**DIGITALNA LOGIKA**

**Boolova algebra**

* Matematična disciplina, ki se uporablja za načrtovanje in analizo obnašanja digitalnih vezij v digitalnih računalnikih in drugih digitalnih sistemih
* Poimenovana po Georgeu Booleu

• Angleški matematik

• Predlagana osnovna načela algebre leta 1854

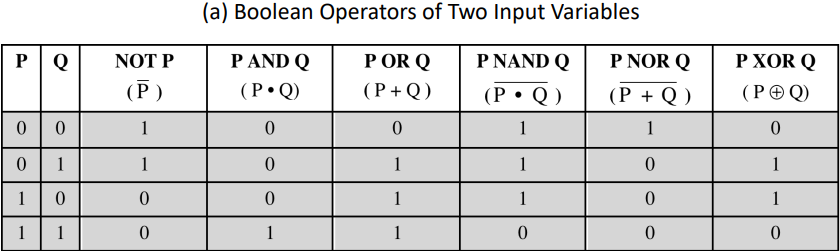
* Claude Shannon (1938) je predlagal Boolovo algebro za reševanje težav pri zasnovi relejno-stikalnih vezij
* Je priročno orodje za:

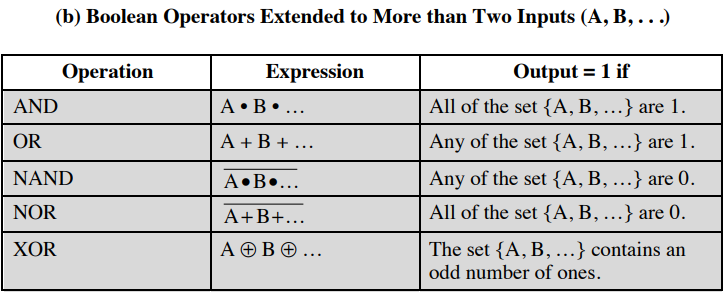
• Analizo ¡ Ekonomičen način opisovanja funkcije digitalnega vezja

• Oblikovanje ¡ Glede na želeno funkcijo je mogoče uporabi = logično algebro za razvoj poenostavljenega izvajanja te funkcije

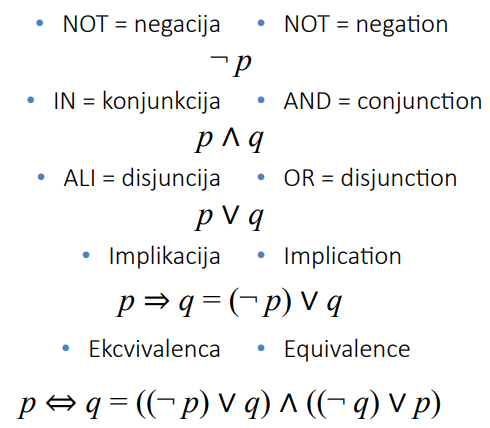
**Osnove Boolove algebre**

* Uporablja logične spremenljivke (operande) in operacije
* Spremenljivke lahko prevzamejo vrednost 1 (TRUE - pravilno) ali 0 (FALSE - napačno)
* Osnovne logične operacije so AND, OR in NOT
* IN ( x∧y, x AND y, Kxy, x\*y, x ● y, xy)
* Rezultat je pravilen, če in samo, če sta oba njegova operanda pravilna
* Če nimamo oklepajev, ima operacija AND prednost pred operacijo OR
* Kjer ni možna dvoumna razlaga, bo operacija AND predstavljena s preprosto povezavo (xy) namesto drugega zapisa operacije
* ALI (x∨y, x OR y, Axy, x+y)
* Rezultat je pravilen, če je eden ali pa sta oba operanda pravilna
* NE (¬x, NOT x, Nx, x, x', !x ̅ )
* Spremeni vrednost operanda

