УНИВЕРЗИТЕТ
У НОВОМ САДУ

ФАКУЛТЕТ
ТЕХНИЧКИХ НАУКА

Трг Доситеја Обрадовића 6, 21000 Нови Сад, Југославија
Деканат: 021 350-413; 021 450-810; Централа: 021 350-122
Рачуноводство: 021 58-220; Студентска служба: 021 350-763
Телефакс: 021 58-133; e-mail: ftndean@uns.ns.ac.yu

PROJEKAT

iz Računarskog projektovanja digitalnih integrisanih kola

TEMA PROJEKTA:

Multiplekser 32/1 realizovan pomoću multipleksera 8/1, koji je realizovan pomoću multipleksera 4/1 i dodatnih logičkih kola.

TEKST PROJEKTA:

Prikazati funkcionalne tabele multipleksera 4/1, 8/1 i 32/1 i objasniti njihov rad.

Nacrtati kako se realizuje multiplekser 8/1 korišćenjem dva multipleksera 4/1 i potrebnih logičkih kola. Voditi računa da realizovani muliplekser treba da ima svoj signal dozvole, tj. EN signal. Nakon toga nacrtati kako se realizuje multiplekser 32/1 korišćenjem isključivo multipleksera 8/1. Voditi računa da realizovani muliplekser treba da ima svoj signal dozvole, tj. EN signal.

U programskom jeziku VHDL napisati kod kojim se realizuje multiplekser 4/1. (**mux4to1.vhd**)

Korišćenjem multipleksera 4/1 napisati kod za multiplekser 8/1. (**mux8to1.vhd**)

Korišćenjem multipleksera 8/1 napisati kod za multiplekser 32/1. (**mux32to1.vhd**)

Napisati kod za testiranje ispravnosti rada multipleksera 32/1. (**mux32to1_tb.vhd**)

Testirati i pokazati ispravnost rada multipleksera 32/1.

Korišćenjem alata *Genus Synthesis Solution* programskog paketa *Cadence* projektovati (sintetizovati) šematik multipleksera 32/1u 0.35 µm AMS (C35B4) tehnologiji.

Pomoću alata *Innovus Implementation System* programskog paketa *Cadence* projektovati (generisati) fizičku realizaciju (lejaut) multipleksera 32/1 u 0.35 µm AMS (C35B4) tehnologiji.

Mentor:
Kristina Nikolić

Student:
Ognjen Višnjić, EE 217/2020

U Novom Sadu, 23.01.2024.

1. Teorijska analiza

Multiplekser je digitalno kolo koje ima više ulaza, ali omogućava prosleđivanje samo jednog od njih na izlazu u zavisnosti od kontrolnog signala. Princip rada multiplekserskog kola se bazira na logičkoj funkciji koja se primenjuje na ulazne signale u skladu sa vrednostima kontrolnog signala. U slučaju multipleksera 32/1, imamo 32 ulaza i koristimo 5-bitni kontrolni signal za odabir željenog ulaza.

Ograničenja multiplekserskih kola uključuju:

- Brzina prenosa podataka: brzina prenosa podataka kroz multiplekser zavisi od unapred definisane brzine rada kola i propagacionog kašnjenja signala kroz njega.
- Veličina kontrolnog signala: broj bitova kontrolnog signala određuje broj ulaza koji se mogu odabrati. Povećanje broja ulaza povećava i širinu kontrolnog signala, što može uticati na brzinu prenosa.
- Osetljivost na šum: kola su osetljiva na elektromagnetni šum i interferenciju, što može uzrokovati netačne rezultate ili gubitak podataka.

Uticaj geometrijskih parametara:

U analizi uticaja geometrijskih parametara, možemo uzeti u obzir širinu i dužinu vodova unutar kola, veličinu tranzistora, kapacitete i otpornike. Na primer, ako analiziramo tranzistor, širina (W) i dužina (L) tranzistora direktno utiču na brzinu prenosa podataka, gde je brzina proporcionalna širini tranzistora i obrnuto proporcionalna dužini tranzistora.

Jednačina koja opisuje ovu zavisnost može izgledati ovako:

$$V_{\text{out}} = \frac{W}{L} \times (V_{\text{in}} - V_{\text{th}}). \quad (1)$$

V_{out} - izlazni napon,
V_{in} - ulazni napon,
V_{th} - prag napona tranzistora.

Ovo je samo jedan primer, a konkretni parametri i jednačine zavise od konkretnih karakteristika kola koje analizirate. Ovaj primer je samo ilustracija kako se može analizirati uticaj geometrijskih parametara na rad kola.

Multiplekser 4/1:

- Ulazi**: D_0 , D_1, D_2, D_3** – 4 ulazna signala.
- Selektor: **S_1,S_0** - 2-bitni selektorski signal (Sel).
- Izlaz: **Y** – izabrani izlaz

Tabela 1. Funkcionalna tabela multipleksera 4/1

S_1S_0	Y
00	D_0
01	D_1
10	D_2
11	D_3

Multiplekser 4/1 radi u zavisnosti od vrednosti S_1,S_0 jedan od ulaza D_0, D_1, D_2, D_3 prosleđuje se na izlaz Y.

