POGLAVLJE 4.
AUTOMATI OPŠTEG TIPA

U ovom poglavlju su ilustrovani postupci implementacije minimalne forme automata opšeg tipa. Opisani su postupci minimizacije broja stanja potpuno i nepotpuno definisanih automata. Implementacija kanoničke strukture automata je prikazana pomoću šema logičkih elemenata kao i pomoću VHDL jezika za opis fizičke arhitekure.

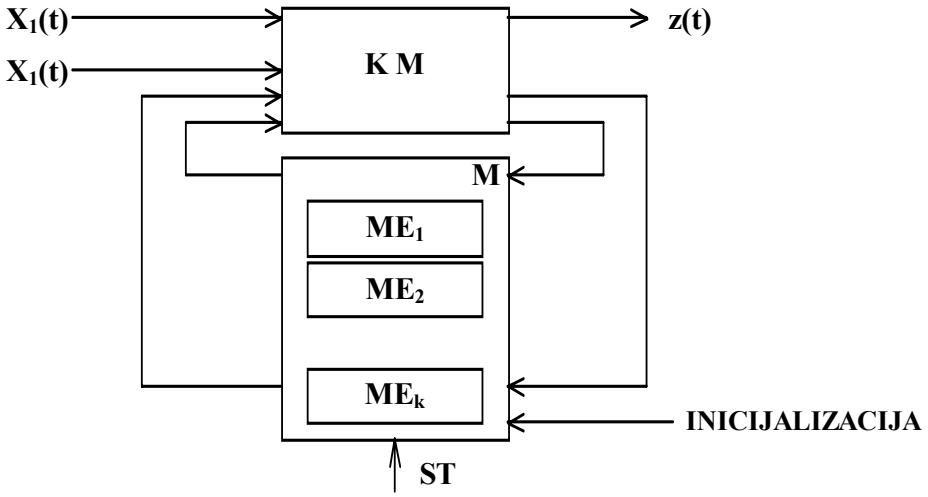
Sekvencijalni sistemi ili automati se realizuju pomoću sekvencijalnih mreža. Jedan od standardnih oblika realizacije sekvencijalnih mreža je kanonička forma realizacije, pri čemu se koristi termin kanonička jer je standardna za sve sekvencijalne sisteme.

Sekvencijalni sistem se sastoji od kombinacione mreže i memorije koja služi za čuvanje stanja. Kombinaciona mreža generiše funkciju pobude memorije i izlaza sekvencijalne mreže.

Pošto je sistem sinhron, signal takta ili sinhronizacije (označen ST) određuje vremenske trenutke u kojima dolazi do formiranja narednog stanja i izdvajanja izlaznog signala. U vreme t unosi se novo stanje u memoriju i tu se čuva sve do trenutka t+1. Signal takta čine periodični impulsi takta.

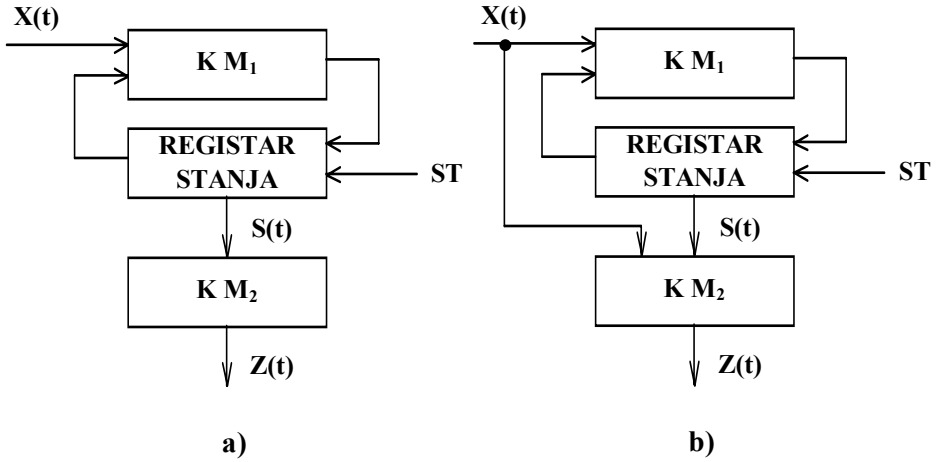
U opštem slučaju, sistem je neophodno dovesti u početno stanje u cilju dobijanja željenog ponašanja na ulazu-izlazu. Inicijalizacija se realizuje nad delom memorije posebnim signalom inicijalizacije.

Memorija se, slično kao i kombinacione mreže, realizuje u obliku elementarnih komponenti, flip-flopova. Stoga se memorija često realizuje u obliku registra stanja ili niza memorijskih automata sa razdvojenim ulazima pobude i izlazima sa kojih se skidaju promenljive stanja. Kanoničku strukturu sekvencijalnog sistema (automata) prikazuje Slika 4.1.



Slika 4.1: Kanonička struktura automata

Kanoničke strukture Milijevog i Murovog automata se razlikuju u tome da kod Milijevog automata, izlaz iz mreže zavisi i od stanja S(t) i od ulaznog signala X(t) što nije slučaj kod Murovog automata. Slika 4.2 a) i b) prikazuje kanoničku strukturu Murovog i Milijevog automata respektivno. Vidi se da su, u slučaju Milijevog automata, na ulaz mreže koja formira izlaz Z(t) dovedeni i stanje S(t) i ulaz X(t), što nije slučaj kod Murovog automata.



Slika 4.2: Kanonička struktura
a) Murov automat
b) Milijev automat

4.1 ZADATAK:

Data je tabela prelaza/izlaza Milijevog automata (Tabela 4.1).

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
W_1	S_2/Z_0	S_1/Z_1	S_2/Z_0	S_3/Z_1	S_6/Z_0	S_8/Z_1	S_6/Z_0	S_4/Z_0
W_2	S_2/Z_1	S_4/Z_0	S_2/Z_1	S_2/Z_0	S_4/Z_1	S_4/Z_0	S_2/Z_1	S_4/Z_1
W_3	S_5/Z_1	S_4/Z_0	S_5/Z_1	S_2/Z_0	S_3/Z_1	S_8/Z_0	S_8/Z_1	S_7/Z_1

Tabela 4.1: Tabela prelaza/izlaza automata

a) Pronaći odgovarajući minimalni ekvivalentni automat.
b) Izvršiti sintezu minimalnog automata u VHDL jeziku za opis fizičke arhitekture.
c) Izvršiti sintezu minimalnog automata kanoničkom metodom pomoću standardnih logičkih kola i D flip-flopova.

Prilikom sinteze automata stanja, ulazne i izlazne signale kodovati prirodnim BCD kodom. Početno stanje automata je stanje S_1.

REŠENJE:

a)

U postupku minimizacije automata koristi se implikaciona tabela. To je specijalna trougaona tabela pomoću koje se dolazi do klasa ekvivalentnih stanja.

Tabela se formira na sledeći način:

- vertikalno (vrste) se unose sva stanja izuzev prvog, a horizontalno (kolone) sva stanja izuzev poslednjeg (radi kraćeg zapisivanja unose se samo indeksi stanja), tako da će za automat sa k-stanja, implikaciona tabela imati k-1 vrsta i isto toliko kolona;
- oznake vrste i kolone se interpretiraju kao koordinate ćelije tabele;
- na osnovu tabele prelaza/izlaza popunjavaju se sve ćelije implikacione tabele;
- vrši se pretraživanje tabele sa ciljem pronalaženja klasa ekvivalentnih stanja

Da bi se ispravno popunila tabela, potrebno je pridržavati se sledećih pravila:

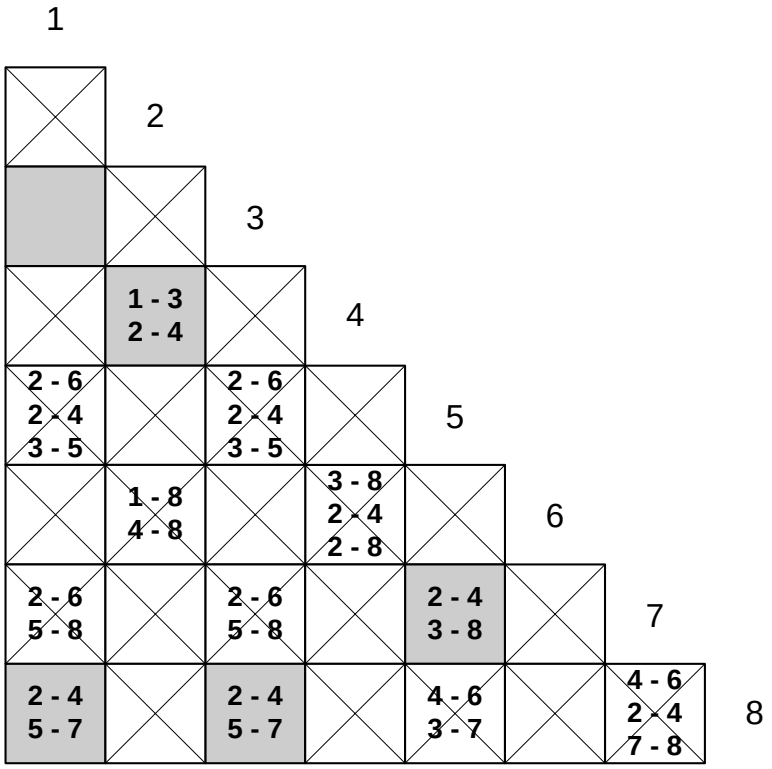
1. U ćelije, čije koordinate čine neekvivalentna stanja (za isti ulazni signal, izlazi su različiti) unosi se znak "X";
2. U ćelije, čije koordinate čine ekvivalentna stanja (za isti ulazni signal, izlazi su isti) unose se parovi indeksa narednih stanja automata za odgovarajuće ulaze (najpre manji, pa veći indeks), i to bez ponavljanja. Ukoliko su stanja ista, ne upisuje se ništa;
3. Ukoliko je ćelija sa koordinatama (i,j) precrtana, moraju se precrtati i ćelije koje i-j sadrže kao obuhvaćeni par;

4. Koordinate neprecrtanih ćelija predstavljaju međusobno ekvivalentna stanja.

Primena ovih pravila će se ilustrovati na nekoliko primera iz ovog zadatka (Slika 4.3):

- ćelija (2,8) → Posmatranjem tabele prelaza/izlaza zadatog automata vidi se da su stanja S_2 i S_8 neekvivalentna, te se na osnovu 1. pravila u ovu ćeliju upisuje znak "X";
- ćelija (4,6) → Iz tabele prelaza/izlaza se vidi da su stanja S_4 i S_6 ekvivalentna, pa se u ćeliju upisuju parovi indeksa narednih stanja, i to 3-8, 2-4 i 2-8 (2.pravilo). Međutim, prilikom kasnijeg pretraživanja tabele, videće se da je ćelija sa koordinatama (2-8) precrtana, pa će se, prema 3. pravilu, i ova ćelija precrtati;
- ćelija (5,7) → Stanja S_5 i S_7 su, prema tabeli prelaza/izlaza, ekvivalentna, pa se u ovu ćeliju (pravilo 3) upisuju parovi 2-4 i 3-8, dok se par 6-6 neće upisati. Kasnije, nakon pretraživanja implikacione tabele, videće se da ćelije sa koordinatama (2,4) i (3,8) nisu precrtane, te će ova ćelija ostati neprecrtana.

Poštujući prethodno navedena pravila, a na osnovu zadate tabele prelaza/izlaza, dobija se implikaciona tabela polaznog automata (Slika 4.3).



Slika 4.3: Implikaciona tabela automata

Sada se posmatranjem kolona implikacione tabele, počev sa desna u levo, pronalaze klase ekvivalentnih stanja.

U cilju pronalaženja klasa ekvivalentnih stanja formira se tabela (Slika 4.4) u čijim vrstama se (odozgo na dole) navode koordinate kolona implikantne tabele, dok se po vrstama upisuju koordinate NEprecrtanih ćelija u odgovarajućim kolonama, ili crtica, kada takvih ćelija nema. Ova tabela se popunjava odozgo na dole, prenoseći pri tom koordinate ekvivalentnih stanja iz prethodne vrste.

Ukoliko se u nekoj od vrsta pojave klase ekvivalentnih stanja oblika (m,n), (n,p), (m,p) umesto njih se, zbog važenja tranzitivnosti, piše jedna klasa ekvivalencije (m,n,p). U ovom slučaju to važi za klase ekvivalentnih stanja (1,3), (3,8), (1,8) koje prelaze u (1,3,8).

7	—
6	—
5	(5,7)
4	(5,7)
3	(5,7),(3,8)
2	(5,7),(3,8),(2,4)
1	(5,7),**(3,8)**,(2,4),**(1,3)**,**(1,8)** ⇔ **(1,3,8)**,(2,4),(5,7)
	(1,3,8),(2,4),(5,7),(6) S_0^* S_1^* S_2^* S_3^*

Slika 4.4: Pomoćna tabela za dobijanje klasa ekvivalentnih stanja

Kako je neophodno da sva stanja polaznog automata budu obuhvaćena nekom od klasa ekvivalentnih stanja, ona stanja koja se nakon ovog postupka ne pojavljuju, ekvivalentna su jedino sama sebi, odnosno svako od njih će formirati posebnu klasu ekvivalencije, kao što je to ovde slučaj sa stanjem S_6.

Stanja minimalnog automata dobiće se tako što se svakoj od klasa ekvivalentnih stanja pridruži po jedno novo stanje, koje je ujedno i stanje minimalnog automata.

Tabela 4.2 predstavlja tabelu prelaza/izlaza minimalnog automata.

	S_0^*	S_1^*	S_2^*	S_3^*
W_1	S_1^*/Z_0	S_0^*/Z_1	S_3^*/Z_0	S_0^*/Z_1
W_2	S_1^*/Z_1	S_1^*/Z_0	S_1^*/Z_1	S_1^*/Z_0
W_3	S_2^*/Z_1	S_1^*/Z_0	S_0^*/Z_1	S_0^*/Z_0

Tabela 4.2: Tabela prelaza/izlaza minimalnog automata

Prikazanim postupkom minimizacije se broj stanja automata smanjio na samo 4, što je dvostruko manje od 8 stanja, koliko ima polazni automat. Dakle, potreban broj memorijskih elemenata (flip-flopova) za implementaciju automata se smanjuje sa 3 na 2

b)

Prilikom sinteze automata treba uočiti tri sastavna modula arhitekture automata koji se realizuje:

- kombinaciona mreža za generisanje izlaznog vektora na osnovu trenutnog stanja i vrednosti ulaznog vektora,
- kombinaciona mreža za generisanje narednog stanja na osnovu trenutnog stanja i vrednosti ulaznog vektora i
- memorijske elemente (flip-flopove) za memorisanje trenutnog stanja automata.

Pošto su vrednosti izlaza obe kombinacione mreže zavisne od istih signala (vrednost ulaznog vektora i kod trenutnog stanja), one se mogu realizovati u okviru jednog VHDL procesa pomoću ugnježđenih `CASE` iskaza. Smeštanje koda trenutnog stanja se može realizovati sa drugim VHDL procesom.

Vodeći računa o kodovanju signala, sinteza automata u VHDL jeziku za opis fizičke arhitekture sledi direktno iz dobijene tabele prelaza/izlaza minimalnog automata.

```
------------------------------------------------------------------
--  Pakovanje sa opisom novoformiranih tipova
--  koji se koriste u realizaciji
--     1. tSTANJA - stanja koja moze da ima automat
--     2. tULAZI  - vrednosti ulaza koja moze da ima automat
--     3. tIZLAZI - vrednosti izlaza koje automat moze da generise
------------------------------------------------------------------
PACKAGE TIPOVI IS
  -- S0="00"; S1="01"; S2="10"; S3="11";
  TYPE tSTANJA IS (S0, S1, S2, S3);
  -- W1="00"; W2="01"; W3="10";
  TYPE tULAZI  IS (W1, W2, W3);
  -- Z0='0';  Z1='1';
  TYPE tIZLAZI IS (Z0, Z1);
END TIPOVI;

LIBRARY ieee;
  USE ieee.std_logic_1164.all;
  USE work.TIPOVI.all;

ENTITY ZAD01 IS PORT (
  -- takt ulaz
  iCLK:   IN  std_logic;
  -- signal za postavljanje pocetnog stanja
  iRESET: IN  std_logic;
  -- vektor ulaznih sugnala, vektor od 2 signala
  iW:     IN  tULAZI;
  -- vektor izlaznih signala, vektor od 1 signala
  oZ:     OUT tIZLAZI );
END ZAD01;

ARCHITECTURE ARH_ZAD01 OF ZAD01 IS
  -- deklaracija signala koji sadrze informaciju o
  -- tekucem i sledecem (narednom) stanju
  SIGNAL sSTANJE, sSLEDECE_STANJE : tSTANJA;
BEGIN
```

```
    -- kombinaciona mreza koja na osnovu trenutnog stanja
    -- i vrednosti ulaza generise kod narednog stanja i
    -- vrednost izlaznog vektora
    PROCESS (iW, sSTANJE) BEGIN
      CASE sSTANJE IS
        WHEN S0 => -- stanje S0
          CASE iW IS
            WHEN W1 => sSLEDECE_STANJE <= S1; oZ <= Z0;
            WHEN W2 => sSLEDECE_STANJE <= S1; oZ <= Z1;
            WHEN W3 => sSLEDECE_STANJE <= S2; oZ <= Z1;
          END CASE;
        WHEN S1 => -- stanje S1
          CASE iW IS
            WHEN W1 => sSLEDECE_STANJE <= S0; oZ <= Z1;
            WHEN W2 => sSLEDECE_STANJE <= S1; oZ <= Z0;
            WHEN W3 => sSLEDECE_STANJE <= S1; oZ <= Z0;
          END CASE;
        WHEN S2 => -- stanje S2
          CASE iW IS
            WHEN W1 => sSLEDECE_STANJE <= S3; oZ <= Z0;
            WHEN W2 => sSLEDECE_STANJE <= S1; oZ <= Z1;
            WHEN W3 => sSLEDECE_STANJE <= S0; oZ <= Z1;
          END CASE;
        WHEN S3 => -- stanje S3
          CASE iW IS
            WHEN W1 => sSLEDECE_STANJE <= S0; oZ <= Z1;
            WHEN W2 => sSLEDECE_STANJE <= S1; oZ <= Z0;
            WHEN W3 => sSLEDECE_STANJE <= S0; oZ <= Z0;
          END CASE;
      END CASE;
    END PROCESS;

    -- flip-flopovi za smestanje koda
    -- trenutnog stanja; postavljanje pocetnog
    -- stanja je sinhrono sa signalom takta
    PROCESS (iCLK) BEGIN
      IF (iCLK'EVENT AND iCLK='1') THEN
        IF (iRESET = '1') THEN -- sinhroni reset
          sSTANJE <= S0;
        ELSE
          sSTANJE <= sSLEDECE_STANJE;
        END IF;
      END IF;
    END PROCESS;

  END ARH_ZAD01;
```

U prethodnom rešenju je kodovanje signala realizovano sa formiranjem tipova nabrajanja sa svaki vektor (`tULAZI`, `tIZLAZI`, `tSTANJA`). Kod tipova nabrajanja se prvom literalu dodeljuje vrednost 0, drugom 1 i tako dalje do poslednjeg literala. Širina vektora kojim se predstavlja signal datog tipa zavisi od broja literala, tj. sa n signala se može predstaviti najviše 2^n literala.

