Увод у организацију и архитектуру рачунара 1

Александар Картељ

kartelj@matf.bg.ac.rs

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

Комбинаторне мреже

Основни појмови

Комбинаторна мрежа

- Логичко коло је *комбинаторна мрежа* ако:
  - логички елементи су међусобно повезани и
  - у сваком тренутку излаз зависи само од вредности улаза у “том истом” тренутку

- Суштина је у одсуству било каквог чувања стања у самој мрежи
- Користи се и назив *комбинаторно коло*
- Сва представљана кола до сада су била комбинаторне мреже

Значај

- Комбинаторне мреже представљају виши ниво апстракције логичких кола
- Омогућавају употребу сложенијих логичких елемената при пројектовању логичких кола
- Њиховом употребом се смањује број потребних елемената при изградњи логичких кола

Начин дефинисања

- У општем случају комбинаторна мрежа се састоји од *n* бинарних улаза и *m* бинарних излаза
- Може се описати помоћу:
  - табеле истинитосних вредности
    - по ред за сваку од 2^n могућих комбинација улазних вредности
    - по колона за сваку улазну и излазну вредност
  - повезаног скупа графичких симбола
  - логичких функција које описују везу улаза и излаза

Врсте комбинаторних мрежа

- Неке од најважнијих врста комбинаторних мрежа:
  - Мултиплексори
  - Демултиплексори
  - Декодери
  - Енкодери
  - Компаратори
  - Сабирачи

Комбинаторне мреже

Преглед основних комбинаторних мрежа

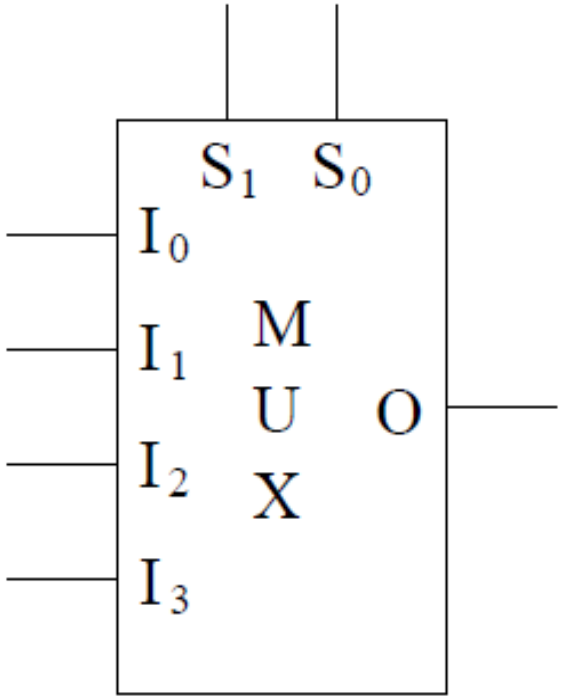
Мултиплексори

- *Мултиплексори* су комбинаторне мреже које имају:
  - 2^n улаза
  - *n селекторских* улаза
  - 1 излаз

- Вредност излаза одговара вредности улаза који је одређен вредношћу селекторских улаза

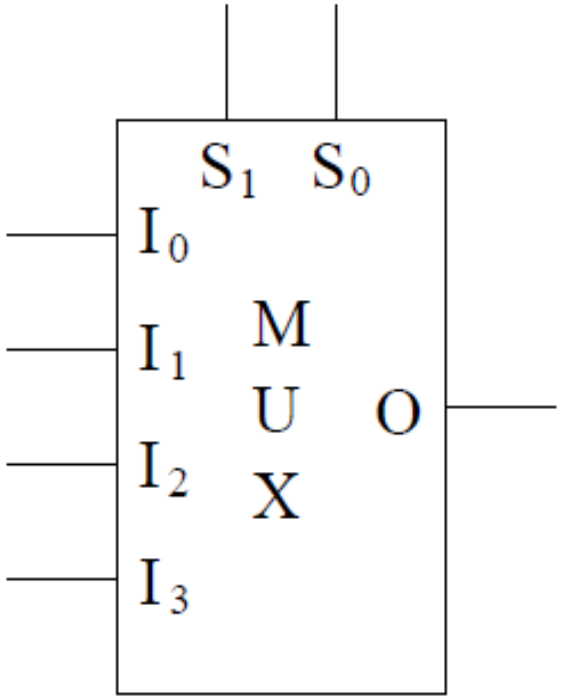
Мултиплексор 4-1

- Графичко представљање и одговарајућа таблица



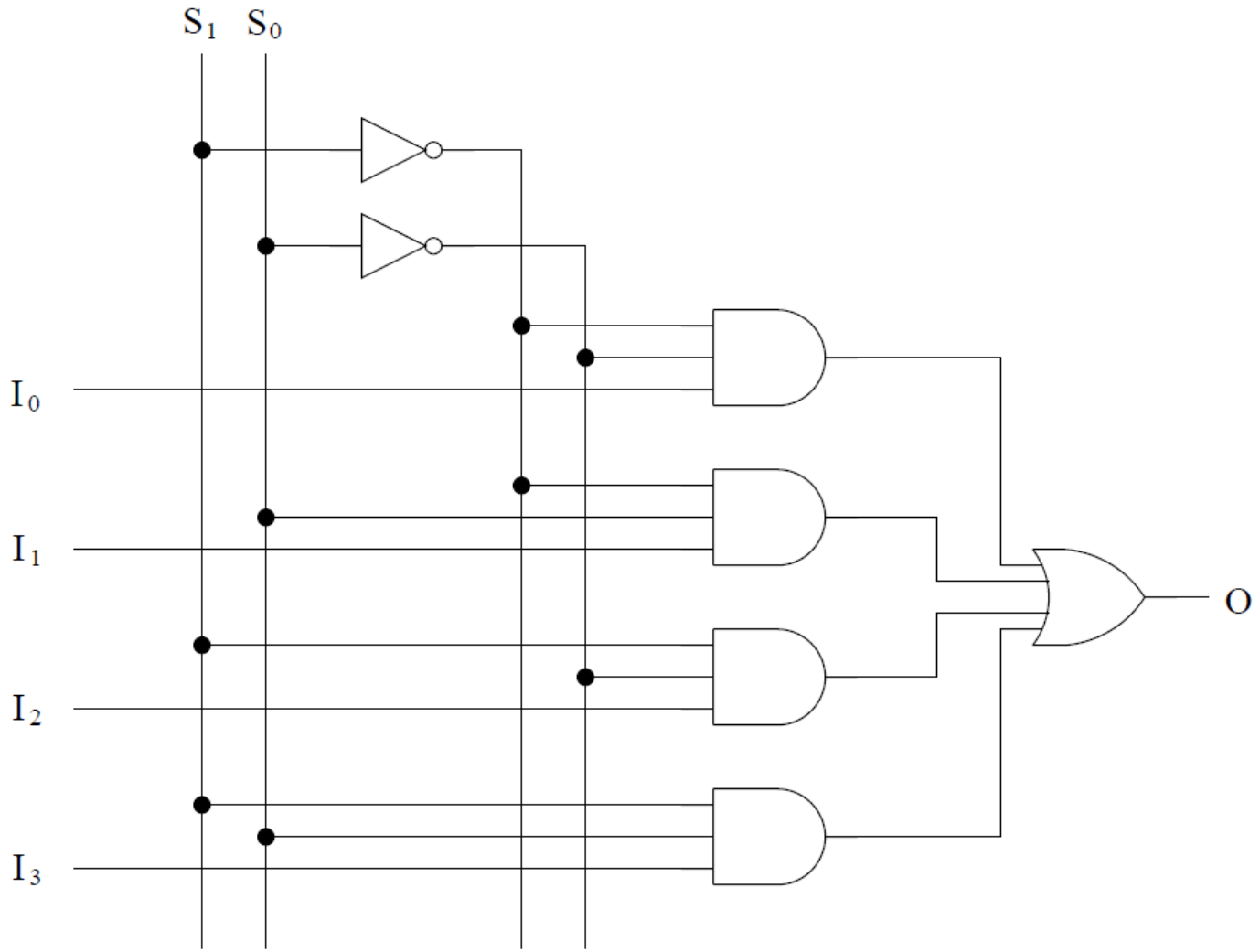
Мултиплексор 4-1 (2)

- Графичко представљање и одговарајућа таблица



S_1	S_0	O
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Имплементација мултиплексора

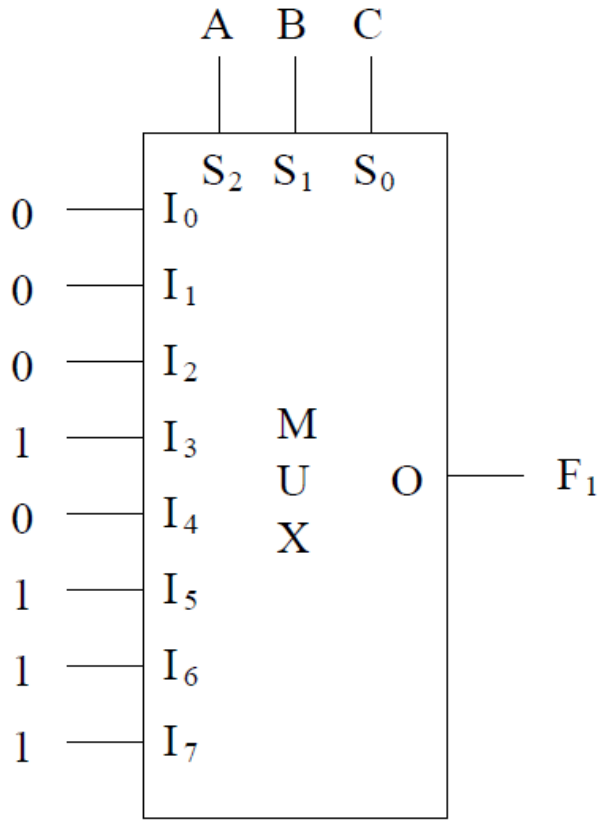


Примена мултиплексора

- Осим основне примене комбиновања више улаза у један излаз (одабирања), бирањем улазних вредности се мултиплексори могу употребљавати за имплементирање функција од селекторских улаза
  - 1. начин: на улазе се доведу константне вредности, тако да одговарају вредностима функције за одговарајуће селекторске улазе
  - 2. начин: процесом редукције се мултиплексором може имплементирати функција са n+1 аргумената
    - (у неким случајевима може и више)