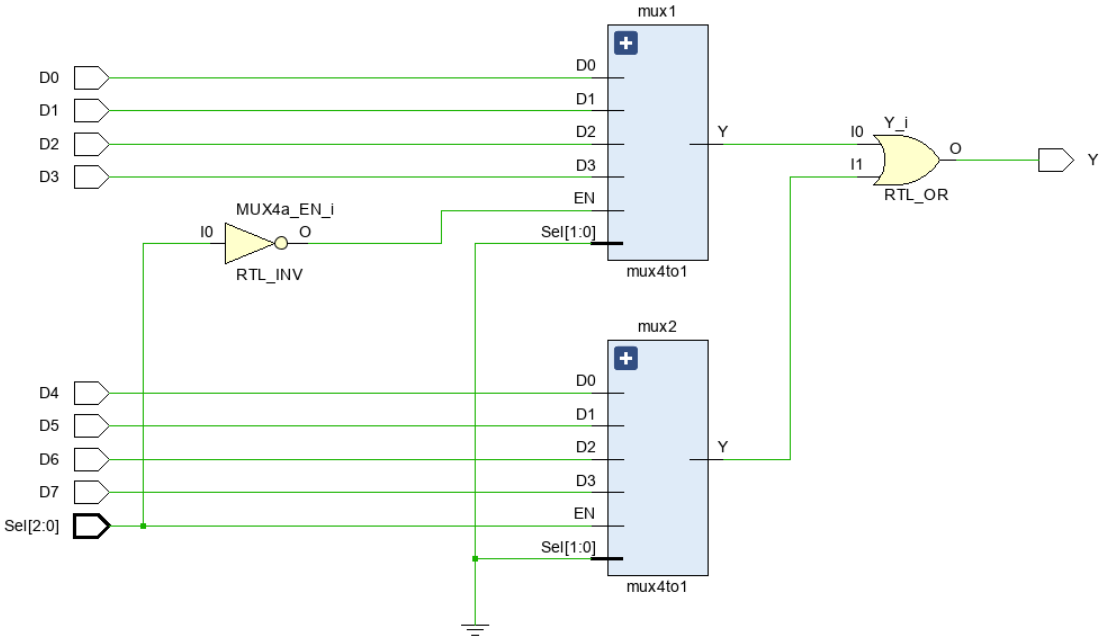
Multiplekser 8/1:

- Ulazi: **D_0**, **D_1**, **D_2**, **D_3**,**D_4**, **D_5**, **D_6**, **D_7** – 8 ulazana signala.
- Selektor: **S_2**,**S_1**,**S_0** - 3-bitni selektorski signal (Sel).
- Izlaz: **Y** – izabrani izlaz.

Tabela 2. Funkcionalna tabela multipleksera 8/1

$S_2S_1S_0$	Y
000	D_0
001	D_1
010	D_2
011	D_3
100	D_4
101	D_5
110	D_6
111	D_7

Multiplekser 8/1 radi u zavisnosti od vrednosti S_2,S_1,S_0 jedan od ulaza D_0, D_1, D_2, D_3,D_4, D_5, D_6, D_7 bira se kao izlaz Y.



Slika 1. Multiplekser 8/1 realizovan sa samo 2 multipleksera 4/1 i potrebnim logickim kolima,

Sel(2) se koristi za kontrolu koji od multipleksera ce da radi i prosleđuje se na signal dozvole, ukoliko je vrednost 0 na mux2 će biti 0 prosleđena na signal dozvole i on nece raditi, a na mux1 ce zbog invertora biti prosleđena 1 i on će raditi, i obratno. Sel(1 do 0) će biti prosleđeni na standardne Sel ulaze multipleksera 4/1 i imaće normalnu funckionalnost.