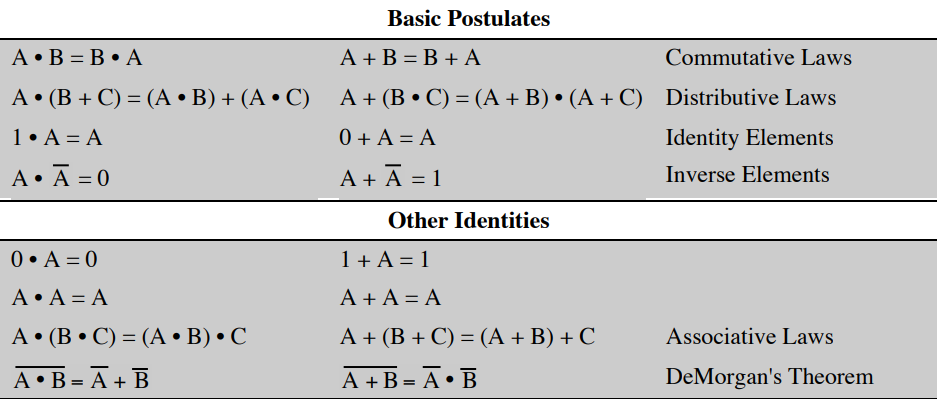
**Resničnostna tabela**

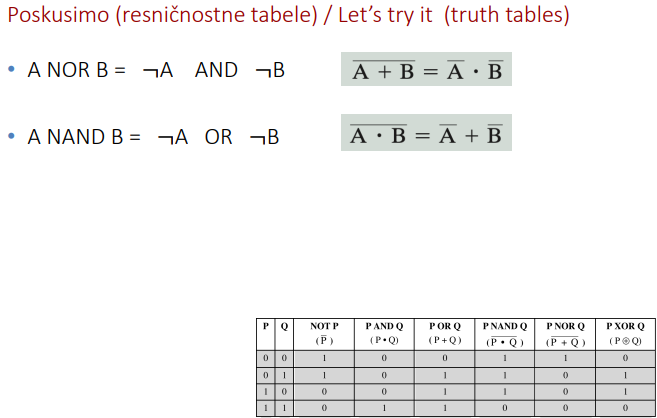
****

**Druga imena, operacije**

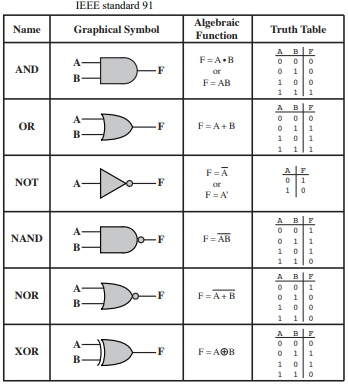
****

**Osnovna pravila Boolove algebre**

****

****

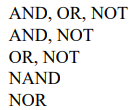
**Logična vrata**

****

* Vsa vrata razen NE imajo lahko več kot 2 vhoda.
* Ko se vhodna vrednost spremeni, se izhodna vrednost spremeni z zamudo zaradi širjenja signala = zakasnitev vrat
* Potrditev signala = prehod iz 0 v 1
* Stanje 1 je bodisi predstavljeno z visoko ali nizko napetostjo, odvisno od izvedbe vezja

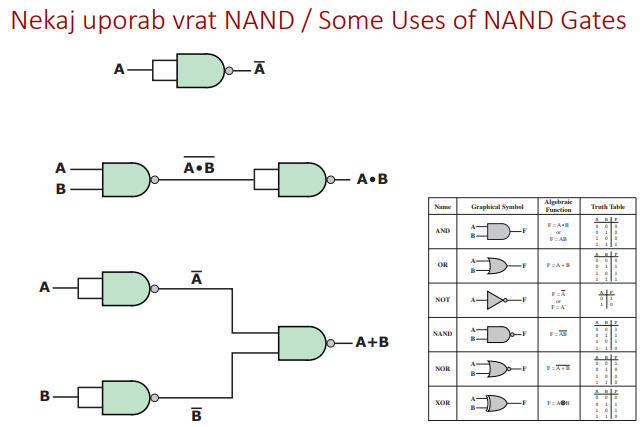
**Funkcionalno polna množica**

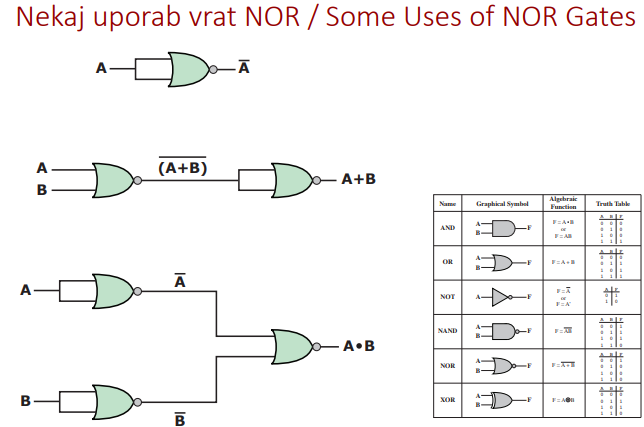
* Katero koli Booleanovo funkcijo je mogoče zapisati z uporabo funkcij v polni množici
* Funkcionalno polne množice

****

* Poskusimo (resničnostne tabele)





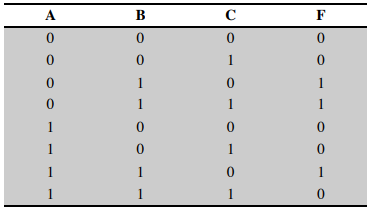


**Kombinacijska vezja**

* Medsebojno povezan niz vrat, katerih izhod je v danem trenutku funkcija vhoda v tistem času
* Pojavu vhoda takoj sledi povaj izhoda, pri čemur pride do zakasnitve vrat
* Vsebuje n binarnih vhodov in m binarnih izhodov

Opredelimo ga lahko na tri načine:

* Resničnostna tabela
* Za vsako od 2n možnih kombinacij vhodnih signalov je navedena binarna vrednost vsakega od m izhodnih signalov
* Grafični simboli
* Upodobljena kot medsebojno povezana vrata
* Boolove enačbe
* Vsak izhodni signal je izražen kot logična funkcija njegovih vhodnih signalov

