PROGRAMIRANJE KOMUNIKACIONOG HARDVERA
– Prilog A –

Prilog A

U okviru ovog priloga će biti izložena kratka rekapitulacija osnovnih pojmova digitalne elektronike. Pod digitalnom elektronikom se tipično podrazumevaju elektronska kola koja rade sa signalima koji imaju ograničen skup mogućih amplituda (tipično su u pitanju binarni signali koji imaju dve moguće amplitude koje odgovaraju logičkoj '0' i '1'). Naravno, zbog nesavršenosti komponenata i radnih uslova, uglavnom se ne posmatra amplituda jer njena vrednost može da varira oko osnovne vrednosti pa se stoga definiše opseg amplituda koji odgovara jednoj logičkoj vrednosti – jedan opseg za logičku '1' i jedan opseg za logičku '0'. Između susednih opsega logičkih vrednosti se nalazi tzv. nedefinisana oblast u kojoj se ne sme naći amplituda signala u stacionarnom režimu (tokom tranzicije iz jedne u drugu logičku vrednost amplituda signala mora kratko da prođe kroz nedefinisanu oblast). Digitalna kola su doživela procvat sa razvojem tehnologije integrisanih kola. Današnja digitalna elektronska kola se sastoje od aktivnih elemenata (tranzistora) i pasivnih elemenata (uglavnom otpornika).

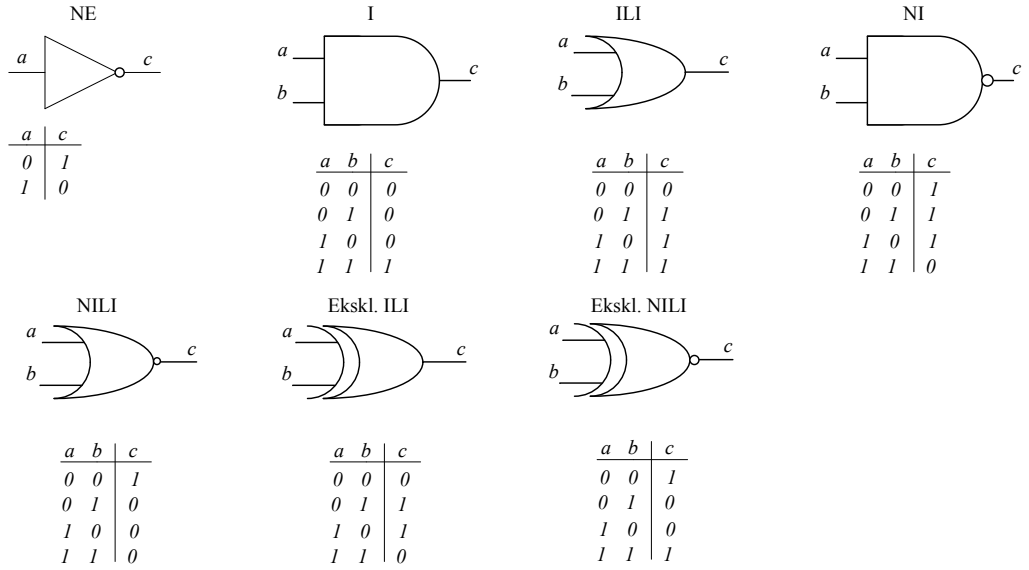
Digitalna elektronska kola obavljaju logičke funkcije koje možemo da grupišemo u dve kategorije:

- kombinaciona kola
- sekvencijalna kola

A.1. Kombinaciona kola

Kombinaciono kolo je ono kolo kod kojega izlazi tog kola zavise samo od vrednosti na ulazima kola. Logička funkcija koju kombinaciono kolo obavlja se može opisati tabelom istinitosti, i takođe u slučaju binarnih kombinacionih kola može se reći da ona podležu zakonima Bulove algebre. U okviru ove sekcije ćemo navesti neka od najpoznatijih kombinacionih kola.

A.1.1. Osnovne logičke funkcije



a	c
0	1
1	0

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

a	b	c
0	0	1
0	1	1
1	0	1
1	1	0

a	b	c
0	0	1
0	1	0
1	0	0
1	1	0

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

a	b	c
0	0	1
0	1	0
1	0	0
1	1	1

Slika A.1.1.1. – Osnovne kombinacione logičke funkcije

U osnovne logičke funkcije spadaju NE, I, ILI, NI, NILI, ekskluzivno ILI i ekskluzivno NILI kolo. Ova kola i njihove tabele istinitosti su prikazane na slici A.1.1.1.

A.1.2. Formiranje kombinacione logike pomoću osnovnih funkcija

Svaka kombinaciona funkcija se može opisati pomoću tabela istinitosti. Na osnovu tabele istinitosti se može formirati kompletna suma proizvoda ili kompletan proizvod suma koje opisuju logičku funkciju opisanu dotičnom tabelom istinitosti. Uzmimo kao primer sledeću tabelu istinitosti za kombinacionu funkciju koja ima četiri ulaza (*a*, *b*, *c* i *d*) i dva izlaza (*e* i *f*).

Tabela A.1.2.1. – Tabela istinitosti za četvoroulaznu kombinacionu funkciju

a	*b*	*c*	*d*	*e*	*f*
0	0	0	0	1	1
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	0	0
1	1	0	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	1	1	1

Na osnovu tabele istinitosti možemo da formiramo kompletnu sumu proizvoda za izlaze *e* i *f* tako što gledamo slučajeve kada su odgovarajući izlazi jednaki ’1’ ili možemo da formiramo kompletan proizvod suma tako što gledamo slučajeve kada su odgovarajući izlazi jednaki ’0’.

Kompletne sume proizvoda za *e* i *f* su:

$$e = \bar{a}\cdot\bar{b}\cdot\bar{c}\cdot\bar{d} + \bar{a}\cdot\bar{b}\cdot c\cdot\bar{d} + \bar{a}\cdot b\cdot\bar{c}\cdot d + \bar{a}\cdot b\cdot c\cdot d + a\cdot\bar{b}\cdot\bar{c}\cdot\bar{d} + a\cdot\bar{b}\cdot c\cdot\bar{d} + a\cdot b\cdot\bar{c}\cdot d + a\cdot b\cdot c\cdot d$$

$$f = \bar{a}\cdot\bar{b}\cdot\bar{c}\cdot\bar{d} + \bar{a}\cdot\bar{b}\cdot c\cdot\bar{d} + \bar{a}\cdot b\cdot c\cdot d + a\cdot\bar{b}\cdot\bar{c}\cdot d + a\cdot b\cdot\bar{c}\cdot\bar{d} + a\cdot b\cdot\bar{c}\cdot d + a\cdot b\cdot c\cdot\bar{d} + a\cdot b\cdot c\cdot d$$

Kompletni proizvodi suma za *e* i *f* su:

$$e = \left(a+b+c+\bar{d}\right)\cdot\left(a+b+\bar{c}+\bar{d}\right)\cdot\left(a+\bar{b}+c+d\right)\cdot\left(a+\bar{b}+\bar{c}+d\right)\cdot\left(\bar{a}+b+c+\bar{d}\right)\cdot\left(\bar{a}+b+\bar{c}+\bar{d}\right)\cdot\left(\bar{a}+\bar{b}+c+d\right)\cdot\left(\bar{a}+\bar{b}+\bar{c}+d\right)$$

