Увод у организацију и архитектуру рачунара 2

Александар Картељ

kartelj@matf.bg.ac.rs

Напомена: садржај ових слајдова је преузет од проф. Саше Малкова

О курсу

Тематика и литература

Архитектура рачунара је...

- "... наука о повезивању хардверских компоненти ради остваривања постављених циљева"
- "... концептуални пројекат и основна оперативна структура рачунара"

План курса

- Логичке основе обраде података
  - Булова алгебра и алгебра логике
  - Логичке функције, методе минимизација
  - Комбинаторне и секвенцијалне мреже
  - Основни градивни блокови рачунара

- Структура савремених дигиталних рачунара
  - Основна организација
  - Архитектура компоненти
    - Процесор
    - Меморије
    - Улазно-излазни подсистеми
  - Начини имплементације
  - Различите архитектуре

Литература за курс

- *Ненад Митић,* ***Увод у организацију рачунара****, Математички факултет, 2009.*
- *Andrew Tanenbaum,* ***Архитектура и организација рачунара****, Микро књига, 2007.*
- *Ненад Митић,* ***Основи рачунарских система****, Математички факултет, 2002.*
- *Sivarama Dandamudi,* ***Fundamentals of Computer Organization and Design****, Springer, 2002.*
- Веб локација наставника: *www.matf.bg.ac.rs/~kartelj*
- Веб локација асистента: *www.matf.bg.ac.rs/~nikola_milev*

Основи дигиталне логике

Булова алгебра

Алгебра логике

- Алгебра логике је структура { S; $\wedge$, $\vee$, ¬ }
  - Константе: S={ $\top$, $\bot$ }
  - Унарна операција: ¬
  - Бинарне операције: $\wedge$, $\vee$

Алгебра логике (2)

- Рачунарска нотација:
  - Константе: 0, 1
  - Променљиве: A, B, C, ...
  - Операције: ~A (или A’...), A·B (или AB), A+B

Законитости

$AB = BA$	$A+B = B+A$	Закон комутације
$(AB)C = A(BC)$	$(A+B)+C$	Закон асоцијације
$A(B+C) = AB+AC$	$A+(BC) = (A+B)(A+C)$	Закон дистрибуције
$1 \cdot A = A$	$0+A = A$	Неутрални елемент
$A\ A' = 0$	$A+A' = 1$	Инверзни елемент

Законитости (2)

$A \cdot 0 = 0$	$A + 1 = 1$	Нула
$A \cdot A = A$	$A + A = A$	Идемпотенција
$A\,(A + B) = A$	$A + (AB) = A$	Апсорпција
$A'' = A$		Двострука негација
$(AB)' = A' + B'$	$(A+B)' = A'\,B'$	Де Морганова правила

Логичке функције

- Свака функција са доменом S^n и кодоменом S назива се **логичка функција**:

$$f: S \times S \times \ldots \times S \rightarrow S$$

Логичке функције (2)

- Колико постоји различитих логичких функција реда *n*
(са *n* аргумената)?

$$2^{2^n}$$

Логичке функције реда 0

- Логичке функције без аргумената:
  - $f_{00}(x) = 0$
  - $f_{01}(x) = 1$

Логичке функције реда 1

Argument	Vrednost		Naziv	Oznaka
A	0	1		
Funkcija				
f_{10}	0	0	nula funkcija	
f_{11}	0	1	identitet	
f_{12}	1	0	negacija	¬A
f_{13}	1	1	jedinična funkcija	

Логичке функције реда 2

Argument	Vrednost			
A	0	0	1	1
B	0	1	0	1
Funkcija				
f_{20}	0	0	0	0
f_{21}	0	0	0	1
f_{22}	0	0	1	0
f_{23}	0	0	1	1
f_{24}	0	1	0	0
f_{25}	0	1	0	1
f_{26}	0	1	1	0
f_{27}	0	1	1	1
f_{28}	1	0	0	0
f_{29}	1	0	0	1
f_{2A}	1	0	1	0
f_{2B}	1	0	1	1
f_{2C}	1	1	0	0
f_{2D}	1	1	0	1
f_{2E}	1	1	1	0
f_{2F}	1	1	1	1

Логичке функције реда 2 (2)

Argument	Vrednost				Naziv	Oznaka
A	0	0	1	1		
B	0	1	0	1		
Funkcija						
f_{20}	0	0	0	0	nula funkcija	
f_{21}	0	0	0	1	konjunkcija	$A \wedge B$
f_{22}	0	0	1	0	negacija implikacije od A ka B	$A \wedge \neg B = \neg(A \Rightarrow B)$
f_{23}	0	0	1	1	prva projekcija	
f_{24}	0	1	0	0	negacija implikacije od B ka A	$\neg A \wedge B = \neg(B \Rightarrow A)$
f_{25}	0	1	0	1	druga projekcija	
f_{26}	0	1	1	0	ekskluzivna disjunkcija	$(A \wedge \neg B) \vee (\neg A \wedge B)$
f_{27}	0	1	1	1	disjunkcija	$A \vee B$
f_{28}	1	0	0	0	Pirsova (Lukašievičeva) funk.	$A \downarrow B$
f_{29}	1	0	0	1	ekvivalencija	$(A \Leftrightarrow B)$
f_{2A}	1	0	1	0	negacija druge projekcije	
f_{2B}	1	0	1	1	implikacija od B na A	$B \Rightarrow A$
f_{2C}	1	1	0	0	negacija prve projekcije	
f_{2D}	1	1	0	1	implikacija od A na B	$A \Rightarrow B$
f_{2E}	1	1	1	0	Šeferova funkcija	$A \uparrow B$
f_{2F}	1	1	1	1	Jedinična funkcija	

Пун систем функција

- Пун систем функција је скуп функција на основу кога се могу извести све остале функције

- Ако се из неког система функција могу извести све функције неког пуног система функција, онда је и тај систем пун систем функција

Неки пуни системи функција

- { $\wedge$, $\vee$, $\neg$ }
  - проверити

Неки пуни системи функција (2)

- { $\wedge$, $\neg$ }
  - проверити
- { $\vee$, $\neg$ }
  - показати
- { $\uparrow$ }
  - показати
- { $\downarrow$ }
  - показати

Нормалне форме функција

- Елементарна конјункција
  - логички израз који не садржи дисјункцију
  - тј. садржи само негацију и конјункцију
  - пример:
    - A B C’ D E’

- Елементарна дисјункција
  - логички израз који не садржи конјункцију
  - тј. садржи само негацију и дисјункцију
  - пример:
    - A + B’ + C + D’ + E

Нормалне форме функција (2)

- Савршена елементарна конјункција
  - елементарна конјункција која садржи све променљиве из скупа променљивих (обично “све аргументе функције”)

- Савршена елементарна дисјункција
  - елементарна дисјункција која садржи све променљиве из скупа променљивих (обично “све аргументе функције”)

Нормалне форме функција (3)

- Дисјунктивна форма
  - логички израз који се састоји од елементарних конјункција међусобно повезаних операцијама дисјункције
- Конјунктивна форма
  - логички израз који се састоји од елементарних дисјункција међусобно повезаних операцијама конјункције

Нормалне форме функција

- Савршена дисјунктивна нормална форма
  - дисјунктивна форма у којој су све конјункције савршене елементарне конјункције
  - СДНФ (енгл. *SOP – sum of products*)

- Савршена конјунктивна нормална форма
  - конјунктивна форма у којој су све дисјункције савршене елементарне дисјункције
  - СКНФ (енгл. *POS – product of sums*)

