulare_DRosca;

architecture Behavioral of Simulare_DRosca is
component AutomatJK is
Port ( R: in std_logic;
       CLK: in std_logic;
       Q: out std_logic_vector(2 downto 0 ));
end component AutomatJK;
signal R, CLK: std_logic;
signal Q: std_logic_vector( 2 downto 0);
begin
UUT: AutomatJK port map ( R => R, CLK => CLK, Q => Q);
R <= '0' after 0 ns, '1' after 40 ns, '0' after 120 ns, '1' after 200ns;
process
begin
CLK <= '0'; wait for 5 ns;
CLK <='1'; wait for 5 ns;
end process;
end Behavioral;
```

Simulare automat:


SIMULATION - Behavioral Simulation - Functional - sim_1 - Simulare_DRosca
Scope
Sources
Name
Design U...
Block Type
Si...
Simulare...
VHDL En...
AutomatU...
VHDL En...
R
CLK
Q[2...
JK.vhd
mux41.vhd
JKTestbench.vhd
Simulare_DRosca.vhd
TestBench_mux41.vhd
Untitled 3
0.000 ns
Value
Q[2:0]
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
0
4
2
5
6
3
1
0
4
0