Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za telekomunikacije

Komutacioni sistemi

4x4 cross-bar multicast komutator sa prioritetnom arbitracijom

Čahtarević Maid (17847)

Kerla Almedina (17805)

Mahovac Nerman (17919)

Šetić Merjema (17986)

Sarajevo, 07.06.2019.g.

Tema 2: 4x4 cross-bar multicast komutator sa prioritetnom arbitracijom

Signali:

- X1 - ulaz, 64-bitna ćelija,
- X2 - ulaz, 64-bitna ćelija,
- X3 - ulaz, 64-bitna ćelija,
- X4 - ulaz, 64-bitna ćelija,
- Y1 - izlaz, 64-bitna ćelija,
- Y2 - izlaz, 64-bitna ćelija.
- Y3 - izlaz, 64-bitna ćelija.
- Y4 - izlaz, 64-bitna ćelija.

Prva četiri bita iz kodne riječi, označena sa ABCD, predstavljaju:

• A - bit kojim se specificira (ako je logička “1”) da je ćeliju potrebno usmjeriti na izlaz Y1,

• B - bit kojim se specificira (ako je logička “1”) da je ćeliju potrebno usmjeriti na izlaz Y2,

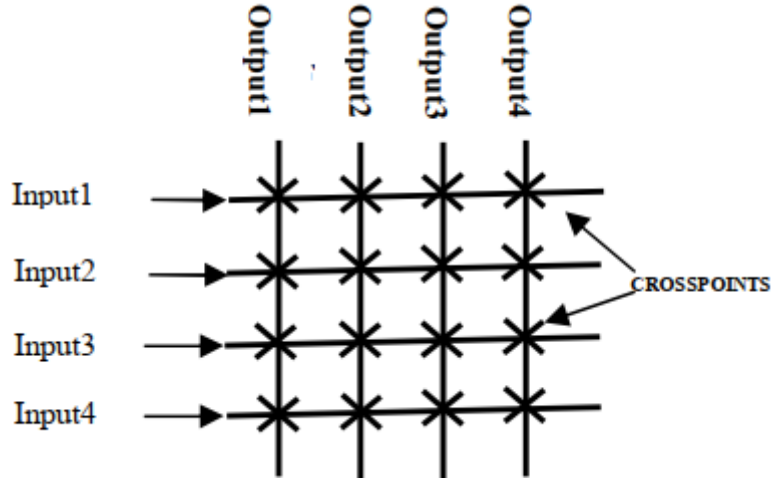
• C - bit kojim se specificira (ako je logička “1”) da je ćeliju potrebno usmjeriti na izlaz Y3,

• D - bit kojim se specificira (ako je logička “1”) da je ćeliju potrebno usmjeriti na izlaz Y4.

Ukoliko je više od jedne ćelije potrebno usmjeriti na jedan izlaz, tada se obavlja usmjeravanje ćelije sa najvećim prioritetom, a ostale se odbacuju. Najveći prioritet ima ćelija koja dolazi sa prvog ulaza, a najmanji prioritet ćelija koja dolazi sa četvrtog ulaza. Potrebno je na konkretnim primjerima analizirati ponašanje komutatora, te simulirati analizirane primjere upotrebom zasebnih VHDL testbench-a. Komutator je potrebno realizirati primjenom tehnika hijerarhijskog modeliranja.

Uvod

Crossbar switch (poznat i kao cross point switch ili matrični switch) je prekidač koji povezuje višestruke ulaze na više izlaza na matrični način. Tačka presijecanja horizontalnih i vertikalnih linija je poznata kao crosspoint. Na svakoj tački presijecanja prekidač kada je zatvoren, povezuje jedan od "M" ulaza sa jednim od "N" izlaza. Kolekcija crossbar-a se može koristiti za implementaciju višeslojnih ili blokirajućih prekidača. Jednostavan model crossbar switch-a je prikazan na slici 1. Svaka horizontalna sabirnica je spojena na ulazni port, a svaka vertikalna sabirnica je spojena na izlazni port. Prema tome, u crossbar switch-u postoji direktan put od svakog ulaza do svakog izlaza.