Улога СКНФ и СДНФ

- Свака логичка функција се може дефинисати у облику СКНФ и СДНФ

- Који је облик бољи зависи од функције, тј. од броја 0 и 1 у резултатима

Алтернативни запис СДНФ

- Сваки улаз се посматра као један бит неозначеног целог броја
  - пример: $F(A_1, A_2, A_3)$ се посматра као запис неозначеног целог броја од 3 бита: $A_1A_2A_3=b_2b_1b_0$
- СДНФ се записује у облику $\sum(n_1,n_2,...,n_k)$, где су $n_1,n_2,...,n_k$ бројеви који одговарају записима оних савршених елементарних конјункција аргумената који се функцијом сликају у 1
  - на пример:
    - $A_1'A_2'A_3 = 001_2 = 1$
    - $A_1A_2'A_3 = 101_2 = 5$
    - $A_1'A_2'A_3 + A_1A_2'A_3 = \sum(1,5)$

Алтернативни запис СКНФ

- Слично као у случају СДНФ
- СДНФ се записује у облику $\prod(n_1,n_2,...,n_k)$, где су $n_1,n_2,...,n_k$ бројеви који одговарају записима савршених елементарних дисјункција аргумената који се функцијом сликају у 1
  - на пример:
    - $(A_1' + A_2' + A_3) * (A_1 + A_2' + A_3) = \prod(1, 5)$

Пример

A	B	C	F
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

Пример (2)

- СДНФ:
  - дисјункција производа који дају 1
  - A’B’C’ + A’B’C + AB’C’

- СКНФ:
  - негација дисјункције производа који дају 0
  - (A’BC’ + A’BC + AB’C + ABC’ + ABC)’
    = (A+B’+C)(A+B’+C’)(A’+B+C’)(A’+B’+C)(A’+B’+C’)

Основи дигиталне логике

Логичка кола и логички елементи

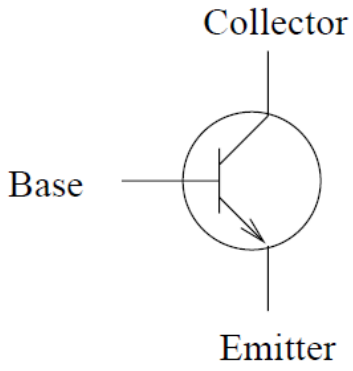
Зашто баш дигитална логика?

- Развој аутоматских рачунских уређаја заснован на дигиталним концептима је
  - једноставнији
  - ефикаснији
  - поузданији
  - јевтинији
  - ...

  него у случају аналогне логике

Транзистори

- Основна јединица имплементације дигиталног рачунара је **транзистор**
  - Емитер је извор електрона (негативни крај)
  - Колектор је сакупљач електрона (позитиван крај)
  - База је попут прекидача:
    - висок потенцијал (уобичајено изнад 2V) омогућава проток
    - низак потенцијал (уобичајено испод 0,8V) спречава проток

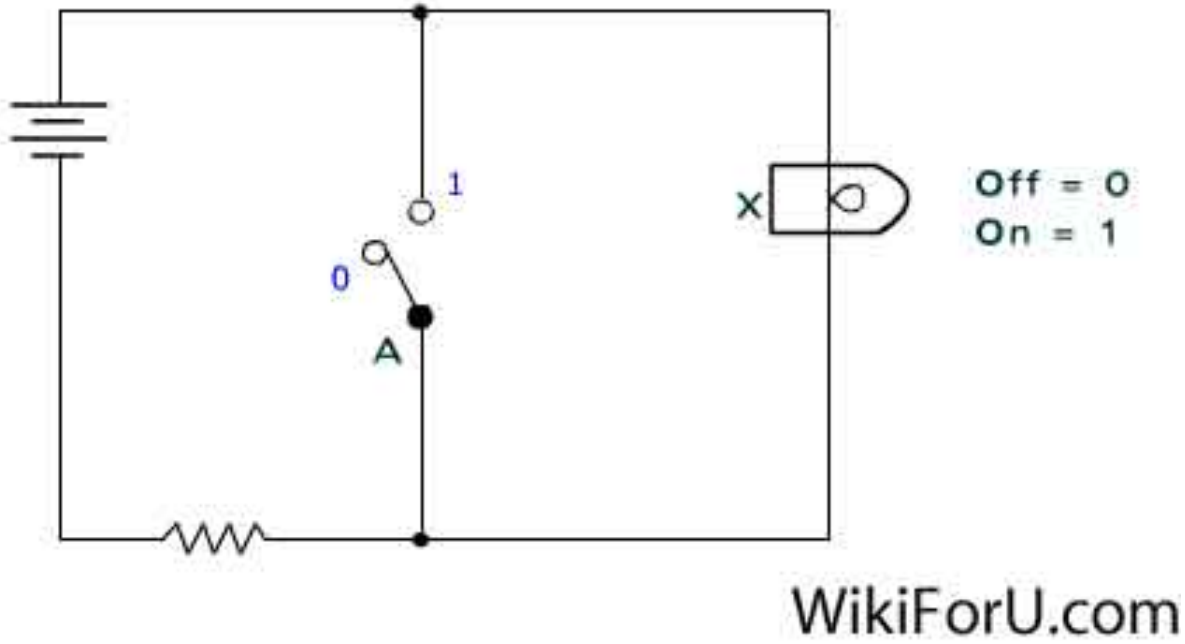


Транзистори (2)

- Уобичајено је:
  - емитери се везују за уземљење
  - извор напајања обезбеђује напон од +5V у односу на уземљење (тачан напон зависи од имплементације)
  - одсуство напона представља вредност 0
  - постојање напона представља вредност 1

Транзистори (3)

- Негација (NOT)– транзисторска и шематска репрезентација:
  - У десном дијаграму је нацртано цело коло са диодом



Логичка кола

- Логичка кола су апстрактна дигитална кола која имплементирају логичке функције
- Представљају апстракцију електричних (оптичких и других) кола

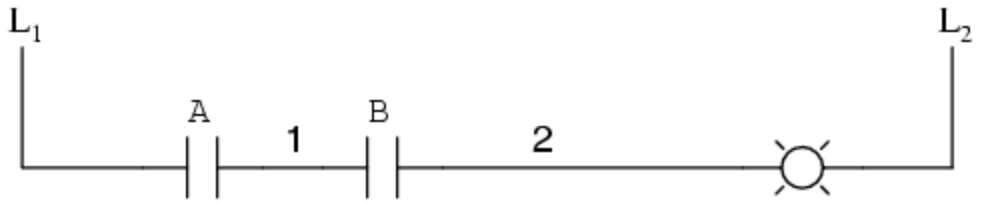
Логички елементи

- Логички елементи су елементарна дигитална кола која имплементирају елементарне логичке функције
- Обично логички елементи имплементирају функције које чине неки пун систем функција

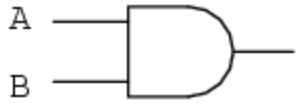
Уобичајени логички елементи

Функција	**Назив логичког елемента**	**Назив на енглеском**
Негација	НЕ-елемент	NOT-element
Конјункција	И-елемент	AND-element
Дисјункција	ИЛИ-елемент	OR-element
Шеферова функција	НИ-елемент	NAND-element
Пирсова функција	НИЛИ-елемент	NOR-element

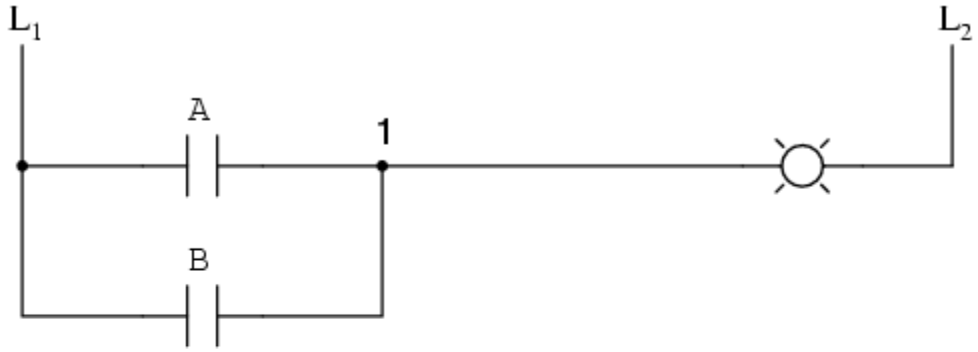
И - елемент



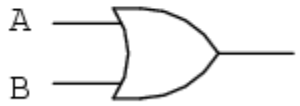
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