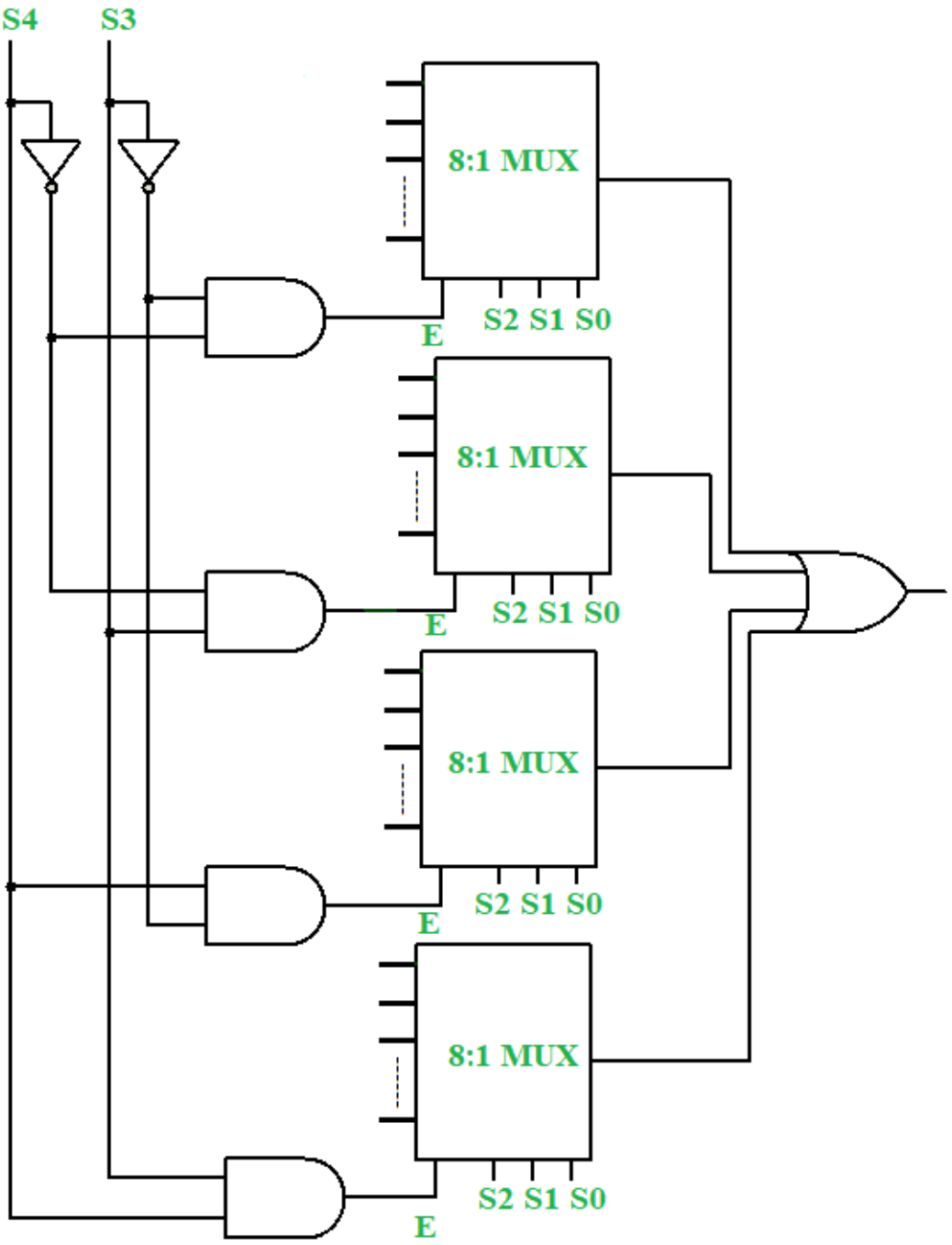
Multiplekser 32/1:

- Ulazi: **D_0**, **D_1**, **D_2**, **D_3,D_4**, **D_5**, **…**, **D_{31}** – 32 ulazna signala.
- Selektor: **S_4,S_3,S_2,S_1,S_0** - 5-bitni selektorski signal (Sel).
- Izlaz: **Y** – izabrani izlaz.

Tabela 3. Funkcionalna tabela multipleksera 32/1

S_4,S_3,S_2,S_1,S_0	Y
00000	D_0
00001	D_1
00010	D_2
00011	D_3
00100	D_4
00101	D_5
00110	D_6
00111	D_7
01000	D_8
01001	D_9
01010	D_{10}
01011	D_{11}
01100	D_{12}
01101	D_{13}
01110	D_{14}
01111	D_{15}
10000	D_{16}
10001	D_{17}
10010	D_{18}
10011	D_{19}
10100	D_{20}
10101	D_{21}
10110	D_{22}
10111	D_{23}
11000	D_{24}
11001	D_{25}
11010	D_{26}
11011	D_{27}
11100	D_{28}
11101	D_{29}
11110	D_{30}
11111	D_{31}

Multiplekser 32/1 radi u zavisnosti od vrednosti S_4,S_3,S_2,S_1,S_0 jedan od ulaza D_0, D_1, D_2, D_3,D_4, D_5, … , D_{31} bira se kao izlaz Y.



Slika 2. Multiplekser 32/1 realizovan sa multiplekserima 8/1

Sel(4) i Sel(3) se koriste za kontrolu koji od multipleksera će da radi i prosleđuje se na signal dozvole kroz I kolo. Sel(2 do 0) će biti prosleđeni na standardne Sel ulaze multipleksera 8/1 i imaće normalnu funckionalnost.

2. Projektovanje digitalnog integrisanog kola

Kod za mux 4/1:

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity mux4to1 is
7       Port (
8           D0, D1, D2, D3 : in STD_LOGIC;
9           Sel : in STD_LOGIC_VECTOR(1 downto 0);
10          EN : in STD_LOGIC;
11          Y : out STD_LOGIC
12      );
13  end mux4to1;
14
15  architecture Behavioral of mux4to1 is
16  begin
17      process(EN, Sel, D0, D1, D2, D3)
18      begin
19          if EN = '1' then
20              case Sel is
21                  when "00" =>
22                      Y <= D0;
23                  when "01" =>
24                      Y <= D1;
25                  when "10" =>
26                      Y <= D2;
27                  when "11" =>
28                      Y <= D3;
29                  when others =>
30                      Y <= '0';
31              end case;
32          else
33              Y <= '0';
34          end if;
35      end process;
36  end Behavioral;
```

Slika 3. VHDL kod 4/1 multipleksera

Ovaj VHDL kod predstavlja 4-na-1 multiplekser (mux4to1). Sledi detaljno objašnjenje koda.

Entitet:

- mux4to1 je naziv entiteta.
- Ulazi (in**): D_0**, D_1**, D_2**, D_3** su četiri ulazna signala, **Sel** je 2-bitni selektroski signal, a **EN** je signal dozvole.
- Izlaz (out): **Y** je izlaz multipleksera.

Arhitektura:

Ovo je arhitektura nazvana "Behavioral" za entitet mux4to1.
Proces ima osetljivost na promene signala EN, Sel, D0, D1, D2 i D3.
Ako je signal dozvole visok ('1'), proces ulazi u case naredbu na osnovu vrednosti selektorskog signala.
U zavisnosti od vrednosti Sel, izlaz Y se postavlja na odgovarajući ulazni signal.
Ako vrednost Sel ne odgovara nijednoj od navedenih vrednosti, postavlja se izlaz Y na '0'.
Ako je signal dozvole nizak ('0'), izlaz Y se postavlja na '0'.
U suštini, ovaj kod implementira 4-na-1 multiplekser gde se izlaz bira na osnovu 2-bitnog selektorskog signala kada je signal dozvole aktivan ('1'). Ako je signal dozvole neaktivan ('0'), izlaz Y se postavlja na '0'.

Kod za mux8/1:

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity mux8to1 is
5       port (
6           D0, D1, D2, D3, D4, D5, D6, D7 : in std_logic;
7           Sel : in std_logic_vector(2 downto 0);
8           En: in std_logic;
9           Y : out std_logic
10      );
11  end mux8to1;
12
13  architecture structural of mux8to1 is
14      -- Component declaration for 4-to-1 multiplexer
15      component mux4to1
16          port (
17              D0, D1, D2, D3 : in std_logic;
18              Sel : in std_logic_vector(1 downto 0);
19              EN : in std_logic;
20              Y : out std_logic
21          );
22      end component;
23
24      -- Signals for intermediate outputs
25      signal MUX4a_out, MUX4b_out : std_logic;
26      signal MUX4a_EN, MUX4b_EN : std_logic;
27      signal zero: std_logic_vector(1 downto 0);
28  begin
29      -- Instantiation of 4-to-1 multiplexers
30      mux1: mux4to1 port map (
31          D0 => D0,
32          D1 => D1,
33          D2 => D2,
34          D3 => D3,
35          Sel => Sel(1 downto 0),
36          EN => MUX4a_EN,
37          Y => MUX4a_out
38      );
39
40      mux2: mux4to1 port map (
41          D0 => D4,
42          D1 => D5,
43          D2 => D6,
44          D3 => D7,
45          Sel => Sel(1 downto 0),
46          EN => MUX4b_EN,
47          Y => MUX4b_out
48      );
49
50      -- Control signal assignment
51      MUX4a_EN <= not Sel(2);
52      MUX4b_EN <= Sel(2);
53
54          Y <= MUX4a_out or MUX4b_out;
55
56
57  end structural;
```

Slika 4. VHDL kod 8/1 multipleksera

Ovaj VHDL kod implementira 8-na-1 multiplekser (mux8to1) pomoću dva 4-na-1 multipleksera (mux4to1). Sledi detaljno objašnjenja koda.

Entitet:

- mux8to1 je naziv entiteta.
- Ulazi (in): **D_0** do **D_7** su osam ulaznih signala, **Sel** je 3-bitni selektorski signal, a **EN** je signal dozvole.
- Izlaz (out): **Y** je izlaz multipleksera.

Arhitektura:

Ovo je arhitektura nazvana "Structural" za entitet mux8to1.

Mux8to1 koristi dva 4-na-1 multipleksera da bi se ostvario 8-na-1 multiplekser.

Prvi 4-na-1 multiplekser (mux1) obrađuje ulazne signale D0 do D3, dok drugi (mux2) obrađuje ulazne signale D4 do D7.

Kontrolni signal Sel(2) se koristi za određivanje koji od ova dva multipleksera će biti aktiviran.

Rezultati izlaza oba multipleksera se kombinuju koristeći logičko "ili" i postavljaju na izlaz Y.

Kontrolni signali MUX4a_EN i MUX4b_EN određuju koji od dva multipleksera će biti aktivan, u zavisnosti od vrednosti Sel(2).

Kada je Sel(2) nizak ('0'), aktiviran je prvi multiplekser, a kada je Sel(2) visok ('1'), aktiviran je drugi multiplekser.

Kod za mux32/1:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mux32to1 is
    Port (
        D0, D1, D2, D3, D4, D5, D6, D7,
        D8, D9, D10, D11, D12, D13, D14, D15,
        D16, D17, D18, D19, D20, D21, D22, D23,
        D24, D25, D26, D27, D28, D29, D30, D31 : in STD_LOGIC;
        Sel : in STD_LOGIC_VECTOR(4 downto 0);
        Y : out STD_LOGIC
    );
end mux32to1;

architecture Behavioral of mux32to1 is

 component mux8to1
        port (
        D0, D1, D2, D3, D4, D5, D6, D7 : in std_logic;
        Sel : in std_logic_vector(2 downto 0);
        En: in std_logic;
        Y : out std_logic
        );
    end component;


    signal MUX8_0_out, MUX8_1_out, MUX8_2_out, MUX8_3_out, sig1,sig2,sig3,sig4 : STD_LOGIC;
begin

    mux1: mux8to1 port map (

        D0 => D0, D1 => D1, D2 => D2, D3 => D3,
        D4 => D4, D5 => D5, D6 => D6, D7 => D7,
        Sel => Sel(2 downto 0),
        EN => sig1,
        Y => MUX8_0_out
        );