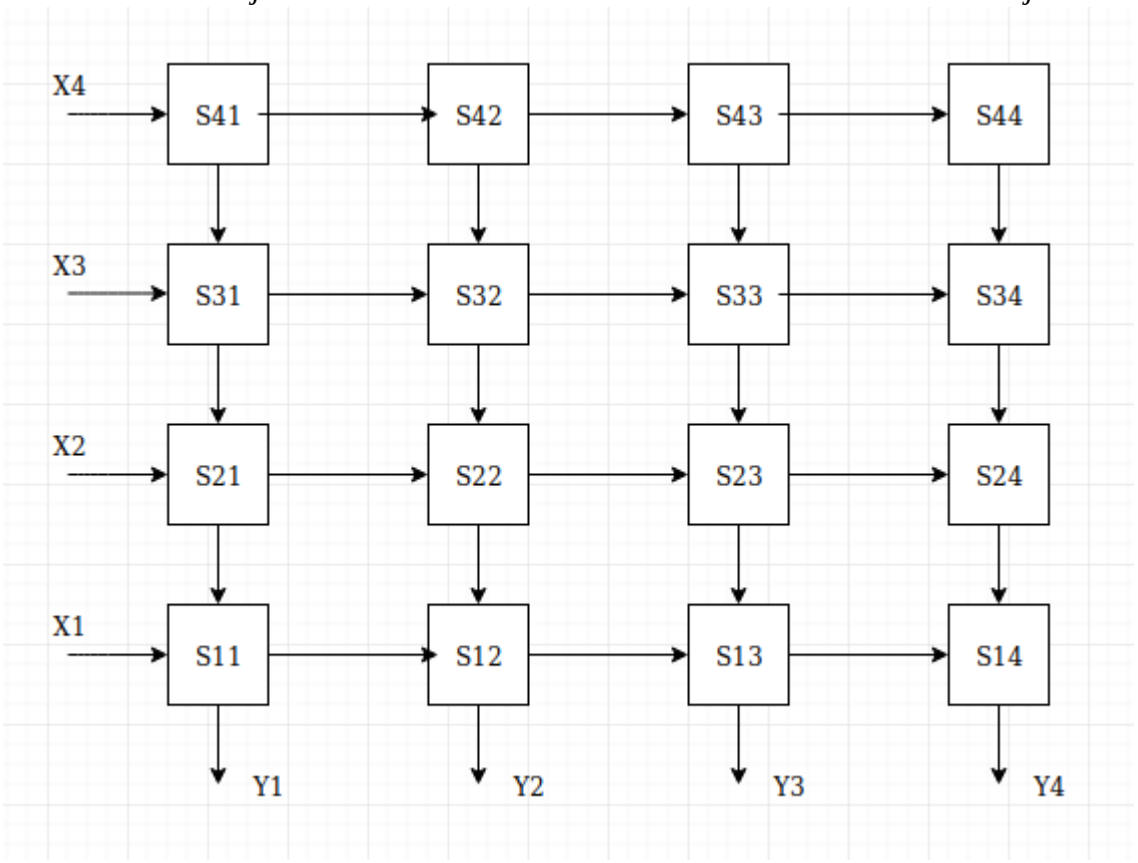


Slika 1. Najjednostavniji model crossbar switch-a

Za izradu našeg projektnog zadatka, koristili smo 4x4 cross-bar multicast komutator. 4x4 cross-bar komutator omogućava da spojimo bilo koji od 4 ulaza, označena sa X1, X2, X3, X4, na bilo koji od 4 izlaza, označena sa Y1, Y2, Y3, Y4.

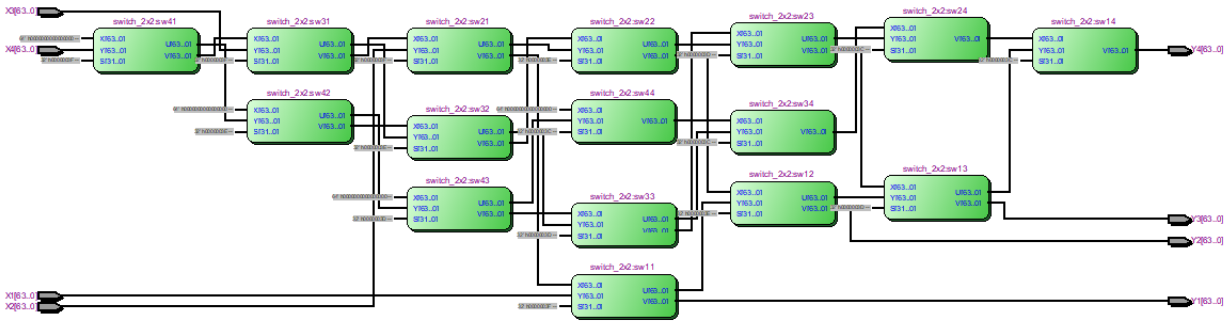
Komutator se može realizovati kao:

1. tri-state buffer
2. multiplekser
3. razne vrste switch-eva



Slika 2. Shematski prikaz 4x4 cross-bar multicast komutatora

U našem projektnom zadatku, 4x4 cross-bar komutator je realiziran pomoću 2x2 multicast switch-eva. Ima ih ukupno 16, te su raspoređeni u četiri stepena gdje svaki stepen sadrži četiri switch-a.