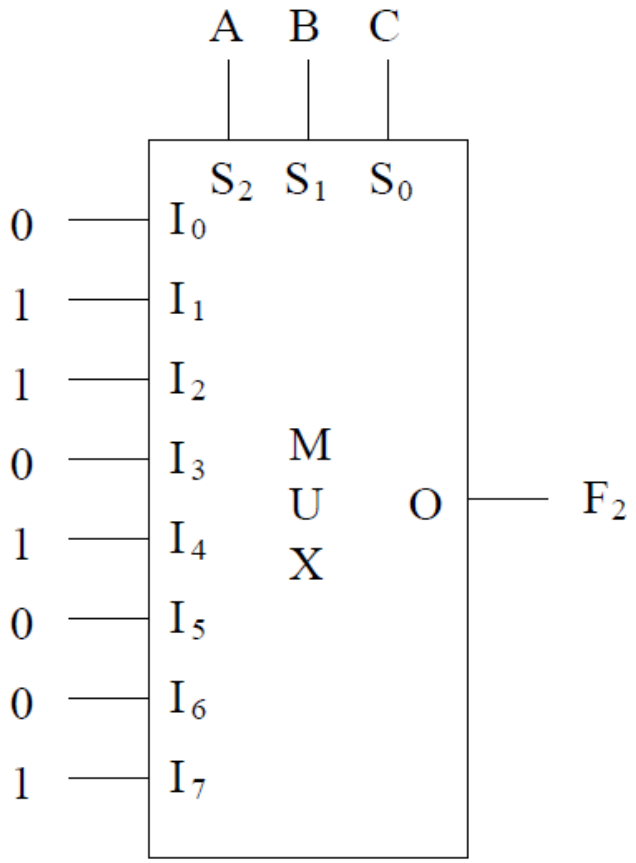
Пример одабирања улаза

- Израчунавање заступљенијег бита на селекторским улазима



Пример одабирања улаза (2)

- Израчунавање парности за селекторске улазе

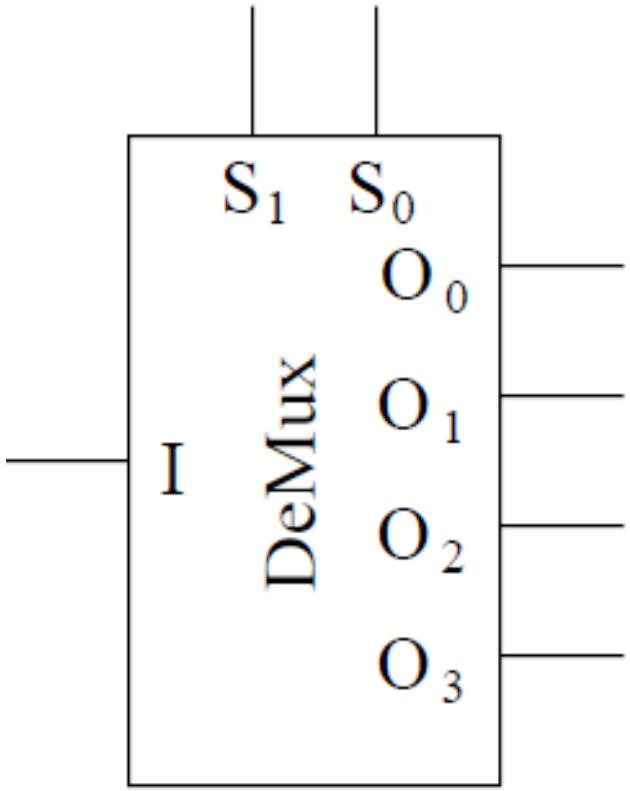


Демултиплексори

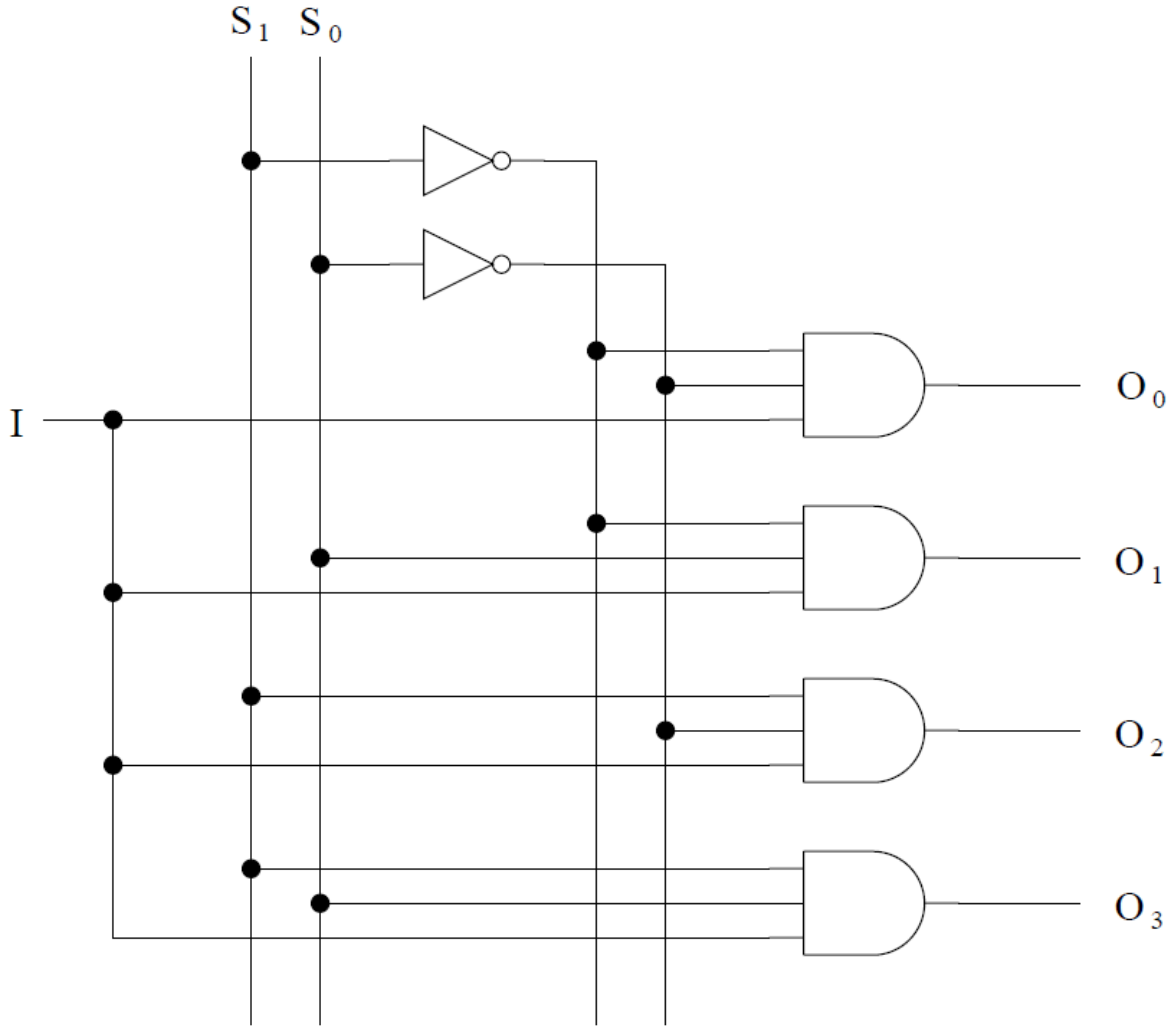
- *Демултиплексори* су комбинаторне мреже које имају:
  - n+1 улаза
    - 1 улазна вредност
    - n селекторских улаза
  - 2^n излаза

- Обављају инверзну функцију мултиплексора
  - Тачно на један излаз се пресликава вредност улаза
  - Сви остали излази имају вредност 0

Демултиплексор 1-4



Имплементација демултиплексора



Декодери

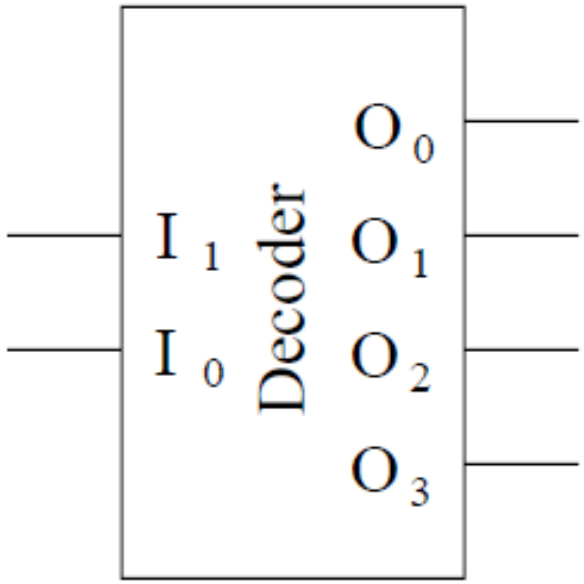
- Декодери су комбинаторне мреже које имају:
  - n улаза
  - највише 2^n излаза

- У сваком тренутку је активан највише један излаз, а у зависности од улаза

Пример декодера

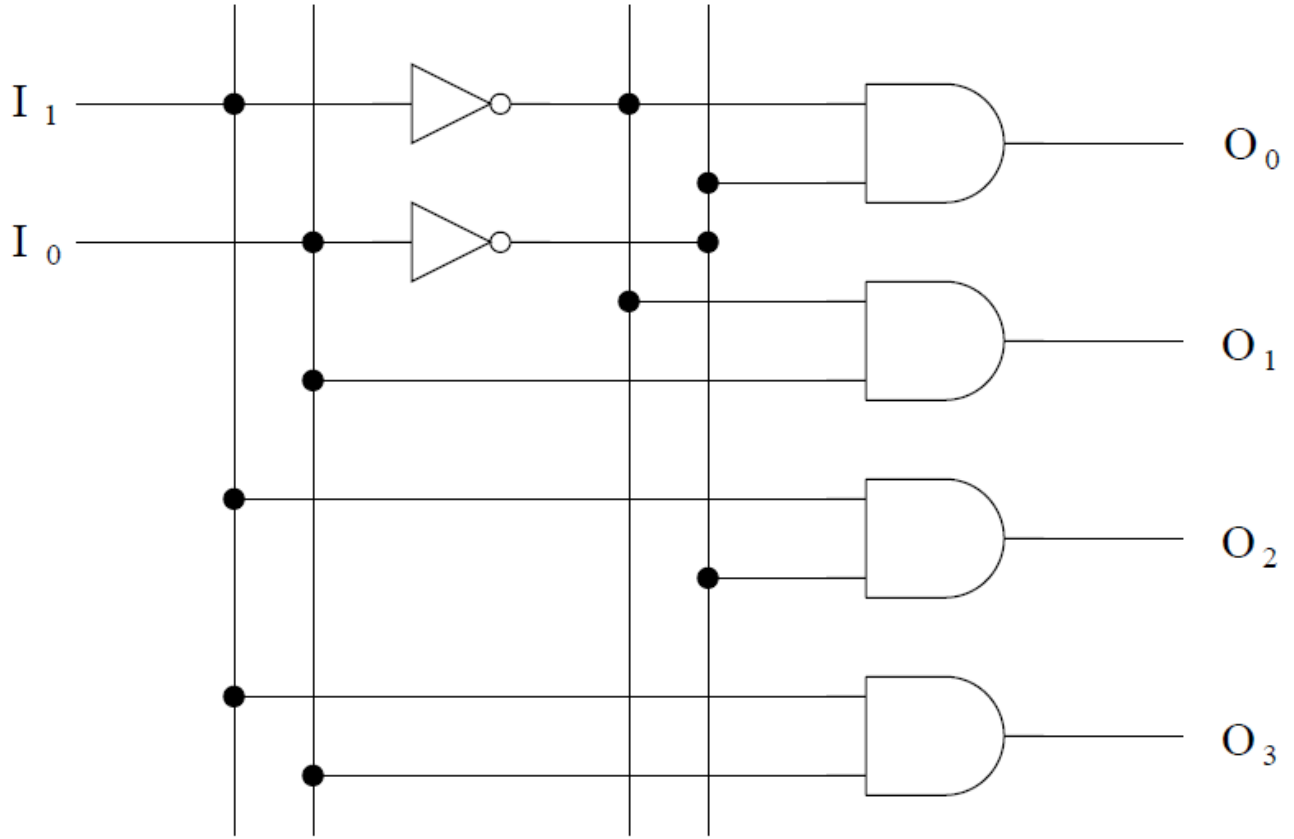
- На основу неозначеног 2-битног целог броја бира одговарајући од 4 улаза:

I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



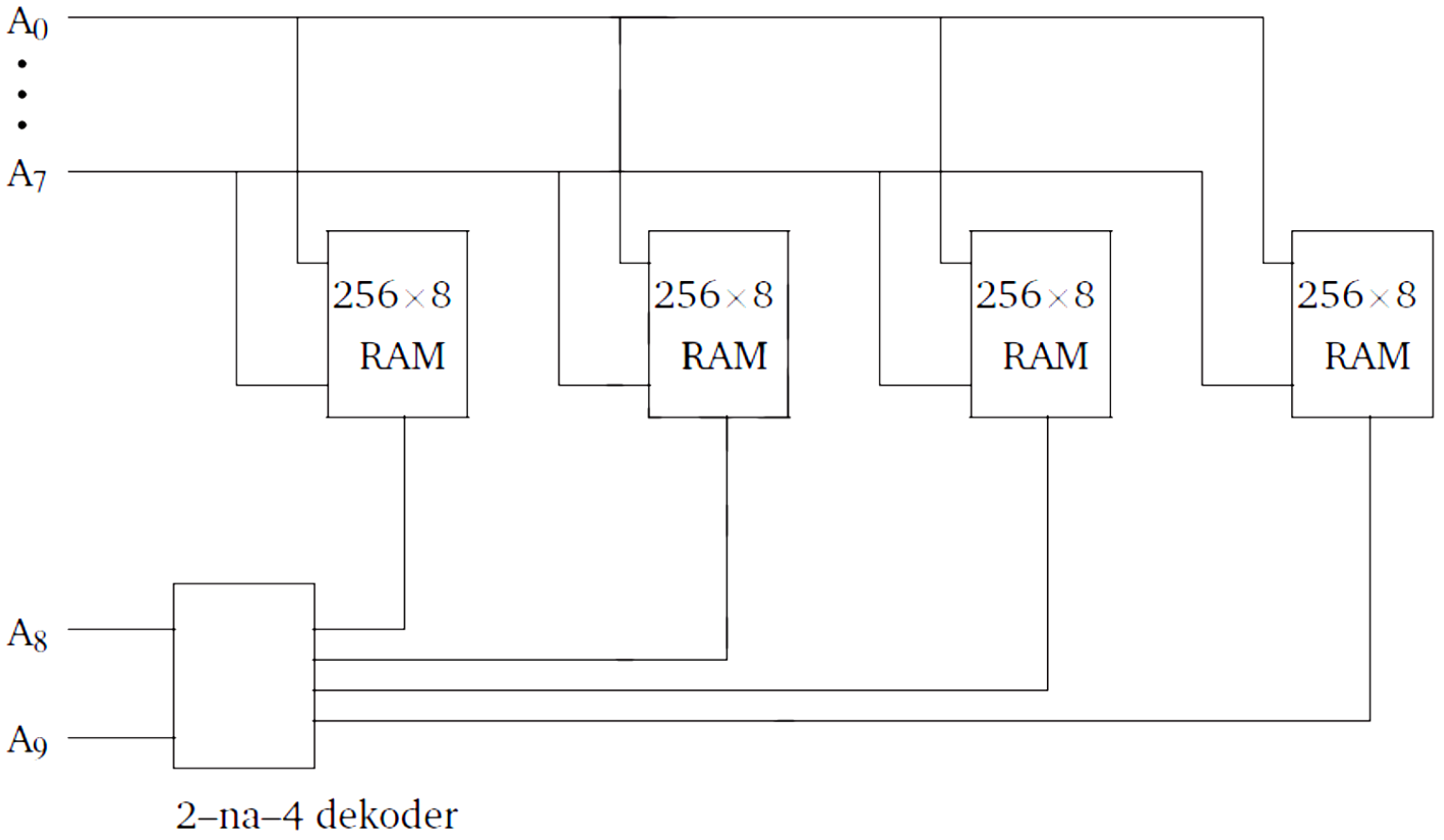
Пример декодера (2)

- На основу неозначеног 2-битног целог броја бира одговарајући од 4 улаза:



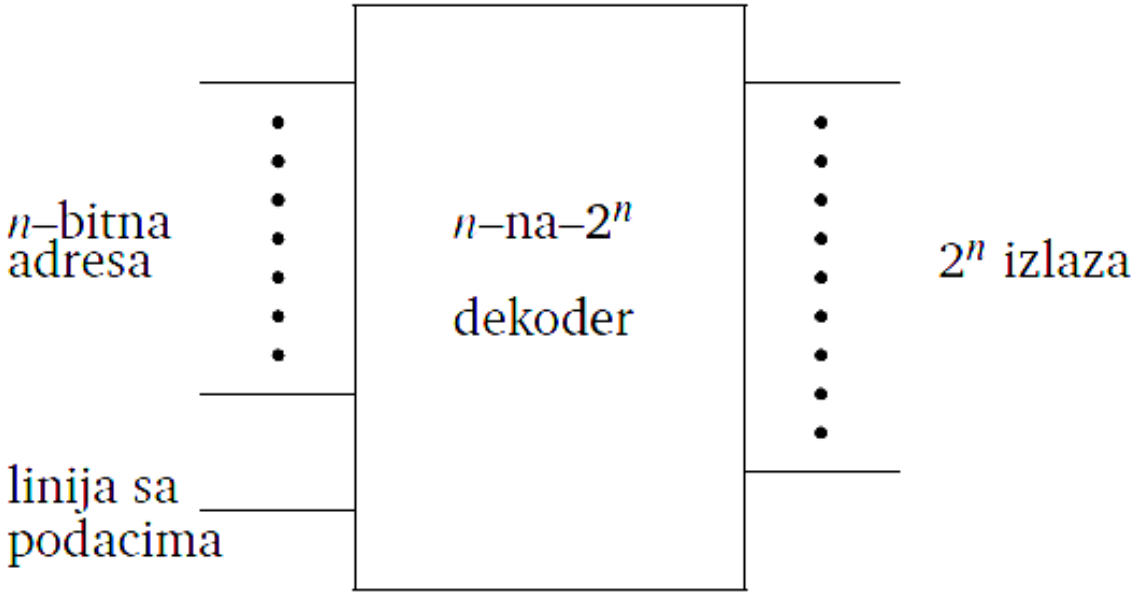
Пример – Декодирање адреса

- Декодирање адреса меморије величине 1KB од четири 256B (8-битна) RAM чипа:



Декодер као демултиплексор

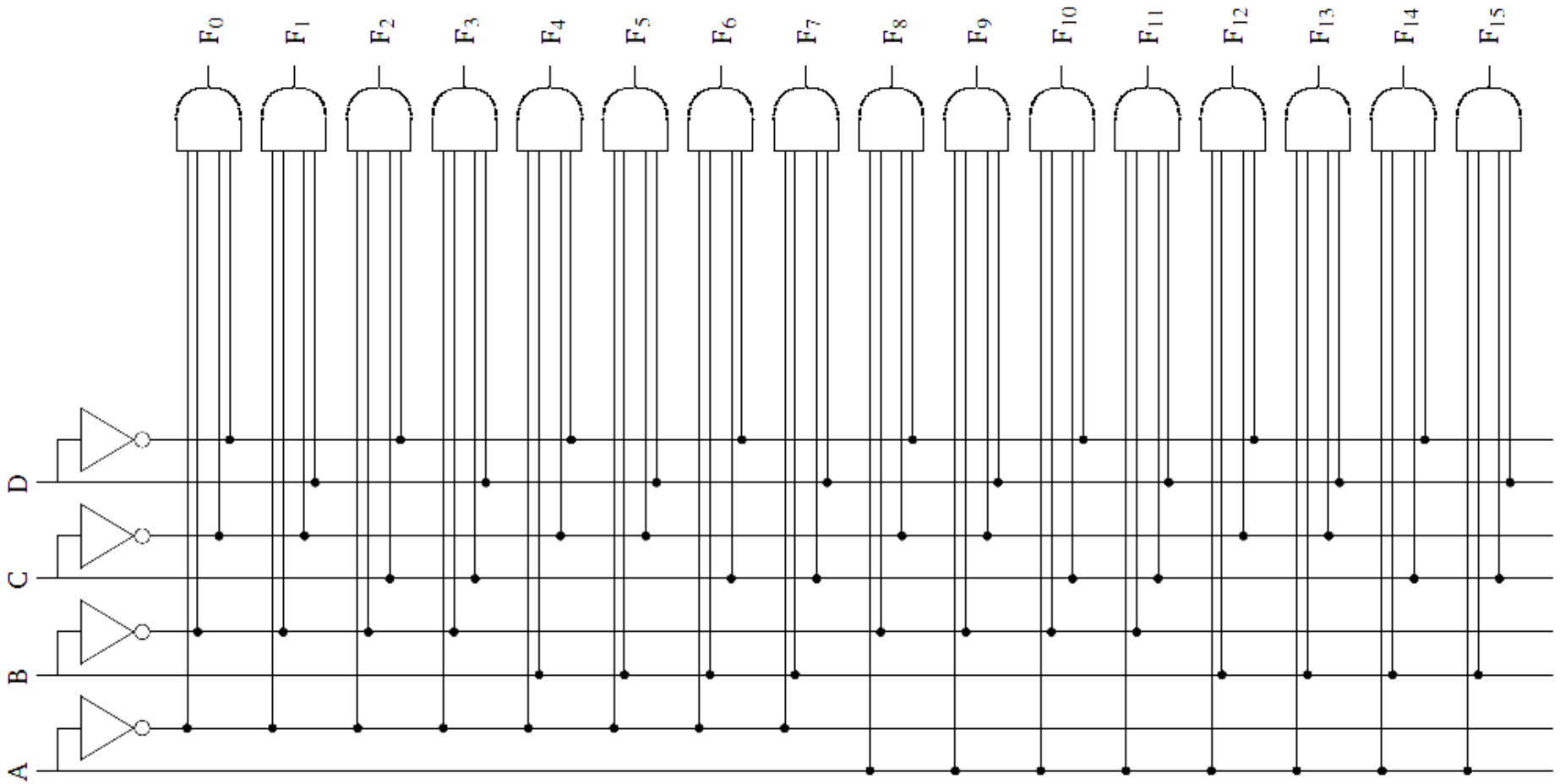
- Декодеру се уводи додатни улаз
  - улаз представља улаз демултиплексора
  - може да се тумачи и као контролни бит декодера:
    - декодер ради ако је улаз активан
    - ако је улаз неактиван, сви излази су неактивни