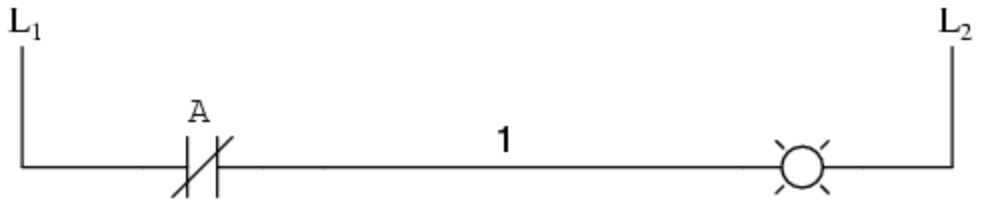
ИЛИ - елемент



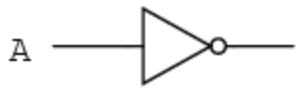
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



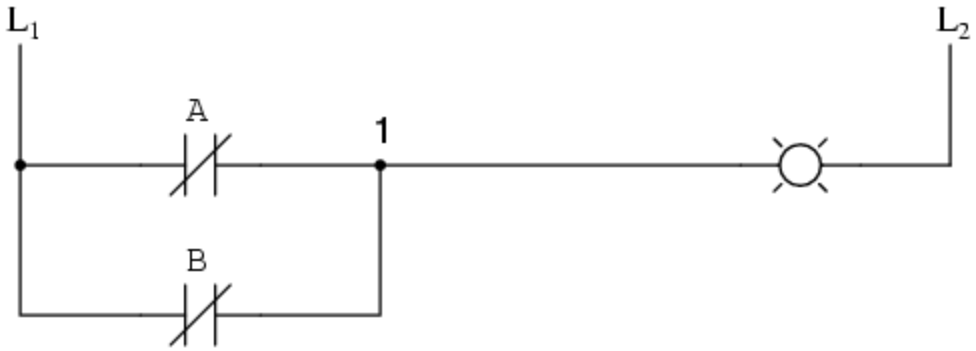
НЕ - елемент



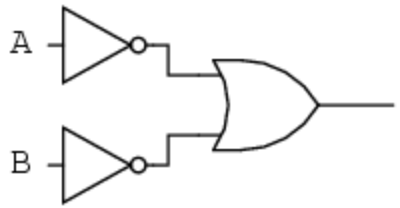
A	Output
0	1
1	0



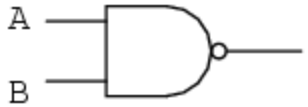
НИ - елемент



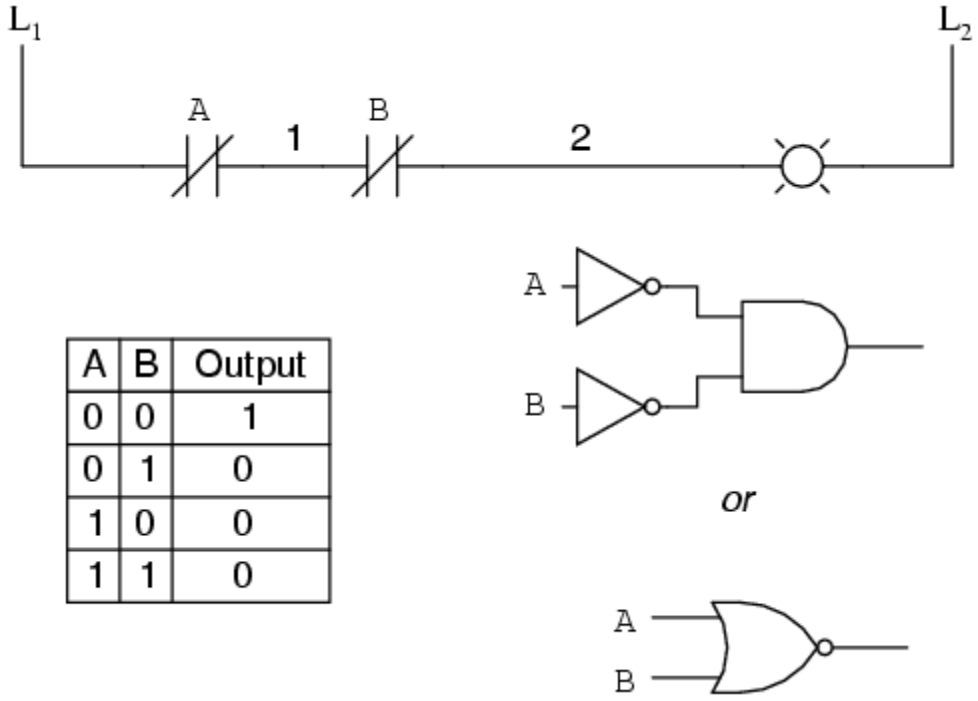
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