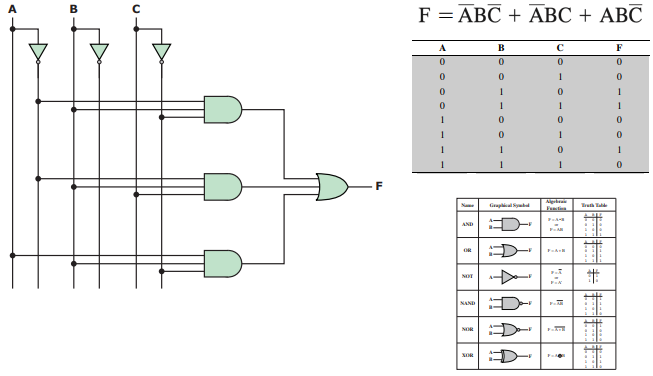
****

Primer

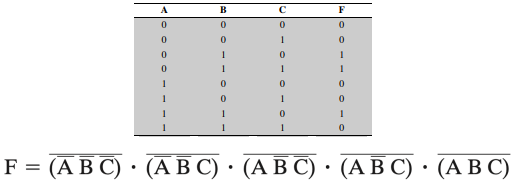
* Boolova enačba (funkcija) ali vsota produktov (SOP)

****

* Grafična predstavitev

****

* Namesto SOP lahko tabelo izrazimo z negacijo vseh možnih kombinacij, kjer je F = 0



* Uporabimo De Morganov teorem



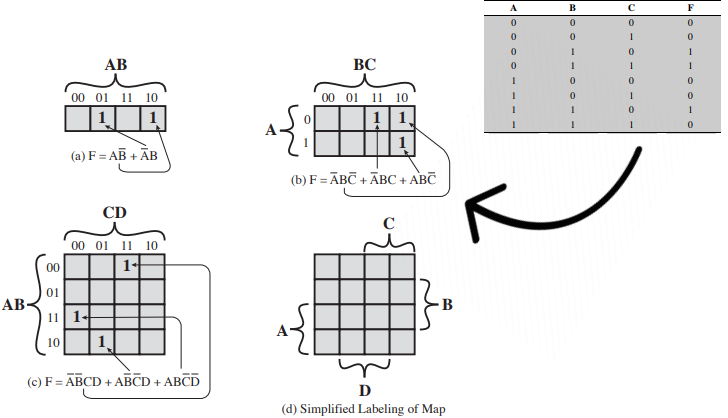
* … (nadaljujemo in dobimo produkt vsot)
* Grafična predstavitev POS s simboli (prezentacija)

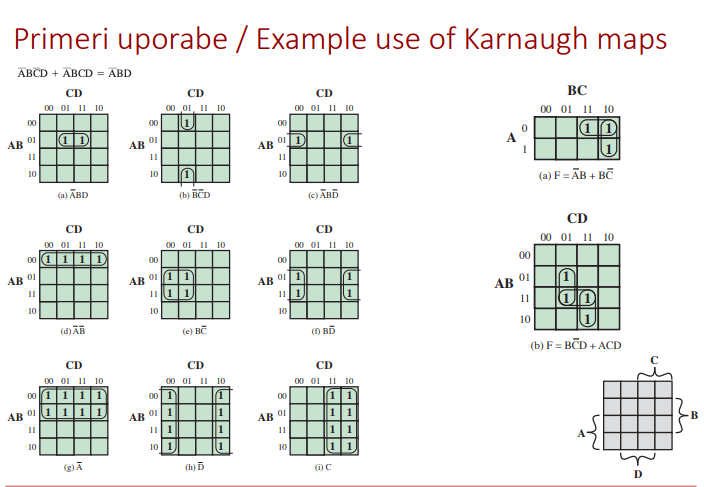
Poenostavitevene metode

* Algebrska poenostavitev
* Karnaughovi diagrami
* Quine–McCluskey tabele
* (NAND in NOR izvedba)

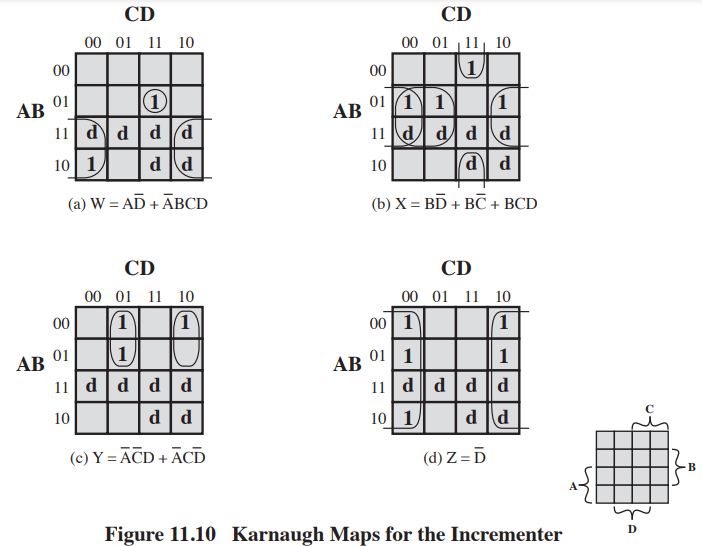
Algebrska poenostavitev (prezentacija)