Пример – Декодирање *BCD* цифара

Ulaz				Izlaz													
				Dekadne cifre										Greška			
A	B	C	D	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Пример – Декодирање *BCD* цифара (2)



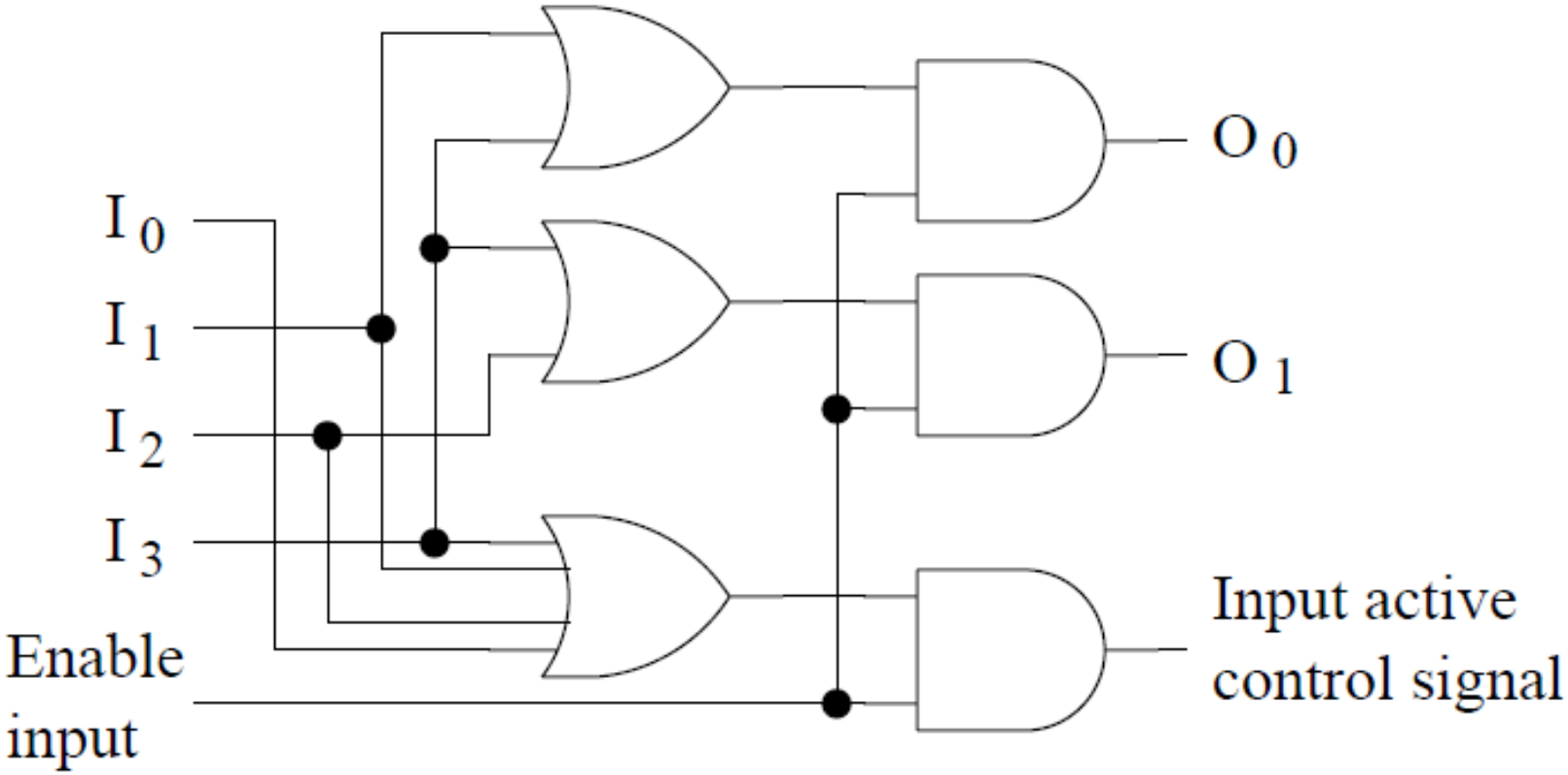
Енкодер

- Енкодери су комбинаторне мреже које имају:
  - 2^n улаза
  - n излаза
- Представљају инверзну операцију декодера
  - У сваком тренутку је активан највише један улаз
  - Излаз је одређен активним улазом
- Обично се додају
  - контролни улаз који укључује енкодер
  - контролни излаз који је активан ако су активни
    - контролни улаз и
    - бар један улазни бит

Пример енкодера

Enable input	I_3	I_2	I_1	I_0	O_1	O_0	Input active control signal
0	X	X	X	X	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	1	0	1
1	1	0	0	0	1	1	1

Пример енкодера (2)

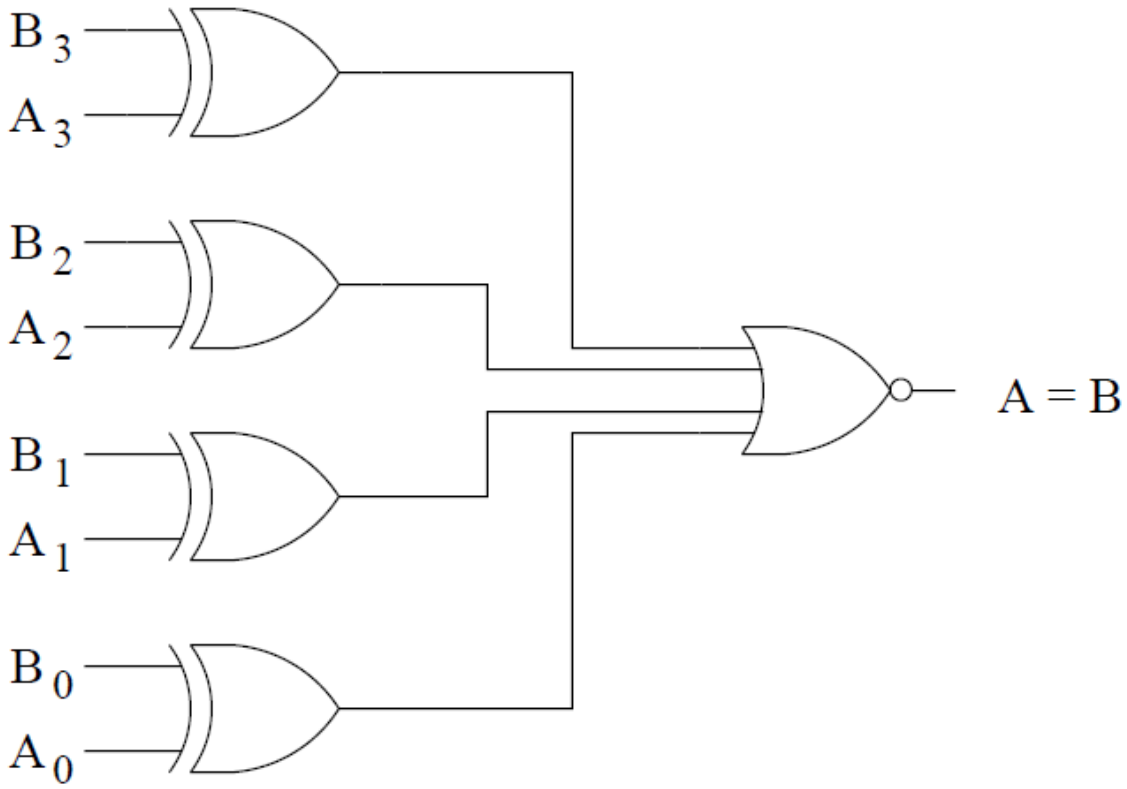


Компаратор

- Компаратори су мреже које пореде два низа битова на одговарајући начин
- Имају:
  - 2 x *n* улаза
  - одговарајући број излаза, зависно од врсте поређења
- Примери:
  - поређење једнакости – довољан је један излаз
  - поређење величине броја – довољна су два излаза, али се обично користе три
  - поређење са проширењем – ако се пореде бројеви који имају *m* x *n* битова, улазна информација обухвата и резултат поређења претходне групе од *n* битова

Компаратори – једнакост

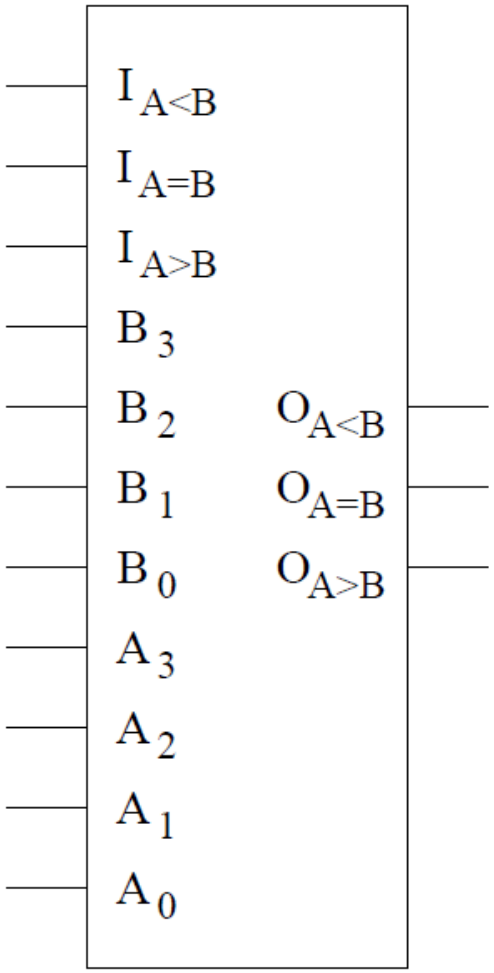
- Поређење једнакости се обично изводи помоћу *XOR* елемената
- Поредимо једнакост два низа од 4 бита:



Компаратори – поређење

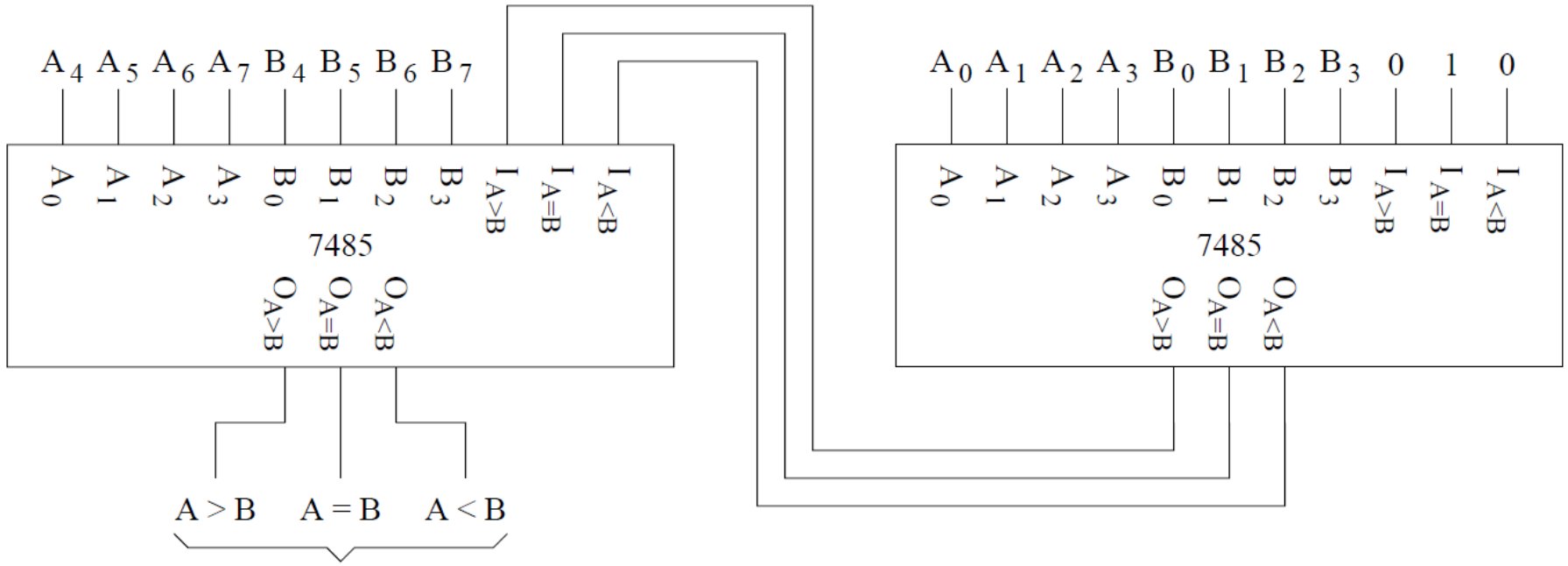
- Поређење два 4-битна броја, за вежбање

Компаратори – поређење са проширењем



Компаратори – поређење са проширењем (2)

- Поређење два неозначена 8-битна цела броја помоћу два компаратора дужине 4:



Сабирачи

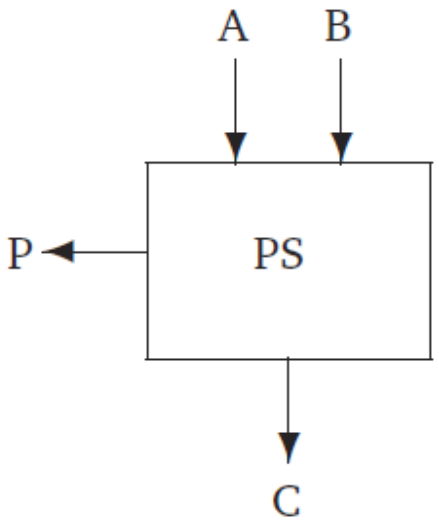
- Сабирачи су комбинаторне мреже које се користе при имплементацији сабирања

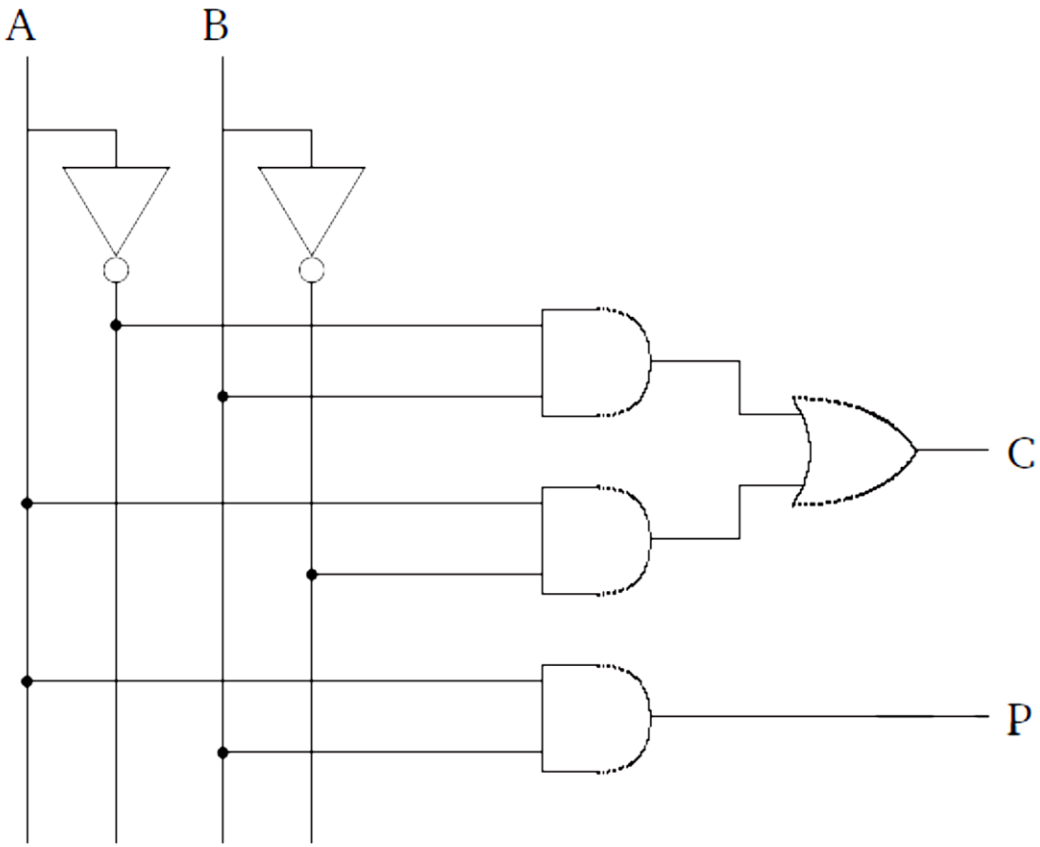
Бинарни полусабирач

- Бинарни полусабирач је комбинаторна мрежа која сабира два једноцифрена бинарна броја
  - Улази су два једноцифрена броја
  - Излази су један бит резултата и један бит преноса

Бинарни полусабирач (2)

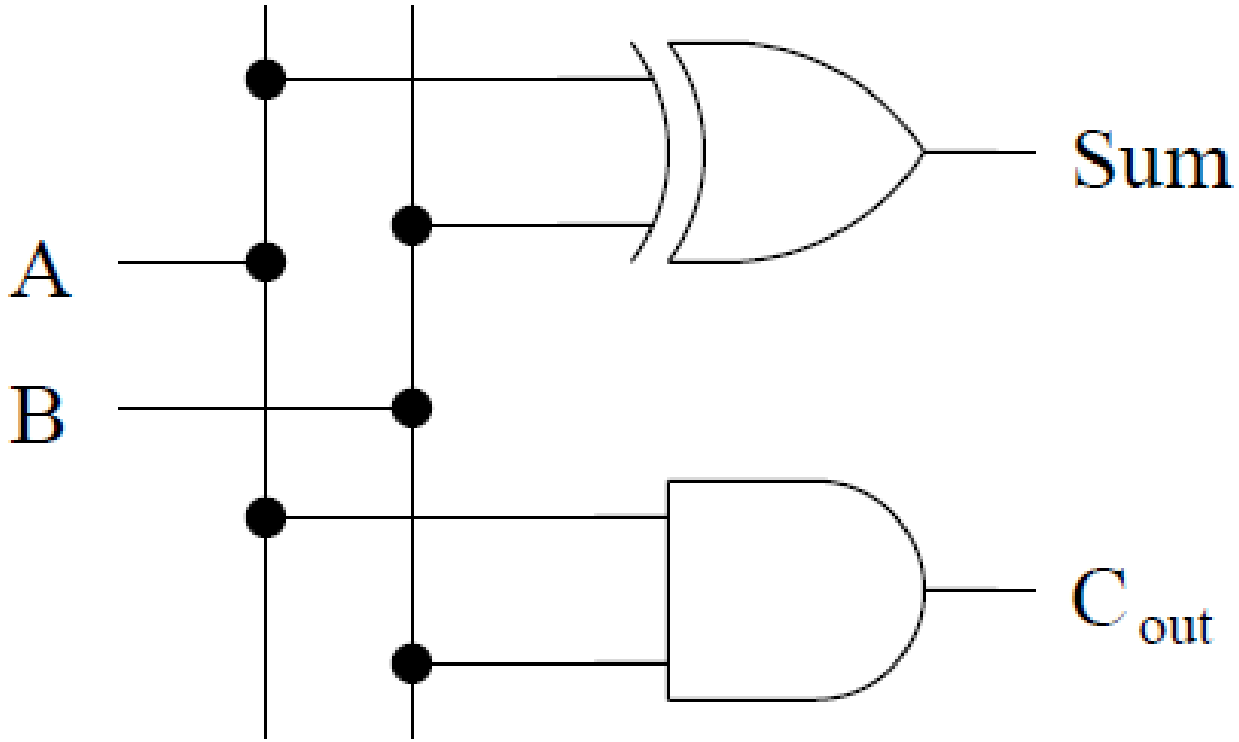
A	B	C	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





Бинарни полусабирач (2)

- Задатак: оптимизовати бинарни полусабирач

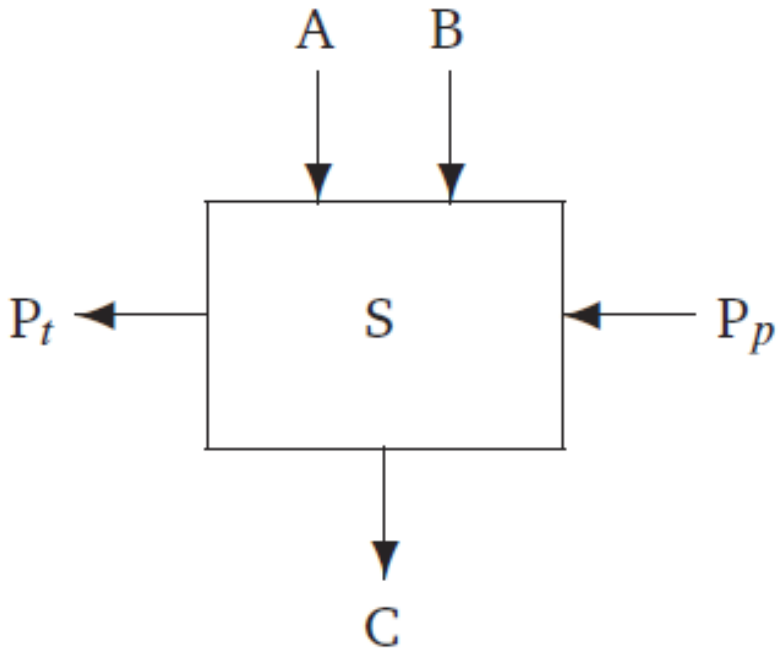


Бинарни сабирач

- Бинарни сабирач је комбинаторна мрежа која сабира два једноцифрена бинарна броја и дати пренос
  - Улази су два једноцифрена броја и један бит преноса
  - Излази су један бит резултата и један бит преноса