Slika 3. Shema komutatora kreirana u programu Altera Quartus

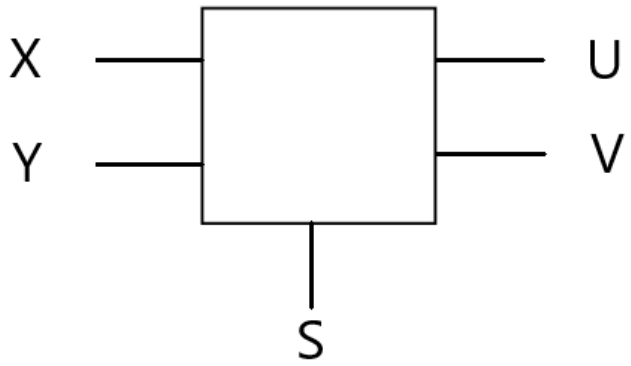
U postavci zadatka je naglašeno da prva četiri bita diktiraju način komutiranja u mreži. Prvi bit svakog od četiri ulaza 64-bitne ćelije je bit koji specificira da li je ćeliju potrebno proslijediti na

neki od izlaza, uz uvažavanje prioriteta. Ulaz sa najvećim prioritetom je X1, dok je onaj sa najmanjim X4. Proslijeđivanjem prvog ulaza po hijerarhiji, svaki sljedeći se odbacuje.

Switch 2x2

Prilikom realizacije projekta koristili smo kopirajući (eng. *Multicast*) komutirajući element 2x2 koji može biti u jednom od sljedećih stanja:

1. bar – gornji ulaz se prosljeđuje na gornji izlaz, a donji ulaz na donji izlaz,
2. copy-up – gornji ulaz se prosljeđuje na oba izlaza,
3. copy-down – donji izlaz se prosljedjuje na oba izlaza,
4. cross – gornji ulaz se prosljedjuje na donji izlaz, a donji ulaz na gornji izlaz.



Slika 4. Switch 2x2

Ulazna varijabla X predstavlja 64-bitnu vrijednost koja se prosljeđuje sa stepena iznad. Druga ulazna varijabla Y predstavlja jednu od ulaznih riječi komutatora (X_1, X_2, X_3, X_4). Ako posmatramo switch prvog stepena, X je varijabla koja je prosljeđena sa drugog stepena dok je Y ulaz X_1. Port V može poprimiti vrijednost X ili Y u ovisnosti od toga koja ulazna varijabla za određeni posmatrani bit A, B, C ili D ima vrijednost 1. Kontrolu koji bit posmatramo vršimo pomoću 6-bitnog selektorskog signala S. S obzirom da su nam od značaja samo prva četiri bita ABCD sa najvećim značajem (MSB) posmatrat ćemo vrijednosti upravljačkog signala u opsegu $S \in [60,\ 61, 62, 63]$. Prvi elementi sva 4 stepena mreže SW11, SW21, SW31 i SW41 će za vrijednost upravljačkog

signala imati vrijednost 63, odnosno u obzir će uzimati samo bit A (bit sa najvećim značajem). Elementi u drugoj koloni mreže SW12, SW22, SW32, SW42 za vrijednost upravljačkog signala uzimaju 62, posmatrajući samo bit B, u elemente treće kolone spadaju switch-evi SW13, SW23, SW33, SW43 sa vrijednošću $S = 61$ (bit C) i elementi posljednje kolone – SW14, SW24, SW34, SW44, odluku o prosljeđivanju donose na osnovu bita D postavljanjem upravljačkog signala na 60. Naravno, bitno je uvažavanje prioriteta prilikom pojave potencijalnog konflikta (X = 1 i Y = 1) tada se na izlaz V uvijek prosljeđuje vrijednost na portu Y. Ukoliko ni X ni Y nemaju vrijednost jedan, u tom slučaju na izlaz izbacujemo stanje visoke impedanse. Na port U se uvijek proslijedi vrijednost na portu Y. Na slici 5. imamo prikaz 4 moguća slučaja. Selektorski signal ćemo smatrati da je 63, odnosno da posmatramo MSB bit.

Prvi karakteristični slučaj – Slika 5.a:

X = 0100000000000000000000000000000000000000000000000000000000000000

Y = 1100000000000000000000000000000000000000000000000000000000000000

S = 63

Drugi karakteristični slučaj – Slika 5.a:

X = 1111000000000000000000000000000000000000000000000000000000000000

Y = 1001000000000000000000000000000000000000000000000000000000000000

S = 63

Treći karakteristični slučaj – Slika 5.b:

X = 1000010101010101010101010101010101010101010101010101010101010101

Y = 0001101010101010101010101010101010101010101010101010101010101010

S = 63

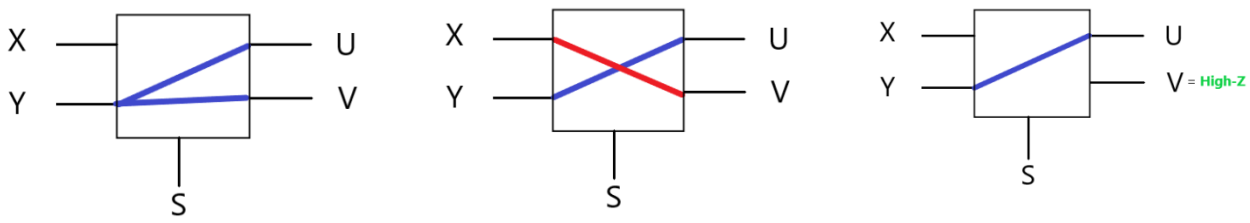
Četvrti karakteristični slučaj – Slika 5.c:

X = 0000111111111111111111111111111111111111111111111111111111111111

Y = 0000000000001000010101111111010100101110110100000001001011111011

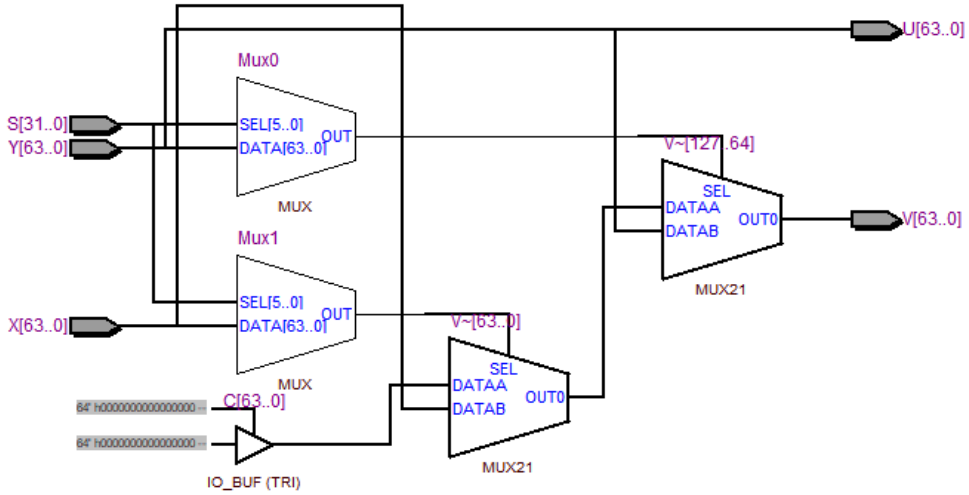
S = 63

V = ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ



a) Y(S) = 1 za svako X(S) *b) Y(S) = 0 I X(S) = 1* *c) Y(S) = 0 I X(S) = 0*

Slika 5. Prikaz komutacije za različite ulazne vrijednosti



Slika 6. Struktura switcha realizovana u programskom paketu Altera Quartus

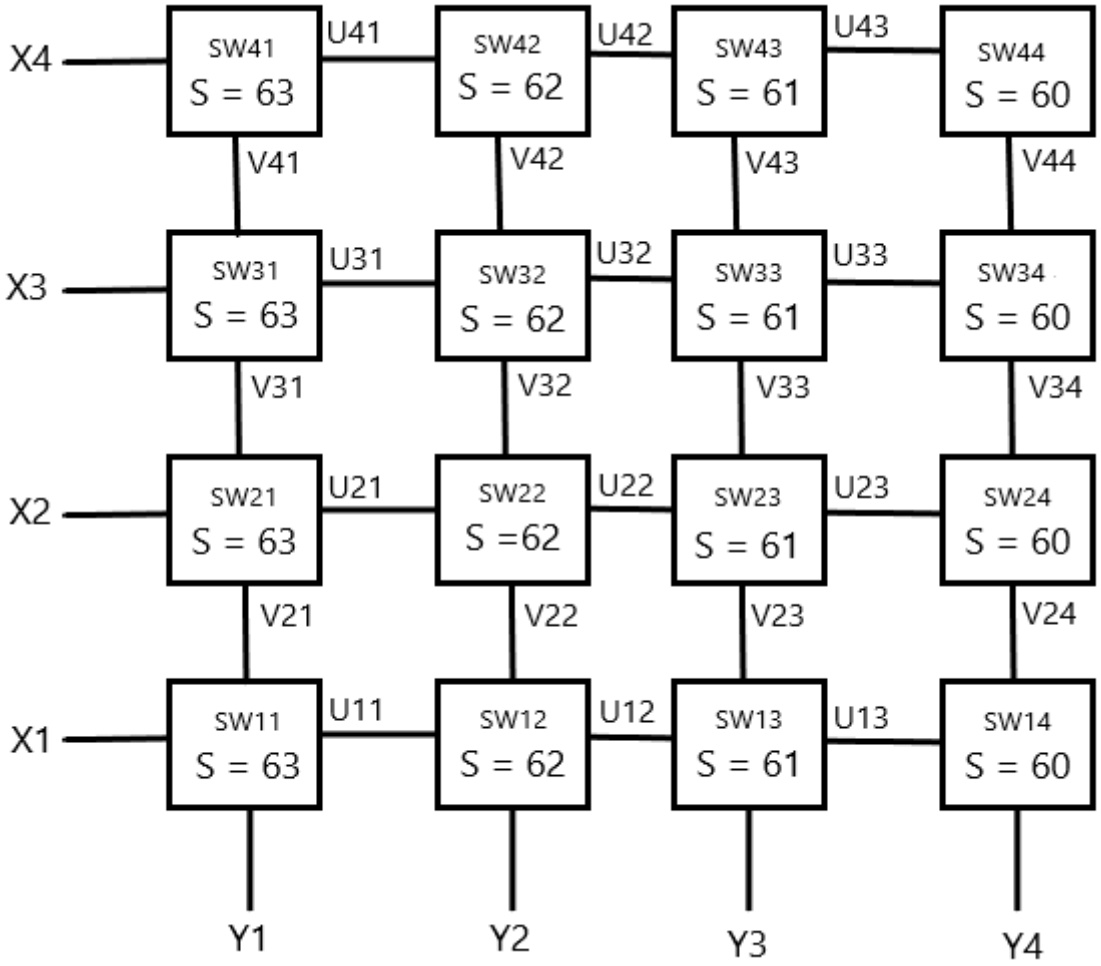
VHDL opis ovog switch-a:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY switch_2x2 IS
        PORT (
                X: IN STD_LOGIC_VECTOR(63 DOWNTO 0);
                Y: IN STD_LOGIC_VECTOR(63 DOWNTO 0);
                S: IN INTEGER;
                U: OUT STD_LOGIC_VECTOR(63 DOWNTO 0);
```