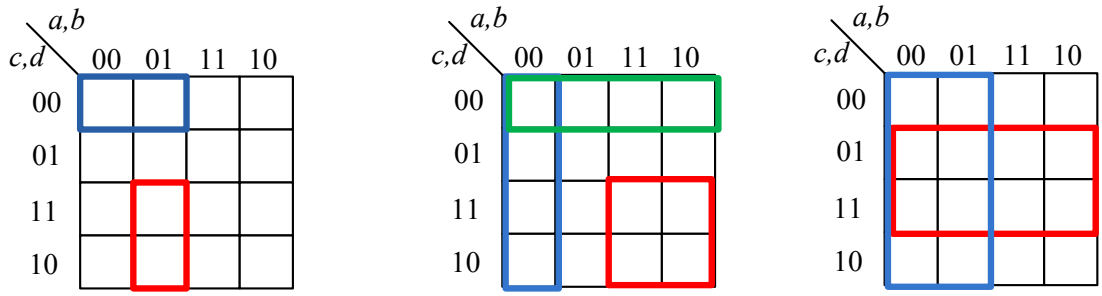
$$f = \left(a+b+c+\bar{d}\right)\cdot\left(a+b+\bar{c}+\bar{d}\right)\cdot\left(a+\bar{b}+c+d\right)\cdot\left(a+\bar{b}+c+\bar{d}\right)\cdot\left(a+\bar{b}+\bar{c}+d\right)\cdot\left(\bar{a}+b+c+\bar{d}\right)\cdot\left(\bar{a}+b+\bar{c}+d\right)\cdot\left(\bar{a}+b+\bar{c}+\bar{d}\right)$$

Kompletni proizvodi suma ili sume proizvoda se mogu dodatno optimizovati tako što se primenom zakona Bulove algebre vrši sažimanje izraza. Tako na primer u slučaju kompletne sume proizvoda za izlaz *e* proces sažimanja bi bio:

$$e = \bar{a}\cdot\bar{b}\cdot\bar{c}\cdot\bar{d} + \bar{a}\cdot\bar{b}\cdot c\cdot\bar{d} + \bar{a}\cdot b\cdot\bar{c}\cdot d + \bar{a}\cdot b\cdot c\cdot d + a\cdot\bar{b}\cdot\bar{c}\cdot\bar{d} + a\cdot\bar{b}\cdot c\cdot\bar{d} + a\cdot b\cdot\bar{c}\cdot d + a\cdot b\cdot c\cdot d =$$
$$= \bar{a}\cdot\bar{b}\cdot\bar{d} + \bar{a}\cdot b\cdot d + a\cdot\bar{b}\cdot\bar{d} + a\cdot b\cdot d = \bar{b}\cdot\bar{d} + b\cdot d$$

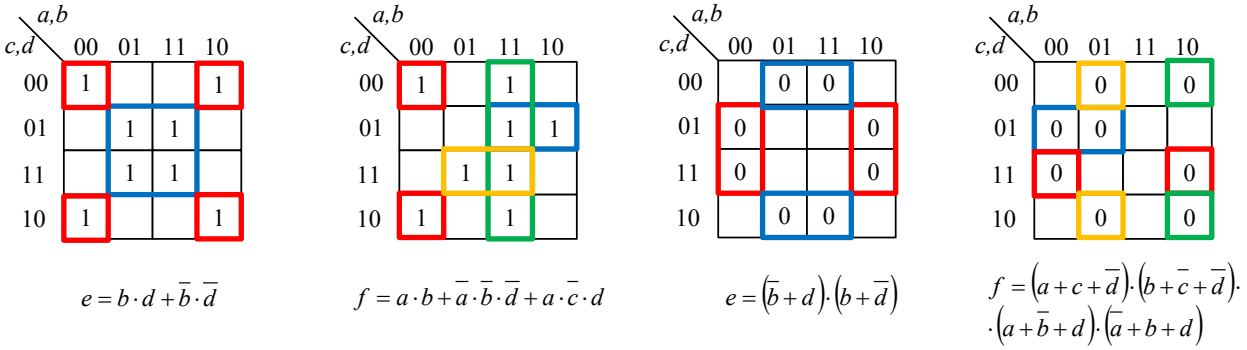
Kao što vidimo, optimizovani izraz za izlaz *e* je znatno prostiji od kompletnog proizvoda suma, čak vidimo da vrednost izlaza *e* zavisi samo od ulaza *b* i *d*. Drugi način sa dobijanje

optimalnog izraza za kombinacionu funkciju je upotreba Karnoovih karti. Deo ulaznih signala predstavlja kolone, a deo ulaznih signala predstavlja redove jedne Karnoove karte. Svako polje Karnoove karte predstavlja vrednost izlaza za jednu kombinaciju vrednosti ulaznih signala. Susedne kolone, odnosno redovi se smeju razlikovati samo za po jedan bit. U slučaju određivanja optimalne sume proizvoda, upisuju se samo jedinice u Karnoovu kartu na odgovarajuće pozicije, a u slučaju određivanja optimalnog proizvoda suma upisuju se samo 0 u Karnoovu kartu. Ukoliko za neke kombinacije ulaza nam nije bitna vrednost izlaza, tada se u ta odgovarajuća polja Karnoove karte upisuje 'X' koji igra ulogu džokera. Optimizacija se radi tako što se grupišu upisane vrednosti po principu prikazanom na slici A.1.2.1 za slučaj četvoroulazne kombinacione funkcije. Moraju biti pokrivene sve upisane '1', ili '0', u zavisnosti da li određujemo optimalnu sumu proizvoda ili proizvod suma, pri čemu džoker znaci samo pomažu u dobijanju optimalnijih izraza, ali oni ne moraju biti svi pokriveni.



Slika A.1.2.1. – Grupisanje članova u Karnoovim kartama

Primenimo Karnoove karte na izlaze *e* i *f* kombinacione funkcije predstavljene tabelom istinitosti datoj u tabeli A.1.2.1. Karnoove karte za formiranje optimalnog proizvoda suma, odnosno sume proizvoda za izlaze *e* i *f* su prikazane na slici A.1.2.2.

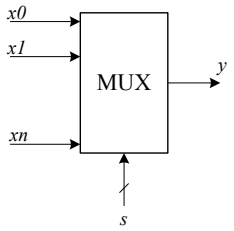


Slika A.1.2.2. – Karnoove karte za izlaze *e* i *f*

Karnoove karte predstavljaju pogodan način za vizuelno nalaženje optimalnih izraza za kombinacione funkcije. Međutim, broj polja u Karnoovim kartama raste eksponencijalno sa brojem ulaza, tako da one nisu pregledne i praktične za kombinacione funkcije sa velikim brojem ulaza.

A.1.3. Multiplekseri

Multiplekseri izigravaju ulogu svojevrsnog prekidača ili sviča. Postoje dva tipa ulaza, korisnički ulazi i kontrolni ulazi (tzv. selektori), i postoji samo jedan izlaz. Funkcija multipleksera je da se jedan od ulaza spoji na izlaz, a koji od ulaza će to biti se bira podešavanjem kontrolnih ulaza. Logički simbol multipleksera je dat na slici A.1.3.1.



Slika A.1.3.1. – Logički simbol multipleksera

U tabeli A.1.3.1. je dat opis rada četvoroulaznog multipleksera. Kao što vidimo selektor *s* određuje koji od ulaza će da bude spojen na izlaz.