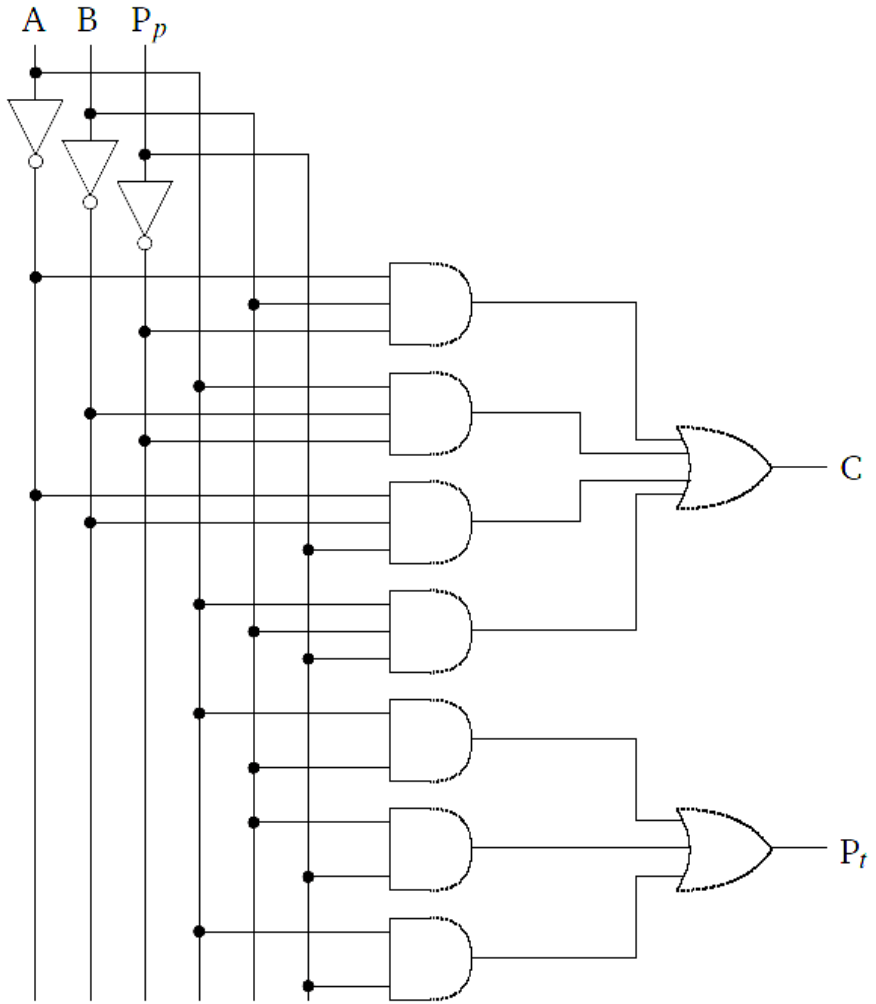
- Задатак: минимизовати бинарни сабирач

Бинарни сабирач (2)



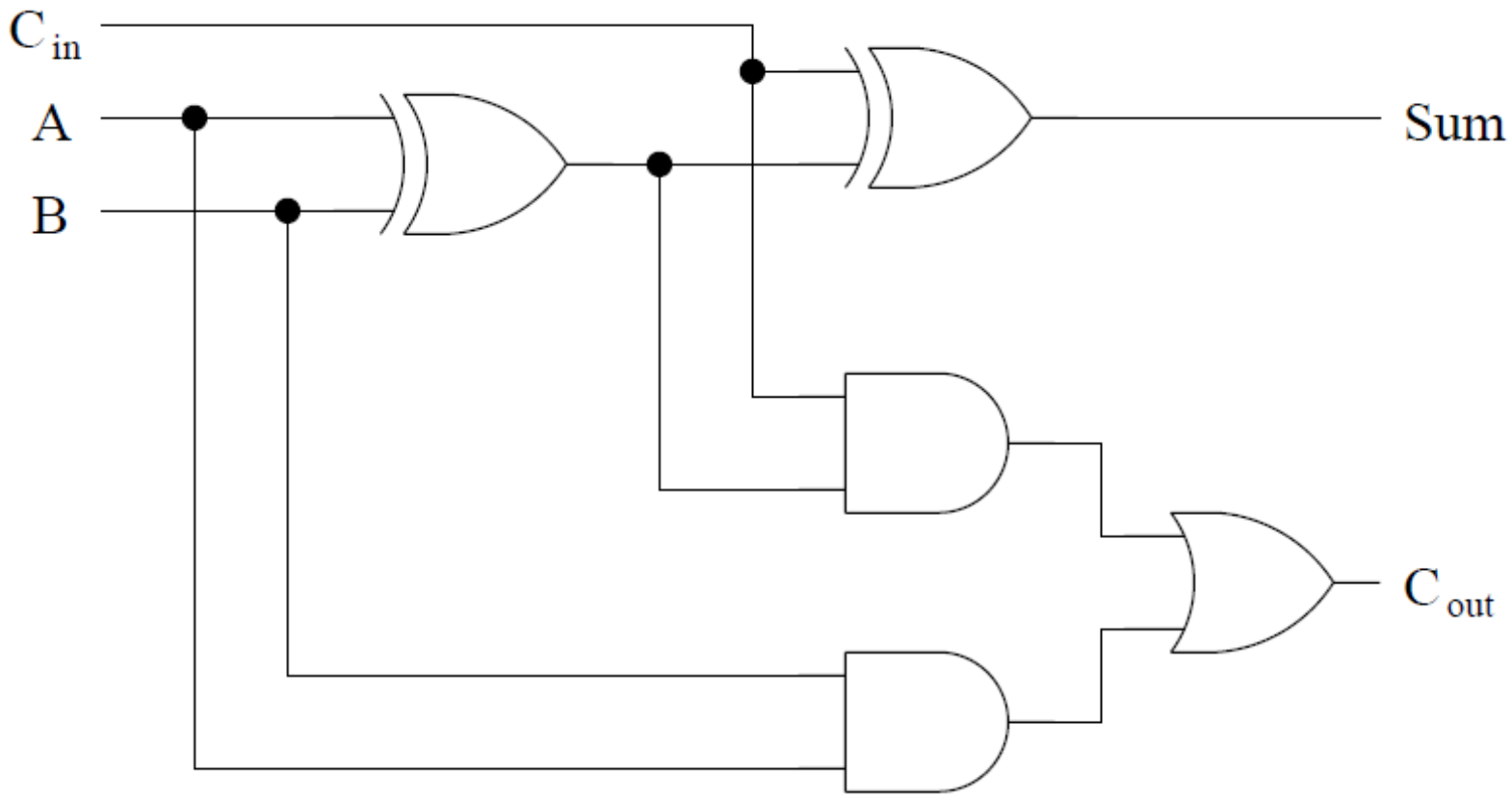
P_p	A	B	C	P_t
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Бинарни сабирач (3)



Бинарни сабирач (4)

- Задатак: оптимизовати бинарни сабирач

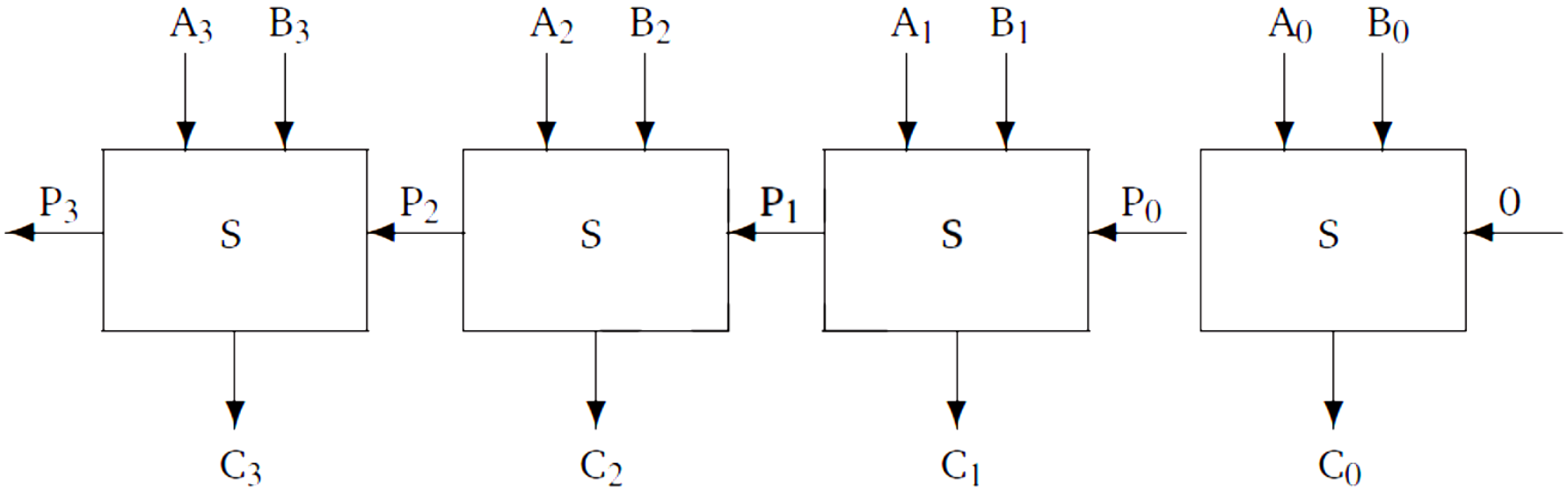


Сложени сабирач

- Сложени сабирач (или *вишебитни сабирач*) је комбинаторна мрежа која сабира два вишецифрена бинарна броја и дати пренос
- Имплементира се помоћу више бинарних сабирача

Сложени сабирач (2)

- Сабирање два 4-битна броја, са прекорачењем



Комбинаторне мреже

Преглед напреднијих комбинаторних мрежа

Програмабилни низ логичких елемената

- Прва интегрисана кола су имала до 10 логичких елемената
- Свако коло је обликовано према намени

- Повећавање степена интеграције и броја уграђених логичких елемената је довело до повећавања броја различитих кола и отежавало производњу
- Одатле је настала потреба за развијање техника за производњу интегрисаних логичких кола опште намене, која се лако могу прилагодити специфичним потребама (тј. *програмирати*)

Програмабилни низ логичких елемената (2)

- Програмабилни низ логичких елемената (енгл. *Programmable Logic Array - PLA*) почива на примени СДНФ