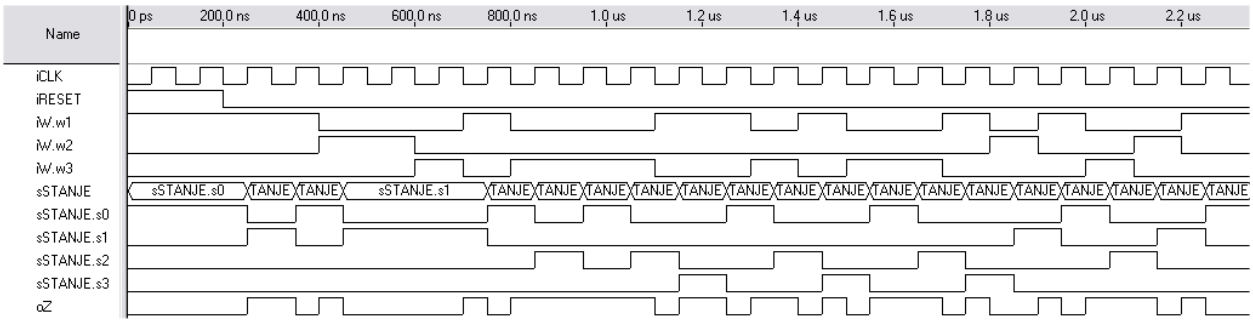
Deklaracija ovih tipova je smeštena u posebno pakovanje, nazvano `TIPOVI`, koje je uključeno u VHDL kodu pre samog opisa entiteta automata. Deklaraciju

novih tipova nije bilo moguće smestiti u okviru zaglavlja arhitekture automata, `ARH_ZAD01`, jer se novoformirani tipovi koriste u okviru opisa ulazno/izlaznih portova automata za predstavljanje ulaznog i izlaznog vektora, znači pre zaglavlja arhitekture.

Arhitektura automata je opisana sa dva već navedena procesa. Potrebno je uočiti da u prvom procesu ne postoji `WHEN OTHERS` iskaz. Naime, ovaj iskaz se mora koristiti kod slučajeva kada navedeni skup vrednosti vektora koji se analizira nije u potpunosti naveden u `CASE` iskazu, kao što je to gotovo uvek slučaj sa vektorima koji su tipa `std_logic_vector`. Pošto se u ovom slučaju koriste tipovi čiji je skup vrednosti lako obuhvatiti u potpunosti sa odgovarajućim `CASE` iskazom to dovodi do situacije da je npr. `WHEN OTHERS` iskaz jednak sa iskazom `WHEN S3`, u `CASE` iskazu za analizu koda trenutnog stanja automata.

Postavljanje početnog stanja se realizuje sinhrono sa signalom takta (`iCLK`) pomoću ulaznog signala `iRESET` koji je aktivan na visokom nivou.

Slika 4.5 prikazuje vrenenski dijagram rada automata koji simulira prolazak automata kroz sva stanja.



Slika 4.5: Simulacija rada realizovanog automata

Na početku vremenskog dijagrama se automat postavi u početno stanje pomoću signala `iRESET`. Nakon toga, u vremenskom trenutku 250ns na rastuću ivicu takt signala `iCLK`, automat prelazi u stanje S_1, jer je vrednost ulaznog vektora jednaka nuli. Pri tome se na izlazu generiše vrednost 1 sve do sledeće rastuće ivice takt signala kada se vraća u stanje S_0 i generiše vrednost 0 na izlazu. U vremenskom trenutku 400ns se menja vrednost ulaznog vektora što prouzrokuje promenu izlaznog vektora bez promene stanja automata. To je zbog toga što izlazni signal nije baferovan, odnosno njega generiše direktno kombinaciona mreža čiji su ulazi vrednost ulaznog vektora i kod trenutnog stanja automata (Milijev automat). Prateći vremenski dijagram do kraja, vidi se da su prikazani prolasci automata kroz sve moguce kombinacije koje prikazuje Tabela 4.2.

c)

Prvi korak kod sinteze automata kanoničkom metodom je kodovanje odgovarajućih promenljivih:

– za kodovanje signala ulaza Wi koristiće se prirodni kod, te će ulazi biti kodovani na sledeći način:

W_1 - 00 W_2 - 01 W_3 - 10

– za kodovanje stanja Si koristiće se prirodni kod, pa će biti:

S_0^* - 00 S_1^* - 01 S_2^* - 10 S_3^* - 11

– za kodovanje izlaza će se takođe koristiti prirodni kod, pa se oni mogu predstaviti na sledeći način:

Z_0 - 0 Z_1 – 1

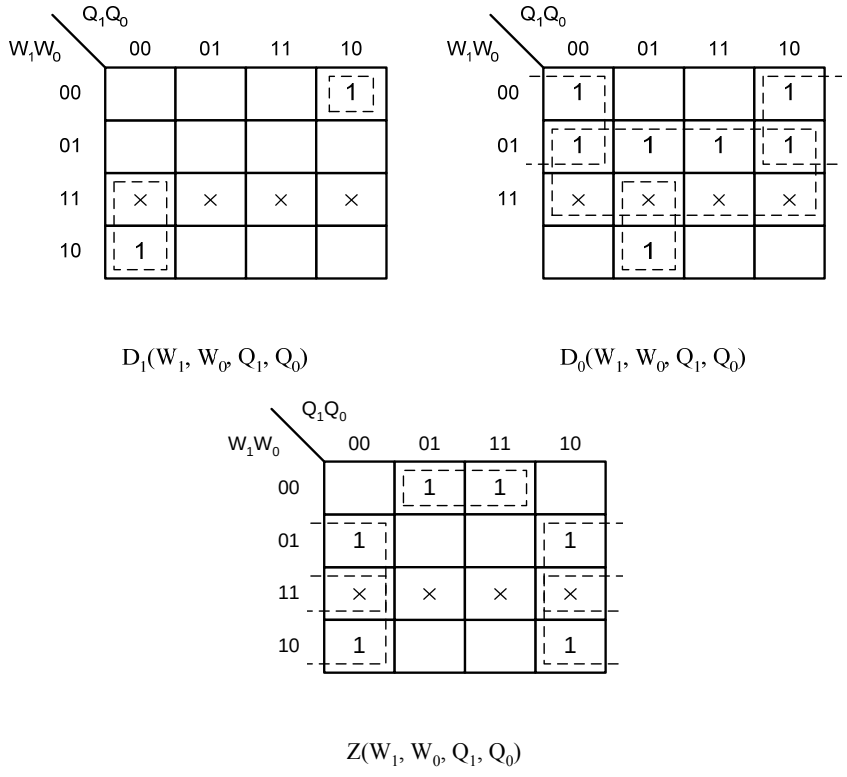
Kodovana tabela prelaza/izlaza minimalnog automata data je kao Tabela 4.3.

	00	01	10	11
00	01/0	00/1	11/0	00/1
01	01/1	01/0	01/1	01/0
10	10/1	01/0	00/1	00/0

Tabela 4.3: Kodovana tabela prelaza/izlaza minimalnog automata

Na osnovu ove tabele formiraju se Karnoove karte (Slika 4.6) za funkcije Q_1, Q_0 i Z, pri čemu Q_1 i Q_0 predstavljaju stanja automata, Z je izlazni signal, a W_1 i W_0 su ulazni signali (sa indeksom 1 je označen bit veće važnosti – MSB bit).

Pri tom, treba imati u vidu da će se za memorijske elemente koristiti D – flip-flopovi, pa važi $Q = D$.



Slika 4.6:Karnoove karte funkcija D_1, D_0 i Z

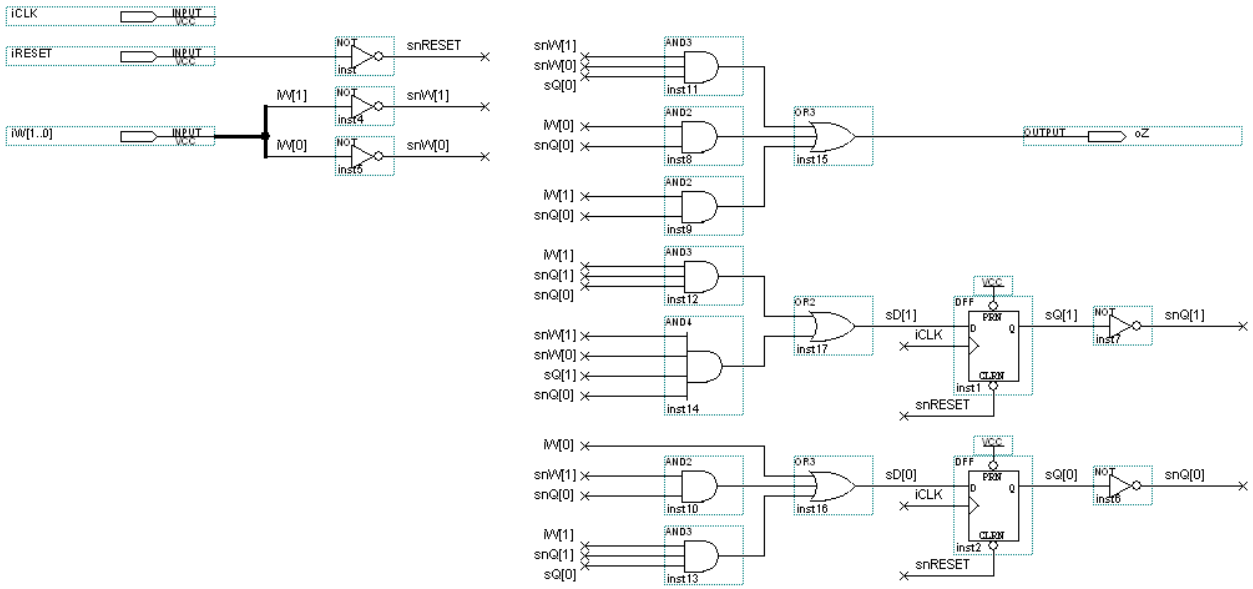
Iz Karnoovih karti se dobijaju sledeće jednačine:

$$D_1 = W_1 \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{W_1} \cdot \overline{W_0} \cdot Q_1 \cdot \overline{Q_0}$$

$$D_0 = W_0 + \overline{W_1} \cdot \overline{Q_0} + W_1 \cdot \overline{Q_1} \cdot Q_0$$

$$Z = \overline{W_1} \cdot \overline{W_0} \cdot Q_0 + W_0 \cdot \overline{Q_0} + W_1 \cdot \overline{Q_0}$$

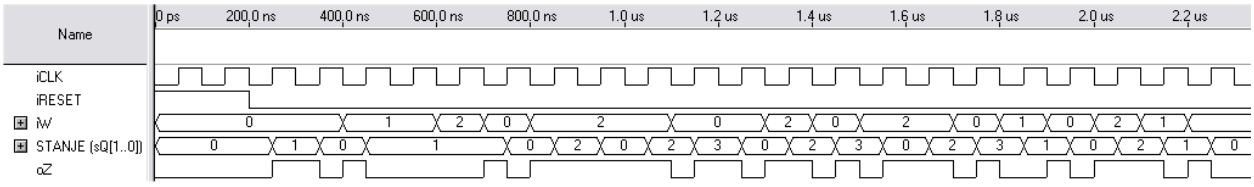
Na osnovu prethodnih jednačina može se formirati logička šema automata, Slika 4.7.



Slika 4.7: Logička šema realizacije automata

D flip-flopovi koji su korišćeni u realizaciji pored standardnih ulaza imaju i ulaze `PRN` i `CLRN`. Pomoću ovih ulaza se flip-flopovi asinhrono postavljaju na vrednost jedan (`PRN`), odnosno nula (`CLRN`), dovođenjem niskog naponskog nivoa na odogovarajući ulaz. Ovi ulazi su pogodni za realizaciju početnog stanja automata. Radi ekvivalencije sa prethodno realizovanim VHDL kodom, i ovde je postavljanje početnog stanja realizovano sa ulaznim signalom `iRESET` koji je aktivan na visokom nivou. Zbog toga je bilo potrebno invertovati njegovu vrednost pre dovođenja na `CLRN` ulaze flip-flopova (kod početnog stanja S_0^* je ”00” te je potrebno oba flip-flopa postaviti na vrednost 0).

Slika 4.8 prikazuje vremenski dijagram rada automata koji je identičan sa vremenskim dijagramom automata realizovanog pomoću VHDL jezika za opis fizičke arhitekture.



Slika 4.8: Vremenski dijagram rada automata

4.2 ZADATAK:

Automat je zadat tabelom prelaza/izlaza (Tabela 4.4).

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
0	$A_1/0$	$A_1/0$	$A_6/1$	$A_5/0$	$A_5/0$	$A_5/0$	$A_3/0$
1	$A_4/0$	$A_7/0$	$A_4/0$	$A_7/0$	$A_2/0$	$A_4/0$	$A_4/0$

Tabela 4.4:Tabela prelaza/izlaza automata

Q(t)	**Q(t+1)**	**J**	**K**
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Tabela 4.5:Tabela prelaza
JK flip-flopa

a) Izvršiti minimizaciju automata kanoničkom metodom
b) Izvršiti sintezu automata pomoću JK flip-flopova i NI kola. Tabela 4.5 prikazuje tablicu prelaza JK flip-flop.
c) Izvršiti sintezu minimalnog automata u VHDL jeziku za opis fizičke arhitekture.

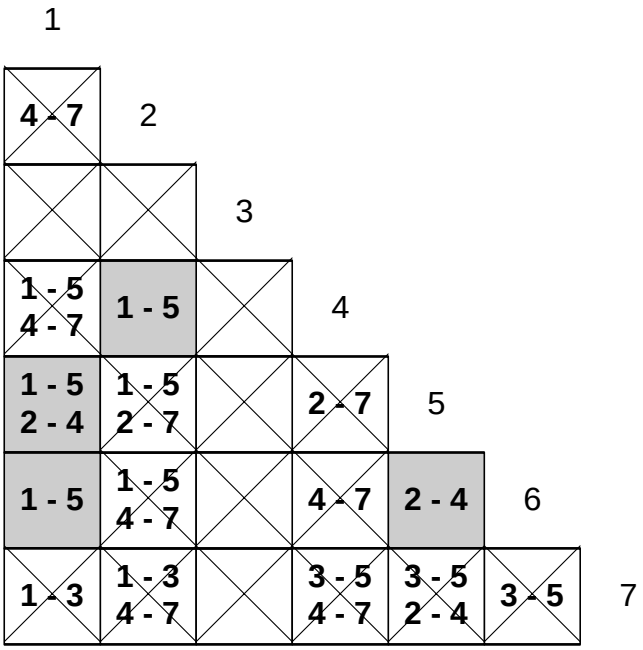
Prilikom sinteze automata stanja, ulazne i izlazne signale kodovati prirodnim BCD kodom. Početno stanje automata je stanje A_1. Izvršiti baferovanje izlaznog signala.

Ako je automat u početnom stanju A_1, odrediti odziv na sledeću pobudnu sekvencu: 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0. Odziv prikazati na odgovarajućim vremenskim dijagramima za obe realizacije minimalnog automata.

REŠENJE:

a)

Minimizacija automata sa izvodi korišćenjem implikacione tablice u kojoj se radi preglednosti umesto punog naziva stanja koriste samo indeksi stanja, Slika 4.9.



Slika 4.9:Implikaciona tablica zadatog automata

Iz implikacione tablice proizilazi tabela ekvivalentnih stanja, Slika 4.10.

6	—
5	(5,6)
4	(5,6)
3	(5,6)
2	(5,6),(2,4)
1	**(5,6)**,(2,4),**(1,5)**,**(1,6)** ⇔ **(1,5,6)**,(2,4)
	(1,5,6), (2,4), (3), (7)

Slika 4.10: Tabela ekvivalentnih stanja

Prema tome, sva stanja automata se mogu rasporediti u četiri klase ekvivalencije:

$$(A_1, A_5, A_6)\ ,\ (A_2\ ,\ A_4)\ ,\ A_3\ ,\ A_7$$

Pridruživanjem jednog reprezenta svakoj od klasa ekvivalencije automata, dolazi se do minimalnog automata koji je ekvivalentan polaznom automatu i ima četiri stanja:

$S_1 = (A_1, A_5, A_6)$
$S_2 = (A_2, A_4)$
$S_3 = A_3$
$S_4 = A_7$

Imajući u vidu gore prikazane relacije koje vladaju između stanja polaznog automata i njemu ekvivalentnog minimalnog automata, dolazimo do tablice prelaza/izlaza minimalnog automata:

	S_1	S_2	S_3	S_4
0	$S_1/0$	$S_1/0$	$S_1/1$	$S_3/0$
1	$S_2/0$	$S_4/0$	$S_2/0$	$S_2/0$

Tabela 4.6:Tablica prelaza/izlaza minimalnog automata

b)

Kada se svakom stanju pridruži prirodni BCD kod prema sledećem obrascu:

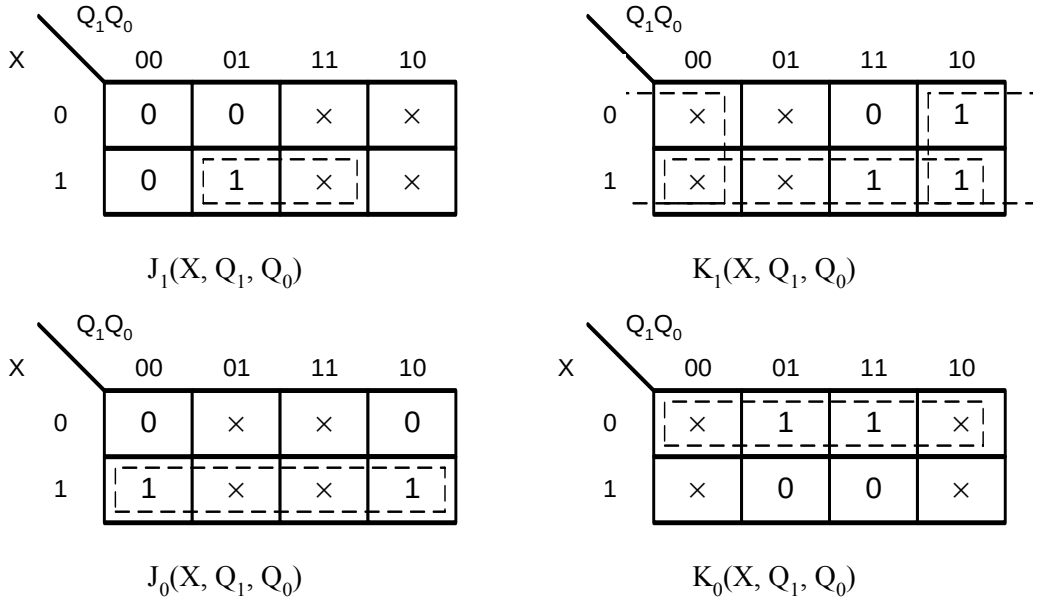
S_1 - 00 S_2 - 01 S_3 - 10 S_4 - 11,

dobija se kodovana tablica prelaza/izlaza minimalnog automata, Tabela 4.7.

	00	01	10	11
0	00/0	00/0	00/1	10/0
1	01/0	11/0	01/0	01/0

Tabela 4.7:Kodovana tablica prelaza/izlaza automata

Za realizaciju automata koji ima četiri stanja dovoljna su dva memorijska elementa (JK flip-flop-a). Bit veće važnosti se označava sa Q_1 a bit manje važnosti sa Q_0. Takođe, ulazni signal je označen sa X, a izlazni signal sa Y. Jednačine pobude flip-flopova dobijaju se na osnovu Karnoovih karti, Slika 4.11.



Slika 4.11: Karnoove karte za dobijanje jednačina pobude flip-flopova

Iz Karnoovih karti se dobijaju sledeće jednačine:

$$J_1 = X \cdot Q_0 \qquad J_0 = X$$

$$K_1 = X + \overline{Q_O} \qquad K_0 = \overline{X}$$

Jednačina izlaznog signala automata, $Y = \overline{X} \cdot Q_1 \cdot \overline{Q_0}$, proizilazi iz sledeće Karnoove karte, Slika 4.12.