* **Karnaughovi diagrami**



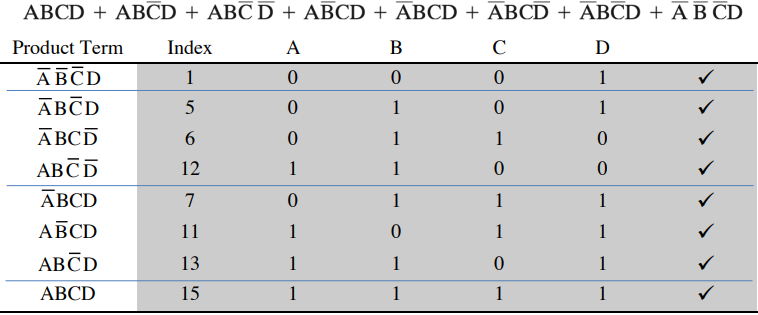


**Dvojiško kodirano desetiško število**



**Quine-McCluskey Metoda**

* Za več kot 4 spremenljivke

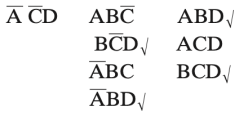
****

1. Združimo po številu negiranih spremenljivk

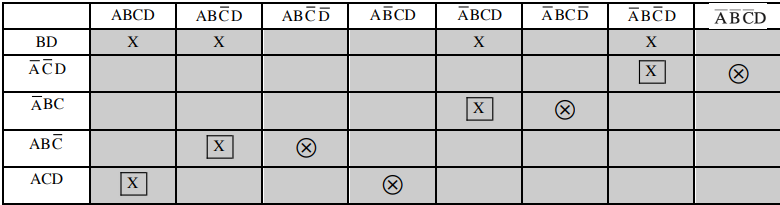
2. Primerjamo vsako vrstico z vsako vrstico v zgornji skupini

3. Če se vrstici razlikujeta le v eni spremenljivki, jo zapišemo brez te spremenljivke in označimo obe vrstici

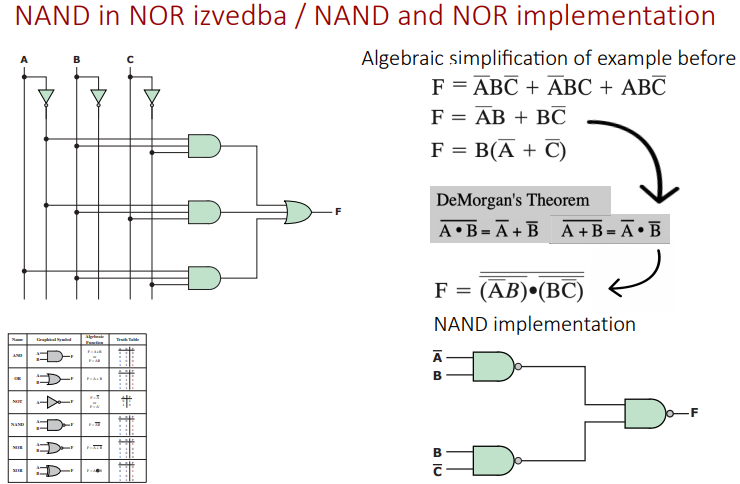
* **Drugi, tretji … korak**
* Združimo, primerjamo, zapišemo in označimo tako kor prej

****

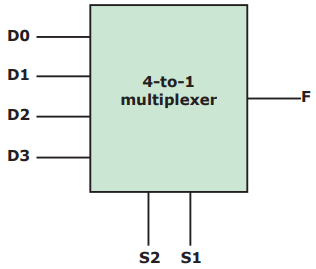
* Nadaljujemo skozi naslednje tabele, dokler ni bila izdelana tabela brez ujemanj.
* **Zadnji korak**

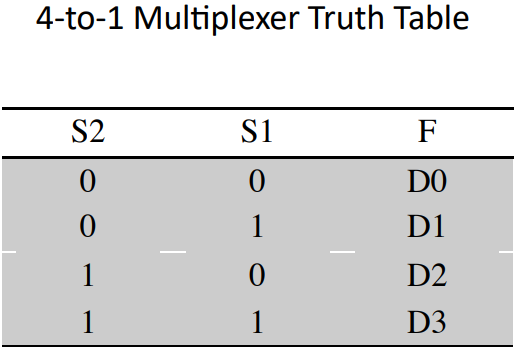
****

****

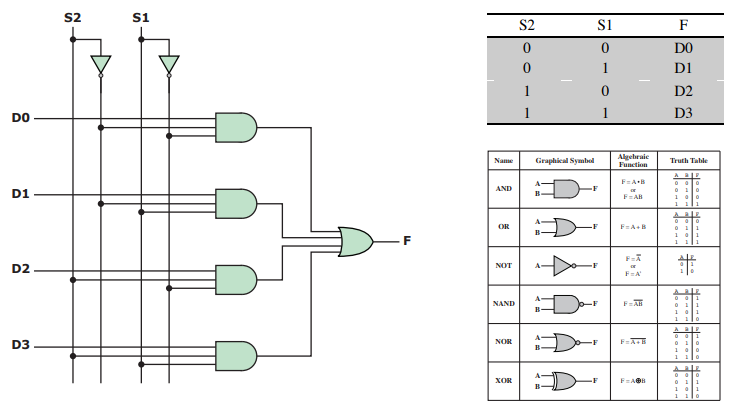
****

* **Kombinacijska vezja – Primeri**
* Multiplekser
* Dekodirnik (razgonetilnik)
* Pomnilnik samo za branje (ROM)
* Seštevalci
* **Multiplekser**