Tabela A.1.3.1. – Princip rada četvoroulaznog multipleksera

s(1..0)	***izlaz***
00	$y = x0$
01	$y = x1$
10	$y = x2$
11	$y = x3$

Na osnovu tabele A.1.3.1 možemo i napisati logičku funkciju četvoroulaznog multipleksera (identična logika se primenjuje i za multipleksere sa više ulaza):

$$y = \overline{s_1} \cdot \overline{s_0} \cdot x_0 + \overline{s_1} \cdot s_0 \cdot x_1 + s_1 \cdot \overline{s_0} \cdot x_2 + s_1 \cdot s_0 \cdot x_3$$

Broj ulaza u multiplekser ne mora biti 2^n i tada se izrazi za nepostojeće ulaze ispuštaju iz logičke funkcije. Na primer, ako bi postojala samo tri ulaza logička funkcija multipleksera bi glasila:

$$y = \overline{s_1} \cdot \overline{s_0} \cdot x_0 + \overline{s_1} \cdot s_0 \cdot x_1 + s_1 \cdot \overline{s_0} \cdot x_2$$

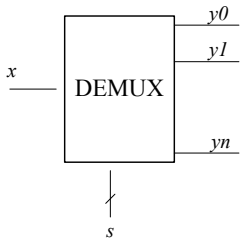
Ulazi i izlaz mogu biti širine jedan bit, a mogu biti i magistrale od više bita. Tada, navedena logička funkcija multipleksera važi za svaki bit tj. liniju magistrale.

A.1.4. Demultiplekseri

Demultiplekseri imaju obrnutu ulogu od multipleksera, tako da sada postoji jedan korisnički ulaz, kontrolni ulazi i više izlaza. Kontrolni ulazi (selektor) biraju na koji od izlaza će da se preslika vrednost sa ulaza. Logički simbol demultipleksera je dat na slici A.1.4.1.

Princip rada četvoroizlaznog demultipleksera je opisan u tabeli A.1.4.1. Na osnovu ove tabele možemo da napišemo logičku funkciju četvoroizlaznog demultipleksera (identična logika se primenjuje i za demultipleksere sa više izlaza):

$$y_0 = \overline{s_1} \cdot \overline{s_0} \cdot x, \quad y_1 = \overline{s_1} \cdot s_0 \cdot x; \quad y_2 = s_1 \cdot \overline{s_0} \cdot x; \quad y_3 = s_1 \cdot s_0 \cdot x$$



Slika A.1.4.1. – Logički simbol demultipleksera

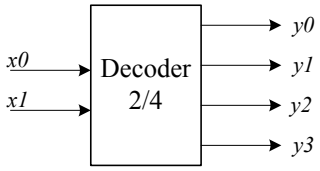
Tabela A.1.4.1. – Princip rada četvoroizlaznog demultipleksera

s(1..0)	*izlaz*
00	*y0 = x*
01	*y1 = x*
10	*y2 = x*
11	*y3 = x*

Ulaz i izlazi mogu biti širine jedan bit, a mogu biti i magistrale od više bita. Tada, navedena logička funkcija demultipleksera važi za svaki bit tj. liniju magistrale.

A.1.5. Dekoderi

Uloga dekodera je da aktivira jedan od izlaza na bazi sadržaja (koda) ulaza. Pod aktiviranjem izlaza se podrazumeva da se dotični izlaz postavi na aktivnu vrednost (na primer '1', ako se '1' podrazumeva u dizajnu kao aktivna vrednost). Jedna od tipičnih primena dekodera je u aktivaciji čipova koji dele adresni prostor i magistralu podataka, tako da se aktivira samo željeni čip i sa njime potom ostvari komunikacija. Logički simbol dekodera 2/4 je dat na slici A.1.5.1.



Slika A.1.5.1. – Logički simbol dekodera 2/4

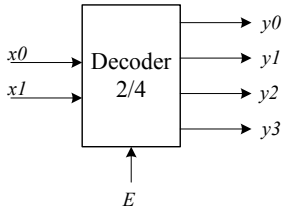
Tabela A.1.5.1. – Princip rada dekodera 2/4

x1x0	*y3y2y1y0*
00	0001
01	0010
10	0100
11	1000

Princip rada dekodera 2/4 je dat u tabeli A.1.5.1. Na osnovu ove tabele možemo da napišemo logičku funkciju dekodera 2/4 (identična logika se primenjuje i za dekodere sa više ulaza i izlaza):

$$y_0 = \overline{x_1} \cdot \overline{x_0}, \quad y_1 = \overline{x_1} \cdot x_0; \quad y_2 = x_1 \cdot \overline{x_0}; \quad y_3 = x_1 \cdot x_0$$

Očigledno dekoder prikazan na slici A.1.5.1. će uvek aktivirati jedan od izlaza. Ako želimo da imamo mogućnost da svi izlazi budu neaktivni uvodi se tzv. signal dozvole *E*. Samo kada je aktivan signal dozvole *E*, dekoder će obavljati svoju funkciju, u suprotnom svi izlazi će biti neaktivni. Umesto, oznake *E* (*Enable*) nekad se koristi oznaka *CS* (*Chip Select*). Logički simbol dekodera 2/4 sa signalom dozvole je dat na slici A.1.5.2, a njegov princip rada je dat u tabeli A.1.5.2.



Slika A.1.5.2. – Logički simbol dekodera 2/4 sa signalom dozvole

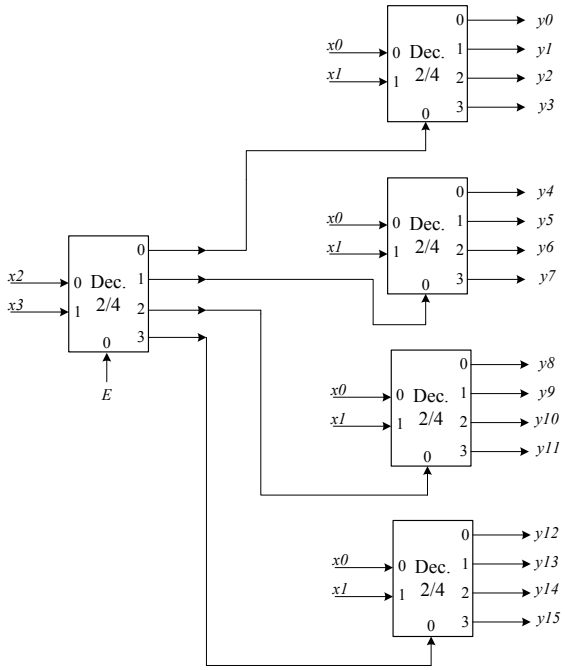
Tabela A.1.5.2. – Princip rada dekodera 2/4 sa signalom dozvole

E	***x1x0***	***y3y2y1y0***
1	00	0001
1	01	0010
1	10	0100
1	11	1000
0	xx	0000

Na osnovu tabele A.1.5.2 možemo da napišemo logičku funkciju dekodera 2/4 sa signalom dozvole (identična logika se primenjuje i za dekodere sa više ulaza i izlaza):

$$y_0 = \overline{x_1} \cdot \overline{x_0} \cdot E, \quad y_1 = \overline{x_1} \cdot x_0 \cdot E; \quad y_2 = x_1 \cdot \overline{x_0} \cdot E; \quad y_3 = x_1 \cdot x_0 \cdot E$$

Signal dozvole može biti sastavljen od više bita i tada svi biti signala dozvole moraju biti aktivni da bi dekoder obavljao svoju funkciju. Ovo je posebno zgodno za slučaj kada se od manjih dekodera prave veći dekoderi. Na slici A.1.5.3 je prikazano kreiranje dekodera 4/16 od dekodera 2/4.