X \ Q_1Q_0	00	01	11	10
0	0	0	0	1
1	0	0	0	0

$Y(X, Q_1, Q_0)$

Slika 4.12: Karnoove karte za dobijanje jednačine izlaza

Pošto za realizaciju automata na raspolaganju stoje NI kola, potrebno je primeniti DeMorganova pravila na jednačine za J_1, K_1 i Y. Zbog toga je:

$$J_1 = X \cdot Q_0 = \overline{\overline{X \cdot Q_0}}$$

$$K_1 = X + \overline{Q_0} = \overline{\overline{X + \overline{Q_0}}} = \overline{\overline{X} \cdot Q_0}$$

$$Y = \overline{X} \cdot Q_1 \cdot \overline{Q_0} = \overline{\overline{\overline{X} \cdot Q_1 \cdot \overline{Q_0}}}$$

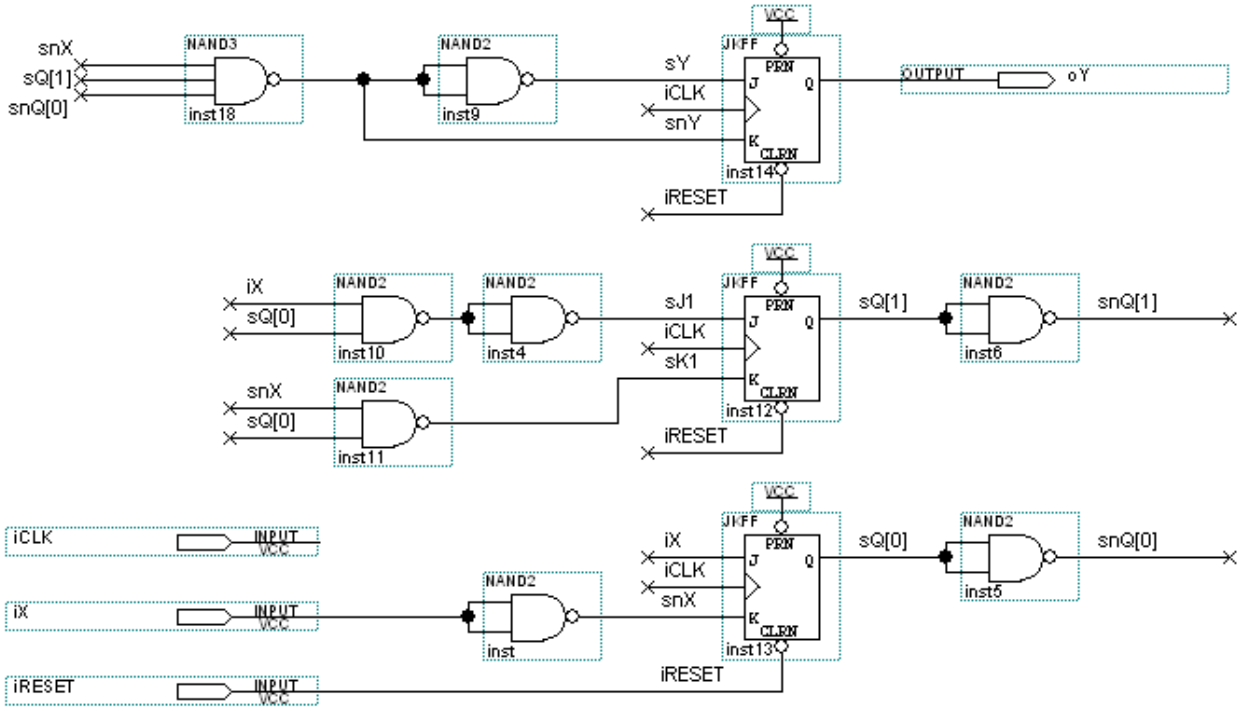
Na osnovu dobijenih jednačina može se formirati logička šema minimalnog automata, Slika 4.13.

Postavljanje početnog stanja automata je realizovano korišćenjem asinhronih ulaza za postavljanje flip-flopa na vrednost 0 (`CLRN`), odnosno 1 (`PRN`). Izlazni signal `sY` je baferovan sa JK flip-flopom koji se uz pomoć jednog invertora (realizovanog sa dvoulaznim NI kolom) ponaša kao D flip-flop.

Na osnovu tabele prelaza stanja minimalnog automata, Tabela 4.6 i Tabela 4.7, može se odrediti odziv automata na zadatu pobudu. Pri tome treba voditi računa o vrednosti izlaznog signala zbog realizovanog baferovanja D flip-flopom. Ponašanje automata za zadatu pobudu prikazuje Tabela 4.8.

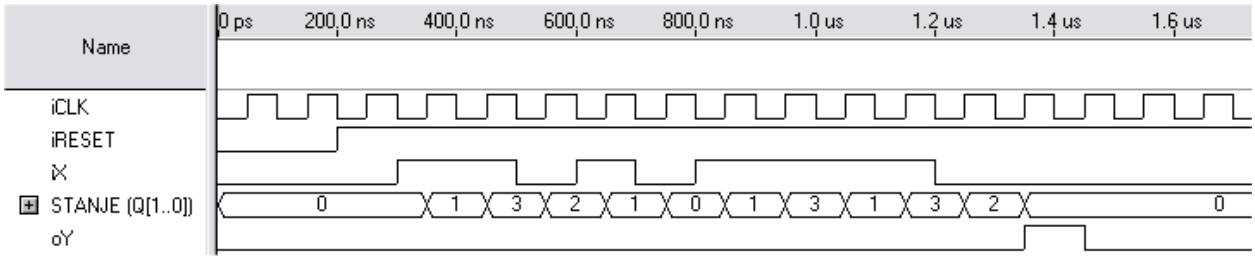
Ulaz		0	1	1	0	1	0	1	1	1	1	0
Stanje	$A_1=S_1$	S_1	S_2	S_4	S_3	S_2	S_1	S_2	S_4	S_2	S_4	S_3
Q_1Q_0	00	00	01	11	10	01	00	01	11	01	11	10
Izlaz		0	0	0	0	0	0	0	0	0	0	0

Tabela 4.8: Stanja automata i njegovi izlazi pri zadatoj pobudi



Slika 4.13: Logička šema realizovanog automata

Rezultati iz gornje tabele se potvrđuju određivanjem vremenskog odziva automata za zadatu pobudu, Slika 4.14.



Slika 4.14: Vremenski odziv automata za datu pobudu

Efekat baferovanja izlaznog signala vidljiv je tek na kraju vremenskog odziva kada se izlazni signal postavlja na vrednost jedan tek posle isteka jedne periode takta, nakon što je automat ušao u stanje S_3 (Q_1Q_0=10), pri čemu je vrednost ulaznog signala jednaka nuli.

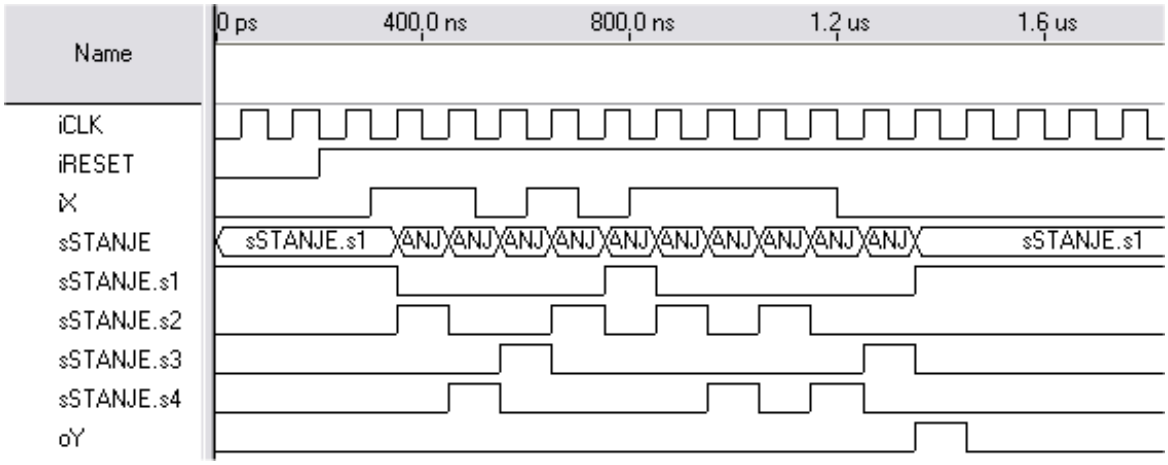
c)

Sinteza VHDL koda minimalnog automata je vrlo slična kao u prethodnom zadatku. Jedina razlika je vezana za baferovanje izlaznog signala. Zbog toga se izlaz kombinacione mreže za generisanje vrednosti izlaznog signala (signal `sY`) ne prosleđuje odmah na izlaz entiteta već na ulaz memorijskog elementa za baferovanje izlaznog signala. Vrednost ovog memorijskog elementa (njegov izlaz) predstavlja vrednost izlaznog signala iz realizovanog entiteta.

```
-------------------------------------------------------------------
--  Pakovanje sa opisom novoformiranih tipova
--  koji se koriste u realizaciji
--     1. tSTANJA - stanja koja moze da ima automat
--     2. tULAZI  - vrednosti ulaza koja moze da ima automat
--     3. tIZLAZI - vrednosti izlaza koje automat moze da generise
-------------------------------------------------------------------
PACKAGE TIPOVI IS
  -- S1="00"; S2="01"; S3="10"; S4="11";
  TYPE tSTANJA IS (S1, S2, S3, S4);
  -- X0='0';  X1='1';
  TYPE tULAZI  IS (X0, X1);
  -- Y0='0';  Y1='1';
  TYPE tIZLAZI IS (Y0, Y1);
END TIPOVI;

LIBRARY ieee;
  USE ieee.std_logic_1164.all;
  USE work.TIPOVI.all;

ENTITY ZAD02 IS PORT (
  iCLK:   IN  std_logic; -- takt ulaz
  iRESET: IN  std_logic; -- signal za postavljanje pocetnog stanja
  iX:     IN  tULAZI;    -- vektor ulaznih sugnala
  oY:     OUT tIZLAZI ); -- vektor izlaznih signala
END ZAD02;

ARCHITECTURE ARH_ZAD02 OF ZAD02 IS
  -- deklaracija signala koji sadrze informaciju o
  -- tekucem i sledecem (narednom) stanju
  SIGNAL sSTANJE, sSLEDECE_STANJE: tSTANJA;
  SIGNAL sY: tIZLAZI; -- signal koji generise vrednost izlaza
BEGIN

  -- kombinaciona mreza koja na osnovu trenutnog stanja
  -- i vrednosti ulaza generise kod narednog stanja i
  -- vrednost izlaznog vektora
  PROCESS (iX, sSTANJE) BEGIN
    CASE sSTANJE IS
      WHEN S1 => -- stanje S1
        CASE iX IS
          WHEN X0 => sSLEDECE_STANJE <= S1; sY <= Y0;
          WHEN X1 => sSLEDECE_STANJE <= S2; sY <= Y0;
        END CASE;
      WHEN S2 => -- stanje S2
        CASE iX IS
          WHEN X0 => sSLEDECE_STANJE <= S1; sY <= Y0;
          WHEN X1 => sSLEDECE_STANJE <= S4; sY <= Y0;
        END CASE;
      WHEN S3 => -- stanje S3
        CASE iX IS
          WHEN X0 => sSLEDECE_STANJE <= S1; sY <= Y1;
          WHEN X1 => sSLEDECE_STANJE <= S2; sY <= Y0;
        END CASE;
      WHEN S4 => -- stanje S4
        CASE iX IS
          WHEN X0 => sSLEDECE_STANJE <= S3; sY <= Y0;
          WHEN X1 => sSLEDECE_STANJE <= S2; sY <= Y0;
```

```
        END CASE;
      END CASE;
    END PROCESS;

    -- flip-flopovi za smestanje koda
    -- trenutnog stanja; postavljanje pocetnog
    -- stanja je sinhrono sa signalom takta
    PROCESS (iCLK) BEGIN
      IF (iCLK'EVENT AND iCLK='1') THEN
        IF (iRESET = '0') THEN -- sinhroni reset
          sSTANJE <= S1;
        ELSE
          sSTANJE <= sSLEDECE_STANJE;
        END IF;
      END IF;
    END PROCESS;

    -- flip-flop za baferovanje izlaznog signala
    PROCESS (iCLK) BEGIN
      IF (iCLK'EVENT AND iCLK='1') THEN
        IF (iRESET = '0') THEN -- sinhroni reset
          oY <= Y0;
        ELSE
          oY <= sY;
        END IF;
      END IF;
    END PROCESS;

  END ARH_ZAD02;
```

Vremenski odziv VHDL realizacije automata za datu pobudu prikazuje Slika 4.15.



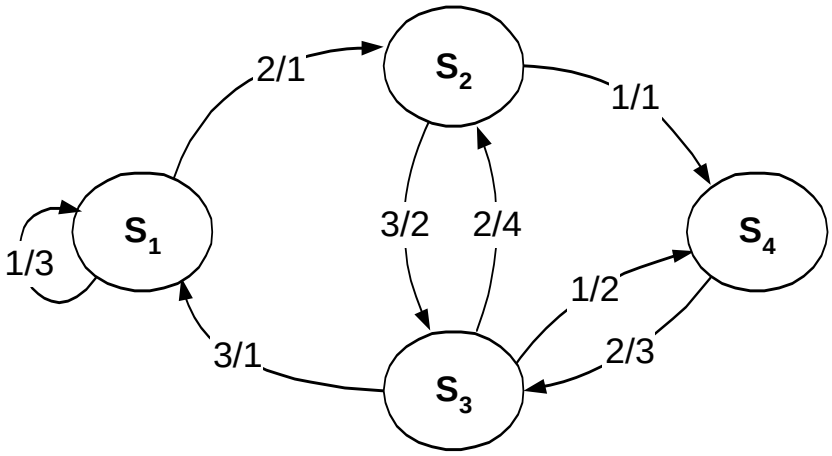
Slika 4.15: Vremenski odziv automata za datu pobudu

4.3 ZADATAK:

Automat je zadat grafom prelaza stanja, Slika 4.16.

a) Pronaći odgovarajući minimalni ekvivalentni automat.

b) Izvršiti sintezu minimalnog automata u VHDL jeziku za opis fizičke arhitekture.
c) Izvršiti sintezu minimalnog automata kanoničkom metodom pomoću standardnih logičkih kola i D flip-flopova.



Slika 4.16: Graf automata

Prilikom sinteze automata stanja, ulazne i izlazne signale kodovati Grejovim kodom. Početno stanje automata je stanje S_4.

REŠENJE:

a)

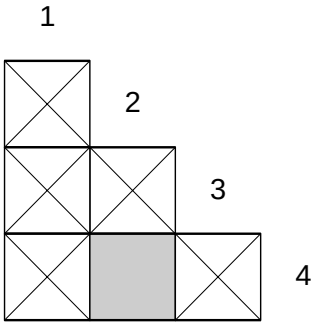
Najpre se na osnovu zadatog grafa formira tabela prelaza/izlaza polaznog automata, Tabela 4.9. Radi preglednosti, u tabeli se pišu samo indeksi odgovarajućih stanja (Si), odnosno izlaznih signala (Zi).

	S_1	S_2	S_3	S_4
W_1	1/3	4/1	4/2	b/b
W_2	2/1	b/b	2/4	3/3
W_3	b/b	3/2	1/1	b/b

Tabela 4.9: Tabela prelaza/izlaza automata

Može se primetiti da je ovde reč o nepotpuno definisanom automatu (nedefinisana stanja označena su sa "b"). U daljem postupku minimizacije koristiće se mogućnost da se nedefinisana naredna stanja i izlazi, po potrebi, odrede tako da budu identična odgovarajućim stanjima i izlazima, a sve u cilju dobijanja minimalnog automata za čiju je realizaciju potreban minimalan broj memorijskih elemenata.

Sada se može formirati implikaciona tabela (Slika 4.17), a zatim i pomoćna tabela ekvivalentnih stanja (Slika 4.18) iz koje se određuju stanja minimalnog automata.



Slika 4.17: Implikaciona tabela

3	—
2	(2,4)
1	(2,4)
	(2,4), (3), (1)
	A_2 A_3 A_1

Slika 4.18: Tabela ekvivalentnih stanja

Na osnovu dobijenih klasa ekvivalentnih stanja formira se tabela prelaza/izlaza minimalnog automata (Tabela 4.10), čime se završava postupak dobijanja minimalnog automata.

	A_1	A_2	A_3
W_1	$A_1/3$	$A_2/1$	$A_2/2$
W_2	$A_2/1$	$A_3/3$	$A_2/4$
W_3	b/b	$A_3/2$	$A_1/1$

Tabela 4.10: Tabela prelaza/izlaza minimalnog automata

b)

VHDL realizacija sledi direktno iz dobijene tablice prelaza/izlaza minimalnog automata. Za razliku od prethodnih zadataka, pri sintezi VHDL koda ovog automata treba voditi računa o kodovanju ulaznih i izlaznih simbola kao i kodovanju stanja automata, pošto je u zadatku zadato Grejovo kodovanje. To se može uraditi definisanjem posebnog tipa, s tim da treba obazrivo navoditi literale za tip koji se realizuje. Na primer, za dobijeni minimalni automat to bi se moglo realizovati na sledeći način:

```
-- A1="00"; A2="01"; DUMMY="10"; A3="11";
TYPE tSTANJA IS (A1, A2, DUMMY, A3);
```

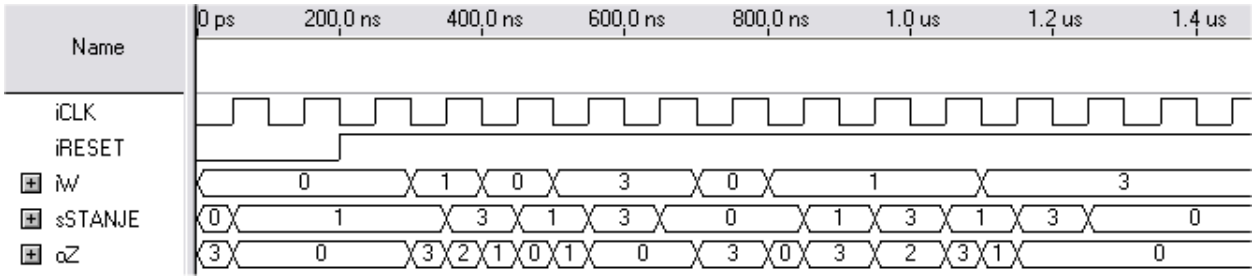
U navedenom VHDL kodu je odabran drugi pristup rešenja problema kodovanja stanja i vrednosti ulaznih i izlaznih simbola. Naime, korišćene su definisane konstante za svaki simbol pojedinačno. Definisane konstante su

std_logic_vector tipa, pa se zbog toga u CASE iskazima javlja WHEN OTHERS iskaz. U tom slučaju automat ostaje u trenutnom stanju i na izlaz postavlja izlazni simbol Z_1. Izuzev ove razlike, realizovani VHDL kod je potpuno analogan sa VHDL realizacijom automata iz prethodnih zadataka.