- Како се свака логичка функција може изразити у облику СДНФ, најопштија форма је да се:
  - сваки улаз “повеже” са НЕ елементом
  - сваки улаз и његова негација “повеже” са И елементима
  - сваки излаз из И елемента се “повеже” са ИЛИ елементима
  - излази ИЛИ елемената представљају излазе логичког кола

Програмабилни низ логичких елемената (3)

- Назив потиче из чињенице да се имплементација састоји од два низа елемената
  - низа И елемената и
  - низа ИЛИ елемената

- Ако има n улаза и m излаза, потребно је:
  - до 2^n И елемената са по $2n$ улаза
  - m ИЛИ елемената са до 2^n улаза

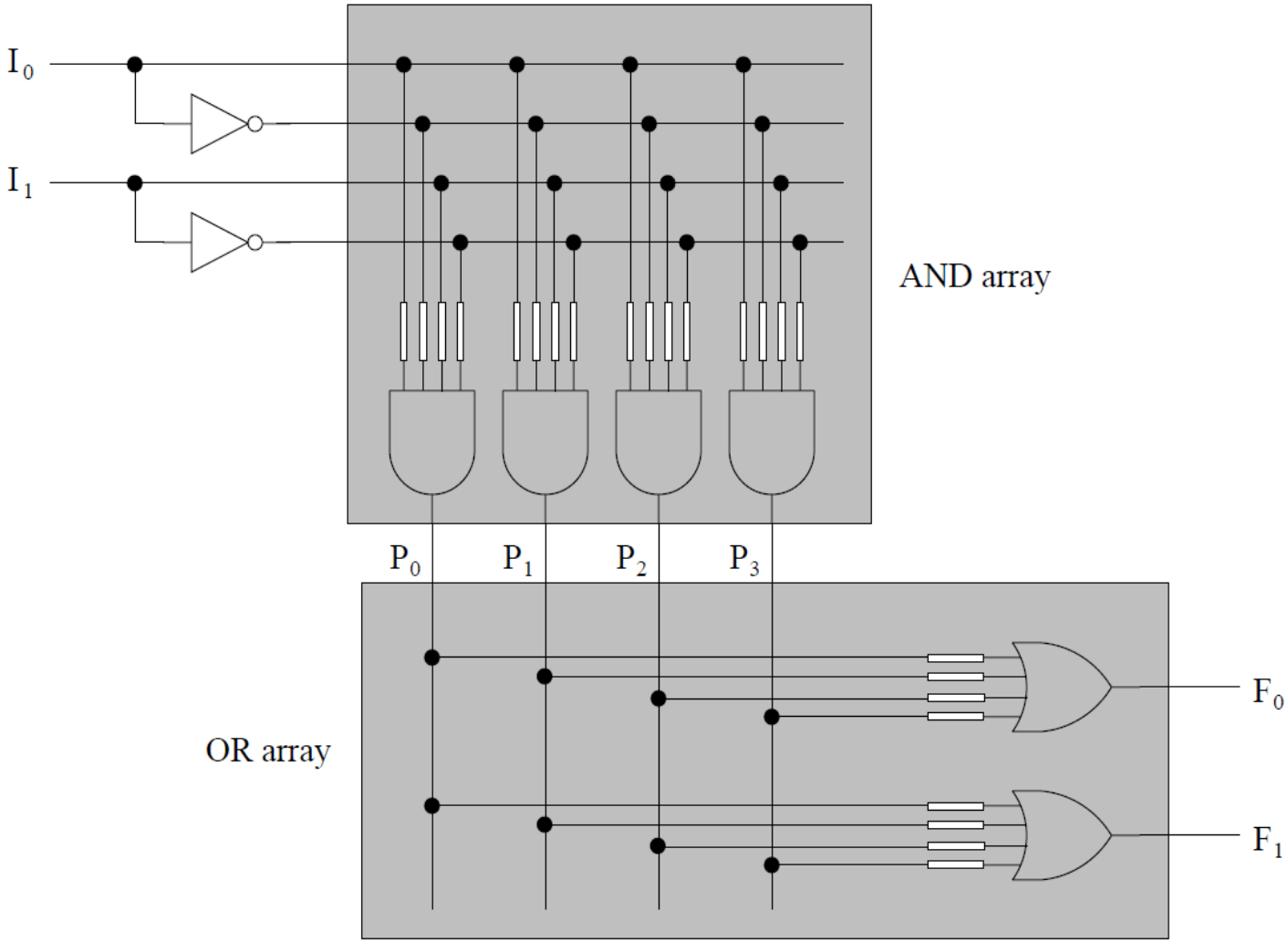
Програмабилни низ логичких елемената (4)

- На сваком од улаза у И и ИЛИ елементе постоји по *прекидач*
- Програмирање се постиже искључивањем прекидача

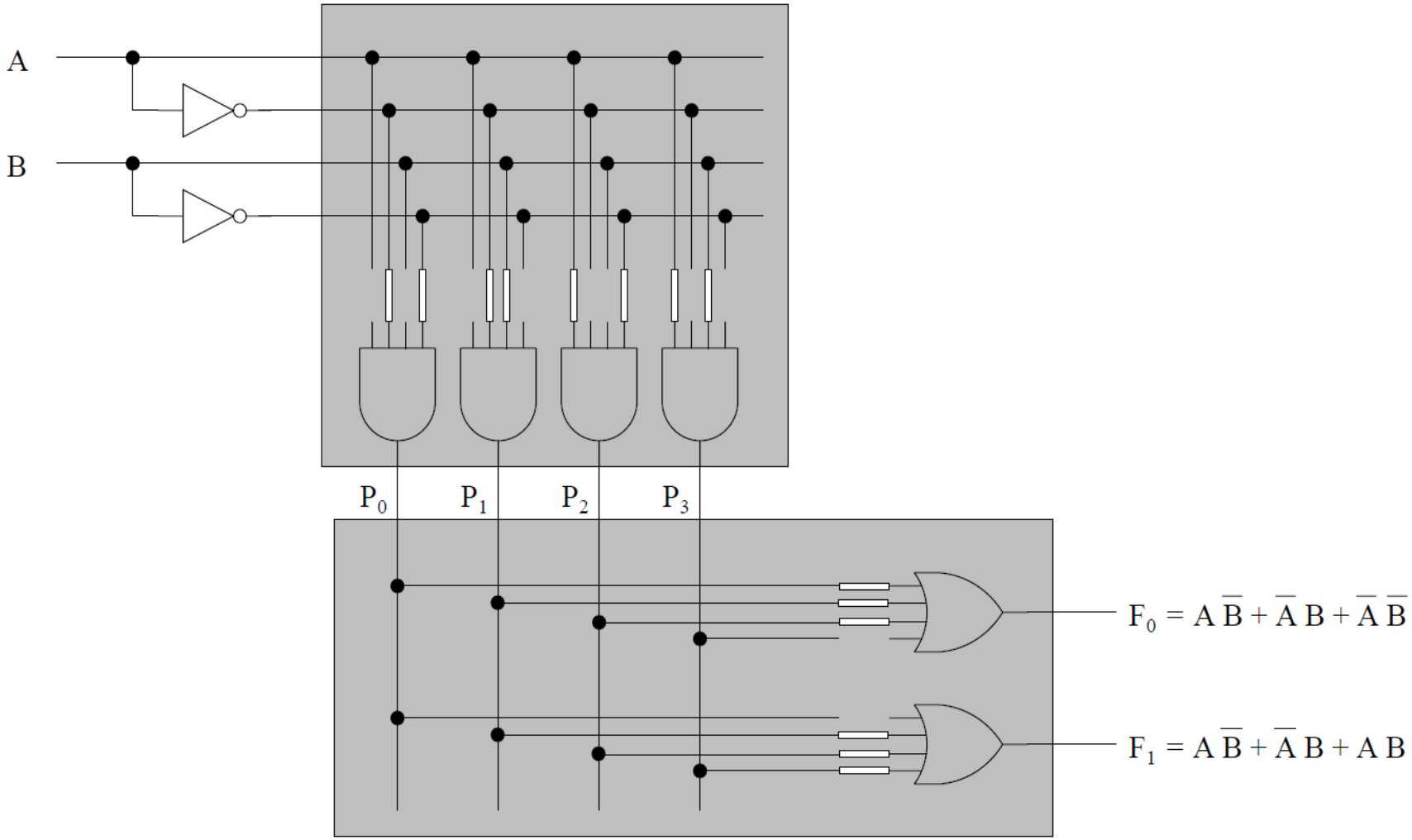
- Прекидачи се по правилу искључују једнократно
  - уобичајено је њихово спаљивање пропуштањем јаке ел. струје

- Имплементацијом се обезбеђује да се искључени прекидачи понашају као активни (1) улази за И кола и неактивни (0) улази за ИЛИ кола

Пример ПНЛЕ



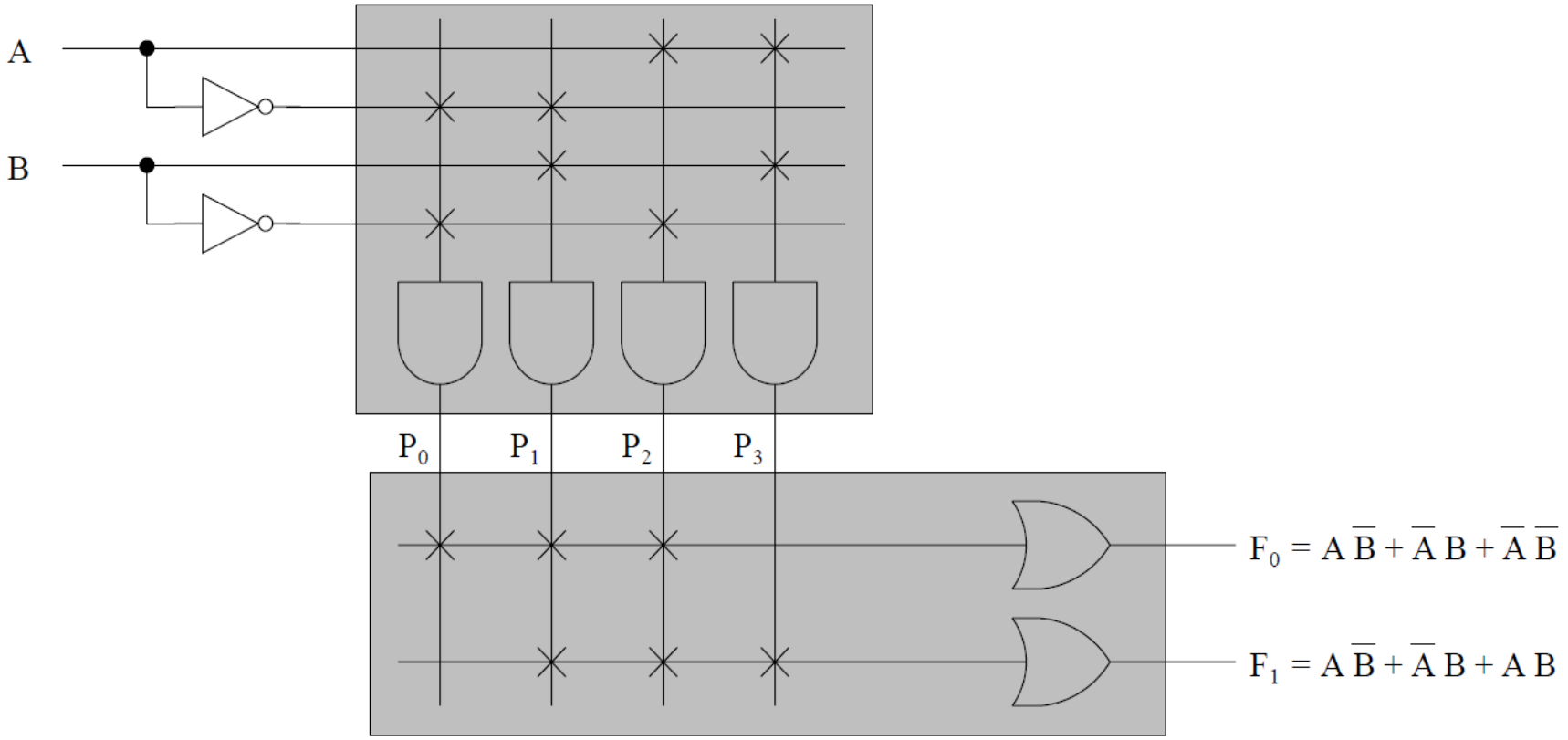
Пример ПНЛЕ (2)