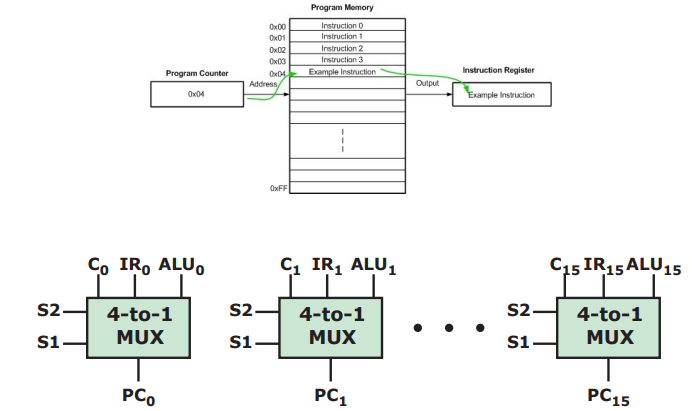
****

****

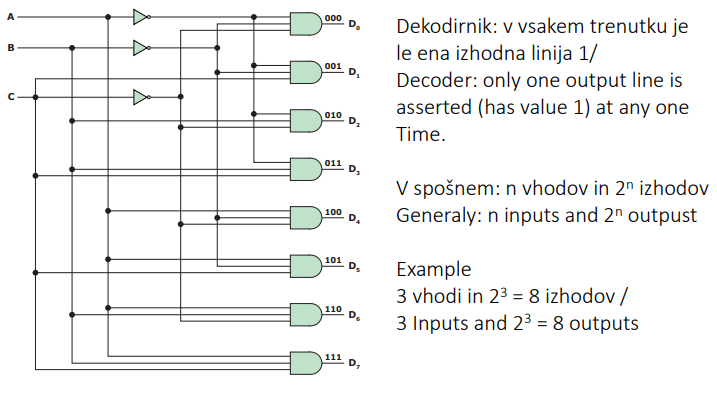
* **Izvedba**

****

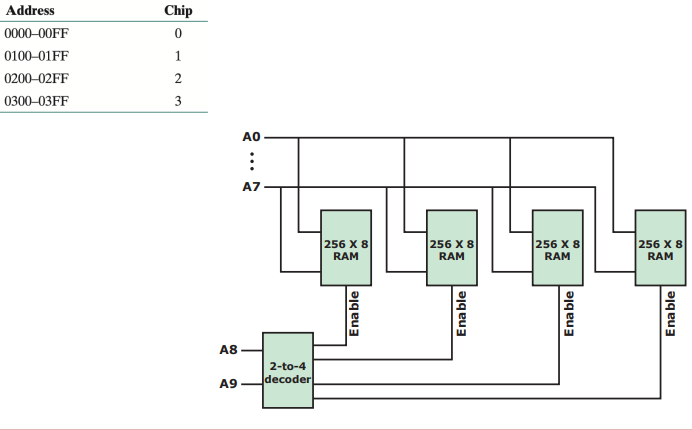
* **Multiplexer Input to Program Counter**

****

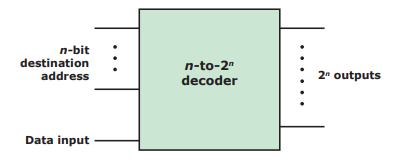
* **Dekodirnik**

****

* **Dekodirnik naslovov**

****

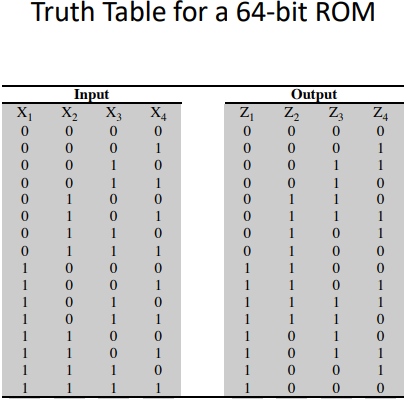
* **Demultiplekser z uporabo dekoderja**

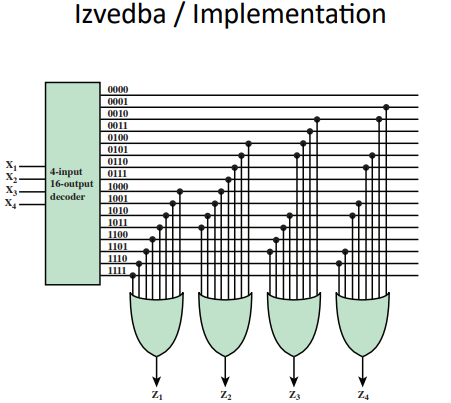
****

* Demultiplekser opravlja inverzno funkcijo multipleksorja; povezuje en vhod z enim od več izhodov.
* Vse 2n izhodne vrstice so povezane z vhodno linijo podatkov. Tako n vhodov deluje kot naslov za izbiro določene izhodne vrstice, vrednost v vhodni vrstici podatkov (0 ali 1) pa se usmeri v to izhodno vrstico.
* **Read-Only Memory (ROM)**

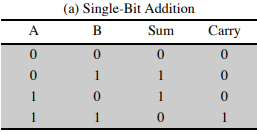
Pomnilnik zgrajen s kombinacijskimi vezji