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY ZAD03 IS PORT (
  iCLK:   IN  std_logic;
  iRESET: IN  std_logic;
  iW:     IN  std_logic_vector(1 downto 0);
  oZ:     OUT std_logic_vector(1 downto 0) );
END ZAD03;

ARCHITECTURE ARH_ZAD03 OF ZAD03 IS
  -- deklaracija konstanti koje se koriste za identifikaciju:
  -- stanja
  CONSTANT cA1: std_logic_vector(1 downto 0) := "00";
  CONSTANT cA2: std_logic_vector(1 downto 0) := "01";
  CONSTANT cA3: std_logic_vector(1 downto 0) := "11";
  -- ulaznih simbola
  CONSTANT cW1: std_logic_vector(1 downto 0) := "00";
  CONSTANT cW2: std_logic_vector(1 downto 0) := "01";
  CONSTANT cW3: std_logic_vector(1 downto 0) := "11";
  -- izlaznih simbola
  CONSTANT cZ1: std_logic_vector(1 downto 0) := "00";
  CONSTANT cZ2: std_logic_vector(1 downto 0) := "01";
  CONSTANT cZ3: std_logic_vector(1 downto 0) := "11";
  CONSTANT cZ4: std_logic_vector(1 downto 0) := "10";
  -- deklaracija signala koji sadrze informaciju o
  -- tekucem i sledecem (narednom) stanju
  SIGNAL sSTANJE, sSLEDECE_STANJE: std_logic_vector(1 downto 0);
BEGIN
  -- kombinaciona mreza koja na osnovu trenutnog stanja
  -- i vrednosti ulaza generise kod narednog stanja i
  -- vrednost izlaznog vektora
  PROCESS (iW, sSTANJE) BEGIN
    CASE sSTANJE IS
      WHEN cA1    => -- stanje A1
        CASE iW IS
          WHEN cW1    => sSLEDECE_STANJE <= cA1; oZ <= cZ3;
          WHEN cW2    => sSLEDECE_STANJE <= cA2; oZ <= cZ1;
          WHEN OTHERS => sSLEDECE_STANJE <= cA1; oZ <= cZ1;
        END CASE;
      WHEN cA2    => -- stanje A2
        CASE iW IS
          WHEN cW1    => sSLEDECE_STANJE <= cA2; oZ <= cZ1;
          WHEN cW2    => sSLEDECE_STANJE <= cA3; oZ <= cZ3;
          WHEN cW3    => sSLEDECE_STANJE <= cA3; oZ <= cZ2;
          WHEN OTHERS => sSLEDECE_STANJE <= cA2; oZ <= cZ1;
        END CASE;
      WHEN OTHERS => -- stanje A3
        CASE iW IS
          WHEN cW1    => sSLEDECE_STANJE <= cA2; oZ <= cZ2;
          WHEN cW2    => sSLEDECE_STANJE <= cA2; oZ <= cZ4;
```

```
            WHEN cW3    => sSLEDECE_STANJE <= cA1; oZ <= cZ1;
            WHEN OTHERS => sSLEDECE_STANJE <= cA3; oZ <= cZ1;
          END CASE;
      END CASE;
    END PROCESS;

    -- flip-flopovi za smestanje koda
    -- trenutnog stanja; postavljanje pocetnog
    -- stanja je sinhrono sa signalom takta
    PROCESS (iCLK) BEGIN
      IF (iCLK'EVENT AND iCLK='1') THEN
        IF (iRESET = '0') THEN -- sinhroni reset
          sSTANJE <= cA2;
        ELSE
          sSTANJE <= sSLEDECE_STANJE;
        END IF;
      END IF;
    END PROCESS;
  END ARH_ZAD03;
```

Slika 4.19 prikazuje vremenski odziv automata na odabrani niz simbola. Niz ulaznih simbola je odabran tako da se simulira prolazak automata kroz sva stanja.



Slika 4.19: Simulacija rada realizovanog automata

Na početku vremenskog dijagrama automat se postavlja u početno stanje A_2. Na prvu sledeću rastuću ivicu takt signala (t=250ns) automat na osnovu ulaznog simbola W_1 ostaje u stanju A_2. U vremenskom trenutku t=350ns automat, na osnovu ulaznog simbola W_2, prelazi u stanje A_3. Dalje ponašanje realizovanog automata je u skladu sa tabelom prelaza/izlaza minimalnog automata (Tabela 4.10). Simulirane prelaske stanja automata tabelarno prikazuje Tabela 4.11.

vreme[ns]		250	350	450	550	650	750	850	950	1050	1150	1250
Ulaz		W_1	W_2	W_1	W_3	W_3	W_1	W_2	W_2	W_2	W_3	W_3
Stanje	A_2	A_2	A_3	A_2	A_3	A_1	A_1	A_2	A_3	A_2	A_3	A_1

Tabela 4.11: Prelasci stanja automata na odabrani niz ulaznih simbola

Vrednost izlaznog vektora pri tome zavisi od vrednosti ulaznog vektora i od trenutnog stanja u skladu sa dobijenom tabelom prelaza/izlaza.

c)

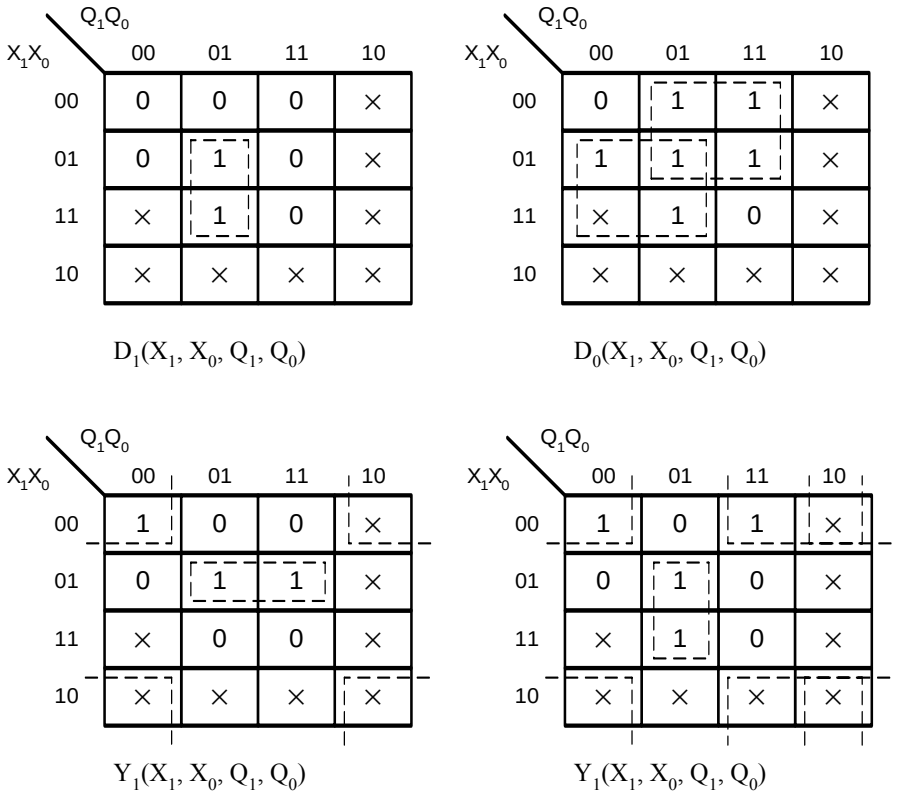
Da bi se formirala kodirana tabela prelaza/izlaza minimalnog automata (Tabela 4.12), potrebno je da se najpre kodiraju ulazni i izlazni signali, kao i stanja tog automata. To će se uraditi na sledeći način koristeći Grejov kod:

Stanja			Ulazi			Izlazi		
	Q_1	Q_0		X_1	X_0		Z_1	Z_0
$A_1 \rightarrow$	0	0	$W_1 \rightarrow$	0	0	$Z_1 \rightarrow$	0	0
$A_2 \rightarrow$	0	1	$W_2 \rightarrow$	0	1	$Z_2 \rightarrow$	0	1
$A_3 \rightarrow$	1	1	$W_3 \rightarrow$	1	1	$Z_3 \rightarrow$	1	1
						$Z_4 \rightarrow$	1	0

	00	01	11
00	00/11	01/00	01/01
01	01/00	11/11	01/10
11	b/b	11/01	00/00

Tabela 4.12: Kodirana tabela prelaza/izlaza minimalnog automata

Na osnovu ovako dobijene kodirane tabele prelaza/izlaza minimalnog automata, može se pristupiti crtanju Karnoovih karti za funkcije koje definišu ulaze u flip-flopove koji sadrže kod trenutnog stanja (D_1, D_0), kao i izlaze (Y_1, Y_0), Slika 4.20.



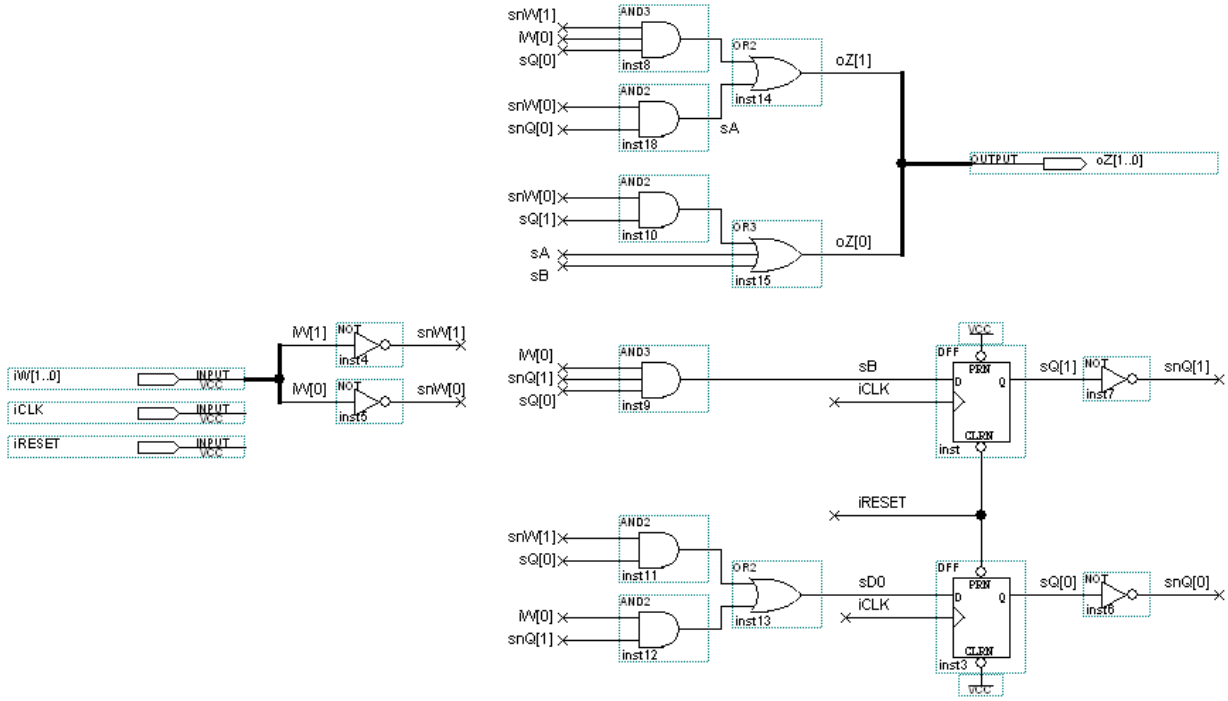
Slika 4.20: Karnoove karte

Minimizacijom, uz pomoć Karoovih karti, dobijaju se sledeće prenosne funkcije:

$$D_1 = X_0 \cdot \overline{Q_1} \cdot Q_0 = B$$
$$D_0 = \overline{X_1} \cdot Q_0 + X_0 \cdot \overline{Q_1}$$
$$Y_1 = \overline{X_1} \cdot X_0 \cdot Q_0 + \overline{X_0} \cdot \overline{Q_0} = \overline{X_1} \cdot X_0 \cdot Q_0 + A$$
$$Y_0 = \overline{X_0} \cdot \overline{Q_0} + X_0 \cdot \overline{Q_1} \cdot Q_0 + \overline{X_0} \cdot Q_1 = A + B + \overline{X_0} \cdot Q_1$$

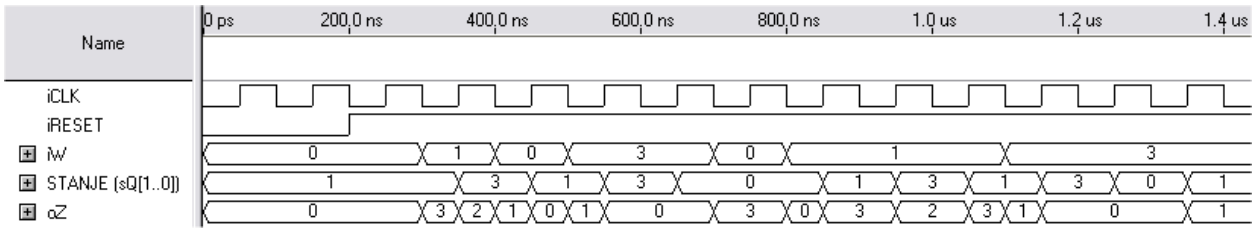
U dobijenim funkcijama se uočava da se izrazi oblika $X_0 \cdot \overline{Q_1} \cdot Q_0$, odnosno $\overline{X_0} \cdot \overline{Q_0}$ pojavljuju na dva mesta, pa postoji mogućnost da se realizujući ih samo jedan put pomoću odgovarajućeg I kola (čiji izlaz je označen sa B, odnosno sa A) uštede dva logička kola, što će se i učiniti.

Slika 4.21 prikazuje sintezu automata pomoću logičkih kola na osnovu dobijenih prenosnih funkcija. Na slici su ulazni signali X_1 i X_0 označeni sa `iW[1]` i `iW[0]` redom., dok su izlazni signali Y_1 i Y_0 označeni sa `iZ[1]` i `iZ[0]` respektivno.



Slika 4.21: Logička šema realizovanog automata

Vremenski odziv automata na odabrani niz ulaznih simbola prikazuje Slika 4.22. Niz ulaznih simbola je odabran tako da se simulira prolazak automata kroz sva stanja i identičan je nizu ulaznih simbola korišćenim prilikom simulacije VHDL realizacije istog automata.



Slika 4.22: Simulacija rada automata

Interesantna je situacija kada se automat nalazi u stanju A_1 i na ulazu se pojavi ulazni simbol W_3. Prema dobijenoj tablici prelaza stanja minimalnog automata nije definisano ponašanje automata u toj specifičnoj situaciji. Međutim, realizovani automat ne sme biti nedeterminisan i mora preći u neko definisano stanje. U ovom slučaju je to stanje zavisno od izvršene minimizacije. Data situacija se pojavljuje u vremenskom trenutku 1,05μs kada automat ulazi u stanje 1. Nakon 50ns, na ulazu se pojavljuje ulazni simbol W_3=11. Na osnovu dobijenih prenosnih funkcija, nakon izvršene minimizacije, sa sledećom rastućom ivicom takt signala realizovani automat prelazi u stanje A_3.

4.4 ZADATAK:

Automat je zadat tabelom prelaza/izlaza, Tabela 4.13.

W(i)	S_0	S_1	S_2	S_3	S_4	S_5	S_6
W_0	S_0/Z_0	S_1/Z_0	b/Z_0	S_3/Z_0	S_0/Z_0	b/Z_0	S_3/Z_0
W_1	b/Z_1	S_4/Z_1	S_3/Z_1	S_2/Z_1	b/Z_1	S_2/Z_1	b/Z_1
W_2	b/b	b/b	S_5/b	b/b	S_5/Z_1	S_0/Z_0	b/b
W_3	S_5/Z_0	b/Z_1	b/Z_0	b/Z_0	S_4/b	S_5/Z_1	S_4/b

Tabela 4.13: Zadata tablica prelaza/izlaza

a) Naći odgovarajući odziv automata na niz ulaznih simbola: W_3 W_1 W_2 W_1 W_1 W_0 W_1 W_2 W_3
b) Naći minimalni automat datog automata. Nacrtati graf minimalnog automata. Naći odgovarajući odziv minimalnog automata na niz ulaznih signala iz tačke a), ako se automat na početku nalazi u stanju kojem odgovara početno stanje originalnog automata.
c) Izvršiti sintezu minimalnog automata u VHDL jeziku za opis fizičke arhitekture.

Prilikom sinteze automata stanja, ulazne i izlazne signale kodovati Grejovim kodom. Početno stanje automata je stanje S_0.

REŠENJE:

a)

Iz zadate tabele prelaza/izlaza automata se primećuje da je automat nepotpuno definisan, pri čemu su nedefinisana stanja označena sa "b". U toku postupka minimizacije ta nedefinisana stanja će se dodeliti onim stanjima pomoću kojih se dobija potreban minimalan automat.

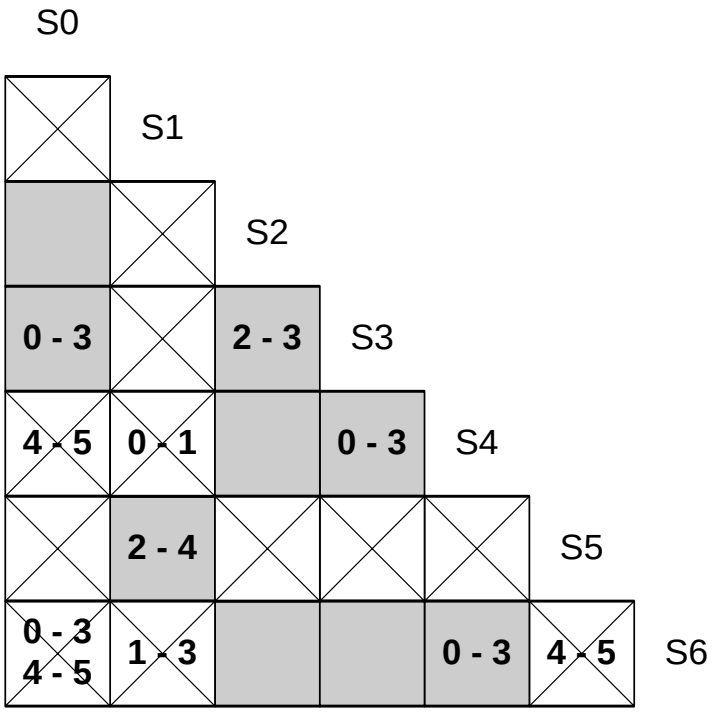
Na osnovu zadate tabele, moguće je odrediti ponašanje automata kada se zada ogovarajući niz ulaznih signala. Zadato početno stanje automata je stanje S_0. Na ulaz se dovodi signal W_3. Iz tabele je moguće zaključiti da automat u narednom trenutku prelazi u stanje S_5, a na izlazu daje signal Z_0. Ponavljajući ovaj postupak za ostale ulazne signale moguće je odrediti ukupni odziv automata na zadat niz ulaznih signala. Da bi ceo postupak bio pregledniji, odgovarajući odziv prikazuje Tabela 4.14.

	W_3	W_1	W_2	W_1	W_1	W_0	W_1	W_2	W_3
S_0	S_5	S_2	S_5	S_2	S_3	S_3	S_2	S_5	S_5
	Z_0	Z_1	b	Z_1	Z_1	Z_0	Z_1	b	Z_1

Tabela 4.14: Tabelarni prikaz odziva automata na zadati niz ulaznih signala

b)

Sada se može pristupiti formiranju ogovarajuće implikacione tablice i pomoćne tablice radi određivanja stanja minimalnog automata, Slika 4.23 i Slika 4.24.



Slika 4.23: Implikaciona tabela

5	—
4	(4,6)
3	**(4,6)**,**(3,4)**,**(3,6)** ⇔ **(3,4,6)**
2	**(3,4,6)**,**(2,3)**,**(2,4)**,**(2,6)** ⇔ **(2,3,4,6)**
1	(2,3,4,6),(1,5)
0	**(2,3,4,6)**,(1,5),**(0,2)**,**(0,3)** ⇔ **(0,2,3)**
	(2,3,4,6),(1,5),(0,2,3)
	S_0* S_1* S_2*

Slika 4.24: Pomoćna tabela ekvivalentnih stanja

Prilikom određivanja klasa ekvivalentnih stanja korišćen je zakon tranzitivnosti. Tako na primer u slučaju klasa (4,6); (3,4) i (3,6) moguće je primenom zakona tranzitivnosti formirati samo sa jednom klasom (3,4,6).

Uočava se da su stanja početnog automata S_2 i S_3 obuhvaćena sa dve klase ekvivalencije S_0* i S_2*. Ovakva situacija se pojavila iz razloga što početni automat nije potpuno definisan zbog čega se prilikom formiranja implikacione tabele, u cilju dobijanja minimalnog automata, za više nedefinisanih stanja pretpostavilo da su ekvivalenta sa istim definisanim stanjima.

Pošto je početno stanje zadatog automata, stanje S_0, obuhvaćeno klasom ekvivalencije (0,2,3), početno stanje minimalnog automata će biti stanje dodeljeno ovoj klasi, tj. stanje S_2*.

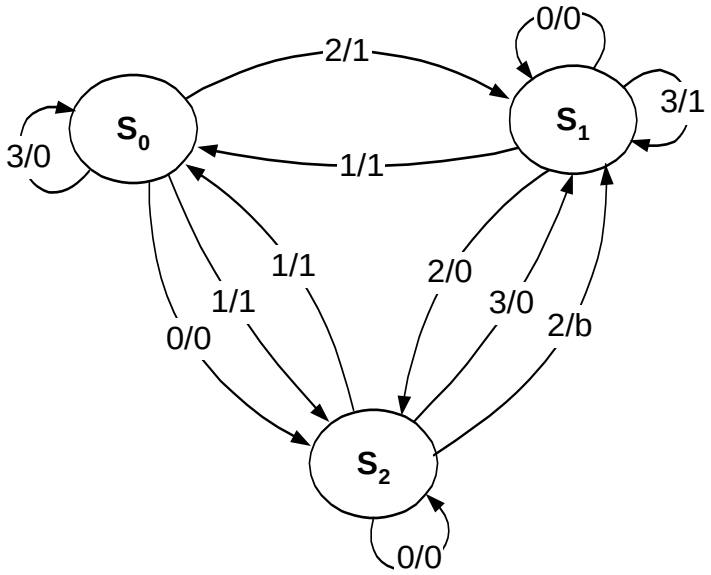
Nakon određivanja stanja minimalnog automata, pristupa se formiranju tablice prelaza/izlaza za minimalni automat, Tabela 4.15.

U ovom slučaju postoje dve specifične situacije. Prva je u slučaju kada se automat nalazi u stanju S_0* i na ulazu se pojavi ulazni simbol W_1. Na osnovu tablice ulaza/izlaza zadatog automata (Tabela 4.13) vidi se da u tom slučaju minimalni automat prelazi u stanje reprezentovano klasom ekvivalencije koja obuhvata stanja S_2 i S_3. Slika 4.24 prikazuje da klase ekvivalencije S_0* i S_2* zadovoljavaju ovaj uslov. U ovakvoj situacije se proizvoljno odabira u koje stanje će preći minimalni automat. U ovom rešenju je odabrano stanje S_2*. Slična situacija postoji kada se na ulazu pojavi ulazni simbol W_1 a automat se nalazi u stanju S_2*.

	S_0*	S_1*	S_2*
W_0	S_2*/Z_0	S_1*/Z_0	S_2*/Z_0
W_1	S_2*/Z_1	S_0*/Z_1	S_0*/Z_1
W_2	S_1*/Z_1	S_2*/Z_0	S_1*/b
W_3	S_0*/Z_0	S_1*/Z_1	S_1*/Z_0

Tabela 4.15: Tabela prelaza/izlaza minimalnog automata

Na osnovu dobijene tabele prelaza/izlaza, formira se graf prelazaka stanja minimalnog automata, Slika 4.25.



Slika 4.25: Graf minimalnog automata

Sada je potrebno naći odziv minimalnog automata na istovetan niz ulaznih signala. Pri tome je početno stanje minimalnog automata ekvivalentno početnom stanju polaznog automata. Tabela 4.16 prikazuje odziv minimalnog automata na zadati niz ulaznih simbola.

	W_3	W_1	W_2	W_1	W_1	W_0	W_1	W_2	W_3
S_2*	S_1*	S_0*	S_1*	S_0*	S_2*	S_2*	S_0*	S_1*	S_1*
	Z_0	Z_1	Z_1	Z_1	Z_1	Z_0	Z_1	Z_1	Z_1

Tabela 4.16: Odziv minimalnog automata na zadati niz ulaznih signala

c)

U nastavku je prikazana VHDL implementacija dobijenog minimalnog automata. Sinteza VHDL koda neposredno sledi iz dobijene tabele prelaza/izlaza minimalnog automata, Tabela 4.15. Pri tome je odabrano da se u slučaju da se automat nalazi u stanju S_2 i na ulazu pojavi ulazni simbol W_2, automat prevodi u stanje S_1 i na izlazu se postavlja izlazni simbol Z_0 umesto nedefinisane vrednosti (b).

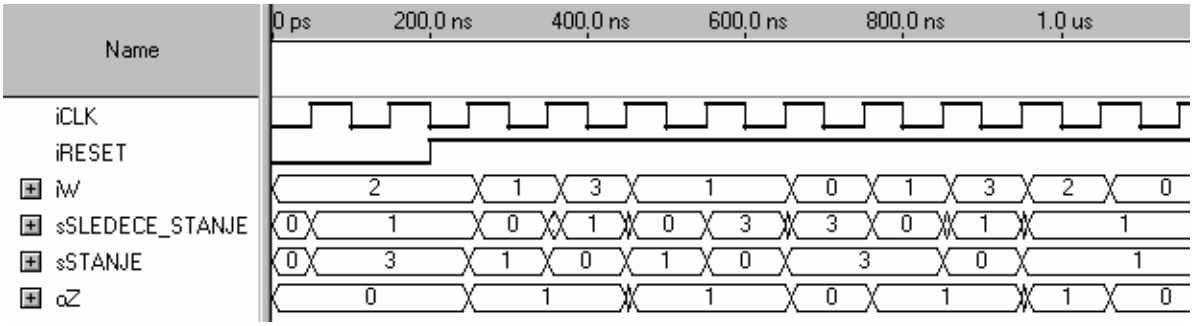
```
------------------------------------------------------------------
--  Pakovanje sa opisom novoformiranih tipova
--  koji se koriste u realizaciji
--     1. tSTANJA - stanja koja moze da ima automat
--     2. tULAZI  - vrednosti ulaza koja moze da ima automat
--     3. tIZLAZI - vrednosti izlaza koje automat moze da generise
------------------------------------------------------------------
```