or

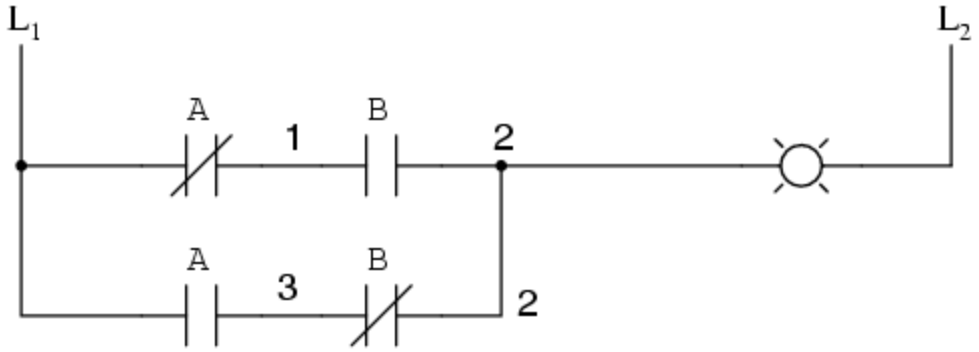


НИЛИ - елемент

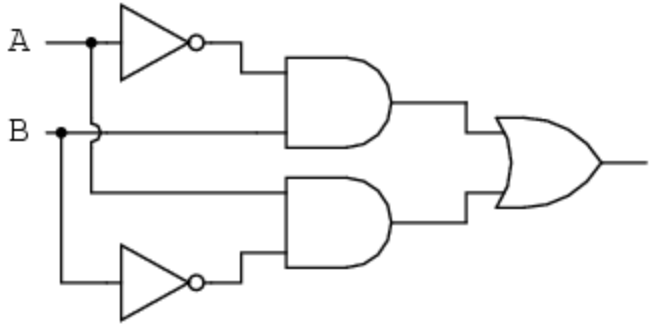


A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

ЕИЛИ (ексклузивно ИЛИ)- елемент



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

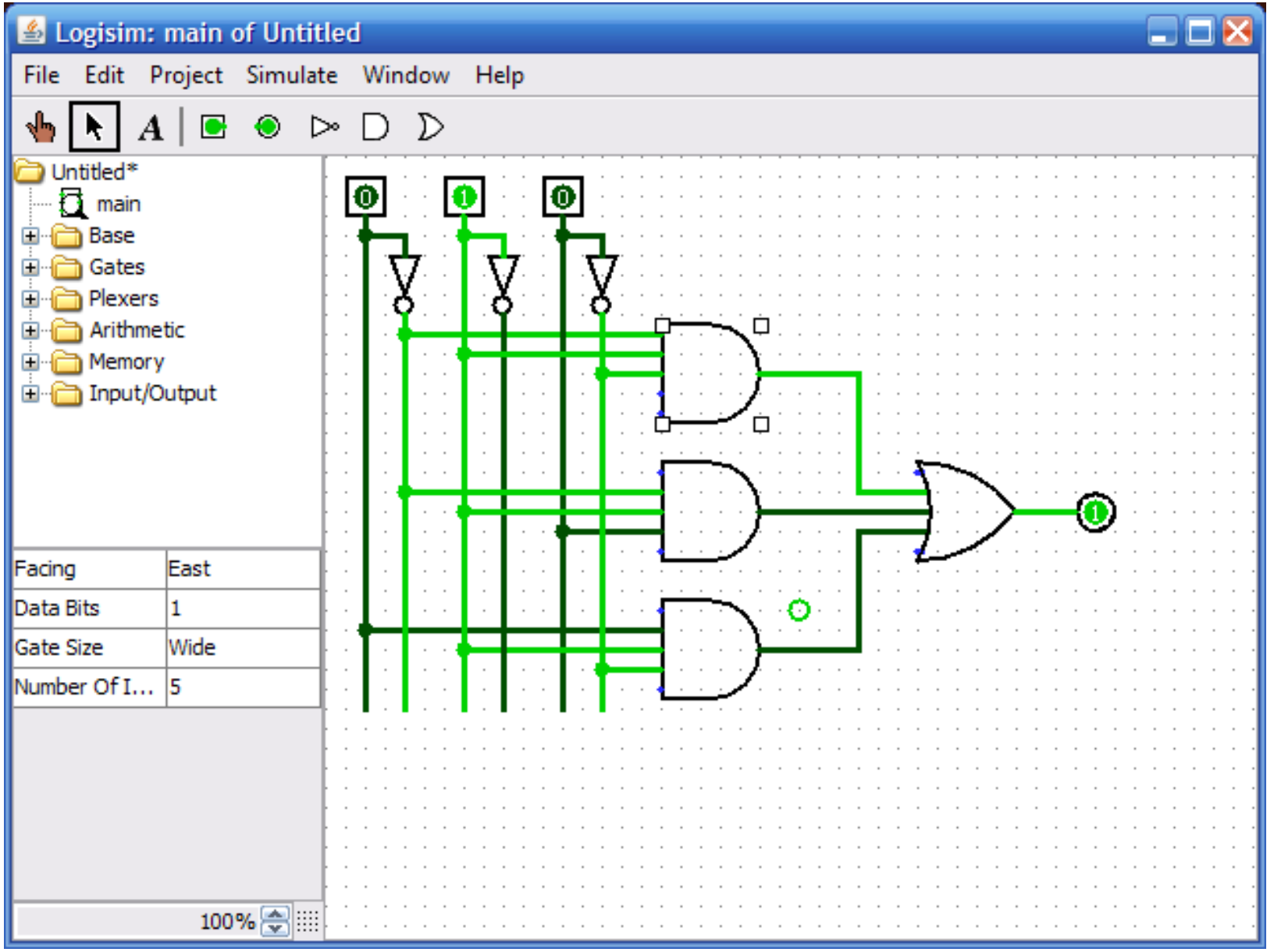


or

Софтвер

- Постоји више програма за вежбање пројектовања логичких кола и симулирање њиховог рада
  - *CEDAR Logic*
    - подржава релативно сложена кола
  - *Logisim*
    - постоји за различите ОС
    - могућност чувања и употребе модула (“кола”)
  - *Logic Circuit Designer*
    - могућност чувања и употребе модула (“кола”)
  - *Logic Gate Simulator*
    - могућност чувања и употребе модула (“кола”)

Logisim


Logisim: main of Untitled
File Edit Project Simulate Window Help
Untitled*
main
Base
Gates
Plexers
Arithmetic
Memory
Input/Output
Facing East
Data Bits 1
Gate Size Wide
Number Of I... 5
100%

Задаци

1. Направити по логичко коло за сваку од бинарних логичких функција
2. Направити логичко коло за изабрану функцију 3 променљиве
3. Направити логичко коло за изабрану функцију 4 променљиве

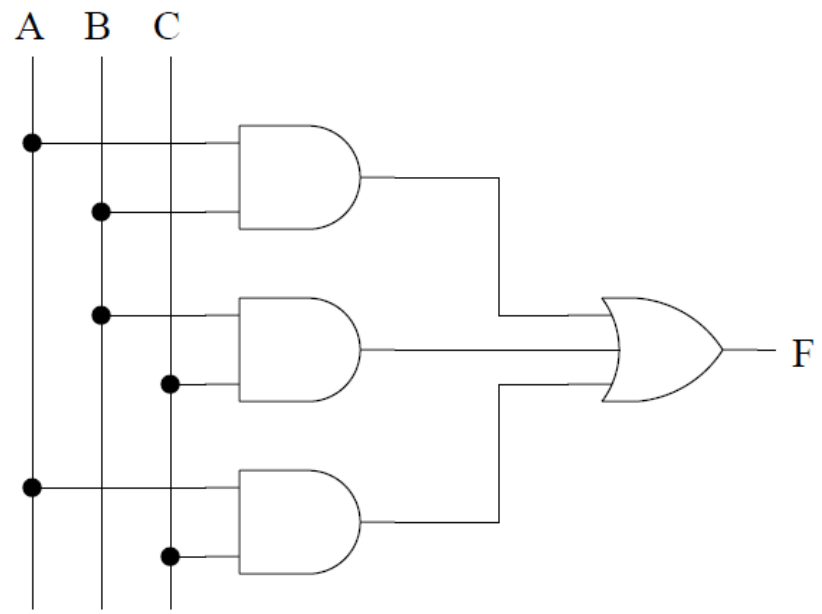
Пројектовање логичких кола

- Одређивање проблема
  - често интегрисано са наредним кораком
- Извођење истинитосне таблице
- Извођење логичког израза
  - обично СДНФ, непосредно из таблице
  - некада и без претходног корака
- Упрошћавање логичког израза
  - свођење на минималан облик
- Обликовање логичког кола

Пример

- Направити логичко коло за функцију три аргумента која израчунава вредност која је заступљенија на улазима:

A	B	C	F_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Пример (2)

- Направити логичко коло за функцију три аргумента која израчунава вредност која је мање заступљена на улазима:

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Основи дигиталне логике

Минимизација логичких функција

Минимизација логичких функција

- Свака логичка функција се може изразити на више различитих начина
- Свака логичка функција се може записати у облику СДНФ и СКНФ
- Да би се записивање и имплементација функција учинили поједноставили, потребно је минимизовати функције
  - **минимизовање логичке функције је проналажење њеног најједноставнијег записа**

Методи минимизације

- Алгебарске трансформације
- Карноове мапе
- Таблична метода Квин-Мек Класког (прескочићемо)