Slika A.1.5.3. – Kreiranje dekodera 4/16 koristeći dekodere 2/4

Kao što vidimo sa viša dva bita *x3* i *x4* aktiviramo signal dozvole na jednom od 4 dekodera u drugoj koloni. Sa niža dva bita *x1* i *x0* potom aktiviramo odgovarajući izlaz na aktiviranom dekoderu. Sa signalom dozvole *E* deaktiviramo kompletan dekoder jer nijedan od

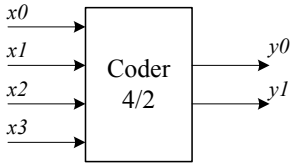
izlaza prvog dekodera neće biti aktivan, a samim tim nijedan od dekodera u drugoj koloni takođe neće biti aktivan jer će svi oni imati neaktivne signale dozvole.

Broj izlaza iz dekodera ne mora biti 2^n i tada se izrazi za nepostojeće izlaze ispuštaju iz logičke funkcije. Na primer ako u dekoder 2/4 ne bi postojao izlaz *y3* tada bi logička funkcija dekodera bila (izraz za *y3* se izostavlja):

$$y_0 = \overline{x_1} \cdot \overline{x_0}, \quad y_1 = \overline{x_1} \cdot x_0; \quad y_2 = x_1 \cdot \overline{x_0}$$

A.1.6. Koderi

Uloga kodera je obrnuta ulozi dekodera. Na ulazima može biti maksimalno jedan aktivan ulaz i njegov položaj (id) se izbacuje na izlaz. Logički simbol kodera 4/2 je dat na slici A.1.6.1.



Slika A.1.6.1. – Logički simbol kodera 4/2

Tabela A.1.6.1. – Princip rada kodera 4/2

x3x2x1x0	***y1y0***
0001	00
0010	01
0100	10
1000	11

Princip rada kodera 4/2 je dat u tabeli A.1.6.1. Na osnovu ove tabele možemo da napišemo logičku funkciju kodera 4/2 (identična logika se primenjuje i za kodere sa više ulaza i izlaza):

$$y_1 = x_3 + x_2, \quad y_0 = x_3 + x_1;$$

Treba primetiti da će na izlazu da stoji vrednost "00" čak i kad su svi ulazni neaktivni. Ako se želi razrešiti i ova situacija može se dodati još jedan izlaz koji bi označavao da li su podaci na izlazu validni (u smislu jedan ulaz je aktivan) ili nisu (svi ulazi su nekativni). Očigledno bi ovaj dodatni izlaz bio jednak sumi svih ulaza. Takođe, broj ulaza ne mora da bude 2^n, već može da bude i manje ulaza. Tada se iz logičkih izraza za izlaze potpunog kodera izbacuju ulazi koji nedostaju. Na primer, ako ulaz *x3* u koderu 4/2 ne bi postojao tada bi izrazi za izlaze *y1* i *y0* bili:

$$y_1 = x_2, \quad y_0 = x_1;$$

Treba primetiti da u slučaju kada je više od jednog ulaza aktivno, koder će izbaciti pogrešnu vrednost. Međutim, često je zgodno imati i koder koji bi imao mogućnost rada i u slučajevima kada je više ulaza istovremeno aktivno. Takvi koderi se nazivaju koderi sa prioritetom, jer na izlaz izbacuju kod ulaza višeg prioriteta. Tabela A.1.6.2 prikazuje princip rada prioritetnog kodera 4/2.

Tabela A.1.6.2. – Princip rada prioritetnog kodera 4/2

x3x2x1x0	*y1y0*
0001	00
001x	01
01xx	10
1xxx	11

Kao što se vidi iz tabele A.1.6.2, kada je ulaz najvišeg prioriteta aktivan izlaz će biti sigurno "11", nezavisno od vrednosti ostalih ulaza. Jedna od tipičnih primena prioritetnog kodera je opsluživanje prekida različitih nivoa (prioriteta).

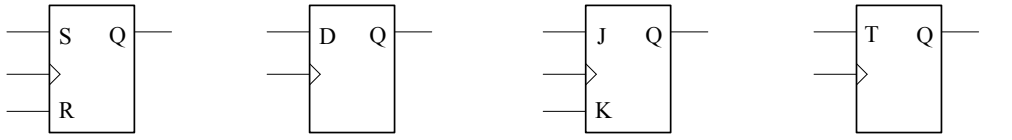
A.2. Sekvencijalna kola

Sekvencijalno kolo je ono kod kojega vrednost na izlazima zavisi od ulaznih signala, ali i trenutnog stanja kola. Očigledno, u sekvencijalnoj logici moramo imati memorijske elemente. Razlikujemo dva tipa sekvencijalnih kola – Murova (*Moore*) sekvencijalna kola i Milijeva (*Mealy*) sekvencijalna kola. U slučaju Murovog sekvencijalnog kola, izlazne vrednosti zavise samo od trenutnog stanja kola, a u slučaju Milijevog sekvencijalnog kola, izlazne vrednosti zavise i od ulaznih vrednosti i od trenutnog stanja kola. Primer Murove logike bi bio brojač koji broji kružno tako da sledeće stanje zavisi samo od trenutnog stanja brojača. Primer Milijeve logike bi bio brojač koji kružno broji, ali koji ima opciju učitavanja tako da kad se aktivira učitavanje izlazna vrednost će postati jednaka ulaznoj vrednosti.

Ulogu memorijskog elementa u sekvencijalnim kolima imaju flip-flopovi. Razlikujemo sinhrone i asinhrone flip-flopove, pri čemu se asinhroni flip-flopovi često nazivaju i lečevi (*latch*).

A.2.1. Sinhroni flip-flopovi

Flip-flopovi menjaju svoje stanje na ivicu signala okidača ili na nivo signala okidača. Signal okidača je sinhroni signal koji se naziva signalom takta (*clock*). U praksi se češće koriste flip-flopovi koji menjaju svoje stanje na ivicu signala okidača (takta). Postoji više vrsta flip-flopova – RS-FF, D-FF, JK-FF i T-FF. Funkcije ovih flip-flopova, njihovi logički simboli i karakteristične jednačine su prikazani na slici A.2.1.1.



S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	?

D	Q_{n+1}
0	0
1	1

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

$Q_{n+1} = S+\overline{R}Q_n$ $\quad$ $Q_{n+1} = D$ $\quad$ $Q_{n+1} = J\overline{Q}_n+\overline{K}Q_n$ $\quad$ $Q_{n+1} = TQ_n+T\overline{Q}_n$

Slika A.2.1.1. – Logički simboli i karakteristične jednačine sinhronih flip-flopova

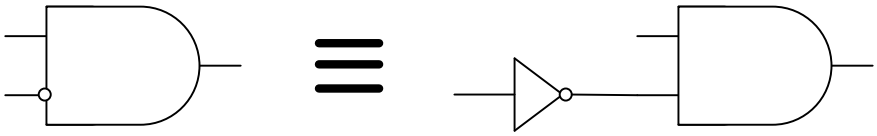
U praksi se najčešće sreću, naročito u slučaju programabilnih čipova, D-FF-ovi koji u principu imaju ulogu kola za kašnjenje. Kao što se vidi sa slike A.2.1.1, ulaz u D-FF se preslikava na izlaz D-FF sa kašnjenjem od jedne periode (ciklusa) signala takta. Takođe je zgodno primetiti da ako bi se držao signal '1' na ulazu T-FF, na izlazu T-FF bi se dobio periodičan signal koji ima periodu dvostruko veću od signala takta (tj. dobio bi se dvostruko sporiji signal takta).