```
PACKAGE TIPOVI IS
  -- S0="00"; S1="01"; S2="11"; S3="10";
  TYPE tSTANJA IS (S0, S1, S3, S2);
  -- W0="00"; W1="01"; W2="11"; W3="10";
  TYPE tULAZI  IS (W0, W1, W3, W2);
  -- Z0='0';  Z1='1';
  TYPE tIZLAZI IS (Z0, Z1);
END TIPOVI;

LIBRARY ieee;
  USE ieee.std_logic_1164.all;
  USE work.TIPOVI.all;

ENTITY ZAD04 IS PORT (
  iCLK:   IN  std_logic;
  iRESET: IN  std_logic;
  iW:     IN  tULAZI;
  oZ:     OUT tIZLAZI );
END ZAD04;

ARCHITECTURE ARH_ZAD04 OF ZAD04 IS
  -- deklaracija signala koji sadrze informaciju o
  -- tekucem i sledecem (narednom) stanju
  SIGNAL sSTANJE, sSLEDECE_STANJE : tSTANJA;
BEGIN

  -- kombinaciona mreza koja na osnovu trenutnog stanja
  -- i vrednosti ulaza generise kod narednog stanja i
  -- vrednost izlaznog vektora
  PROCESS (iW, sSTANJE) BEGIN
    CASE sSTANJE IS
      WHEN S0 => -- stanje S0
        CASE iW IS
          WHEN W0 => sSLEDECE_STANJE <= S2; oZ <= Z0;
          WHEN W1 => sSLEDECE_STANJE <= S2; oZ <= Z1;
          WHEN W2 => sSLEDECE_STANJE <= S1; oZ <= Z1;
          WHEN W3 => sSLEDECE_STANJE <= S0; oZ <= Z0;
        END CASE;
      WHEN S1 => -- stanje S1
        CASE iW IS
          WHEN W0 => sSLEDECE_STANJE <= S1; oZ <= Z0;
          WHEN W1 => sSLEDECE_STANJE <= S0; oZ <= Z1;
          WHEN W2 => sSLEDECE_STANJE <= S2; oZ <= Z0;
          WHEN W3 => sSLEDECE_STANJE <= S1; oZ <= Z1;
        END CASE;
      WHEN S2 => -- stanje S2
        CASE iW IS
          WHEN W0 => sSLEDECE_STANJE <= S2; oZ <= Z0;
          WHEN W1 => sSLEDECE_STANJE <= S0; oZ <= Z1;
          WHEN W2 => sSLEDECE_STANJE <= S1; oZ <= Z0;
          WHEN W3 => sSLEDECE_STANJE <= S1; oZ <= Z0;
        END CASE;
      WHEN OTHERS => -- stanje S3: nedefinisano u automatu
         sSLEDECE_STANJE <= S2; oZ <= Z0; -- pocetno stanje
    END CASE;
  END PROCESS;
```

```
    -- flip-flopovi za smestanje koda
    -- trenutnog stanja; postavljanje pocetnog
    -- stanja je sinhrono sa signalom takta
    PROCESS (iCLK) BEGIN
      IF (iCLK'EVENT AND iCLK='1') THEN
        IF (iRESET = '0') THEN -- sinhroni reset
          sSTANJE <= S2;
        ELSE
          sSTANJE <= sSLEDECE_STANJE;
        END IF;
      END IF;
    END PROCESS;

END ARH_ZAD04;
```

U prethodnom VHDL kodu su svi signali kodovani definisanjem novog tipa signala koji odgovara Grejovom kodovanju svakog signala. Kod definisanja tipa signala za reprezentovanje stanja automata uveden je četvrti literal koji predstavlja nepostojeće stanje S_3. Ovo je bilo potrebno zbog zadatog Grejovog kodovanja stanja automata. Zbog toga je u procesu koji opisuje kombinacionu mrežu za određivanje sledećeg stanja uveden iskaz `WHEN OTHERS` koji je aktivan u slučaju da se automat nađe u tom stanju, npr po uključenju napajanja. U tom slučaju se automat prevodi u početno stanje S_2.

Vremenski odziv automata na zadati niz ulaznih simbola prikazuje Slika 4.26. Na ulaz iW se dovodi zadata sekvenca signala. Početno stanje automata je S_2*=11. Kada se na ulaz dovede W_3=10, tada na rastuću ivicu takta automat prelazi u novo stanje, S_1*=01, a na izlazu je Z_0=00. Slično je i sa ostalim ulaznim signalima.



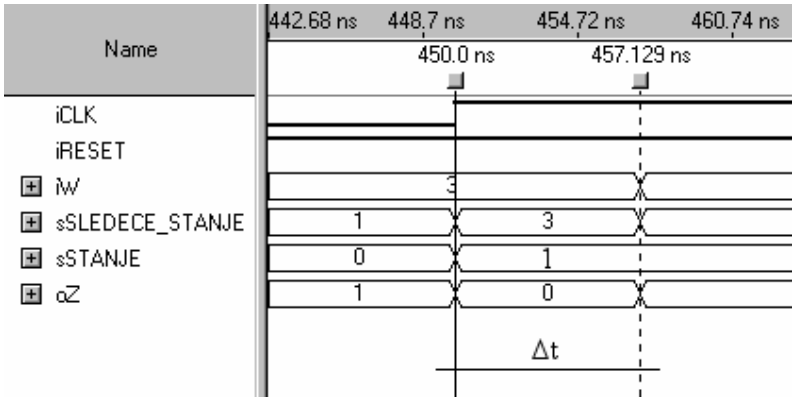
Slika 4.26: Vremenski odziv automata na zadati niz ulaznih simbola

Kratkotrajni »gličevi« koji se javljaju na izlaznom vektoru (`oZ`) i vektoru koji definiše sledeće stanje automata (`sSLEDECE_STANJE`) su posledica intervala Δt za koji ulazni simbol (`iW`) zadržava vrednost iza rastuće ivice takt signala. Ovi »gličevi« se javljaju na izlazu kombinacionih mreža koje kao ulazne vrednosti imaju trenutno stanje automata (`sSTANJE`) i ulazne simbole automata, zbog promene trenutnog stanja nakon rastuće ivice takt signala.

Slika 4.27 prikazuje jednu takvu situaciju.

U vremenskom trenutku kada se pojavljuje rastuća ivica takt signala `iCLK`, t=450ns, na ulaznom vektoru se nalazi vrednost 3 i automat prelazi u stanje 1. U

trenutku kada se pojavljuje rastuća ivica takt signala, ne sme biti promene vrednosti ulaznog vektora, radi pravilnog očitavanja njegove vrednosti u svim tačkama digitalnog sistema (u ovom slučaju automata) gde je potrebno. Zbog toga se vrednost ulaznog vektora zadržava za vremenski interval Δt, u ovom slučaju Δt=7,129ns, iza rastuće ivice takt signala, i tek posle se menja njegova vrednost. Zbog promene stanja nakon rastuće ivice takta i produžavanja vrednosti ulaznog simbola, kombinacione mreže koje za ulaze imaju ove dve vrednosti u međuvremenu reaguju i menjaju vrednost na njihovom izlazu, što dovodi do pojavljivanja „gličeva“.



Slika 4.27: Prikaz intervala Δt

Vremenski dijagrami, Slika 4.26 i Slika 4.27, u rešenju ovog zadatka su kreirani uz pomoć Altera Quartus II programskog paketa verzija 2.2.

4.5 ZADATAK:

Murov automat je zadat tablicom prelaza/izlaza, Tabela 4.17.

	1/α	**2/α**	**3/β**	**4/γ**	**5/β**	**6/γ**	**7/α**	**8/β**
$\mathbf{X_1}$	2	1	6	8	5	8	3	4
$\mathbf{X_2}$	7	7	2	1	3	2	4	1
$\mathbf{X_3}$	8	3	4	7	6	7	8	6

Tabela 4.17: Tablica prelaza/izlaza automata

Potrebno je:

a) Pronaći odgovarajući ekvivalentni minimalni automat. Nacrtati graf minimalnog automata
b) Izvršiti sintezu minimalnog automata u VHDL jeziku za opis fizičke arhitekture.

Prilikom sinteze automata stanja, ulazne i izlazne signale kodovati prirodnim BCD kodom. Početno stanje automata je stanje S_1.

REŠENJE:

U postupku minimizacije Murovih automata može se primeniti ista tehnika koja je korišćena u prethodnim zadacima za minimizaciju Milijevih automata.

U ovom rešenju će se demonstrirati drugi postupak minimizacije koji koristi specifičnost Murovih automata da njihov izlaz zavisi samo od trenutnog stanja automata.

a)

Pošto kod Murovog automata signali na izlazu zavise samo od trenutnog stanja automata, on se zadaje takvom tablicom prelaza u kojoj se pored svakog stanja Si dodaje i njemu odgovarajući izlazni signal $Y_i=f(S_i)$.

Postupak pronalaženja minimalnog automata započinje sa pregrupisavanjem stanja Murovog automata, tako da stanja koja imaju isti izlaz budu jedno pored drugog.

Ako se sa **a** označi grupa svih stanja sa izlazom α; sa **b** onih stanja sa izlazom β i sa **c** stanja sa izlazom γ, može se formirati nova pregrupisana tabela prelaza/izlaza Murovog automata, Tabela 4.18.

grupa	**a/α**			**b/β**			**c/γ**	
stanje	**1**	**2**	**7**	**3**	**5**	**8**	**4**	**6**
$\mathbf{X_1}$	2(a)	1(a)	3(b)	6(c)	5(b)	4(c)	8(b)	8(b)
$\mathbf{X_2}$	7(a)	7(a)	4(c)	2(a)	3(b)	1(a)	1(a)	2(a)
$\mathbf{X_3}$	8(b)	3(b)	8(b)	4(c)	6(c)	6(c)	7(a)	7(a)
	a_1		b_1	c_1	d_1	c_1	e_1	

Tabela 4.18: Pregrupisana tabela zadatog automata

U prethodnoj tabeli je pored svakog sledećeg stanja u zagradi označeno kojoj novoformiranoj grupi stanja ono pripada. Uz pomoć ove dodatne oznake se može uočiti koja stanja za iste ulaze prelaze u istu grupu stanja. Tako na primer stanja 1 i 2 za ulaze X_1, X_2 i X_3 prelaze redom u grupe stanja označene sa (a), (a) i (b). Pošto oba stanja pripadaju istoj grupi, tj. imaju istu vrednost izlaza, i imaju iste prelaze stanja ova dva stanja formiraju klasu ekvivalentnih stanja i mogu se zameniti sa jednim stanjem, označenim sa a_1.

Daljom pretragom tabele formiraju se klase ekvivalentnih stanja označene sa (a_1...e_1). Tabela 4.19 prikazuje tabelu prelaza/izlaza zadatog Murovog automata sa označenim klasama ekvivalentnih stanja. Kao i u prethodnom koraku u tabeli su pored svakog sledećeg stanja u zagradama označene klase ekvivalencije kojima dato stanje pripada.

klasa	**a_1/α**		**b_1/α**	**c_1/β**		**d_1/β**	**e_1/γ**	
stanje	**1**	**2**	**7**	**3**	**8**	**5**	**4**	**6**
$\mathbf{X_1}$	$2(a_1)$	$1(a_1)$	$3(c_1)$	$6(e_1)$	$4(e_1)$	$5(d_1)$	$8(c_1)$	$8(c_1)$
$\mathbf{X_2}$	$7(b_1)$	$7(b_1)$	$4(e_1)$	$2(a_1)$	$1(a_1)$	$3(c_1)$	$1(a_1)$	$2(a_1)$
$\mathbf{X_3}$	$8(c_1)$	$3(c_1)$	$8(c_1)$	$4(e_1)$	$6(e_1)$	$6(e_1)$	$7(b_1)$	$7(b_1)$

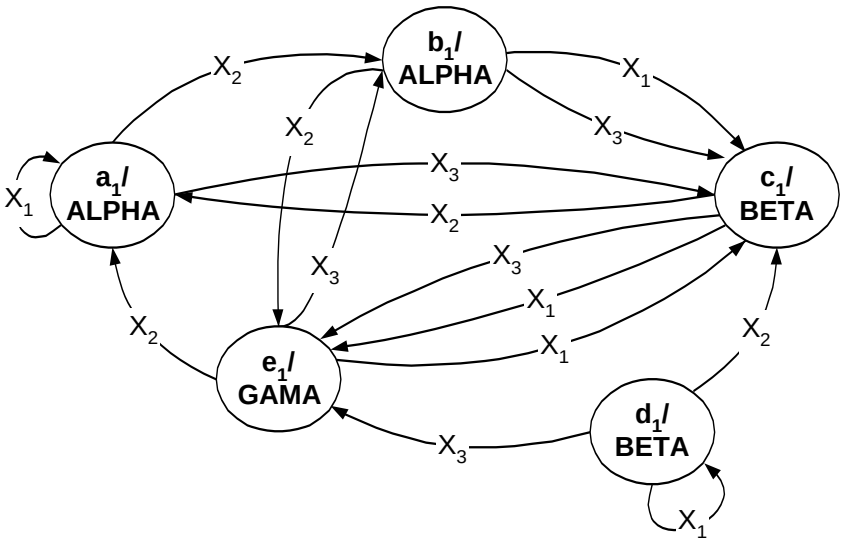
Tabela 4.19: Tabela prelaza/izlaza Murovog automata

Ako se ponovo posmatraju kolone stanja koja pripadaju istim grupama stanja i koja će dati jednake izlaze, može se uočiti da su među sobom jednake (te se mogu zameniti samo jednom od njih) prva i druga kolona; četvrta i peta; sedma i osma. Ponovnim pretraživanjem se uočava da je tako nastali Murov automat minimalan. Sada se formira tabela minimalnog automata. U njoj će se umesto broja stanja samo naznačiti grupa kojoj to stanje pripada. (Tabela 4.20).

	a_1/α	**b_1/α**	**c_1/β**	**d_1/β**	**e_1/γ**
$\mathbf{X_1}$	a_1	c_1	e_1	d_1	e_1
$\mathbf{X_2}$	b_1	e_1	a_1	c_1	a_1
$\mathbf{X_3}$	c_1	c_1	e_1	e_1	b_1

Tabela 4.20: Tabela prelaza/izlaza minimalnog automata

Na osnovu dobijene tabele prelaza/izlaza minimalnog automata formira se graf prelazaka stanja u zavisnosti od ulaznog simbola, Slika 4.28. Na grafu je u okviru simbola stanja označena i vrednost izlaza koji se generiše u tom stanju, dok su na strelicama koje označavaju prelaske stanja označene vrednosti ulaznih simbola za koje se događaju određeni prelasci stanja.



Slika 4.28: Graf prelazaka stanja minimalnog Murovog automata

b)

VHDL kod minimalnog Murovog automata sledi direktno iz dobijene tabele prelaz/izlaza ili na osnovu formiranog grafa prelaza stanja. Za razliku od VHDL opisa Milijevih automata, ovde je potrebno razdvojiti opis kombinacionih mreža za generisanje narednog stanja i izlaza. To je zbog toga što vrednost izlaza zavisi samo od trenutnog stanja, a ne i od vrednosti ulaza. Stoga u navedenom VHDL kodu postoje tri procesa:

1. Proces koji opisuje kombinacionu mrežu za generisanje narednog stanja na osnovu trenutnog stanja i vrednosti ulaza.
2. Proces koji opisuje kombinacionu mrežu za generisanje vrednosti izlaza na osnovu trenutnog stanja.
3. Proces koji opisuje flip-flopove za memorisanje trenutnog stanja automata.

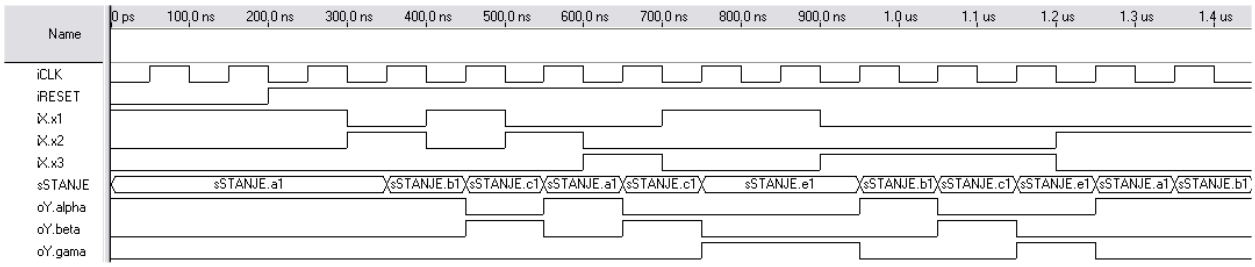
```
-----------------------------------------------------------------
--  Pakovanje sa opisom novoformiranih tipova
--  koji se koriste u realizaciji
--     1. tSTANJA - stanja koja moze da ima automat
--     2. tULAZI  - vrednosti ulaza koja moze da ima automat
--     3. tIZLAZI - vrednosti izlaza koje automat moze da generise
-----------------------------------------------------------------
PACKAGE TIPOVI IS
  -- A1="000"; B1="001"; C1="010"; D1="011"; E1="100";
  TYPE tSTANJA IS (A1, B1, C1, D1, E1);
  -- X1="00"; X2="01"; X3="11";
  TYPE tULAZI  IS (X1, X2, X3);
  -- ALPHA="00"; BETA="01; GAMA="10";
  TYPE tIZLAZI IS (ALPHA, BETA, GAMA);
END TIPOVI;