   mux2: mux8to1 port map (

        D0 => D8, D1 => D9, D2 => D10, D3 => D11,
        D4 => D12, D5 => D13, D6 => D14, D7 => D15,
        Sel => Sel(2 downto 0),
        EN => sig2,
        Y => MUX8_1_out
       );

    mux3: mux8to1 port map (

        D0 => D16, D1 => D17, D2 => D18, D3 => D19,
        D4 => D20, D5 => D21, D6 => D22, D7 => D23,
        Sel => Sel(2 downto 0),
        EN => sig3,
        Y => MUX8_2_out
       );

    mux4: mux8to1 port map (

        D0 => D24, D1 => D25, D2 => D26, D3 => D27,
        D4 => D28, D5 => D29, D6 => D30, D7 => D31,
        Sel => Sel(2 downto 0),
        EN => sig4,
        Y => MUX8_3_out
       );


      sig1<= (not Sel(3) and not Sel(4));
      sig2<= ( Sel(3) and not Sel(4));
      sig3<= (not Sel(3) and  Sel(4));
      sig4<= ( Sel(3) and  Sel(4));

        Y <= MUX8_0_out or MUX8_1_out or MUX8_2_out or MUX8_3_out;


end Behavioral;
```

Slika 5. VHDL kod 32/1 multipleksera

Ovaj VHDL kod implementira 32-na-1 multiplekser (mux32to1) koristeći četiri 8-na-1 multipleksera (mux8to1). Sledi detaljno objašnjenje koda.

Entitet:

- mux32to1 je naziv entiteta.
- Ulazi (in): **D_0** do **D_{31}** su 32 ulazna signala, **Sel** 5-bitni selektorski signal .
- Izlaz (out): **Y** je izlaz multipleksera.

Arhitektura:

Ovo je arhitektura nazvana "Behavioral" za entitet mux32to1.

U arhitekturi Behavioral koriste se četiri 8-na-1 multipleksera da bi se realizovao 32-na-1 multiplekser.

Svaki od četiri multipleksera procesuira osam od ukupno 32 ulazna signala.

Kontrolni signali (Sig1 do Sig4) određuju koji od četiri multipleksera će biti aktiviran.

Izlazi multipleksera se kombinuju logičkim "ili" operacijama i postavljaju na izlaz Y.

3. Simulaciona provera rezultata

Na slici se nalazi kod iz testbench kola.

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mux32to1_tb is
end mux32to1_tb;

architecture testbench of mux32to1_tb is
    signal D0, D1, D2, D3, D4, D5, D6, D7,
           D8, D9, D10, D11, D12, D13, D14, D15,
           D16, D17, D18, D19, D20, D21, D22, D23,
           D24, D25, D26, D27, D28, D29, D30, D31 : STD_LOGIC;

    signal Sel : STD_LOGIC_VECTOR(4 downto 0);
    signal Y : STD_LOGIC;
    -- Instantiate the mux32to1 component
    component mux32to1
        Port (
            D0, D1, D2, D3, D4, D5, D6, D7,
            D8, D9, D10, D11, D12, D13, D14, D15,
            D16, D17, D18, D19, D20, D21, D22, D23,
            D24, D25, D26, D27, D28, D29, D30, D31 : in STD_LOGIC;
            Sel : in STD_LOGIC_VECTOR(4 downto 0);
            Y : out STD_LOGIC
        );
    end component;
begin
    -- Instantiate the mux32to1 component
    mux32to1_inst : mux32to1
        Port Map (
            D0 => D0, D1 => D1, D2 => D2, D3 => D3,
            D4 => D4, D5 => D5, D6 => D6, D7 => D7,
            D8 => D8, D9 => D9, D10 => D10, D11 => D11,
            D12 => D12, D13 => D13, D14 => D14, D15 => D15,
            D16 => D16, D17 => D17, D18 => D18, D19 => D19,
            D20 => D20, D21 => D21, D22 => D22, D23 => D23,
            D24 => D24, D25 => D25, D26 => D26, D27 => D27,
            D28 => D28, D29 => D29, D30 => D30, D31 => D31,
            Sel => Sel,
            Y => Y
        );
        -- Initialize inputs
        D0 <= '0'; D1 <= '1'; D2 <= '0'; D3 <= '1';
        D4 <= '0'; D5 <= '1'; D6 <= '0'; D7 <= '1';
        D8 <= '0'; D9 <= '1'; D10 <= '0'; D11 <= '1';
        D12 <= '0'; D13 <= '1'; D14 <= '0'; D15 <= '1';
        D16 <= '0'; D17 <= '1'; D18 <= '0'; D19 <= '1';
        D20 <= '0'; D21 <= '1'; D22 <= '0'; D23 <= '1';
        D24 <= '0'; D25 <= '1'; D26 <= '0'; D27 <= '1';
        D28 <= '0'; D29 <= '1'; D30 <= '0'; D31 <= '1';

          Sel<= "00000" after 0 ns,
         "00001" after 100 ns,
        "00010" after 200 ns,
       "00011" after 300 ns,
        "00100" after 400 ns,
        "00101" after 500 ns,
       "00110" after 600 ns,
       "00111" after 700 ns,
       "01000" after 800 ns,
       "01001" after 900 ns,
       "01010" after 1000 ns,
       "01011" after 1100 ns,
       "01100" after 1200 ns,
       "01101" after 1300 ns,
       "01110" after 1400 ns,
       "01111" after 1500 ns,
       "10000" after 1600 ns,
       "10001" after 1700 ns,
       "10010" after 1800 ns,
       "10011" after 1900 ns,
       "10100" after 2000 ns,
        "10101" after 2100 ns,
        "10110" after 2200 ns,
       "10111" after 2300 ns,
       "11000" after 2400 ns,
      "11001" after 2500 ns,
      "11010" after 2600 ns,
      "11011" after 2700 ns,
      "11100" after 2800 ns,
      "11101" after 2900 ns,
     "11110" after 3000 ns,
     "11111" after 3100 ns,
     "00000" after 3200 ns;
end testbench;
```