Алгебарске трансформације

- Почивају на примени законитости и правила идентитата Булове алгебре

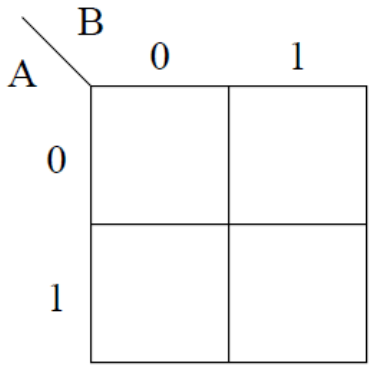
Пример

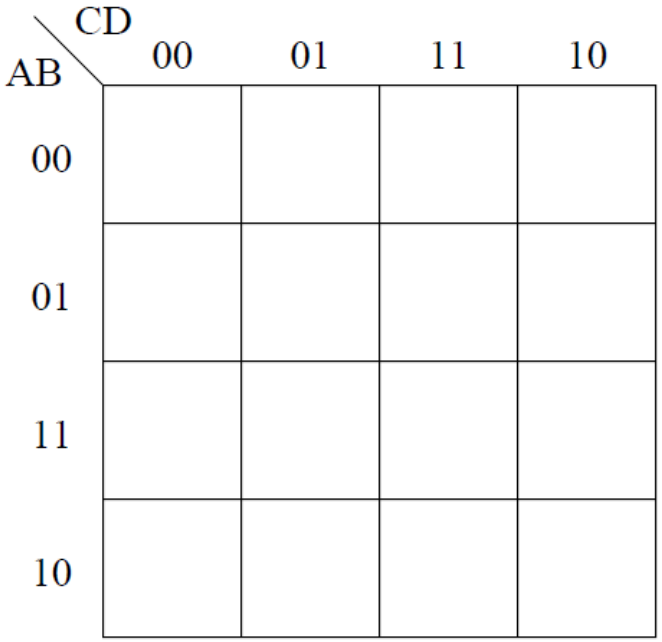
- Пример: пронаћи минималан запис функције
  - F = A’BC’ + A’BC + ABC’
- корак 1: идемпотенција и комутација
  - F = A’BC’ + A’BC + ABC’ + A’BC’
- корак 2: закон дистрибуције
  - F = A’B(C’ + C) + (A + A’)BC’
- корак 3: закон о инверзним елементима
  - F = A’B + BC’
- корак 4: закон дистрибуције
  - F = B(A’ + C’)

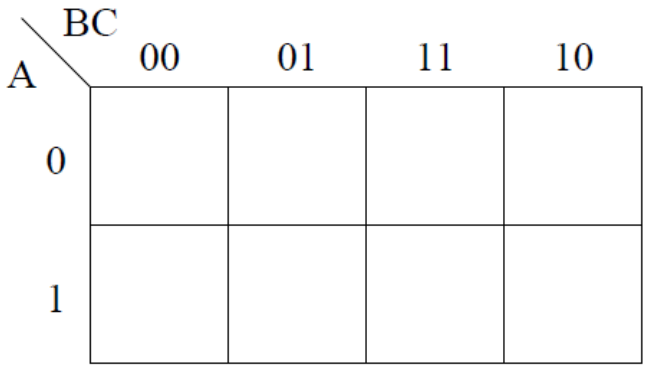
Карноове мапе

- Почивају на представљању табела у облику који се лако може оптимизовати
  - једноставно за аутоматизацију
- Основна идеја:
  - прави се вишедимензиона мрежа
  - на свакој димензији се наводе највише по два аргумента функције
  - вредности аргумената се наводе таквим редом да се мења по тачно један бит (Хамингова дистанца 1):
    - 00, 01, 11, 10

Примери







Карноове мапе (2)

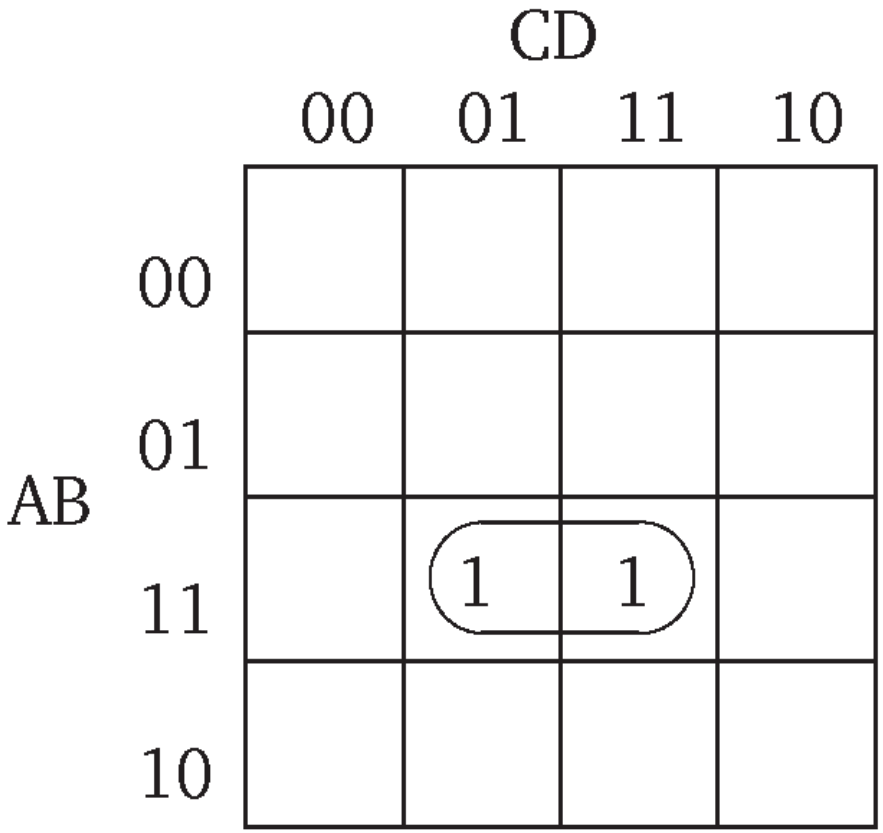
- Основу за упрошћавање представља правило:
  - $A_1A_2...A_k...A_n + A_1A_2...A_k'...A_n = A_1A_2...A_{k-1}A_{k+1}...A_n$

- Како се суседни квадрати у Карноовим мапама увек разликују за по највише један бит, ако одговарају истим вредностима функције онда се одговарајући бит може елиминисати
- И квадрати на крајњим супротним странама се сматрају за суседне

Карноове мапе (3)

- СДНФ има по дисјункт за сваки квадрат у мапи који садржи јединицу
- Ако два суседна квадрата имају садрже јединицу, два дисјункта којима они одговарају се могу заменити једним, који не садржи аргумент који се разликује
- ...слично и за четири квадрата, 4x1 или 2x2

Пример

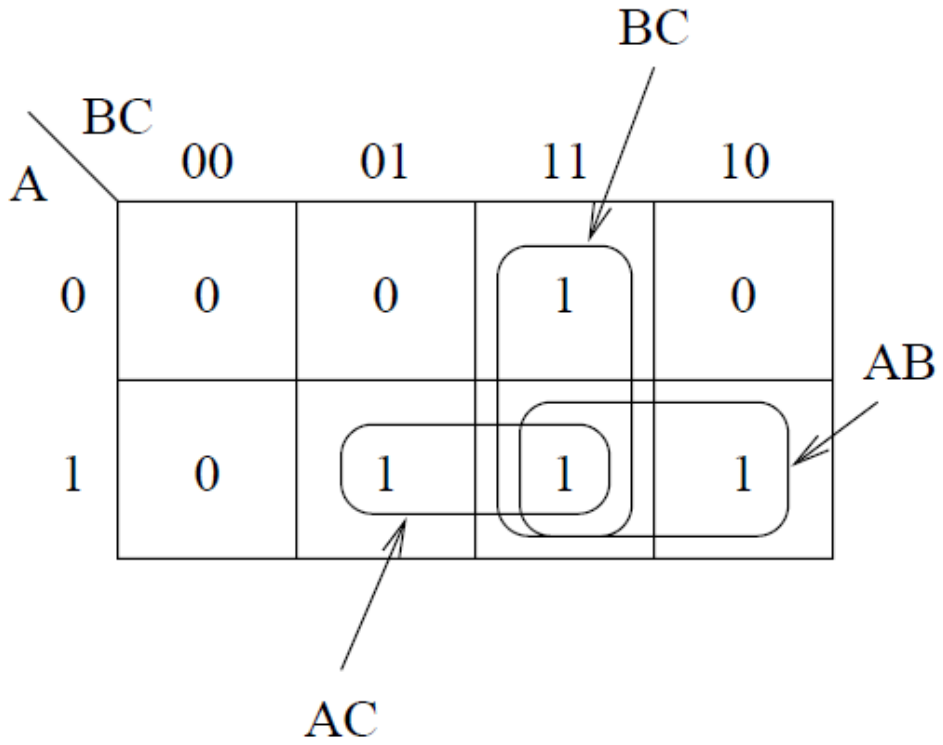


Карноове мапе (4)