```
                V: OUT STD_LOGIC_VECTOR(63 DOWNTO 0)

        );

END switch_2x2;


ARCHITECTURE arch_switch_2x2 OF switch_2x2 IS

SIGNAL A,B: STD_LOGIC;

SIGNAL C: STD_LOGIC_VECTOR(63 DOWNTO 0);

BEGIN

        A <= Y(S);

        B <= X(S);

        C <= "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ";

        U <= Y;

        V <= Y WHEN (A = '1') ELSE

           X WHEN (B = '1') ELSE

                C;

END ARCHITECTURE;
```

4x4 komutator

U uvodnom dijelu projektne dokumentacije smo okvirno opisali rad komutatora. Na slici 7. imamo prikaz svih signala ovog komutatora. Posmatramo prva četiri bita sa svakog od ulaza, te u skladu sa prethodno definisanim ponašanjem switcha, odgovarajući ulazi će biti prosljeđeni na odgovarajuće izlaze ili će se eventualno na izlazu pojaviti stanje visoke impedanse.



Slika 7. Shematski prikaz 4x4 komutatora sa pomoćnim signalima

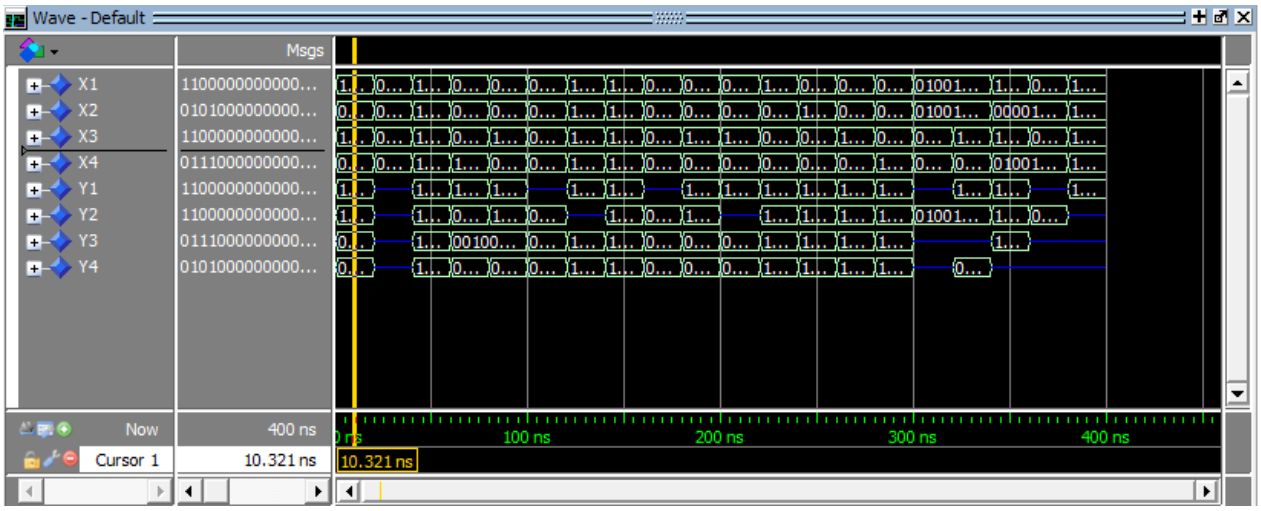
Primjer 1. Ulazi su zadani na sljedeći način:

X1 = 11000000000000000000000000000000000000000000000000000000000000000000

X2 = 01010000000000000000000000000000000000000000000000000000000000000000

X3 = 11000000000000000000000000000000000000000000000000000000000000000000

X4 = 01110000000000000000000000000000000000000000000000000000000000000000



Slika 8. Testbench za primjer 1

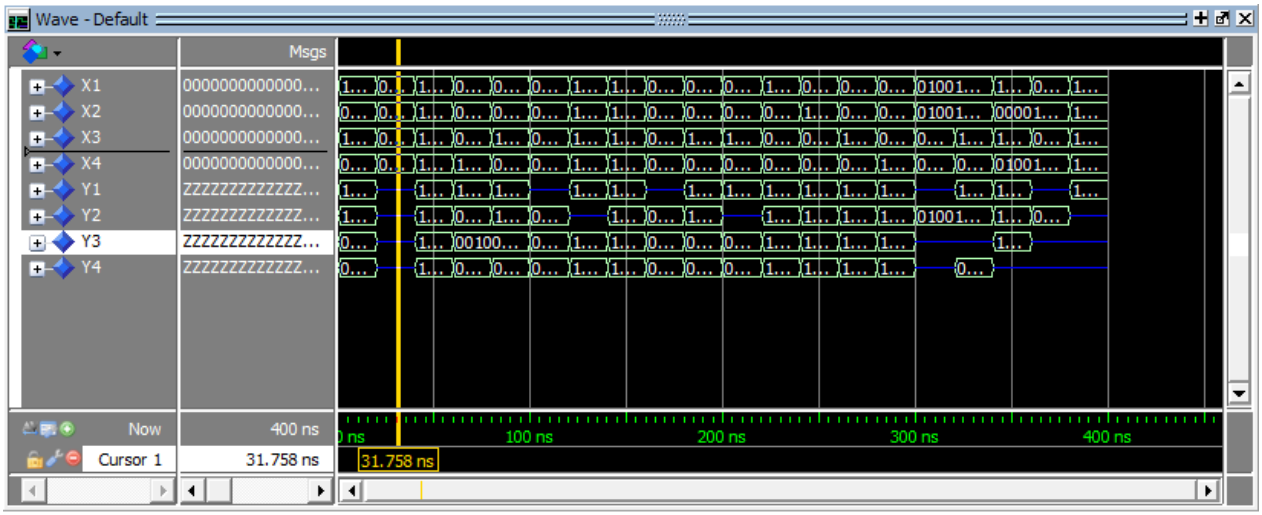
Primjer 2. Stanje visoke impedance

X1 = 00000000000000000000000000000000000000000000000000000000000000000000

X2 = 00000000000000000000000000000000000000000000000000000000000000000000

X3 = 00000000000000000000000000000000000000000000000000000000000000000000

X4 = 00000000000000000000000000000000000000000000000000000000000000000000



Slika 9. Testbench za primjer 2

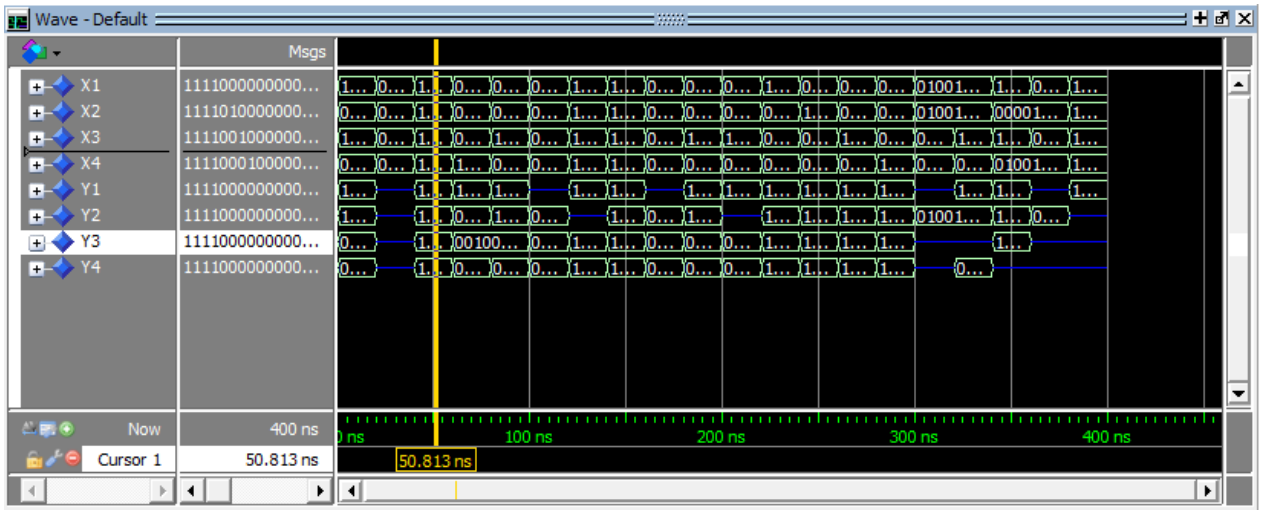
Primjer 3. Provjera prioritetnog ulaza

X1 = 11110000000000000000000000000000000000000000000000000000000000000000

X2 = 11110100000000000000000000000000000000000000000000000000000000000000

X3 = 11110010000000000000000000000000000000000000000000000000000000000000