Uzlazna ivica takta ili nivo '1' signala takta

Silazna ivica takta ili nivo '0' signala takta

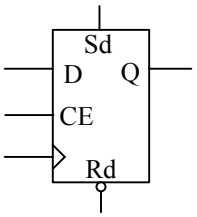
Slika A.2.1.2. – Simbolička oznaka aktivne vrednosti signala takta

Flip-flop menja svoje stanje ili na uzlaznu ili na silaznu ivicu takta, ukoliko je u pitanju flip-flop koji menja stanje na ivicu signala takta. Flip-flop menja svoje stanje ili na vrednost '0' ili na vrednost '1' signala takta, ukoliko je u pitanju flip-flop koji menja stanje na nivo signala takta. Simboličke oznake za sve navedene slučajeve su prikazane na slici A.2.1.2. Treba primetiti da ukoliko je '0' aktivna vrednost signala (bilo kog, ne samo takta) stavlja se kružić na ulazu tog signala u određeno logičko kolo da bi se signalizirala takva situacija. Dotični kružić kao da predstavlja inverzni bafer kroz koji je prošao signal kao što je prikazano u primeru sa slike A.2.1.3.



Slika A.2.1.3. – Primer simboličke oznake za aktivnu vrednost '0' i njen ekvivalent u vidu inverznog bafera

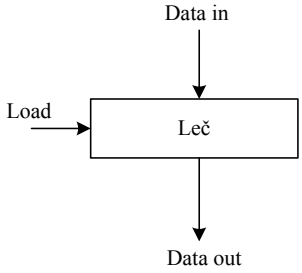
Flip-flopovi mogu imati i dodatne ulaze uz one prikazane na slici A.2.1.1. To su asinhroni reset (*Rd*), asinhroni set (*Sd*) i signal dozvole za takt (*CE*). Asinhroni reset daje mogućnost reseta stanja flip-flopa nezavisno od signala takta. Dok god je asinhroni reset aktivan na izlazu flip-flopa se nalazi '0'. Kada se deaktivira signal asinhronog reseta, flip-flop nastavlja da radi u skladu sa svojom karakterističnom jednačinom. Asinhroni set je analogan signalu asinhronog reseta, samo što se na izlazu flip-flopa sada pojavljuje vrednost '1'. Očigledno, ne smeju da istovremeno budu aktivni i asinhroni set i asinhroni reset. Signal dozvole za takt se koristi da zamrzne rad flip-flopa. Kad je signal dozvole aktivan flip-flop radi regularno u skladu sa svojom karakterističnom jednačinom. Kada se signal dozvole *CE* deaktivira, tada flip-flop 'zamrzava' svoje trenutno stanje i ne reaguje na signal takta. Svaki od navedena tri signala može i ne mora da bude prisutan u flip-flopu. Najviši prioritet imaju signali asinhronog reseta i seta, a tek potom signal dozvole za takt. Na slici A.2.1.4 je prikazan D-FF koji ima sva tri navedena signala, pri čemu je u primeru sa slike, za signale *Sd* i *CE* je stavljeno da je aktivna vrednost '1', a za signal *Rd* je stavljeno da je aktivna vrednost '0' (naravno dozvoljene su i druge kombinacije aktivnih vrednosti).



Slika A.2.1.4. – D-FF sa signalima asinhronog seta i reseta i signalom dozvole za takt

A.2.2. Lečevi

Lečevi (*latch*) predstavljaju asinhrone flip-flopove. Lečevi u stvari imaju ulogu transparentnih registara. Naime, oni reaguju samo na nivo signala dozvole (u ovom slučaju se ne koristi termin signal okidača jer je u pitanju asinhrono kolo). Kada je signal dozvole aktivan, na izlaz leča se preslikava signal sa ulaza, dok kad je signal dozvole neaktivan, izlaz leča je zamrznut u trenutnom stanju tj. izlaz ima vrednost sa ulaza neposredno pre isključivanja signala dozvole. Signal dozvole je asinhron, pa je otuda i leč asinhrono kolo. Logički simbol leča je prikazan na slici A.2.2.1. Signal dozvole se često naziva i signal učitavanja (*Load*).

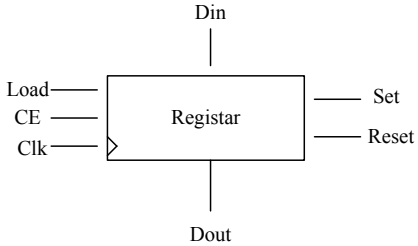


Slika A.2.2.1. – Logički simbol leč kola

Lečevi su pogodni za korišćenje u slučaju asinhrone logike, odnosno kad ne koristimo signal takta, a da nam pri tome trebaju memorijski elementi u dizajniranom hardveru. S druge strane, ako dizajniramo sinhronu logiku, trebalo bi izbegavati lečeve i koristiti registre kreirane od sinhronih flip-flopova.

A.2.3. Registri

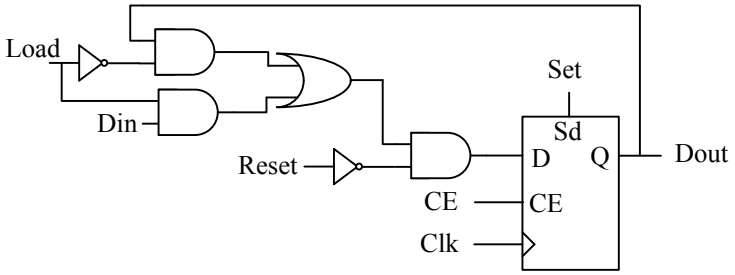
U ovoj sekciji razmatramo sinhrone registre koji rade sinhrono sa signalom takta. Registri podrazumevaju memorijske elemente koji čuvaju željene podatke. Logički simbol registra je dat na slici A.2.3.1.