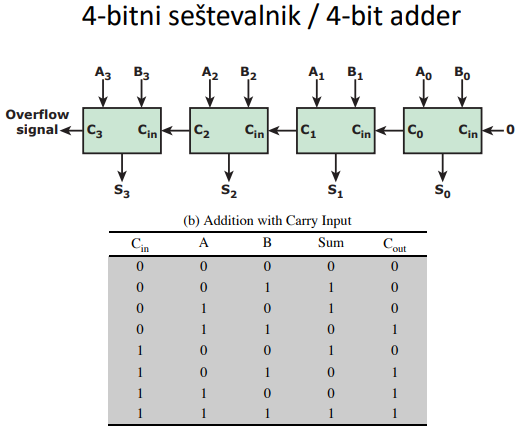
* Kombinacijska vezja pogosto imenujemo "brezpomnilna" vezja, ker je njihov izhod odvisen samo od njihovega trenutnega vhoda in zgodovina predhodnih vhodov ni ohranjena
* Pomnilniška enota, ki izvaja samo postopek branja
* Binarni podatki, shranjeni v ROM-u, so stalni in se ustvarjajo med postopkom izdelave
* Dani vhod v ROM (naslovne vrstice) vedno ustvari enak izhod (podatkovne vrstice)
* Ker so izhodi samo funkcija sedanjih vhodov, je ROM kombinacijsko vezje