LIBRARY ieee;
  USE ieee.std_logic_1164.all;
  USE work.TIPOVI.all;

ENTITY ZAD05 IS PORT (
  iCLK:   IN  std_logic;
  iRESET: IN  std_logic;
  iX:     IN  tULAZI;
  oY:     OUT tIZLAZI
);
END ZAD05;

ARCHITECTURE ARH_ZAD05 OF ZAD05 IS
  -- deklaracija signala koji sadrze informaciju o
  -- tekucem i sledecem (narednom) stanju
  SIGNAL sSTANJE, sSLEDECE_STANJE : tSTANJA;
BEGIN

  -- kombinaciona mreza koja na osnovu trenutnog stanja
  -- i vrednosti ulaza generise kod narednog stanja
  PROCESS (iX, sSTANJE) BEGIN
    CASE sSTANJE IS
      WHEN A1 =>
        CASE iX IS
          WHEN X1 => sSLEDECE_STANJE <= A1;
```

```
            WHEN X2 => sSLEDECE_STANJE <= B1;
            WHEN X3 => sSLEDECE_STANJE <= C1;
          END CASE;
        WHEN B1 =>
          CASE iX IS
            WHEN X1 => sSLEDECE_STANJE <= C1;
            WHEN X2 => sSLEDECE_STANJE <= E1;
            WHEN X3 => sSLEDECE_STANJE <= C1;
          END CASE;
        WHEN C1 =>
          CASE iX IS
            WHEN X1 => sSLEDECE_STANJE <= E1;
            WHEN X2 => sSLEDECE_STANJE <= A1;
            WHEN X3 => sSLEDECE_STANJE <= E1;
          END CASE;
        WHEN D1 =>
          CASE iX IS
            WHEN X1 => sSLEDECE_STANJE <= D1;
            WHEN X2 => sSLEDECE_STANJE <= C1;
            WHEN X3 => sSLEDECE_STANJE <= E1;
          END CASE;
        WHEN E1 =>
          CASE iX IS
            WHEN X1 => sSLEDECE_STANJE <= E1;
            WHEN X2 => sSLEDECE_STANJE <= A1;
            WHEN X3 => sSLEDECE_STANJE <= B1;
          END CASE;
      END CASE;
    END PROCESS;

    -- kombinaciona mreza za odredjivanje vrednosti
    -- izlaznog vektora na osnovu trenutnog stanja
    PROCESS (sSTANJE) BEGIN
      CASE sSTANJE IS
        WHEN A1 => oY <= ALPHA;
        WHEN B1 => oY <= ALPHA;
        WHEN C1 => oY <= BETA;
        WHEN D1 => oY <= BETA;
        WHEN E1 => oY <= GAMA;
      END CASE;
    END PROCESS;

    -- flip-flopovi za smestanje koda
    -- trenutnog stanja; postavljanje pocetnog
    -- stanja je sinhrono sa signalom takta
    PROCESS (iCLK) BEGIN
      IF (iCLK'EVENT AND iCLK='1') THEN
        IF (iRESET = '0') THEN -- sinhroni reset
          sSTANJE <= A1;
        ELSE
          sSTANJE <= sSLEDECE_STANJE;
        END IF;
      END IF;
    END PROCESS;
  END ARH_ZAD05;
```

Slika 4.29 prikazuje vremenski odziv minimalnog automata na određeni niz simbola. Prelaske stanja na simulirani niz ulaznih simbola zajedno sa vrednošću izlaza prikazuje Tabela 4.21.

	X_3	X_2	X_1	X_2	X_3	X_1	X_1	X_3	X_3	X_3	X_2	X_2	X_2
a_1	a_1	b_1	c_1	a_1	c_1	e_1	e_1	b_1	c_1	e_1	a_1	b_1	e_1
	α	α	β	α	β	γ	γ	α	β	γ	α	α	γ

Tabela 4.21: Niz ulaznih simbola koji prikazuje simulacija rada automata



Slika 4.29: Simulacija rada minimalnog automata

4.6 ZADATAK:

Lift opslužuje dva sprata. Na svakom spratu ima dugme za poziv, a u kabini postoje dva dugmeta za odabiranje željenog sprata. Nacrtati graf i izvršiti sintezu konačnog automata koji opisuje lift imajući u vidu da:

a) Pritisak na svako dugme predstavlja poseban ulazni signal
b) Izlazne signale predstavlja rad ili mirovanje motora
c) Stanje automata predstavlja mirovanje lifta na nekom spratu

Qn	Qn+1	T
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 4.22: Prelasci stanja T flip-flopa

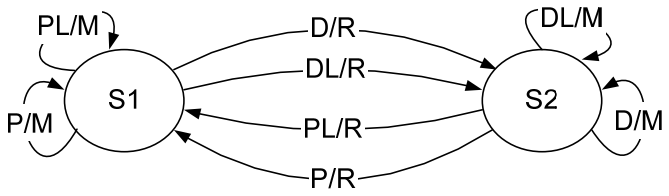
U sintezi automata kao memorijske elemente koristiti T flip-flopove i standardna logička kola. Tabela 4.22 prikazuje prelaske stanja T flip-flopa. Izlazne signale baferovati sa D flip-flopom. Sve signale kodovati sa Grejovim kodom.

REŠENJE:

Na osnovu teksta zadatka mogu se formirati odgovarajući ulazi, stanja i izlazi automata za opsluživanje lifta.

ULAZI	STANJA	IZLAZI
P - dugme na prvom spratu	S1 - prvi sprat	R - motor radi
PL - dugme u liftu na prvom spratu	S2 - drugi sprat	M - motor miruje
D - dugme na drugom spratu		
DL - dugme na u liftu na drugom spratu		

Zatim se može nacrtati graf odgovarajućeg automata što prikazuje Slika 4.30.



Slika 4.30: Graf prelazaka stanja lifta

Automat prikazan prethodnim grafom može se prikazati i u formi tabele prelaza/izlaza (Tabela 4.23).

	S1	S2
P	S1/M	S1/R
PL	S1/M	S1/R
D	S2/R	S2/M
DL	S2/R	S2/M

Tabela 4.23: Tabela prelaza/izlaza

Može se uočiti da je ovde reč o potpuno definisanom automatu.

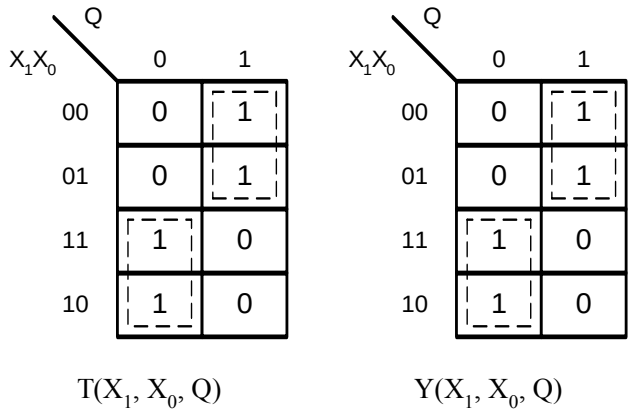
U daljem postupku pristupa se minimizaciji automata. Prvo je potrebno izvršiti kodovanje stanja, ulaza i izlaza. Na osnovu zadatog načina kodiranja pomoću Grejovog koda formira se kodirana tabela prelaza/izlaza (Tabela 4.24):

Ulazi → X_1X_0	Stanja → Q	Izlazi → Y
P → 0 0	S1 → 0	M → 0
PL → 0 1	S2 → 1	R → 1
D → 1 1		
DL → 1 0		

	0	1
00	0/0	0/1
01	0/0	0/1
11	1/1	1/0
10	1/1	1/0

Tabela 4.24: Kodirana tabela prelaza/izlaza

Uz pomoć ovako dobijene tabele može se pristupiti crtanju Karnoovih karti za funkciju prelaza (T) i funkciju izlaza (Y), Slika 4.31. Prilikom popunjavanja Karnoove karte za minimizaciju funkcije prelaza potrebno je obratiti pažnju na tabelu prelaza T flip-flopa.



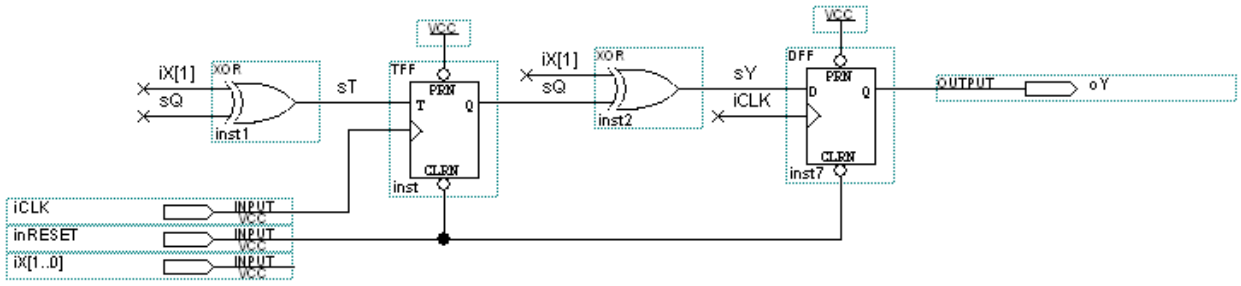
Slika 4.31: Karnoove karte

Minimizacijom uz pomoć Karnoovih karti dobijaju se sledeće funkcije:

$$T = \overline{X_1} \cdot Q + X_1 \cdot \overline{Q} = X_1 \oplus Q$$

$$Y = \overline{X_1} \cdot Q + X_1 \cdot \overline{Q} = X_1 \oplus Q$$

Na osnovu dobijenih funkcija, pristupa se formiranju šeme logičkih elemenata traženog automata. To prikazuje Slika 4.32.



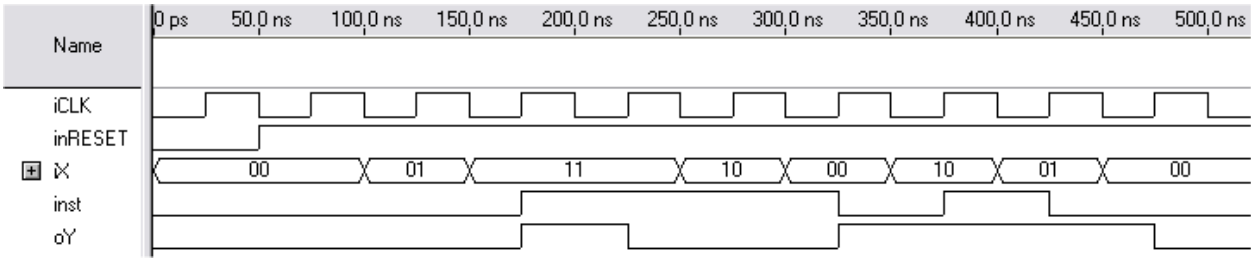
Slika 4.32: Logička šema automata za opsluživanje lifta

Ispravnost rada automata iz ovog zadatka proverava se uz pomoć digitalne simulacije svih prelazaka stanja realizovanog automata. Simulirane prelaska stanja i dobijene vrednosti izlaza prikazuje Tabela 4.25.

Ulazi		P	PL	D	D	DL	P	DL	PL
Stanja	S1	S1	S1	S2	S2	S2	S1	S2	S1
Izlazi		M	M	R	M	M	R	R	R

Tabela 4.25: Tabela simuliranih prelaza/izlaza

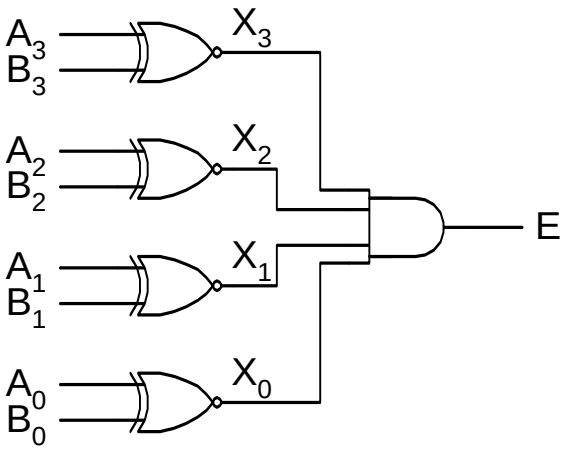
Slika 4.33 prikazuje digitalnu simulacija sintetizovanog automata za opsluživanje lifta.



Slika 4.33: Simulacija rada realizovanog automata

4.7 ZADATAK:

Potrebno je kolo za određivanje jednakosti dva četvorobitna broja koje prikazuje Slika 4.34, zameniti digitalnim sistemom koji serijski ispituje pojedinačne parove bita { (A_0, B_0), (A_1, B_1), (A_2, B_2) i (A_3, B_3) }.



Slika 4.34: Kolo za određivanje jednakosti dva četvorobitna broja

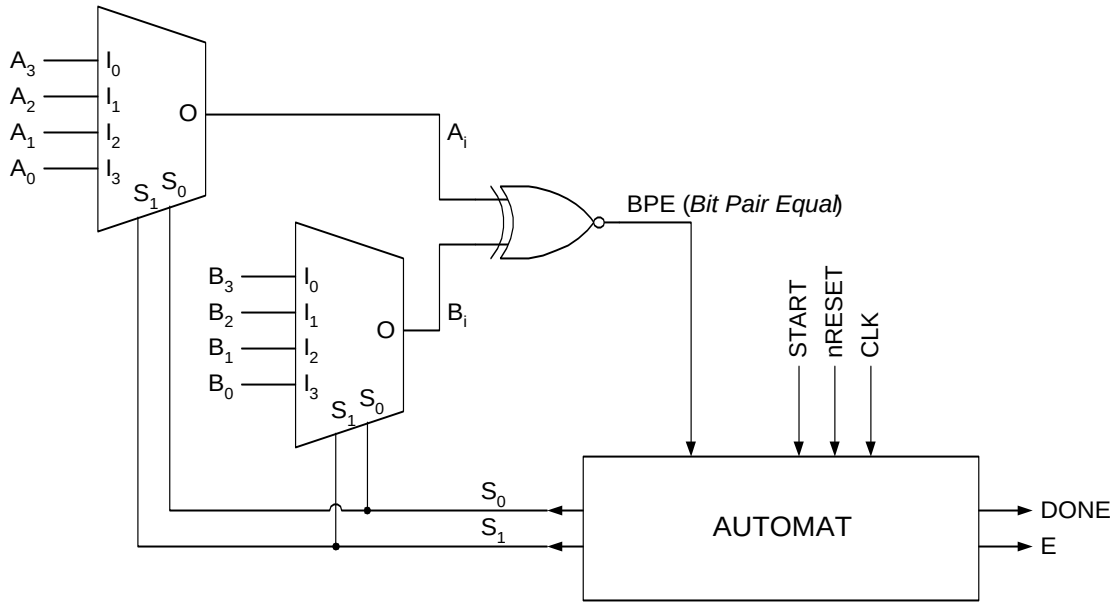
Blok šemu ciljnog digitalnog sistema prikazuje Slika 4.35. U ovom rešenju se za odabir parova bita koriste dva multipleksera 4×1, dok je upravljanje radom sistema kontrolisano automatom čiji dijagram prelazaka stanja prikazuje Slika 4.36.

Potrebno je definisati prelaze stanja automata i implementirati digitalni sistem u VHDL jeziku za opis fizičke arhitekture.

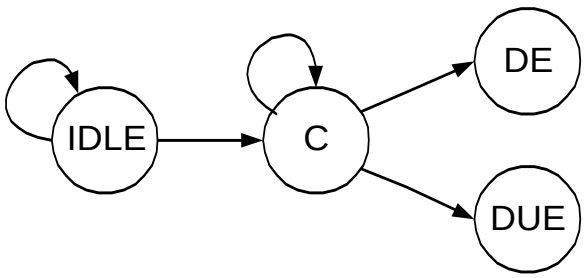
POJAŠNJENJE:

Početno stanje automata je stanje IDLE gde se automat nalazi sve dok START signal nije aktivan. Nakon aktiviranja START signala, START=1, automat prelazi u stanje C (*Compute*). U ovom stanju ostaje najviše 4 takta radi poređenja sva četiri para bita. Iz stanja C automat prelazi u stanje DE (*Done-Equal*) ako su

svi parovi bita jednaki, ili u stanje DUE (*Done-UnEqual*) ako bilo koji par bita nije jednak. Kada automat jednom uđe u stanje DE ili DUE ostaje u njemu do trenutka inicijalizacije automata. U oba stanja (DE i DUE) automat generiše DONE signal (DONE=1). U stanju DE generiše se signal E=1, dok je u stanju DUE njegova vrednost jednaka nuli (E=0).



Slika 4.35: Blok dijagram digitalnog sistema za određivanje jednakosti dva četvorobitna broja



Slika 4.36: Dijagram prelazaka stanja automata

Potrebno je obratiti pažnju da automat nemora da poredi preostale bite ako se ustanovi jedan par ne jednakih bita.

Za brojanje iteracija iskoristiti dvobitni brojač sa signalom dozvole brojanja koji generiše signal identifikacije kraja ciklusa brojanja. Vrednost brojača se briše ulaskom u stanje IDLE, dok se u stanju C brojač uvećava za jedan. Izlaz brojača se može iskoristiti za kontrolu adresnih ulaza multipleksera.

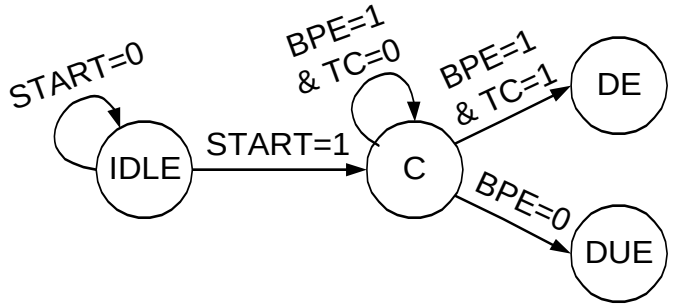
REŠENJE:

Sa ciljem jednostavnije realizacije automata za upravljanje adresnim ulazima multipleksera iskoristiti će se dvobitni brojač. Inicijalno stanje brojača je 0. Ukoliko je aktivan signal dozvole brojanja (`sCE`), brojač broji na gore. Vrednost

brojača direktno adresira adresne ulaze oba multipleksera, dok automat upravlja radom brojača generisanjem signala dozvole brojanja. Za indikaciju kraja ciklusa brojanja, čime se ukazuje da su upoređena sva četiri para bita, brojač generiše signal `sTC` koji predstavlja ulaz u automat.

Sa ovom izmenom u arhitekturi digitalnog sistema automat treba da prelazi iz stanja u stanje na sledeći način. Inicijalno stanje automata je stanje `IDLE` gde se automat nalazi sve dok `iSTART` signal nije aktivan (`iSTART=0`). Nakon aktiviranja `iSTART` signala, `iSTART=1`, automat prelazi u stanje `C`. U ovom stanju automat ostaje sve dok su poređeni biti jenaki (signal `sBPE` je jednak jedinici) i dok se ne stigne do kraja ciklusa brojanja (signal `sTC` jednak nuli). Ukoliko se stiglo do kraja ciklusa brojanja (signal `sTC` jednak jedinici) i poslednji poređeni par bita je jednak (signal `sBPE` je jednak jedinici), automat iz stanja `C` prelazi u stanje `DE`. Ukoliko se u bilo kom trenutku u stanju `C` detektuje da su poređeni biti različiti (signal `sBPE` je jednak nuli) automat odmah prelazi u stanje `DUE`. Slika 4.37 prikazuje graf prelazaka stanja automata sa definisanim prelazima stanja.

Automat generiše tri izlazna signala: `sCE` za upravljanje radom brojača, `oE` koji ukazuje da li su poređene četvorobitne reči jednake i `oDONE` koji ukazuje da li je završena procedura poređenja četvorobitnih reči. Ovi izlazni signali zavise samo od trenutnog stanja, pa se automat realizuje kao Murov automat.



Slika 4.37: Označeni prelasci stanja na dijagramu

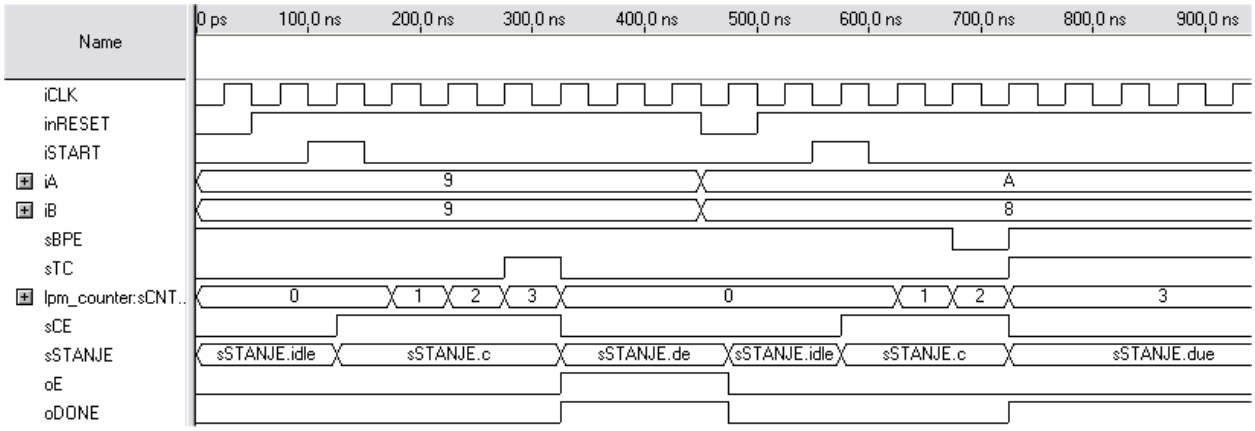
Na osnovu navedenih informacija sledi VHDL implementacija traženog digitalnog sistema. Automat i brojač su realizovani sa sinhronim postavljanjem u inicijalno stanje ulaznim signalom `inRESET`. Odabrani biti četvorobitnih reči koji se porede (izlazi multipleksera) su predstavljeni signali `sA_BIT` i `sB_BIT`. Njihove vrednosti se porede sa jednim XNOR kolom koje generiše signal `sBPE` (*Bit Pair Equal*). Adresiranje bita je realizovano tako da poređenje kreće od bita najveće važnosti. Pošto je inicijalno stanje brojača nula, na ulaze multipleksera sa adresom nula su dovedeni biti `iA(3)` i `iB(3)`. Dalje redom slede biti sve do bita sa težinom 0, tj. `iA(0)` i `iB(0)`, koji su dovedeni na ulaze multipleksera sa adresom 3.

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;
  USE ieee.numeric_std.all;

ENTITY KOMPARATOR IS PORT (
  iCLK:      IN  STD_LOGIC;
  inRESET:   IN  STD_LOGIC;
  iSTART:    IN  STD_LOGIC;
  iA, iB:    IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
  oE, oDONE: OUT STD_LOGIC );
END KOMPARATOR;

ARCHITECTURE ARH_KOMPARATOR OF KOMPARATOR IS
  -- tip koji predstavlja sva stanja automata za poredjenje brojeva
  TYPE tSTANJA IS (IDLE, C, DE, DUE);
  -- deklaracija signala koji sadrze informaciju o
  -- tekucem i sledecem (narednom) stanju
  SIGNAL sSTANJE, sSLEDECE_STANJE : tSTANJA;
  -- brojac ciklusa (jedan ciklus <=> poredjenje jednog bita)
  SIGNAL sCNT: UNSIGNED(1 DOWNTO 0);
  -- indikacija kraja ciklusa poredjenja
  SIGNAL sTC: STD_LOGIC;
  -- dozvola brojanja brojaca
  SIGNAL sCE: STD_LOGIC;
  -- indikacija da li su dva bita jednaka
  SIGNAL sBPE: STD_LOGIC;
  -- pojedinacni biti ulaznih parametara
  SIGNAL sA_BIT, sB_BIT: STD_LOGIC;
BEGIN

  -- multiplekser za biranje bita prvog parametra
  PROCESS (iA, sCNT) BEGIN
    CASE sCNT IS
      WHEN "00"   => sA_BIT <= iA(3);
      WHEN "01"   => sA_BIT <= iA(2);
      WHEN "10"   => sA_BIT <= iA(1);
      WHEN OTHERS => sA_BIT <= iA(0);
    END CASE;
  END PROCESS;

  -- multiplekser za biranje bita drugog parametra
  PROCESS (iB, sCNT) BEGIN
    CASE sCNT IS
      WHEN "00"   => sB_BIT <= iB(3);
      WHEN "01"   => sB_BIT <= iB(2);
      WHEN "10"   => sB_BIT <= iB(1);
      WHEN OTHERS => sB_BIT <= iB(0);
    END CASE;
  END PROCESS;

  sBPE <= sA_BIT XNOR sB_BIT; -- provera jednakosti dva bita

  -- brojac ciklusa
  PROCESS (iCLK) BEGIN
    IF (iCLK'EVENT AND iCLK='1') THEN
      IF (inRESET='0') THEN  -- sinhroni reset
        sCNT <= "00";
      ELSE
```

```
      IF (sCE='1') THEN
        sCNT <= sCNT + 1;
      END IF;
    END IF;
  END IF;
END PROCESS;

-- provera kraja ciklusa
PROCESS (sCNT) BEGIN
  IF (sCNT="11") THEN
    sTC <= '1';
  ELSE
    sTC <= '0';
  END IF;
END PROCESS;

-------------
-- AUTOMAT --
-------------
-- kombinaciona mreza koja na osnovu trenutnog stanja
-- i ulaznih signala automata generise kod narednog stanja
PROCESS (iSTART, sBPE, sTC, sSTANJE) BEGIN
  CASE sSTANJE IS
    WHEN IDLE =>
      IF (iSTART='1') THEN
        sSLEDECE_STANJE <= C;
      ELSE
        sSLEDECE_STANJE <= IDLE;
      END IF;
    WHEN C    =>
      IF ((sTC='0') AND (sBPE='1')) THEN
        -- nije kraj ciklusa i poredjeni biti su jednaki
        sSLEDECE_STANJE <= C;
      ELSIF ((sTC='1') AND (sBPE='1')) THEN
        -- kraj ciklusa i poredjeni biti su jednaki
        sSLEDECE_STANJE <= DE;
      ELSE
        -- poredjeni biti su razliciti
        sSLEDECE_STANJE <= DUE;
      END IF;
    WHEN DE   =>
      sSLEDECE_STANJE <= DE;
    WHEN DUE  =>
     sSLEDECE_STANJE <= DUE;
  END CASE;
END PROCESS;

-- flip-flopovi za smestanje koda trenutnog stanja
PROCESS (iCLK) BEGIN
  IF (iCLK'EVENT AND iCLK='1') THEN
    IF (inRESET = '0') THEN -- sinhroni reset
      sSTANJE <= IDLE;
    ELSE
      sSTANJE <= sSLEDECE_STANJE;
    END IF;
  END IF;
END PROCESS;
```

```
    -- kombinaciona mreza za generisanje izlaza automata
    PROCESS (sSTANJE) BEGIN
      CASE sSTANJE IS
        WHEN IDLE => sCE   <= '0';
                     oE    <= '0';
                     oDONE <= '0';
        WHEN C    => sCE   <= '1';
                     oE    <= '0';
                     oDONE <= '0';
        WHEN DE   => sCE   <= '0';
                     oE    <= '1';
                     oDONE <= '1';
        WHEN DUE  => sCE   <= '0';
                     oE    <= '0';
                     oDONE <= '1';
      END CASE;
    END PROCESS;

END ARH_KOMPARATOR;
```