Slika A.2.3.1. – Logički simbol registra

Pored signala takta (*Clk*) i signala izlaznih podataka (*Dout*), koji su uvek prisutni, registar može da sadrži i sledeće signale: signal ulaznih podataka (*Din*), signal učitavanja (*Load*), signal seta (*Set*), signal reseta (*Reset*) i signal dozvole za takt (CE). Izlazni podaci (*Dout*) daju trenutni sadržaj registra. Ulazni podaci (*Din*) ne moraju da budu prisutni ukoliko je u pitanju registar sa

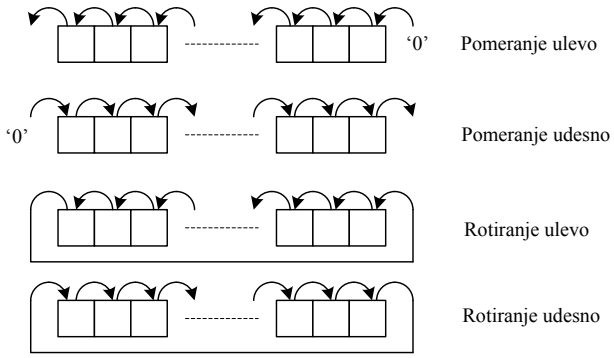
sadržajem koji je konstantan i ne menja se u toku vremena, ali tipično se ne koriste registri za čuvanje konstantnog sadržaja već se takav sadržaj kreira od izvora logičkih ’0’ (*GND*) i ’1’ (*VCC*). Kada su ulazni podaci prisutni tada je prisutan i signal učitavanja kojim se aktivira učitavanje novog sadržaja (koji je prisutan na ulazu) u registar. Signali seta i reseta mogu biti sinhroni ili asinhroni (pri tome set i reset mogu biti različiti po tom pitanju, tj. jedan može biti sinhron, a jedan asinhron). U slučaju aktiviranja signala sinhronog seta ili reseta, sadržaj registra se na prvu narednu ivicu takta setuje na sve ’1’ ili resetuje na sve ’0’. Asinhroni set i reset rade po identičnom principu kao i u slučaju sinhronih flip-flopova. Naravno, signali seta i reseta su opcioni, u smislu može da postoji samo jedan od njih, oba ili nijedan. Takođe, mogu postojati istovremeno i asinhroni i sinhroni set signali (isto važi i za reset signale), ali to se uglavnom ne koristi. Kao kod sinhronih flip-flopova, i kod registara ne bi smeli da budu istovremeno aktivni i set i reset signal. Signal dozvole za takt je takođe opcion, a ako postoji onda je njegova uloga identična njegovoj funkciji u slučaju sinhronih flip-flopova. To znači da kad je signal dozvole neaktivan, registar ’zamrzava’ svoje stanje i tada može da reaguje samo na asinhroni set ili reset signal. Jednobitni registar sa svim navedenim signalima baziran na D-FF je prikazan na slici A.2.3.2. Na slici A.2.3.2 je prikazana varijanta sa sinhronim resetom i asinhronim setom, a svi kontrolni signali (*Load*, *CE*, *Set*, *Reset*) su aktivni na logičkoj ’1’.



Slika A.2.3.2. – Struktura jednobitnog registra

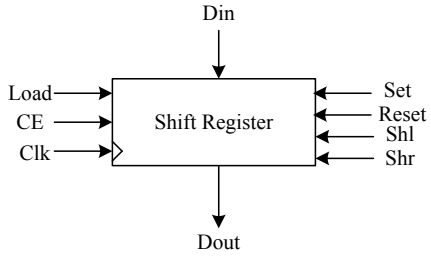
A.2.4. Pomerački registri

Pomerački (*shift*) registri su posebna klasa registara. Ovi registri mogu da imaju sve opcije kao i obični registri, sa dodatnom mogućnošću pomeranja ili rotiranja sadržaja registra. Pomeranje/rotiranje može da se radi u samo jednu stranu (ulevo ili udesno) ili da postoji mogućnost izbora smera pomeranja/rotiranja. U slučaju pomeranja sadržaja upražnjena mesta se tipično popunjavaju nulama, ali mogu da se implementiraju i druge opcije za popunjavanje sadržaja upražnjenih mesta. Principi pomeranja i rotiranja sadržaja registra su prikazani na slici A.2.4.1. U primerima sa slike se vrši pomeranje/rotiranje za jedno mesto.



Slika A.2.4.1. – Principi pomeranja i rotiranja sadržaja pomeračkih registara

Dodatni kontrolni signali za pomeračke registre u odnosu na obične registre su: signal za pomeranje/rotiranje ulevo (*Shl*) i signal za pomeranje/rotiranje udesno (*Shr*). Ukoliko je pomeranje/rotiranje moguće u samo jednu stranu, biće prisutan samo jedan od dva navedena signala. Ukoliko su moguća pomeranja/rotiranja u obe strane postoji i varijanta koja koristi signal za pomeranje/rotiranje (*Shift*) i signal za izbor smera za pomeranje/rotiranje (*Dir*). U slučaju kada je signal za pomeranje/rotiranje aktivan vrši se pomeranje sadržaja registra, u suprotnom registar zadržava trenutno stanje. Ukoliko postoji i signal za učitavanje (*Load*), taj signal ima prioritet nad signalima za pomeranje/rotiranje. Ukoliko se kod bidirekcionih pomeračkih registara (koji mogu da pomeraju sadržaj u obe strane) koristi varijanta sa signalima *Shl* i *Shr*, tada ne smeju ta dva signala biti istovremeno aktivna. Veličina pomeraja (za koliko mesta će sadržaj registra biti pomeren) je uglavnom statička tj. ne može da se menja i tipično iznosi jedan. Neki pomerački registri omogućavaju i da se dinamički definiše veličina pomeraja, ali s obzirom da takvo svojstvo povećava hardverske resurse ovakva opcija se koristi samo kada je zaista neophodna. Primer logičkog simbola za pomerački registar je dat na slici A.2.4.2.



Slika A.2.4.2. – Logički simbol pomeračkog registra

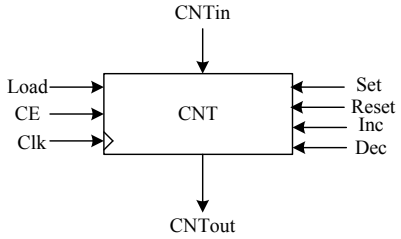
Pomerački registri se mogu koristiti za implementiranje funkcija množenja i deljenja sadržaja pomeračkog registra sa brojevima koji su stepena dvojke (2^n), gde je n veličina pomeraja. Pomeranje udesno odgovara operaciji deljenja, a pomeranje ulevo operaciji množenja. U oba slučaja se podrazumeva da se upražnjena mesta popunjavaju sa nulama. Na primer, ako je n=1, tada je pomeranje udesno za jedno mesto ekvivalentno deljenju sadržaja pomeračkog registra sa 2, a pomeranje ulevo za jedno mesto je ekvivalentno množenju sadržaja pomeračkog registra sa 2.

A.2.5. Brojači

Funkcija brojača je, kao što i sam njihov naziv kaže, da izvršavaju funkciju brojanja. Brojanje može da bude sa korakom jedan, sa većim korakom ili sam redosled brojeva može da bude definisan proizvoljno. Takođe, brojanje može da se radi u jednom smeru, ali i oba smera (brojanje unapred i unazad). Najčešće korišćeni brojači su oni sa korakom jedan, pri čemu se koriste sve tri varijante podjednako često: sa brojanjem unapred (inkrementiranje), sa brojanjem unazad (dekremetiranje) i sa mogućnošću dinamičkog izbora smera brojanja. Takođe, često korišćeni brojači su oni koji broje u skladu sa Grejevim kodom, koji je karakterističan po tome da se susedna stanja (brojevi) razlikuju samo po jednom bitu, čime se sprečava pojava tzv. gličeva (*glitch*).