X4 = 11110001000000000000000000000000000000000000000000000000000000000000



Slika 9. Testbench za primjer 3

Primjer 4. Provjera za ulaz sa najmanjim prioritetom

X1 = 00001111111111111111111111111111111111111111111111111111111111111111

X2 = 00001111111111111111111111111111111111111111111111111111111111111111

X3 = 00001111111111111111111111111111111111111111111111111111111111111111

X4 = 11111111111111111111111111111111111111111111111111111111111111111111

Slika 10. Testbench za primjer 4

Konflikt između toga koji od ulaza se proslijedi na koji izlaz, odnosno problem prioriteta, smo riješili vrlo jednostavno. Naime, prilikom dizajniranja same sheme prioritetni ulazi su postavljeni od dna prema vrhu. Time smo postigli da, prilikom dizajna switch-a, uvijek jedan od ulaza dobija prioritet. Ranijim navođenjem uslova za proslijeđivanje prilikom opisa switch-a u VHDL-u je dati uslov ostvaren.

Kao što je već napomenuto, posljednjih 60 bita ne igra značajnu ulogu u prosljeđivanju u ovom slučaju, stoga ih nećemo uzimati u obzir tokom daljnjeg razmatranja. Ulazi i izlazi su 64-bitne vrijednosti, ali radi lakše prezentacije gledat ćemo samo prva četiri bita ulaza i izlaza.

X_1	X_2	X_3	X_4	A	B	C	D	Y_1	Y_2	Y_3	Y_4
1100	0101	1100	0111	$X_1(63)$	$X_1(62)$	$X_4(61)$	$X_2(60)$	X_1	X_1	X_4	X_2
0000	0000	0000	0000	Z	Z	Z	Z	Z	Z	Z	Z
1111	1111	1111	1111	$X_1(63)$	$X_1(62)$	$X_1(61)$	$X_1(60)$	X_1	X_1	X_1	X_1
0000	0101	0010	1011	$X_4(63)$	$X_2(62)$	$X_3(61)$	$X_2(60)$	X_4	X_2	X_3	X_2
0010	0011	1100	0101	$X_3(63)$	$X_3(62)$	$X_1(61)$	$X_2(60)$	X_3	X_3	X_1	X_2
0100	0101	0110	0111	Z	$X_1(62)$	$X_3(61)$	$X_2(60)$	Z	X_1	X_3	X_2
1000	1001	1010	1011	$X_1(63)$	Z	$X_3(61)$	$X_2(60)$	X_1	Z	X_3	X_2
1100	1101	1110	1111	$X_1(63)$	$X_1(62)$	$X_3(61)$	$X_2(60)$	X_1	X_1	X_3	X_2
0001	0010	0011	0100	Z	$X_4(62)$	$X_2(61)$	$X_1(60)$	Z	X_4	X_2	X_1
0000	0000	1100	0011	$X_3(63)$	$X_3(62)$	$X_4(61)$	$X_4(60)$	X_3	X_3	X_4	X_4
0001	0000	1001	0010	$X_3(63)$	Z	$X_4(61)$	$X_1(60)$	X_3	Z	X_4	X_1
1111	0000	0000	0000	$X_1(63)$	$X_1(62)$	$X_1(61)$	$X_1(60)$	X_1	X_1	X_1	X_1