Slika 4.38 prikazuje simulaciju rada realizovanog digitalnog sistema za dva slučaja. Prvo je prikazan slučaj kada su ulazni brojevi jednaki ($A=B=9_{HEX}$). Aktiviranjem signala `iSTART` u trenutku t=100ns započinje procedura poređenja. Sa prvom rastućum ivicom takt signala `iCLK` automat prelazi iz stanja `IDLE` (kod 0) u stanje `C` (kod 1). Sledi poređenje parova bita počev od para bita `iA(3)` i `iB(3)`. Sa svakom rastućom ivicom takt signala, vrednost brojača `sCNT` se uvećava za jedan pošto je u stanju `C` aktivan signal dozvole brojanja `sCE`. U poslednjem ciklusu poređenja, t=275ns÷325ns, poredi se par bita `iA(0)` i `iB(0)`. Takođe je aktivan i signal indikacije kraja ciklusa brojanja `sTC`. Pošto su biti jednaki, `sBPE=1`, automat prelazi u stanje `DE` (kod 2) gde generiše izlazne signale `oE` i `oDONE`. Celokupan digitalni sistem ostaje u stanju `DE` sve do aktiviranja signala `inRESET` čime se automat postavlja u inicijalno stanje.



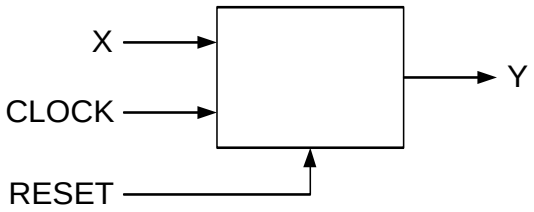
Slika 4.38: Simulacija rada realizovanog digitalnog sistema

Drugi slučaj prikazuje poređenje brojava `iA`=A_{HEX} i `iB`=8_{HEX}. Aktiviranjem iSTART signala, t=550ns, započinje poređenje parova bita počev od bita sa

najvećom težinom. Razlika ulaznih brojeva je detektovana na paru bita `iA(1)=1` i `iB(1)=0`. To se desilo u vremenskom trenutku t=700ns gde je signal `sBPE` primio vrednost nula. Iz tog razloga automat odmah sa sledećom rastućom ivicom takt signala prelazi u stanje DUE (kod 3) gde se ukazuje kraj ciklusa poređenja (`oDONE=1`) i da su ulazni brojevi različiti (`oE=0`).

4.8 ZADATAK:

Isprojektovati sinhronu sekvencijalnu mrežu (Slika 4.39) koja detektuje ulaznu sekvencu X=1101. Obezbediti pravilno detektovanje preklapajućih sekvenci.



Slika 4.39: Blok šema sekvencijalne mreže koju treba isprojektovati

Primer odnosa ulazno/izlaznih signala, gde su sa masnim slovima označeni nizovi ulaznih signala koji odgovaraju traženoj ulaznoj sekvenci:

X= 1 0 1 0 **1 1 0 1 1 0 1** 1 **1 1 0 1**
Y= 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1
——→ vreme

Sekvencijalnu mrežu isprojektovati u VHDL jeziku za opis fizičke arhitekture:

a) koristeći Milijev automat
b) koristeći Murov automat

REŠENJE:

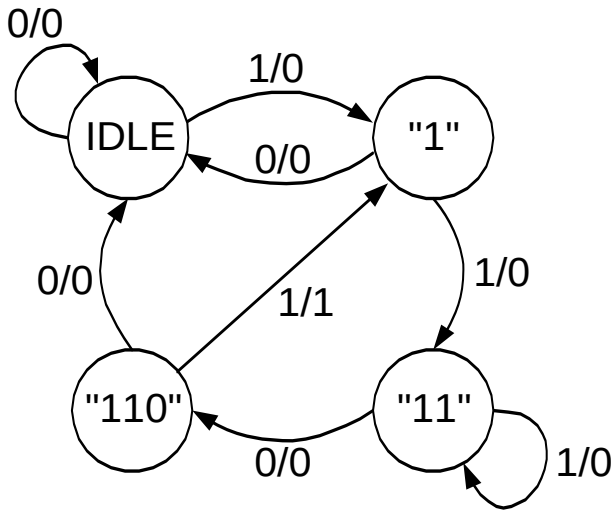
a)

Prvo se definiše automat za detekciju tražene ulazne sekvence. Postoji jedan ulazni signal (X) preko kojeg se prima niz ulaznih simbola, i jedan izlazni signal (Y) koji označava da je primljena tražena ulazna sekvenca. Potrebno je odrediti koja stanja ima automat i koji su prelasci između stanja, vodeći računa o pravilnom detektovanju ulazne sekvence uzimajući u obzir i preklapajuće sekvence.

Neka je početno stanje automata IDLE. Dok god je na ulazu prisutna vrednost 0 automat ostaje u ovom stanju i na izlazu generiše vrednost 0. Ako se na ulazu pojavi vrednost 1 automat treba da pređe u novo stanje pošto je dektovan prvi bit

tražene ulazne sekvence. Vrednost izlaza ostaje 0. Neka se novo stanje označi sa "1", čime se označava da je primljen prvi bit ulazne sekvence. U slučaju da se nakon jedinice na ulazu pojavi vrednost 0, prekinuta je tražena sekvenca i automat treba da se vrati u stanje IDLE ne menjajući vrednost izlaza (Y=0). U suprotnom slučaju, tj. ako je u sledećoj periodi takt signala takođe detektovana vrednost jedan na ulazu automata, automat treba da pređe u novo stanje koje označava da su primljena prva dva ulazna simbola tražene sekvence. Pošto su prva dva bita jednaka 1, neka se novo stanje označi sa "11". Na izlazu je još uvek prisutna vrednost 0. Ako je i sledeći detektovani ulazni simbol jednak 1, automat ne menja svoje stanje i vrednost izlaza, pošto su i dalje detektovane dve jedinice koje čine prva dva tražena ulazna simbola, bez obzira što je i pre njih na ulazu bila prisutna vrednost 1. Prijemom ulazne vrednosti 0 u stanju "11", detektuje se i treći simbol tražene ulazne sekvence pa je potrebno da automat pređe u novo stanje "110". Vrednost izlaza je ostaje 0. Kada se u narednoj periodi primi ulazna vrednost 0, tražena ulazna sekvenca je prekinuta i automat prelazi u stanje IDLE ne menjajući vrednost izlaza (Y=0). Međutim, kada se u stanju "110" na ulazu pojavi vrednost 1, primljena je kompletna ulazna sekvenca i na izlazu Y se generiše vrednost 1. Pri tome automat treba da pređe u stanje "1" pošto je nakon nule primljena prva jedinica koja predstavlja prvi ulazni simbol tražene ulazne sekvence. Na ovaj način se rešava problem preklapajućih sekvenci, gde kraj jedne sekvence istovremeno predstavlja i njen početak.

Slika 4.40 prikazuje graf prelazaka stanja prethodno opisanog Milijevog automata.



Slika 4.40: Graf prelazaka stanja Milijevog automata za detekciju ulazne sekvence X=1101

Nakon definisanja prelazaka stanja Milijevog automata za detekciju ulazne sekvence X=1101, prelazi se na formiranje odgovarajućeg VHDL koda. Opisana stanja su u VHDL kodu koji sledi označena na sledeći način:

Stanje	Oznaka	Binarni kod
IDLE	IDLE	00
"1"	S1	01
"11"	S11	10
"110"	S110	11

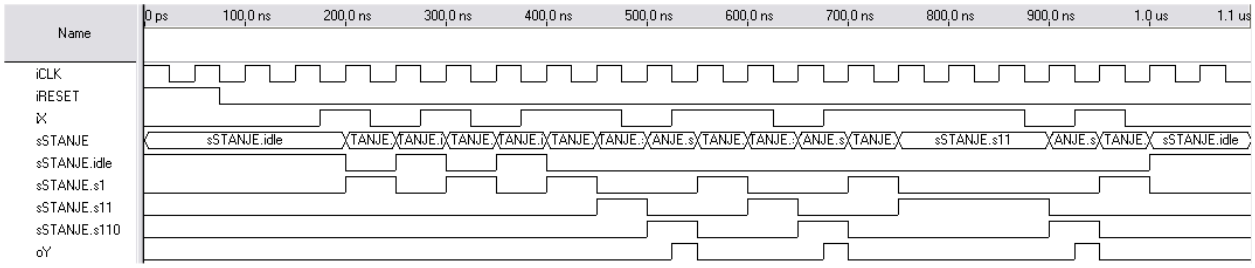
Automat je u VHDL-u realizovan sa dva procesa. U prvom procesu se na osnovu trenutnog stanja automata (sSTANJE) i vrednosti ulaznog signala iX definišu dve kombinacione mreže. Prva od njih određuje kod sledećeg stanja automata (sSLEDECE_STANJE), dok druga kombinaciona mreža definiše vrednost izlaznog signala oY.

Drugi proces opisuje memorijske elemente (flip-flopove) za pamćenje koda trenutnog stanja automata. Postavljanje početnog stanja je realizovano ulaznim signalom iRESET sinhrono sa signalom takta iCLK.