Kontrolni signali brojača su identični onima pomeračkog registra, samo umesto signala *Shl* i *Shr*, se koriste signali *Inc* i *Dec*. Signal *Inc* aktivira brojanje unapred (inkrementiranje), a signal *Dec* aktivira brojanje unazad (dekrementiranje). Očigledno, ova dva signala ne smeju biti istovremeno aktivna. Kada su oba signala neaktivna, brojač zadržava svoje trenutno stanje. Naravno, u slučaju brojača u jednom smeru, biće prisutan samo jedan od ta dva signala ili signal

Inc ili signal *Dec*, u zavisnosti u kom smeru se vrši brojanje. Kao i kod pomeračkih registara, postoji i druga varijanta kada su moguća oba smera. U drugoj varijanti umesto *Inc* i *Dec* signala se koriste signali *Dir* i *CntE*, gde *Dir* definiše smer brojanja, a *CntE* aktivira brojanje. Kada signal *CntE* nije aktivan brojač zadržava svoje trenutno stanje. Logički simbol brojača koji ima mogućnost brojanja u obe strane je prikazan na slici A.2.5.1, pri čemu je u pitanju varijanta sa signalima *Inc* i *Dec*. Treba napomenuti da u slučaju kad postoji i signal za učitavanje (*Load*), taj signal ima prioritet nad kontrolnim signalima za brojanje. Takođe, mogu se definisati i brojači koji neprestano broje, i u takvim slučajevima nisu potrebni signali za aktiviranje brojanja, a jedino u slučaju dvosmernih neprestanih brojača potreban je signal *Dir* za zadavanje smera brojanja.



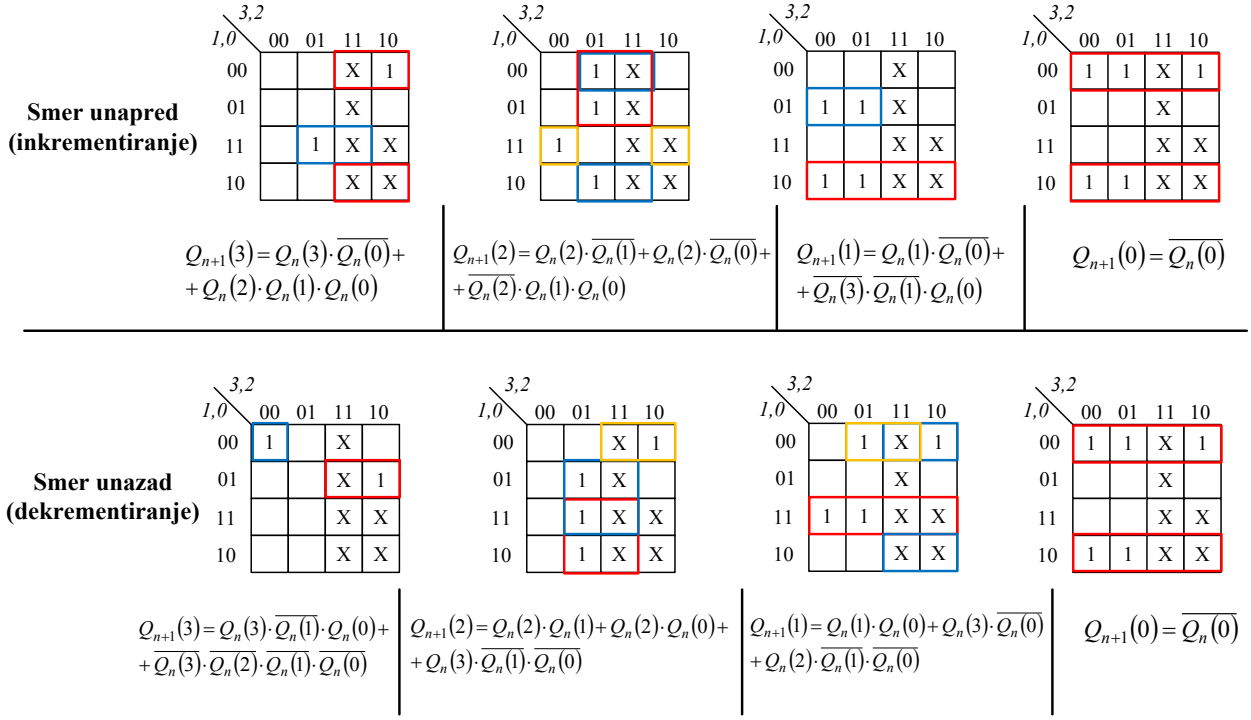
Slika A.2.5.1. – Logički simbol brojača

Kreirajmo strukturu četvorobitnog brojača koji ima mogućnost brojanja unapred i unazad, kao i učitavanja, sinhronog seta i asinhronog reseta. Asinhroni reset je aktivan na logičkoj ’0’, a svi ostali kontrolni signali su aktivni na logičkoj ’1’. Koristimo varijantu sa signalima *Dir* i *CntE*. Ako je vrednost signala *Dir* jednaka ’1’, smer brojanja je unapred, u suprotnom smer brojanja je unazad. Brojač broji kroz sva moguća stanja sa korakom jedan. Stanja brojača su od "0000" do "1001", tj. u pitanju je dekadni brojač. Kreirajmo tabelu prelaza stanja brojača za slučaj inkrementiranja (tabela A.2.5.1) i tabelu prelaza stanja brojača za slučaj dekrementiranja (tabela A.2.5.1). Za nedozvoljena stanja brojača ćemo staviti ’*don’t care*’ vrednost tj- ’X’.

Tabela A.2.5.1. – Tabla prelaza stanja četvorobitnog brojača za smer inkrementiranja

	Dir=’1’ i CntE=’1’	***Dir=’0’ i CntE=’1’***
$CNTout_n(3..0)$	***$CNTin_{n+1}(3..0)$***	***$CNTin_{n+1}(3..0)$***
0000	0001	1001
0001	0010	0000
0010	0011	0001
0011	0100	0010
0100	0101	0011
0101	0110	0100
0110	0111	0101
0111	1000	0110
1000	1001	0111
1001	0000	1000
1010	XXXX	XXXX
1011	XXXX	XXXX
1100	XXXX	XXXX
1101	XXXX	XXXX
1110	XXXX	XXXX
1111	XXXX	XXXX

Na osnovu tabele prelaza stanja brojača možemo formirati Karnoove karte za sva četiri bita stanja brojača i za smer unapred i za smer unazad. Formirane Karnoove karte, kao i izrazi za bite brojača *CNTout*(*i*) su prikazani na slici A.2.5.2. Radi preglednijih izraza umesto *CNTout* na slici A.2.5.2 je stavljena oznaka *Q*.