0000	1111	0000	0000	$X_2(63)$	$X_2(62)$	$X_2(61)$	$X_2(60)$	X_2	X_2	X_2	X_2
0000	0000	1111	0000	$X_3(63)$	$X_3(62)$	$X_3(61)$	$X_3(60)$	X_3	X_3	X_3	X_3
0000	0000	0000	1111	$X_4(63)$	$X_4(62)$	$X_4(61)$	$X_4(60)$	X_4	X_4	X_4	X_4
0100	0100	0100	0100	Z	$X_1(62)$	Z	Z	Z	X_1	Z	Z
0100	0100	1100	0101	$X_3(63)$	$X_1(62)$	Z	$X_4(60)$	X_3	X_1	Z	X_4

1010	**0000**	**1110**	**0100**	$X_1(63)$	$X_3(62)$	$X_1(61)$	Z	**X_1**	**X_3**	**X_1**	**Z**
0000	**0000**	**0000**	**0100**	Z	$X_4(62)$	Z	Z	**Z**	**X_4**	**Z**	**Z**

VHDL opis glavnog programa (4x4 komutator):

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY commutator_4x4 IS
        PORT (
                X1: IN STD_LOGIC_VECTOR(63 DOWNTO 0);
                X2: IN STD_LOGIC_VECTOR(63 DOWNTO 0);
                X3: IN STD_LOGIC_VECTOR(63 DOWNTO 0);
                X4: IN STD_LOGIC_VECTOR(63 DOWNTO 0);
                Y1: OUT STD_LOGIC_VECTOR(63 DOWNTO 0);
                Y2: OUT STD_LOGIC_VECTOR(63 DOWNTO 0);
                Y3: OUT STD_LOGIC_VECTOR(63 DOWNTO 0);
                Y4: OUT STD_LOGIC_VECTOR(63 DOWNTO 0)
        );
END commutator_4x4;

ARCHITECTURE arch_commutator_4x4 OF commutator_4x4 IS
  COMPONENT switch_2x2 IS
    PORT (
      X: IN STD_LOGIC_VECTOR(63 DOWNTO 0);
      Y: IN STD_LOGIC_VECTOR(63 DOWNTO 0);
      S: IN INTEGER;
      U: OUT STD_LOGIC_VECTOR(63 DOWNTO 0);
      V: OUT STD_LOGIC_VECTOR(63 DOWNTO 0)
    );
  END COMPONENT;
        SIGNAL U11, U12, U13: STD_LOGIC_VECTOR(63 DOWNTO 0);
        SIGNAL U21, U22, U23: STD_LOGIC_VECTOR(63 DOWNTO 0);
        SIGNAL U31, U32, U33: STD_LOGIC_VECTOR(63 DOWNTO 0);
        SIGNAL U41, U42, U43: STD_LOGIC_VECTOR(63 DOWNTO 0);
        SIGNAL V21, V22, V23, V24: STD_LOGIC_VECTOR(63 DOWNTO 0);
        SIGNAL V31, V32, V33, V34: STD_LOGIC_VECTOR(63 DOWNTO 0);
        SIGNAL V41, V42, V43, V44: STD_LOGIC_VECTOR(63 DOWNTO 0);
BEGIN
-- prvi stepen
        sw11: switch_2x2
        PORT MAP (
                X => V21,
                Y => X1,
                S => 63,
                U => U11,