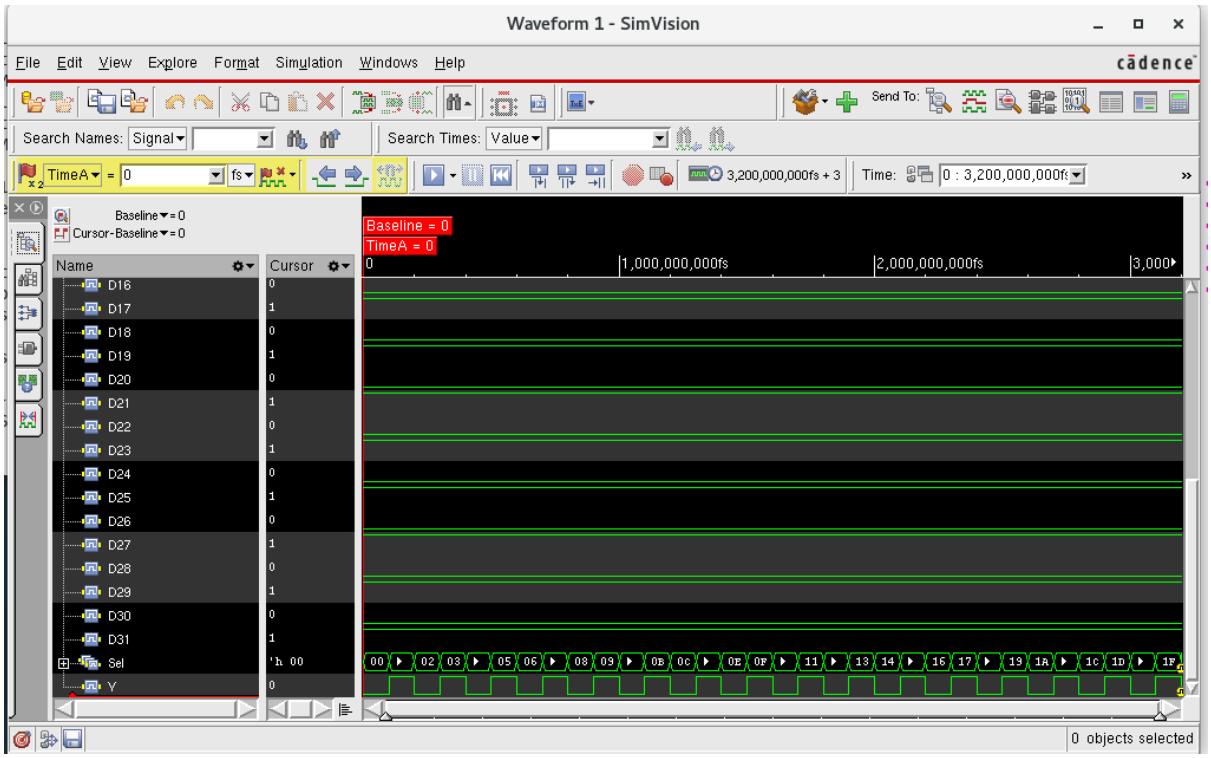
Slika 6. VHDL kod 32/1 multipleksera (testbench)

Ovaj VHDL kod implementira testbench 32-na-1 multipleksera (mux32to1_tb) koristeći Sledi detaljno objašnjenje koda.

Entitet:

- Mux32to1_tb je naziv entiteta.
- Nemamo konkretno ulaze i izlaze nego inicijalizujemo mux32to1.

Arhitektura:
U arhitekturi testbench pravimo test benč entitet za mux32to1.
Koristimo signale D_0 do D_{31} za ulaze i Sel za selektor.
Instanciramo komponentu mux32to1 i mapiramo ulaze i izlaze.
Inicijalizujemo ulazne signale i simuliramo promene selektora tokom vremena.



Slika 7. Rezultati Simulacije

Ovaj VHDL kod simulira 32-na-1 multiplekser (mux32to1) pomoću testbencha. U nastavku imate detaljno objašnjenje rezultata simulacije korak po korak.

- Inicijalizacija (vreme = 0 ns):

Svi ulazni signali (D0 do D31) inicijalizuju se sa naizmeničnim vrednostima '0' i '1'.
Signal za selekciju (Sel) postavljen je na "00000" (binarno) na vremenu 0 ns. Multiplekser je u početnom stanju, a izlaz (Y) zavisi od početne vrednosti signala za selekciju. U ovom slučaju, izlaz je vrednost D0 (koja je '0').

- Korak 1 (vreme = 100 ns):

Sel se ažurira na "00001".
Izlaz (Y) sada zavisi od vrednosti D1 (koja je '1').