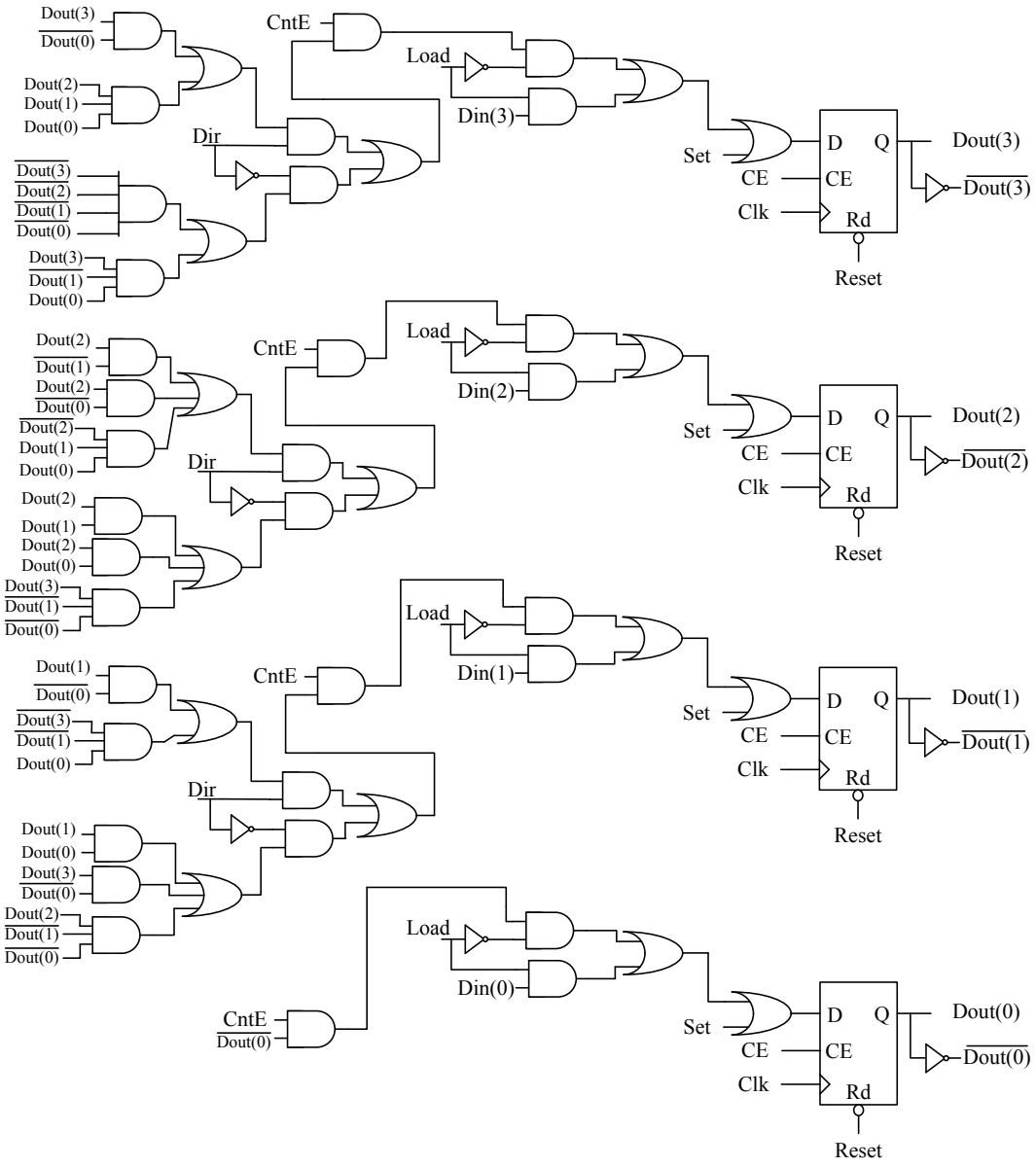


Slika A.2.5.2. – Karnoove karte za bite brojača za slučajeve inkrementiranja i dekrementiranja

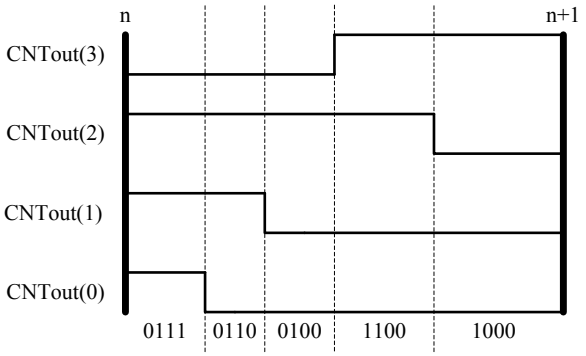
Na osnovu dobijenih izraza iz Karnoovih karata možemo da kreiramo strukturu brojača koja je prikazana na slici A.2.5.3. Važno je napomenuti da su zabranjena stanja označena sa 'X' iskorišćena za formiranje optimalnijih izraza na osnovu Karnoovih karti. Međutim, treba biti oprezan sa takvom upotrebom zabranjenih stanja. Uvek se mora predvideti šta će se desiti i šta treba uraditi ako se sistem nađe u zabranjenom stanju. Na primer, može se uvesti dodatna provera koja će proveravati da li je brojač u zabranjenom stanju i ako jeste signalizirati grešku izlaznim signalom greške (*Error*). Takođe, može se desiti da kad jednom brojač uđe u zabranjeno stanje, on nikad ne izađe iz zabranjenih stanja što takođe nije željeno. Uvek je jedna od opcija da po ulasku u zabranjeno stanje, odmah u sledećem ciklusu brojač pređe u neko dozvoljeno stanje, na primer u stanje "0000". Ako se izabere ova opcija tada se sadržaj tabele A.2.5.1 menja jer se umesto 'XXXX' kod zabranjenih stanja stavlja vrednost dozvoljenog stanja u koje brojač treba da pređe iz zabranjenih stanja. Naravno, ovo utiče na strukturu Karnoovih karata pa se dobijaju novi izrazi za sledeća stanja brojača u slučaju inkrementiranja i dekrementiranja.

Na kraju objasnimo i pojavu gliča. U realnosti sva električna kola imaju konačna fizička kašnjenja, za razliku od idealnih kola koja nemaju kašnjenja. Takođe, i same linije koje spajaju kola imaju svoju fizičku dužinu pa samim tim i kašnjenje. Veličina kašnjenja se razlikuje od kola do kola, a takođe i od veličine kombinacione logike kroz koju signal prolazi (suma kašnjenja svih kola kroz koja signal prolazi na putu sa najvećim kašnjenjem kroz kombinacionu logiku). Usled različitih kašnjenja, može doći do situacije da u slučaju višeizlazne logike, neki izlazi dobijaju nove vrednosti pre ostalih izlaza, čime dolazi do pojave 'lažnih' stanja na izlazu, tzv.

gličeva. Na slici A.2.5.4 je prikazan primer kada brojač menja stanje iz "0111" u stanje "1000", pri čemu je u primeru uzeto da najbržu promenu ima Dout(0), pa Dout(1), pa Dout(3) i na kraju Dout(2) koji ima najsporiju promenu svoje vrednosti. Kao što se vidi na izlazu, pojavljuju se tri 'lažne' međuvrednosti. Ako neki uređaj koristi dotične izlazne vrednosti, važno je da takav uređaj ne pogreši i uzme 'lažne' vrednosti kao tačne. U slučaju sinhrone sekvencijalne logike sve promene moraju da se okončaju do početka narednog ciklusa takta, u suprotnom će sekvencijalna logika raditi neispravno. Otuda je najbolje rešenje da se uređaj koji koristi izlazne vrednosti brojača iz primera takođe sinhronizuje na signal takta, čime će se obezbediti da uvek uzima vrednosti brojača na signal takta (tj. na početku novog ciklusa) čime će se obezbediti da se uvek uzimaju korektne vrednosti. Ako je uređaj asinhron onda ovakvo rešenje nije moguće i tada treba osmisliti način za izbegavanje pogrešnog očitavanja brojača iz primera. Upravo u ovakvim slučajevima su pogodni Grejevi brojači jer se kod njih susedna stanja razlikuju samo za jedan bit pa ne može doći do pojave gličeva.



Slika A.2.5.3. – Struktura četvorobitnog dvosmernog dekadnog brojača



Slika A.2.5.4. – Primer pojave gliča