                V => Y1
```

```
        );
        sw12: switch_2x2
        PORT MAP (
                X => V22,
                Y => U11,
                S => 62,
                U => U12,
                V => Y2
        );
        sw13: switch_2x2
        PORT MAP (
                X => V23,
                Y => U12,
                S => 61,
                U => U13,
                V => Y3
        );
        sw14: switch_2x2
        PORT MAP (
                X => V24,
                Y => U13,
                S => 60,
                V => Y4
        );
-- drugi stepen
        sw21: switch_2x2
        PORT MAP (
                X => V31,
                Y => X2,
                S => 63,
                U => U21,
                V => V21
        );
        sw22: switch_2x2
        PORT MAP (
                X => V32,
                Y => U21,
                S => 62,
                U => U22,
                V => V22
        );
        sw23: switch_2x2
        PORT MAP (
                X => V33,
                Y => U22,
                S => 61,
                U => U23,
                V => V23
```

```
	);
	sw24: switch_2x2
	PORT MAP (
		X => V34,
		Y => U23,
		S => 60,
		V => V24
	);
-- treci stepen
	sw31: switch_2x2
	PORT MAP (
		X => V41,
		Y => X3,
		S => 63,
		U => U31,
		V => V31
	);
	sw32: switch_2x2
	PORT MAP (
		X => V42,
		Y => U31,
		S => 62,
		U => U32,
		V => V32
	);
	sw33: switch_2x2
	PORT MAP (
		X => V43,
		Y => U32,
		S => 61,
		U => U33,
		V => V33
	);
	sw34: switch_2x2
	PORT MAP (
		X => V44,
		Y => U33,
		S => 60,
		V => V34
	);
-- cetvrti stepen
	sw41: switch_2x2
	PORT MAP (
		X => (OTHERS => 'X'),
		Y => X4,
		S => 63,
		U => U41,
		V => V41
```

```
	);
	sw42: switch_2x2
	PORT MAP (
		X => (OTHERS => 'X'),
		Y => U41,
		S => 62,
		U => U42,
		V => V42
	);
	sw43: switch_2x2
	PORT MAP (
		X => (OTHERS => 'X'),
		Y => U42,
		S => 61,
		U => U43,
		V => V43
	);
	sw44: switch_2x2
	PORT MAP (
		X => (OTHERS => 'X'),
		Y => U43,
		S => 60,
		V => V44
	);
END ARCHITECTURE;
```