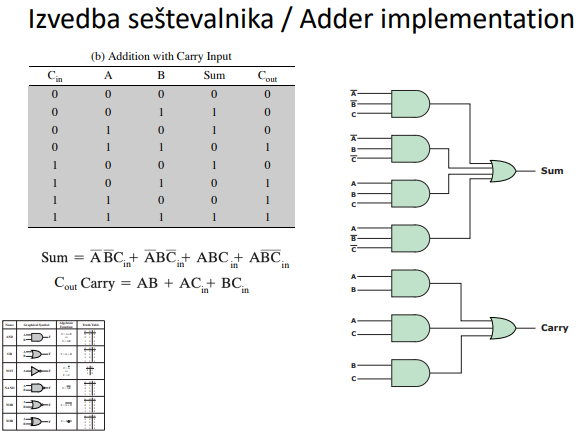
****

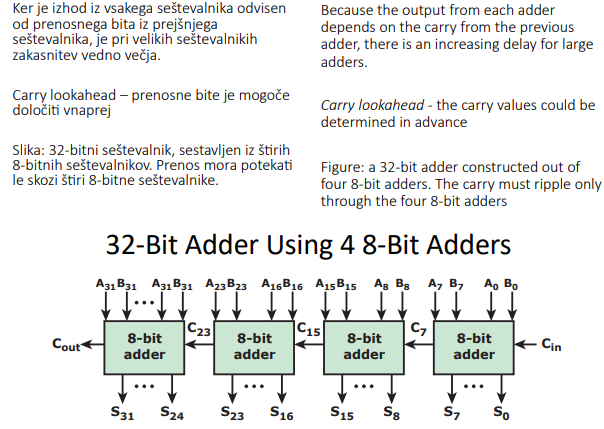
****

* **Seštevalniki**
* Resničnostna tabela 1-bitnega binarnega seštevanja

****

****

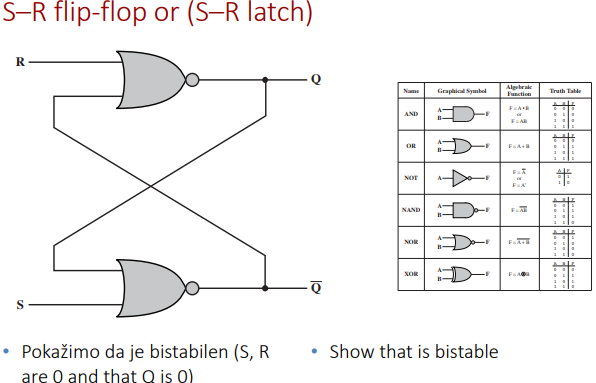
****

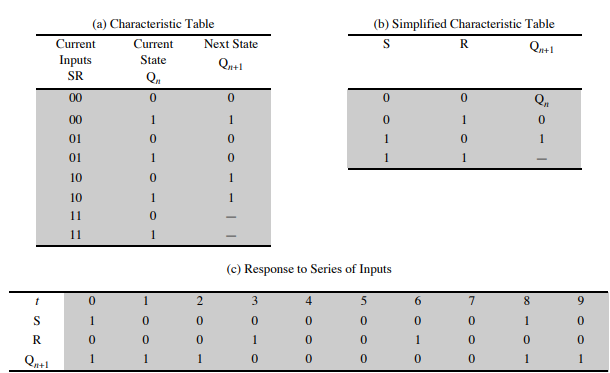
****

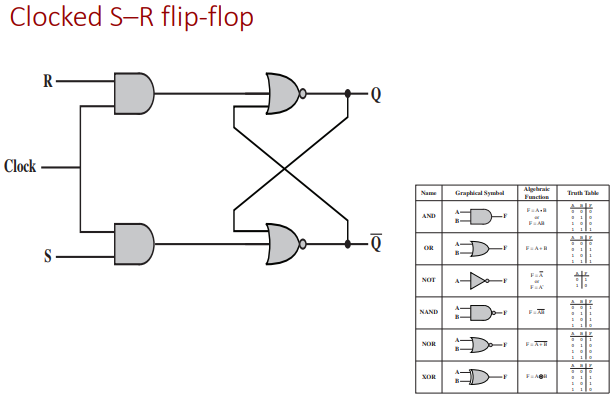
* **Zaporedno vezje**
* Trenutni izhod ni odvisen samo od trenutnega vhoda, ampak tudi od pretekle zgodovine vhodov – to je od trenutnega vhoda in on trenutnega stanja vezja.
* Uporablja kombinacijska vezja
* **Flip-flopi (bistabilna vezja) / Flip-Flops**
* Najenostavnejša oblika zaporednega vezja
* Obstajajo različni flipl-flopi, ki si delijo dve lastnosti:

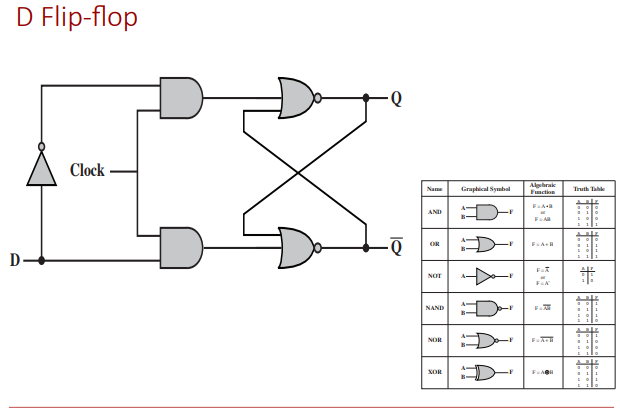
1. Flip-flop je bistabilna naprava. Obstaja v enem od dveh stanj in, če vnosa ni, ostane v tem stanju. Tako lahko flip-flop deluje kot 1-bitni pomnilnik.

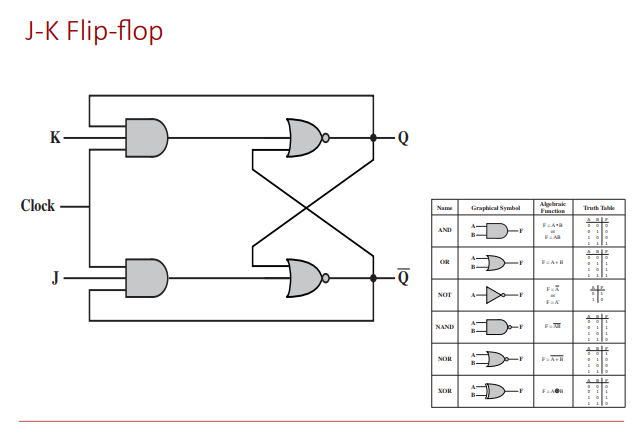
2. Flip-flop ima dva izhoda, ki sta vedno dopolnitvi drug drugega.