- Koraci 2 do 31 (vreme = 200 ns do 3200 ns):

Signal za selekciju (Sel) nastavlja da se menja pri svakom koraku i uveczava za 1.

Pri svakom koraku, izlaz (Y) odgovara vrednosti izabranog ulaza na osnovu binarne vrednosti Sel.

Sekvenca izabranih ulaza ponavlja se svakih 100 ns (od D0 do D31 redom).

- Korak 32:

Signal za selekciju (Sel) ponovo je postavljen na "00000", čime se izlaz (Y) vraća na vrednost D0.

- Završetak (vreme = 3100 ns):

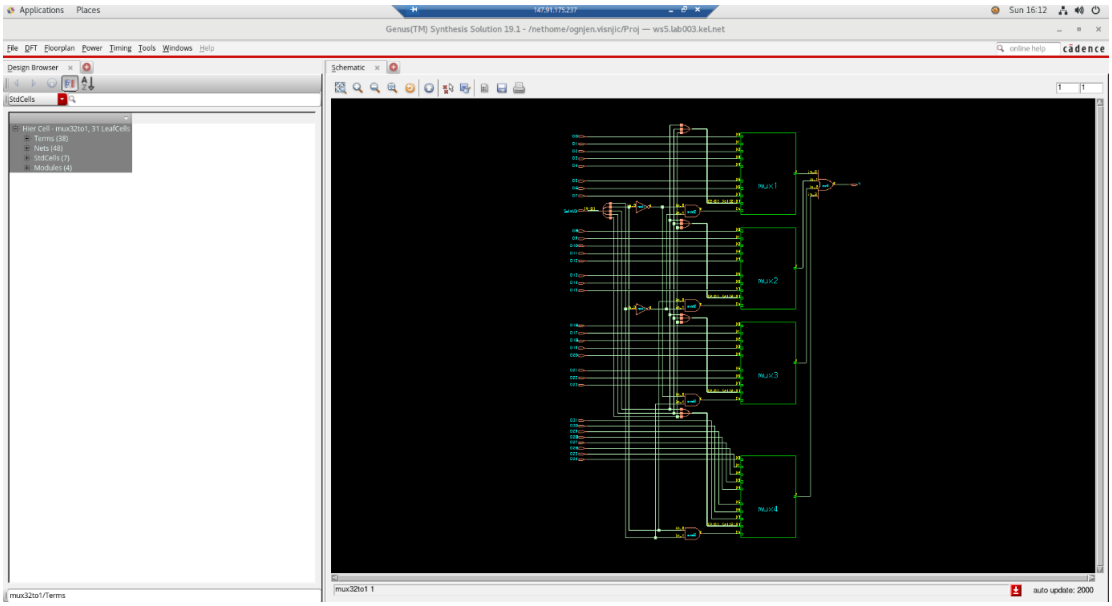
Simulacija se završava nakon 3100 ns.

Ukratko, simulacija prikazuje ponašanje 32-na-1 multipleksera tokom vremena. Signal za selekciju prolazi kroz sve moguće binarne vrednosti od "00000" do "11111", čime se multiplekser podređuje odabranom ulazu pri svakom koraku. Izlaz (Y) odražava vrednost izabranog ulaza. Testbench simulira kompletan ciklus odabira ulaza, vraćajući se na početno stanje pre završetka simulacije.

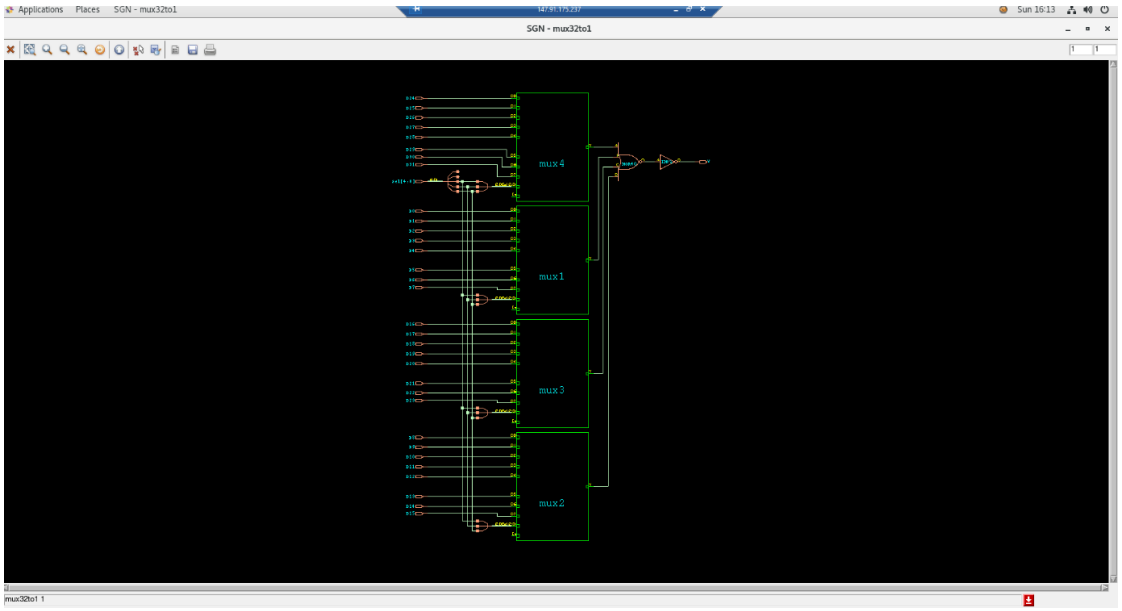
4. Lejaut i šematik projektovanog kola

Na slici 8 nalazi nam se šematik koji smo generisali uz pomoć alata *Genus Synthesis Solution*.