- Уопштено:
  - област реда 0 је јединични квадрат
  - област реда *n*+1 је унија две суседне области реда *n*
  - ако две суседне области реда *n* садрже јединице, оне се замењују једном облашћу реда *n*+1
  - претходни корак се примењује докле год је могуће
  - почев од области највишег реда
    - ако област није у потпуности обухваћена другим областима вишег или истог реда, прави се дисјункт
      - дисјункт се састоји само од аргумената који су константни за дату област
  - провери се редундантност
  - добијени израз је минимизована функција

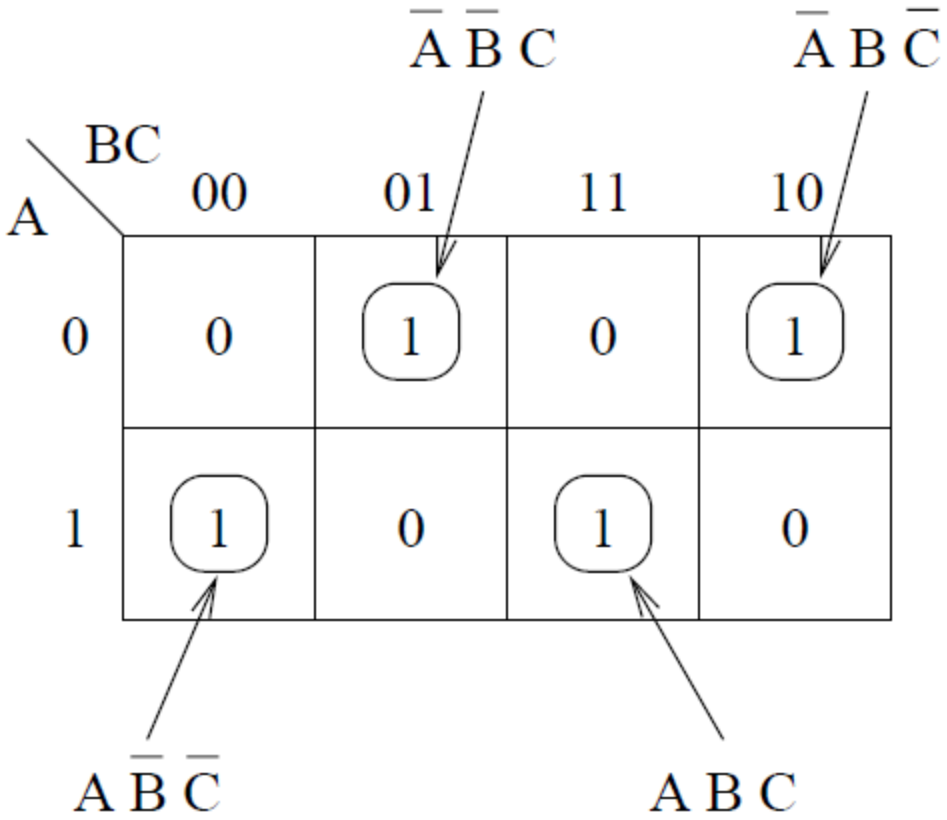
Пример

- Функција три аргумента рачуна ону вредност која је заступљенија у аргументима



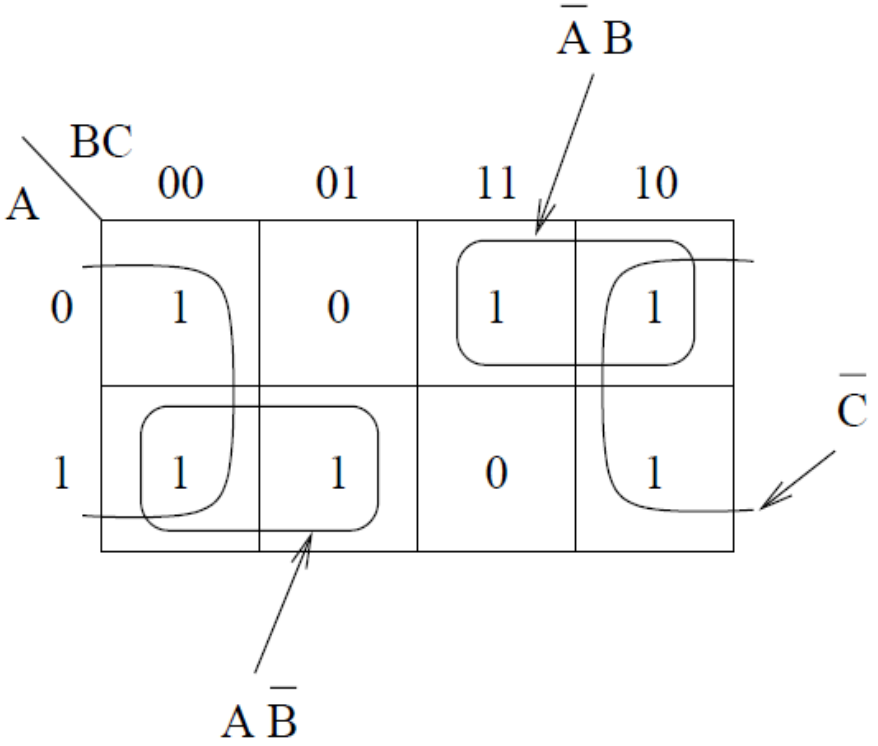
Пример

- Функција три аргумента рачуна парност

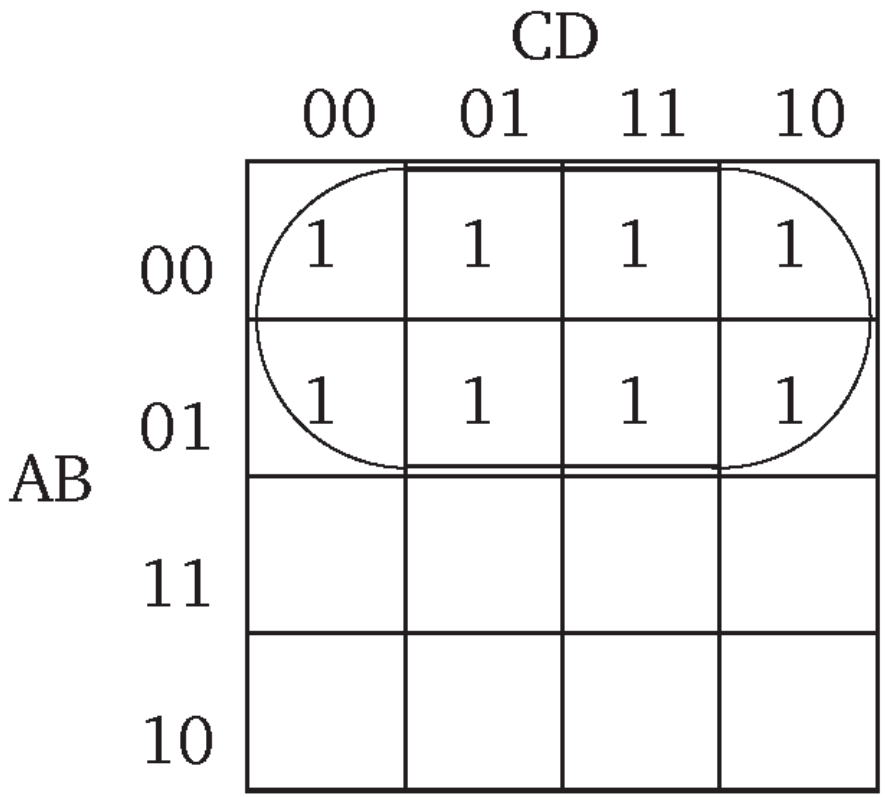


Пример

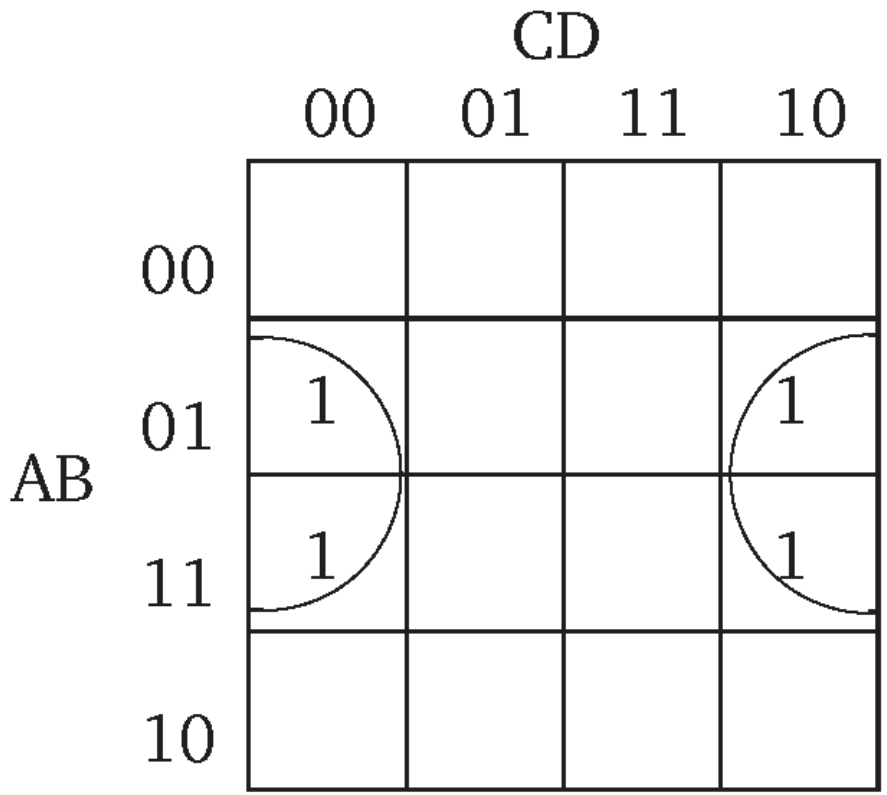
- Пример функције три аргумента код које постоје суседне области на супротним крајевима мапе