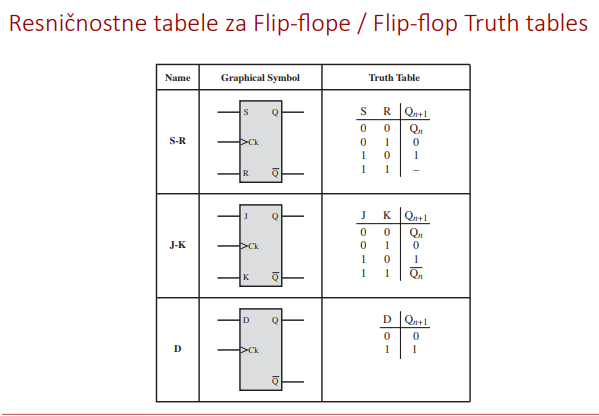
****

****

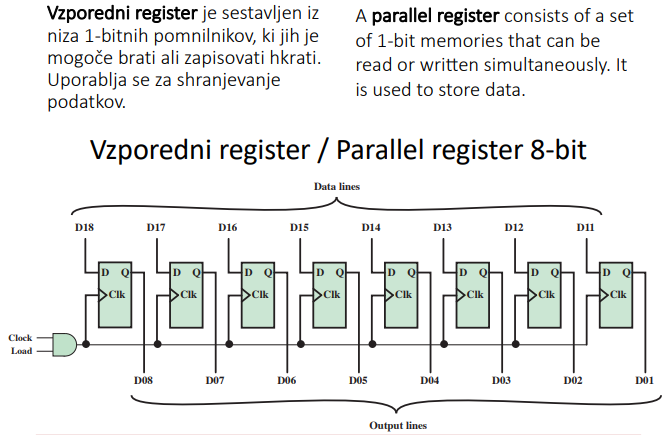
****

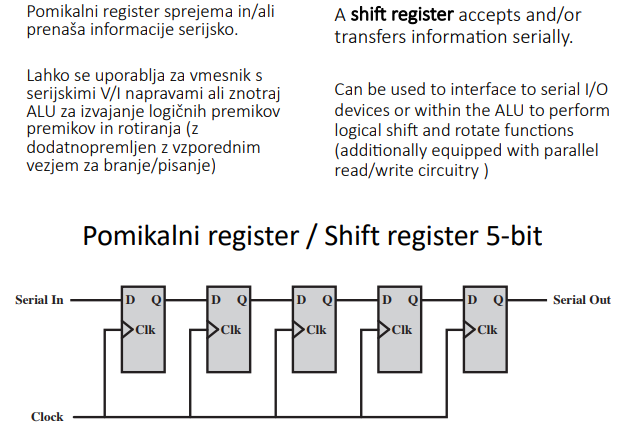
****

****

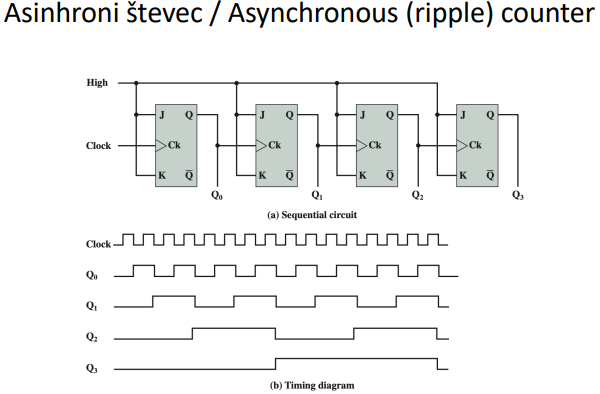
****

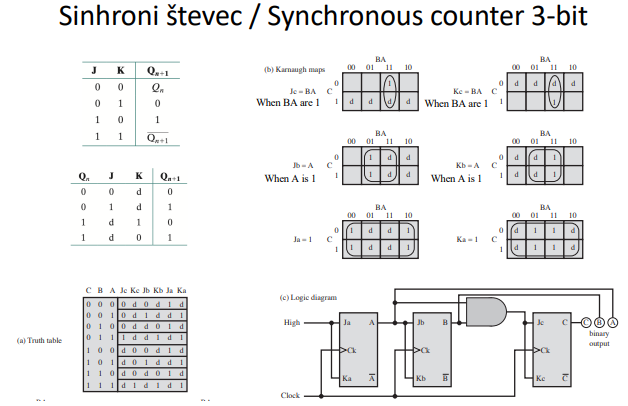
* **Uporaba Flip-Flopov**
* Registri
* Vzporedni registri
* Pomikalni registri (registri za pomik)
* Števci
* Valovni števec
* Sinhroni števec

****

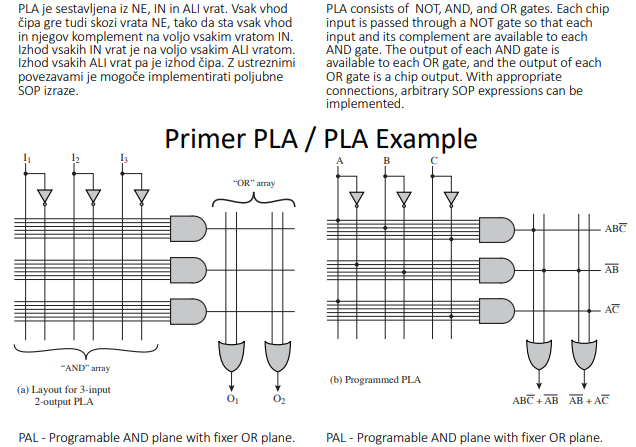
****

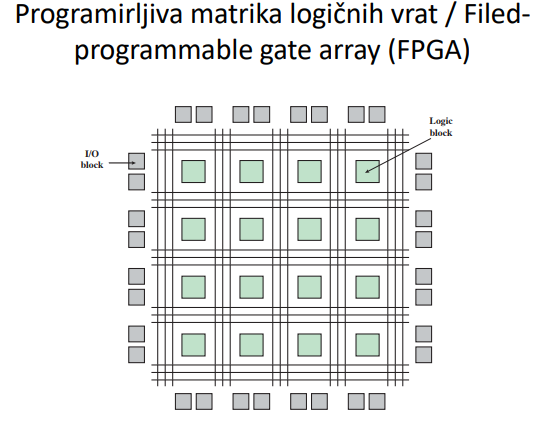
* **Števec**
* Register, katerega vrednost se povečuje za 1 modul zmogljivost registra
* Ko je dosežena največja vrednost, naslednji prirast nastavi vrednost števca na 0
* Primer števca v CPU-ju je programski števec
* Lahko je izveden:
* Asinhrono
* Sorazmerno počasen, ker izhod enega flip-flopa sproži spremembo stanja naslednjega flip-flopa
* Sinhrono
* Hkratno vse flip-flopi spremenijo stanje ¡ Ker je hitrejši, je takšen, ki se uporablja v procesorjih

****

****

* **Programirljiva logična naprava**
* Ena ali nekaj vrat na čipu zahteva veliko čipov. Z integracijo lahko damo več čipov na vezje in dodamo še povezave med njimi – ceneje, manjše, hitreje.
* Taki čipi narejeni po meri so dragi. Zato se splača izdelati splošne čipe, ki jih imenujemo Programirljiva logična naprava
* Preprosta
* Programirljivo logično polje (PLA)
* Programirljivo polje logike (PAL)
* Zapletena PLD
* Programirljiva matrika logičnih vrat

****

****