Na slici 9 nalazi nam se šematik nakon sinteze koji smo generisali takođe uz pomoć alata *Genus Synthesis Solution*.

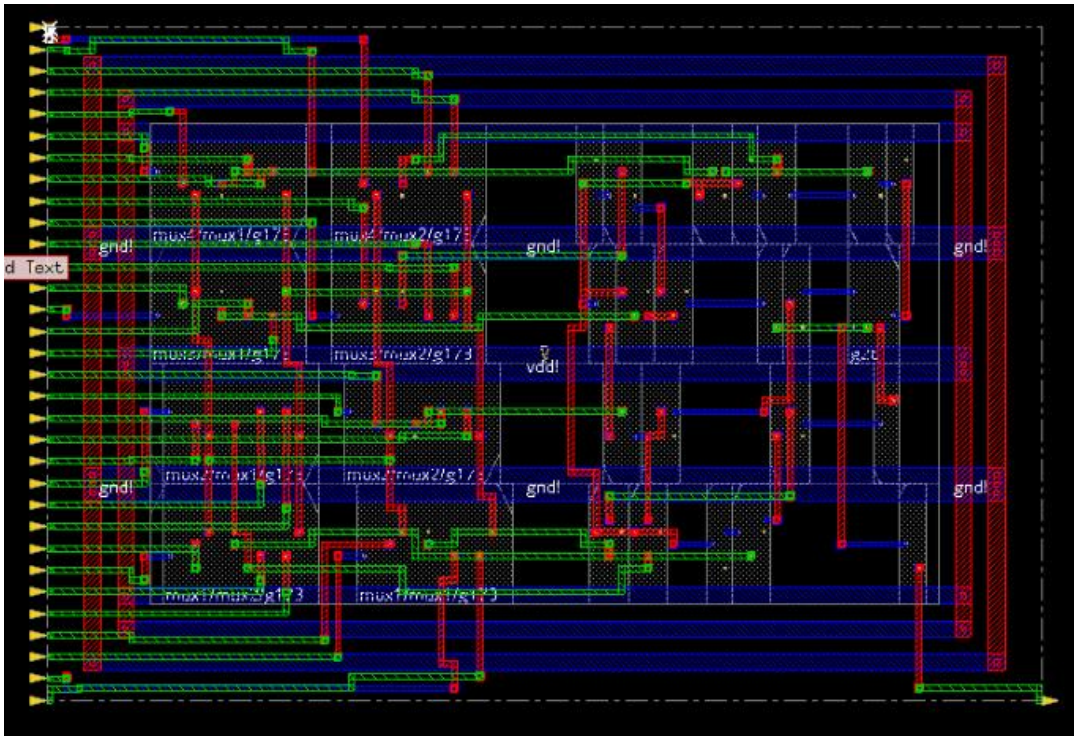


Slika 8. Generisani šematik kola



Slika 9. Generisani šematik kola nakon sinteze

Na slici 10 nalazi nam se lejaut koji smo generisali uz pomoć alata *Innovus Implementation System*.



Slika 10. Generisani lejaut kola

5. Zaključak

U okviru ovog projekta, analizirali smo digitalno kolo multipleksera 32/1, razvili VHDL kod za njegovu implementaciju, napisali test-benč za proveru ispravnosti rada, simulirali ga i donosimo sledeće zaključke.

- Funkcionalnost kola:

Multiplekser 32/1 je digitalno kolo koje omogućava izbor jednog od 32 ulaza na osnovu kontrolnog signala.

Implementiran je korišćenjem pet multipleksera 8/1, što omogućava jednostavnu i modularnu strukturu.

- Ispravnost rada:

Test-benč je uspešno prošao testove sa različitim kombinacijama ulaznih signala i kontrolnih signala.

Multiplekser 32/1 je odgovorio očekivano, izlazni signal se pravilno menja u skladu sa selektovanim ulazom.

- Prednosti:

Modularna struktura kola omogućava lakše održavanje i proširivost.

VHDL kod je pisan na način koji omogućava jasno razumevanje strukture i funkcionalnosti.

- Mane:

Višestruki slojevi multipleksera mogu dovesti do povećane zakašnjenja signala, što može uticati na brzinu rada kola.

Potrebno je pažljivo upravljati veličinom kontrolnog signala kako bi se očuvala brzina prenosa podataka.

- Poboljšanja:

Analiza i optimizacija geometrijskih parametara tranzistora mogla bi poboljšati brzinu prenosa podataka.

Dodatna implementacija optimizacija u VHDL kodu može doprineti efikasnosti kola.

- Zaključak:

Multiplekser 32/1 je efikasno implementiran i testiran kroz simulaciju.

Razumevanje geometrijskih parametara tranzistora i njihov uticaj na performanse kola ključno je za dalju optimizaciju.

Projekat predstavlja dobar temelj za dalje istraživanje i poboljšanja u smislu brzine i efikasnosti.