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY DETEKTOR IS PORT (
  iCLK:   IN  STD_LOGIC;
  iRESET: IN  STD_LOGIC;
  iX:     IN  STD_LOGIC;
  oY:     OUT STD_LOGIC );
END DETEKTOR;

ARCHITECTURE ARH_DETEKTOR OF DETEKTOR IS
  -- tip koji predstavlja sva stanja detektora
  TYPE tSTANJA IS (IDLE, S1, S11, S110);
  -- deklaracija signala koji sadrze informaciju o
  -- tekucem i sledecem (narednom) stanju
  SIGNAL sSTANJE, sSLEDECE_STANJE : tSTANJA;
BEGIN
  -- kombinaciona mreza koja na osnovu trenutnog stanja
  -- i vrednosti ulaza generise kod narednog stanja i
  -- vrednost izlaznog signala
  PROCESS (iX, sSTANJE) BEGIN
    CASE sSTANJE IS
      WHEN IDLE  => IF (iX='1') THEN
                     sSLEDECE_STANJE <= S1;   oY <= '0';
                    ELSE
                     sSLEDECE_STANJE <= IDLE; oY <= '0';
                    END IF;
      WHEN S1    => IF (iX='1') THEN
                     sSLEDECE_STANJE <= S11;  oY <= '0';
                    ELSE
                     sSLEDECE_STANJE <= IDLE; oY <= '0';
                    END IF;
      WHEN S11   => IF (iX='1') THEN
                     sSLEDECE_STANJE <= S11;  oY <= '0';
                    ELSE
                     sSLEDECE_STANJE <= S110; oY <= '0';
                    END IF;
```

```
        WHEN S110  => IF (iX='1') THEN
                       sSLEDECE_STANJE <= S1;   oY <= '1';
                      ELSE
                       sSLEDECE_STANJE <= IDLE; oY <= '0';
                      END IF;
      END CASE;
    END PROCESS;

    -- flip-flopovi za smestanje koda trenutnog stanja
    PROCESS (iCLK) BEGIN
      IF (iCLK'EVENT AND iCLK='1') THEN
        IF (iRESET = '1') THEN -- sinhroni reset
          sSTANJE <= IDLE;
        ELSE
          sSTANJE <= sSLEDECE_STANJE;
        END IF;
      END IF;
    END PROCESS;
  END ARH_DETEKTOR;
```

Slika 4.41 prikazuje vremenski dijagram koji ilustruje rad realizovanog automata na primeru iz teksta zadatka. Vidi se da su detektovane sve tri sekvence, uključujući i preklapajuću sekvencu.



Slika 4.41: Ilustracija rada realizovanog Milijevog automata

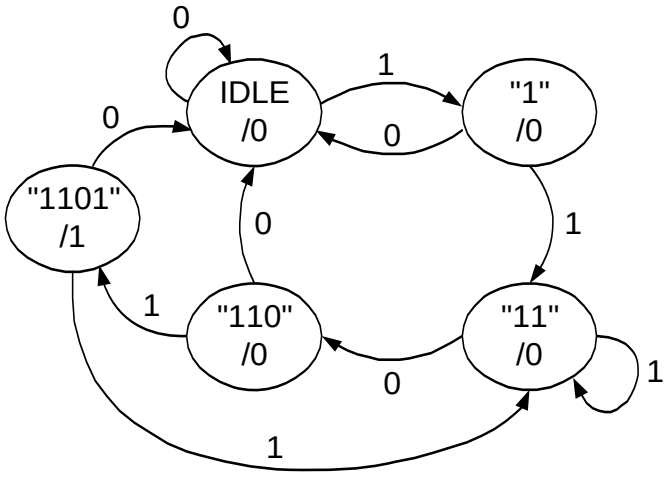
b)

Isto kao i kod realizacije Milijevog automata, i kod realizacije Murovog automata je prvo potrebno definisati stanja automata i odgovarajuće prelaske između stanja. Potrebno je napomeniti da za razliku od Milijevog automata gde vrednost izlaza zavisi od trenutnog stanja i vrednosti ulaznog signala, kod Murovog automata vrednost izlaza zavisi samo od trenutnog stanja.

Neka se i u ovom slučaju početno stanje označi sa IDLE. Vrednost izlaza u ovom stanju je 0. Dok god je na ulazu prisutna vrednost 0 automat ostaje u ovom stanju. Ako se na ulazu pojavi vrednost 1 automat treba da pređe u novo stanje pošto je dektovan prvi bit tražene ulazne sekvence. Neka se novo stanje označi sa "1. Izlazni signal i u ovom stanju ima vrednost 0. U slučaju da se nakon jedinice na ulazu pojavi vrednost 0, prekinuta je tražena sekvenca i automat treba da se vrati u stanje IDLE. U suprotnom slučaju, tj. ako je u sledećoj periodi takt signala takođe detektovana vrednost jedan na ulazu automata, automat treba da pređe u novo stanje koje označava da su primljena prva dva ulazna simbola tražene sekvence.

Neka se novo stanje označi sa "11". Na izlazu je i u ovom stanju prisutna vrednost 0. Ako je i sledeći detektovani ulazni simbol jednak 1, automat nemenja svoje stanje, pošto su i dalje detektovane dve jedinice koje čine prva dva tražena ulazna simbola. Prijemom ulazne vrednosti 0 u stanju "11", detektuje se i treći simbol tražene ulazne sekvence pa je potrebno da automat pređe u novo stanje "110". Vrednost izlaza u novom stanju ostaje 0. Kada se u narednoj periodi primi ulazna vrednost 0, tražena ulazna sekvenca je prekinuta i automat prelazi u stanje IDLE. Kada se u stanju "110" na ulazu pojavi vrednost 1 primljena je kompletna ulazna sekvenca i automat treba da pređe u novo stanje da bi na izlazu Y generisao vrednost 1. Neka se novo stanje označi sa "1101". Ako se u stanju "1101" primi vrednost jedan detektovana je preklapajuća sekvenca pa automat treba da pređe u stanje "11" pošto su na ulazu automata primljene dve uzastopne jedinice koje predstavljaju prva dva ulazna simbola tražene sekvence. Prijemom ulazne vrednosti 0 u stanju "1101" automat prelazi u stanje IDLE pošto je tražena sekvenca prekinuta.

Slika 4.42 prikazuje graf prelazaka stanja prethodno opisanog Murovog automata, gde se vidi da Murov automat ima jedno stanje više od realizovanog Milijevog automata.



Slika 4.42: Graf prelazaka stanja Murovog automata za detekciju ulazne sekvence X=1101

Nakon definisanja prelazaka stanja Murovog automata za detekciju ulazne sekvence X=1101, prelazi se na formiranje odgovarajućeg VHDL koda. Opisana stanja su u VHDL kodu koji sledi označena na sledeći način:

Stanje	Oznaka	Binarni kod
IDLE	`IDLE`	000
"1"	`S1`	001
"11"	`S11`	010
"110"	`S110`	011
"1101"	`S1101`	100

Automat je u VHDL-u realizovan sa tri procesa. U prvom procesu se na osnovu trenutnog stanja automata (`sSTANJE`) i vrednosti ulaznog signala `iX` definiše kombinaciona mreža koja određuje kod sledećeg stanja automata (`sSLEDECE_STANJE`).

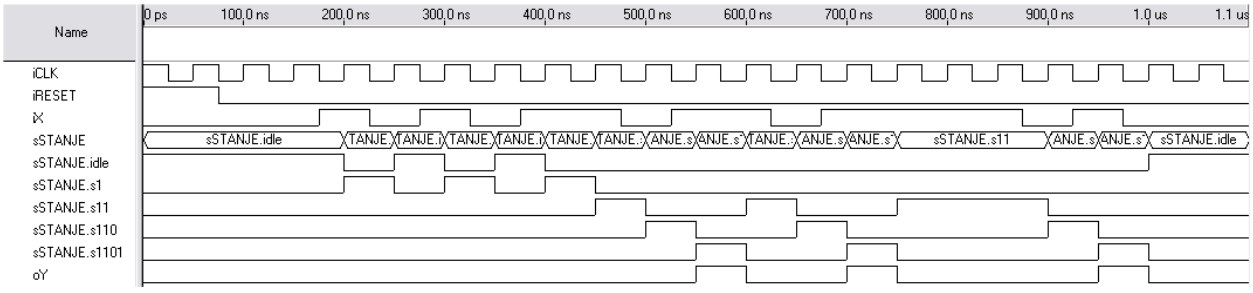
Drugi proces opisuje memorijske elemente (flip-flopove) za pamćenje koda trenutnog stanja automata. Postavljanje početnog stanja je realizovano ulaznim signalom `iRESET` sinhrono sa signalom takta `iCLK`.

Definisanje vrednosti izlaznog signala `oY` u zavisnosti od trenutnog stanja je realizovano kombinacionom mrežom u trećem procesu.

```
LIBRARY ieee;
  USE ieee.std_logic_1164.all;

ENTITY DETEKTOR IS PORT (
  iCLK:   IN  STD_LOGIC;
  iRESET: IN  STD_LOGIC;
  iX:     IN  STD_LOGIC;
  oY:     OUT STD_LOGIC
);
END DETEKTOR;

ARCHITECTURE ARH_DETEKTOR OF DETEKTOR IS
  -- tip koji predstavlja sva stanja detektora
  TYPE tSTANJA IS (IDLE, S1, S11, S110, S1101);
  -- tip koji predstavlja sva stanja detektora
  SIGNAL sSTANJE, sSLEDECE_STANJE : tSTANJA;
BEGIN

  -- kombinaciona mreza koja na osnovu trenutnog stanja
  -- i ulaznih signala automata generise kod narednog stanja
  PROCESS (iX, sSTANJE) BEGIN
    CASE sSTANJE IS
      WHEN IDLE  => IF (iX='1') THEN
                     sSLEDECE_STANJE <= S1;
                    ELSE
                     sSLEDECE_STANJE <= IDLE;
                    END IF;
      WHEN S1    => IF (iX='1') THEN
                     sSLEDECE_STANJE <= S11;
                    ELSE
                     sSLEDECE_STANJE <= IDLE;
                    END IF;
      WHEN S11   => IF (iX='1') THEN
                     sSLEDECE_STANJE <= S11;
                    ELSE
                     sSLEDECE_STANJE <= S110;
                    END IF;
      WHEN S110  => IF (iX='1') THEN
                     sSLEDECE_STANJE <= S1101;
                    ELSE
                     sSLEDECE_STANJE <= IDLE;
                    END IF;
      WHEN S1101 => IF (iX='1') THEN
                     sSLEDECE_STANJE <= S11;
```

```
                    ELSE
                     sSLEDECE_STANJE <= IDLE;
                    END IF;
      END CASE;
    END PROCESS;

    -- flip-flopovi za smestanje koda trenutnog stanja
    PROCESS (iCLK) BEGIN
      IF (iCLK'EVENT AND iCLK='1') THEN
        IF (iRESET = '1') THEN -- sinhroni reset
          sSTANJE <= IDLE;
        ELSE
          sSTANJE <= sSLEDECE_STANJE;
        END IF;
      END IF;
    END PROCESS;

    -- kombinaciona mreza za generisanje izlaza automata
    PROCESS (sSTANJE) BEGIN
      CASE sSTANJE IS
        WHEN IDLE  => oY <= '0';
        WHEN S1    => oY <= '0';
        WHEN S11   => oY <= '0';
        WHEN S110  => oY <= '0';
        WHEN S1101 => oY <= '1';
      END CASE;
    END PROCESS;

  END ARH_DETEKTOR;
```

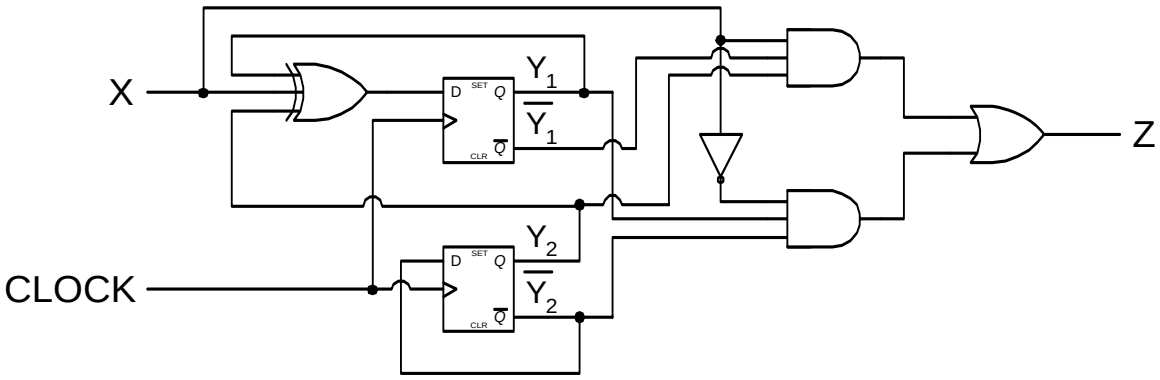
Slika 4.43 prikazuje vremenski dijagram koji ilustruje rad realizovanog automata na primeru iz teksta zadatka. Vidi se da su detektovane sve tri sekvence, uključujući i preklapajuću sekvencu.



Slika 4.43: Ilustracija rada realizovanog Murovog automata

4.9 ZADATAK:

Slika 4.44 prikazuje logičku šemu sekvencijalnog sistema sa jednim ulazom (X) i jednim izlazom (Z).



Slika 4.44: Logička šema sekvencijalnog sistema

Za dati sistem je potrebno:

- odrediti jednačine prelazaka stanja i jednačinu izlaza,
- definisati tabelu prelazaka stanja/izlaza i
- formirati graf prelazaka stanja.

REŠENJE:

Direktno sa logičke šeme se mogu odrediti jednačine prelazaka stanja, odnosno jednačine koje definišu signale na ulazima flip-flopova:

$$D_1 = X \oplus Y_1 \oplus Y_2$$

$$D_2 = \overline{Y_2}$$

Jednačina izlaza se takođe očitava direktno sa logičke šeme:

$$Z = X \cdot \overline{Y_1} \cdot Y_2 + \overline{X} \cdot Y_1 \cdot \overline{Y_2}$$

Pošto vrednost izlaza zavisi od stanja flip-flopova i od vrednosti ulaznog signala, može se zaključiti da se u ovom slučaju radi o Milijevom automatu. Imajući to u vidu, na osnovu određenih jednačina formira se kodovana tabela prelazaka stanja, Tabela 4.26.

Stanje		**ulaz X**	
Y_1	**Y_2**	0	1
0	0	01/0	11/0
0	1	10/0	00/1
1	0	11/1	01/0
1	1	00/0	10/0

Tabela 4.26: Kodovana tabela prelazaka stanja/izlaza

Tabela 4.26 prikazuje na osnovu trenutnog stanja automata (Y_1Y_2) i vrednosti ulaza X koje je sledeće stanje (D_1D_2) i vrednost izlaza Z u formatu *sledeće stanje/izlaz*.

Radi jednostavnijeg prepoznavanja stanja uvode se oznake (nazivi) za odgovarajuće kodove:

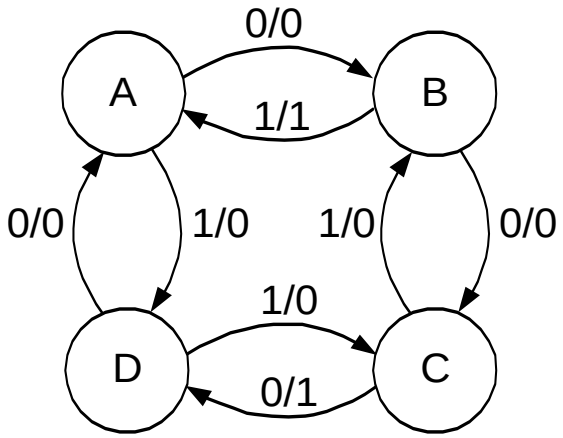
Oznaka stanja	Y_1	Y_2
A	0	0
B	0	1
C	1	0
D	1	1

Na osnovu ovih informacija sledi formiranje tabele prelazaka stanja/izlaza, Tabela 4.27.

Stanje	ulaz X	
	0	1
A	B/0	D/0
B	C/0	A/1
C	D/1	B/0
D	A/0	C/0

Tabela 4.27: Tabela prelazaka stanja/izlaza

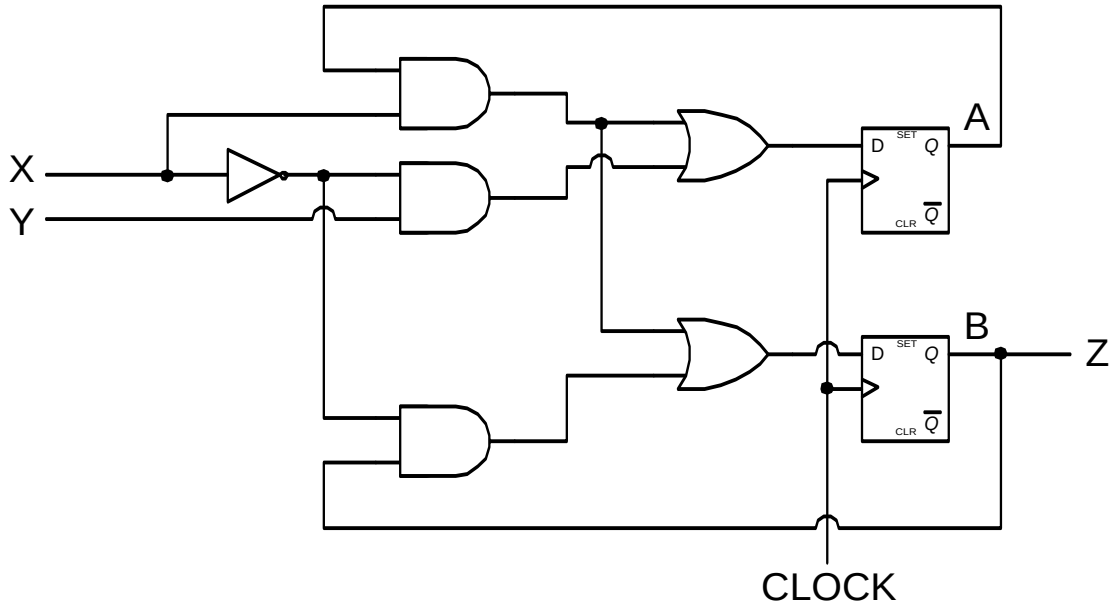
Graf prelazaka stanja automata se formira jednostavnom transfomacijom tabele prelazaka stanja/izlaza. Slika 4.45 prikazuje formirani graf prelazaka stanja zadatog Milijevog automata.



Slika 4.45: Graf prelazaka stanja zadatog automata

4.10 ZADATAK:

Slika 4.46 prikazuje logičku šemu sekvencijalnog sistema sa dva ulaza (X i Y) i jednim izlazom (Z).



Slika 4.46: Logička šema sekvencijalnog sistema

Za dati sistem je potrebno:

- odrediti jednačine prelazaka stanja i jednačinu izlaza,
- definisati kodovanu tabelu prelazaka stanja i
- formirati graf prelazaka stanja.

REŠENJE:

Direktno sa logičke šeme se mogu odrediti jednačine prelazaka stanja, odnosno jednačine koje definišu signale na ulazima flip-flopova:

$$D_A = AX + \overline{X}Y$$

$$D_B = AX + B\overline{X}$$

Jednačina izlaza se takođe očitava direktno sa logičke šeme:

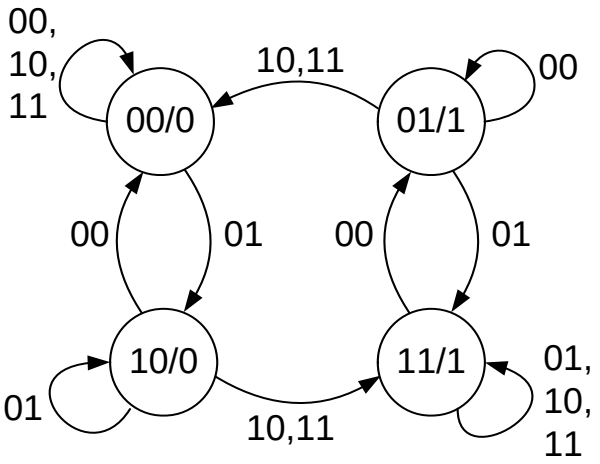
$$Z = B$$

Pošto vrednost izlaza zavisi samo od stanja flip-flopova, može se zaključiti da se u ovom slučaju radi o Murovom automatu. Imajući to u vidu, na osnovu određenih jednačina formira se kodovana tabela prelazaka stanja. Pošto automat ima dva ulaza i dva flip-flopa za pamćenje trenutnog stanja, postoji ukupno 16 različitih kombinacija vrednosti ulaznih signala i stanja flip-flopova. Kodovanu tabelu prelazaka stanja prikazuje Tabela 4.28.

Graf prelazaka stanja automata se formira jednostavnom transfomacijom tabele prelazaka stanja. Slika 4.47 prikazuje formirani graf prelazaka stanja zadatog Murovog automata.

Trenutno stanje		**Ulazi**		**Sledeće stanje**		**Izlaz**
A	B	X	Y	D_A	D_B	Z
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Tabela 4.28: Tabela prelazaka stanja



Slika 4.47: Graf prelazaka stanja zadatog automata