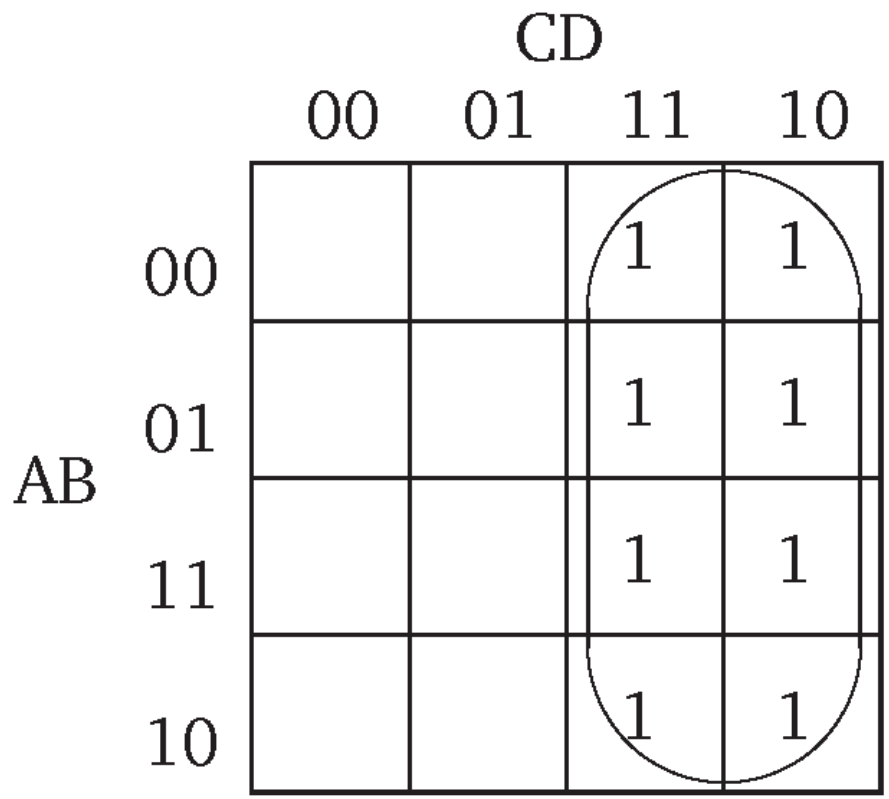
Пример



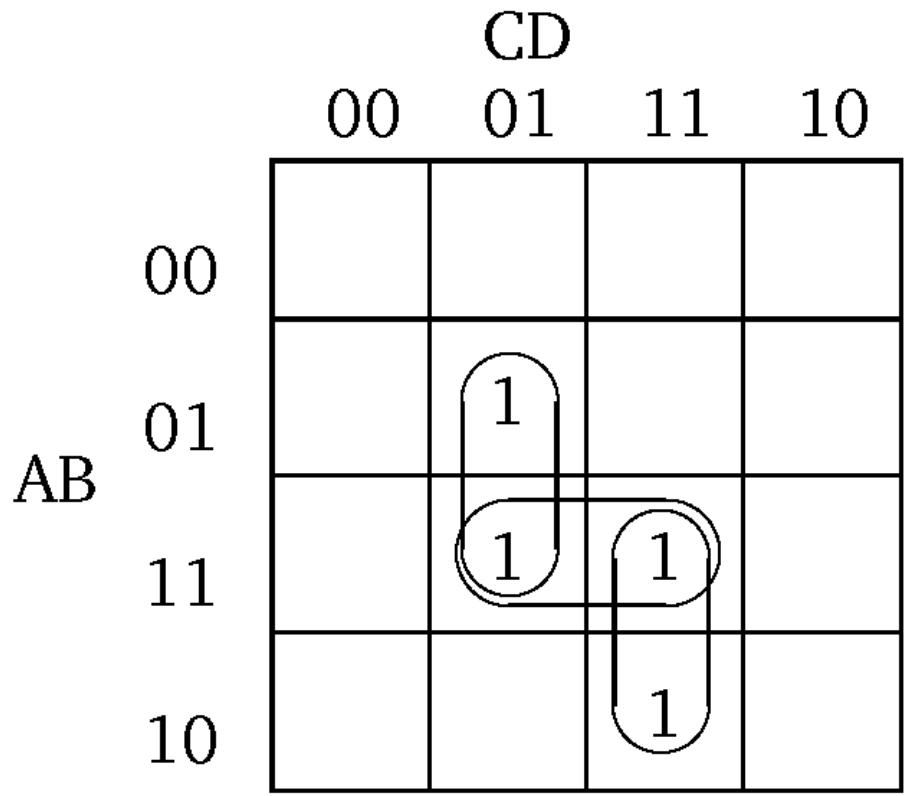
Пример



Пример



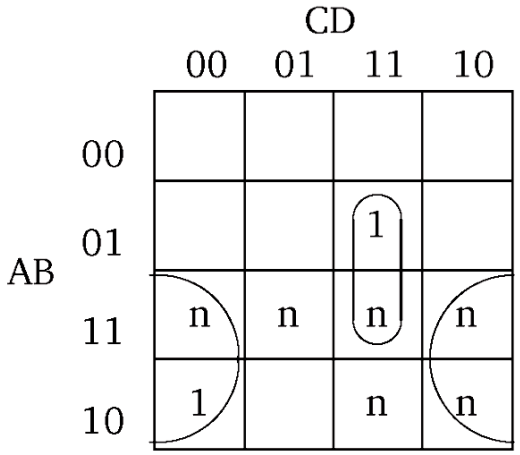
Пример



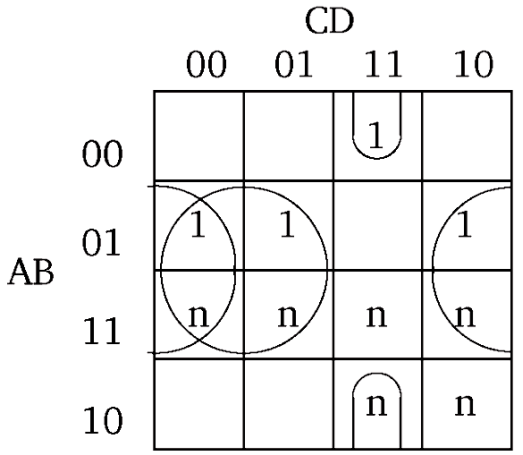
Непотпуне мапе

- Ако је функција дефинисана само за неки домен аргумената, онда се мапа може правити са три вредности: 0, 1 и недефинисано.
- Недефинисани квадрати се могу слободно употребљавати као да садрже било 1 било 0, тако да се добије мање дисјунката

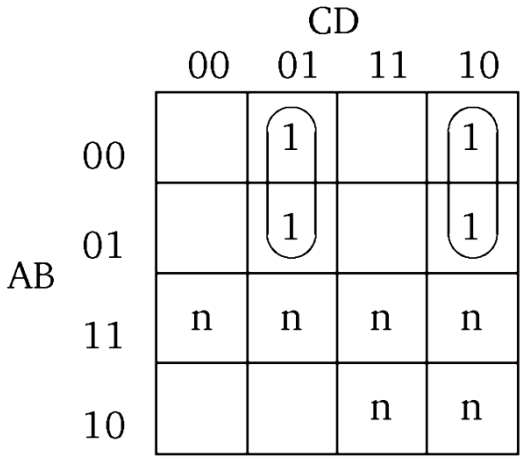
Непотпуне мапе (2)



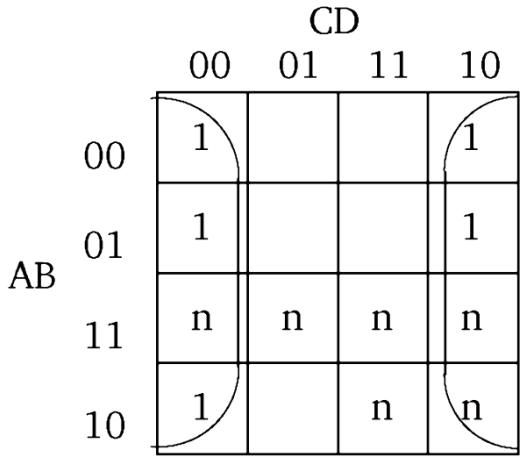
a) $W = A\overline{D} + BCD$



b) $X = B\overline{D} + B\overline{C} + \overline{B}CD$

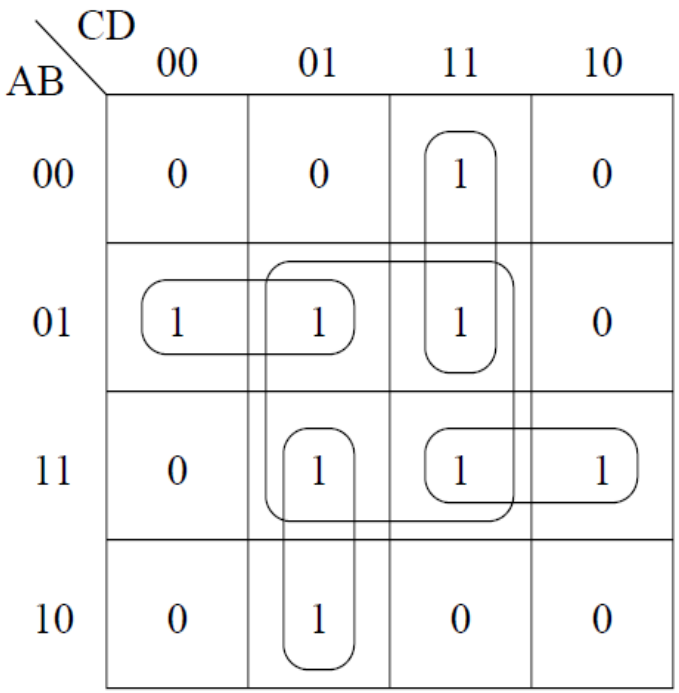


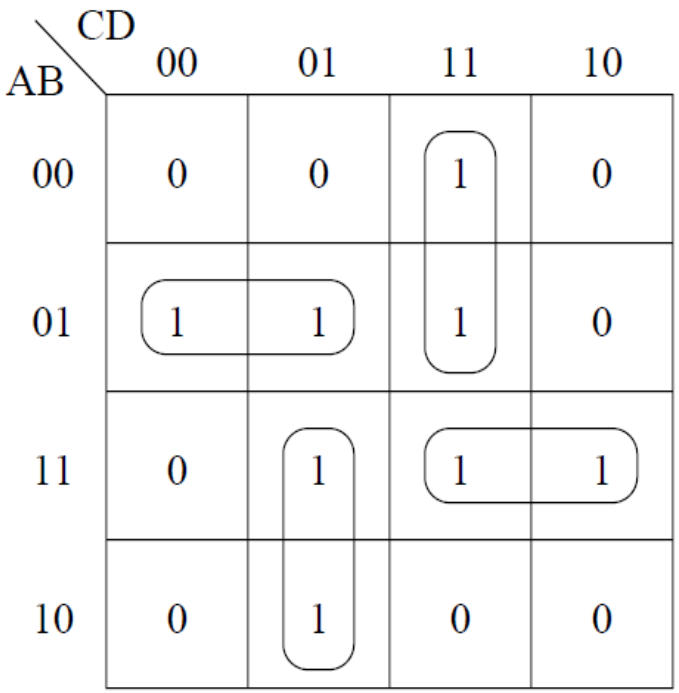
c) $Y = \overline{A}C\overline{D} + \overline{A}\overline{C}\overline{D}$



d) $Z = \overline{D}$

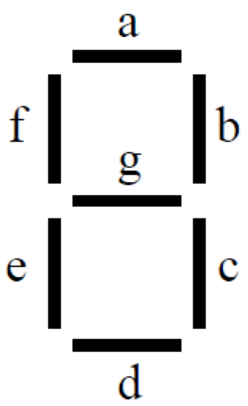
Пример редундантности





Седмоделни дисплеј

- 4 улаза
  - битови једне BCD цифре
- 7 излаза
  - за сваки сегмент дисплеја по један бит, који означава да ли је укључен или искључен



Седмоделни дисплеј (2)

Улаз					Излаз						
	Код 8421				Сегменти дисплеја						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0				1			
1	0	0	0	1				0			
2	0	0	1	0				1			
3	0	0	1	1				1			
4	0	1	0	0				0			
5	0	1	0	1				1			
6	0	1	1	0				1			
7	0	1	1	1				0			
8	1	0	0	0				1			
9	1	0	0	1				1			
...	...	...	...	...				...			

Седмоделни дисплеј (3)

- Минимизација сегмента *d* са и без недефинисаних вредности