Поједностављена нотација

- Да би се олакшало приказивање ПНЛЕ, уводи се нотација која претпоставља
  - изостављање прекидача из приказа
  - означавање проходних пресека водова са X

Пример ПНЛЕ (2)



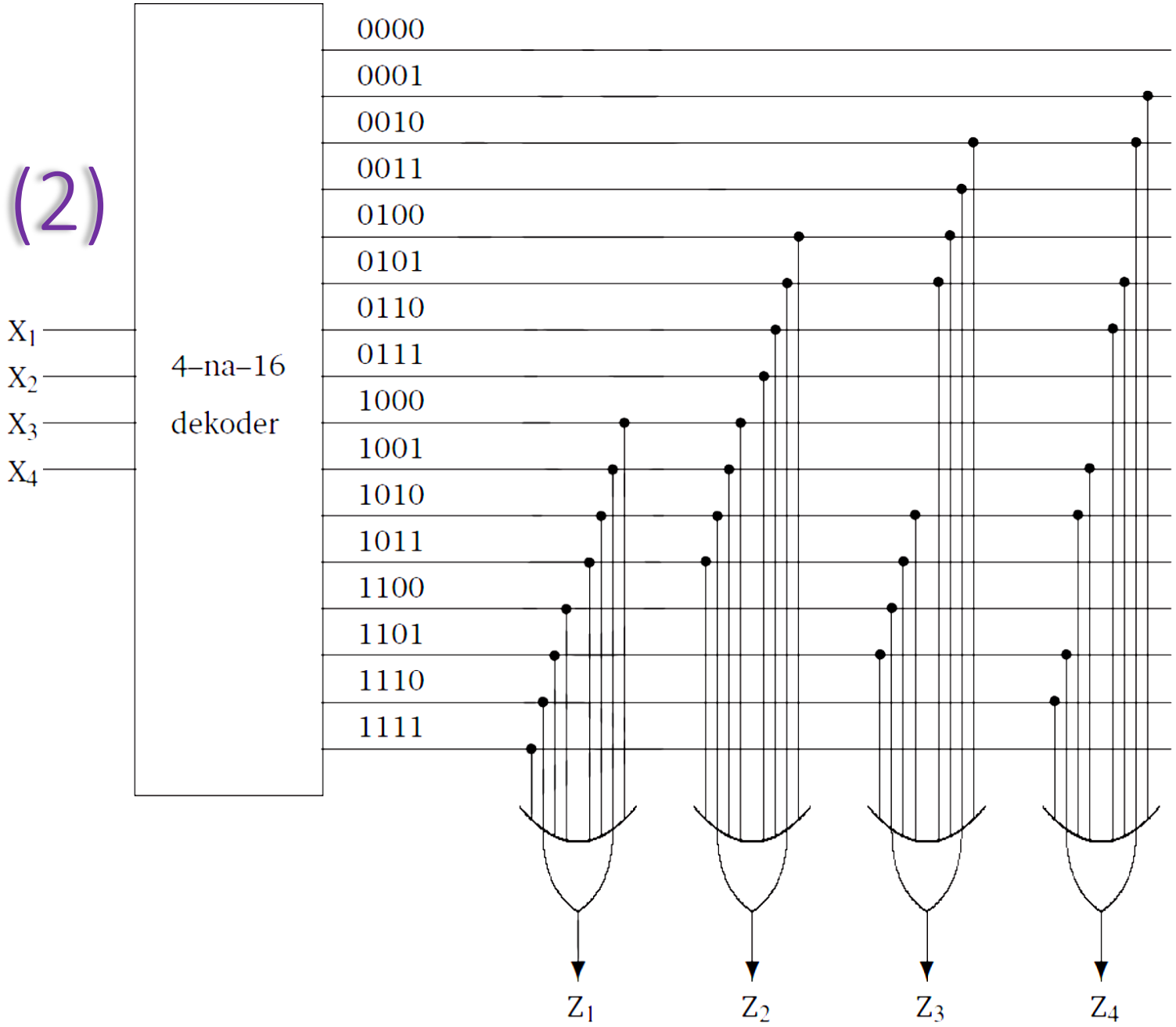
Меморија само за читање

- Комбинаторне мреже се називају и “мреже без меморије”, зато што резултати зависе искључиво од улаза
- Ипак, могу се употребљавати за имплементацију меморија које служе само за читање (*ROM*)
- Ако је дужина адресе n битова, а податка m битова, онда се имплементира помоћу:
  - декодера n-на-2^n, који препознаје адресу
  - низа од m ИЛИ елемената, који дају битове податка у зависности од адресе

Пример *ROM*-а

Ulaz				Izlaz			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Пример *ROM*-а (2)



Аритметичко-логичка јединица

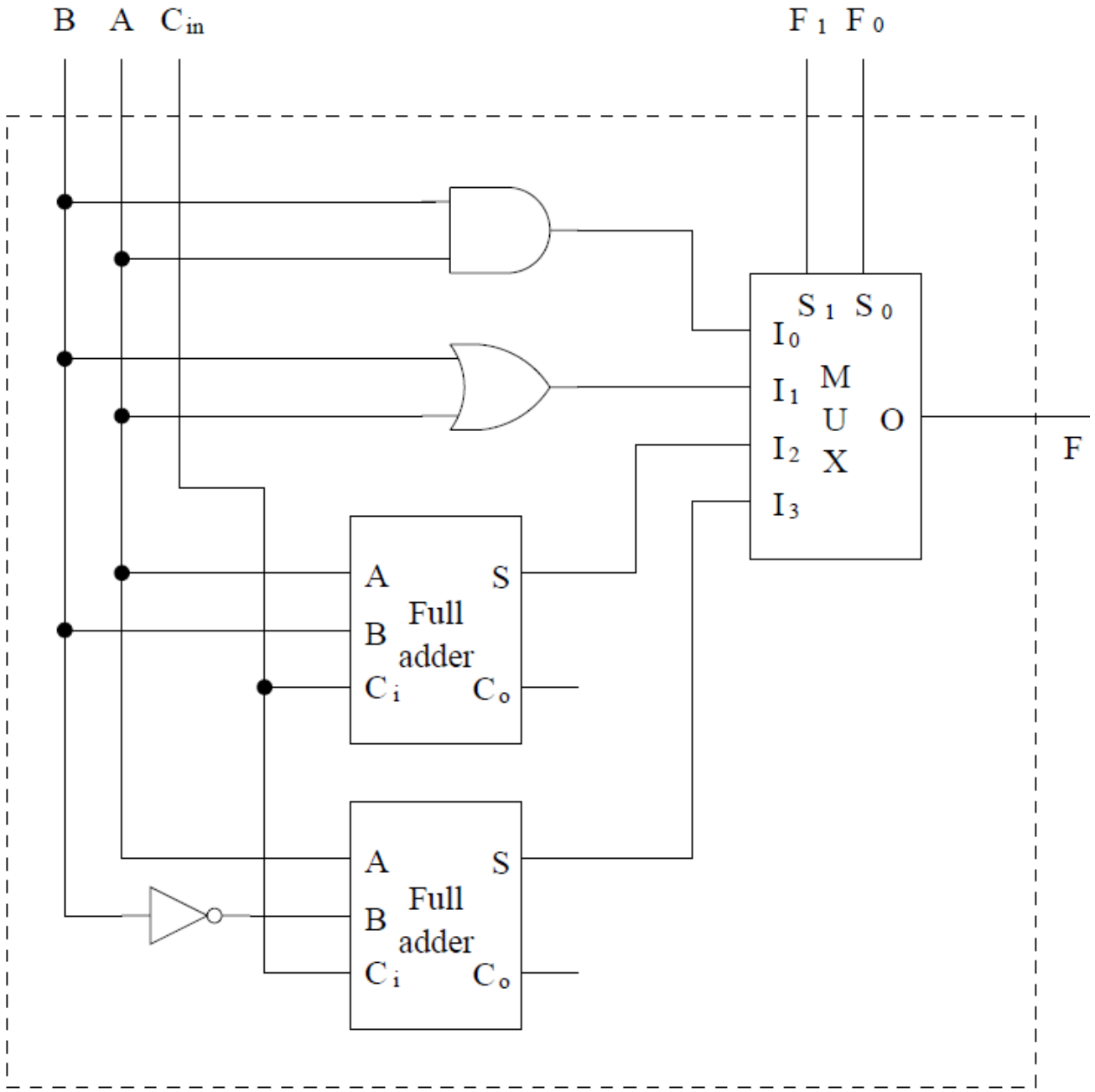
- До сада је изложено довољно материјала да можемо да направимо једноставну аритметичко-логичку јединицу (АЛЈ) (енгл. *arithmetic logic unit*)
- Задатак:
  - имплементирати коло које у зависности од улаза F_0 и F_1 рачуна конјункцију, дисјункцију, збир или разлику два улазна бита *A* и *B*

F_1	F_0	F
0	0	A ∧ B
0	1	A ⊼ B
1	0	A + B
1	1	A - B

Решење

- Идеја:
  - Резултат се рачуна помоћу мултиплексора који има као улаз резултате свих подржаних операција, а као селекторске улазе F_0 и F_1
  - Конјункција и дисјункција се рачунају непосредном применом елемената И и ИЛИ
  - За сабирање се употребљава сабирач са улазима *A* и *B*
  - За одузимање се употребљава сабирач са *A* и *B'*
    - допуна до пуног комплемента ће доћи као претходни пренос

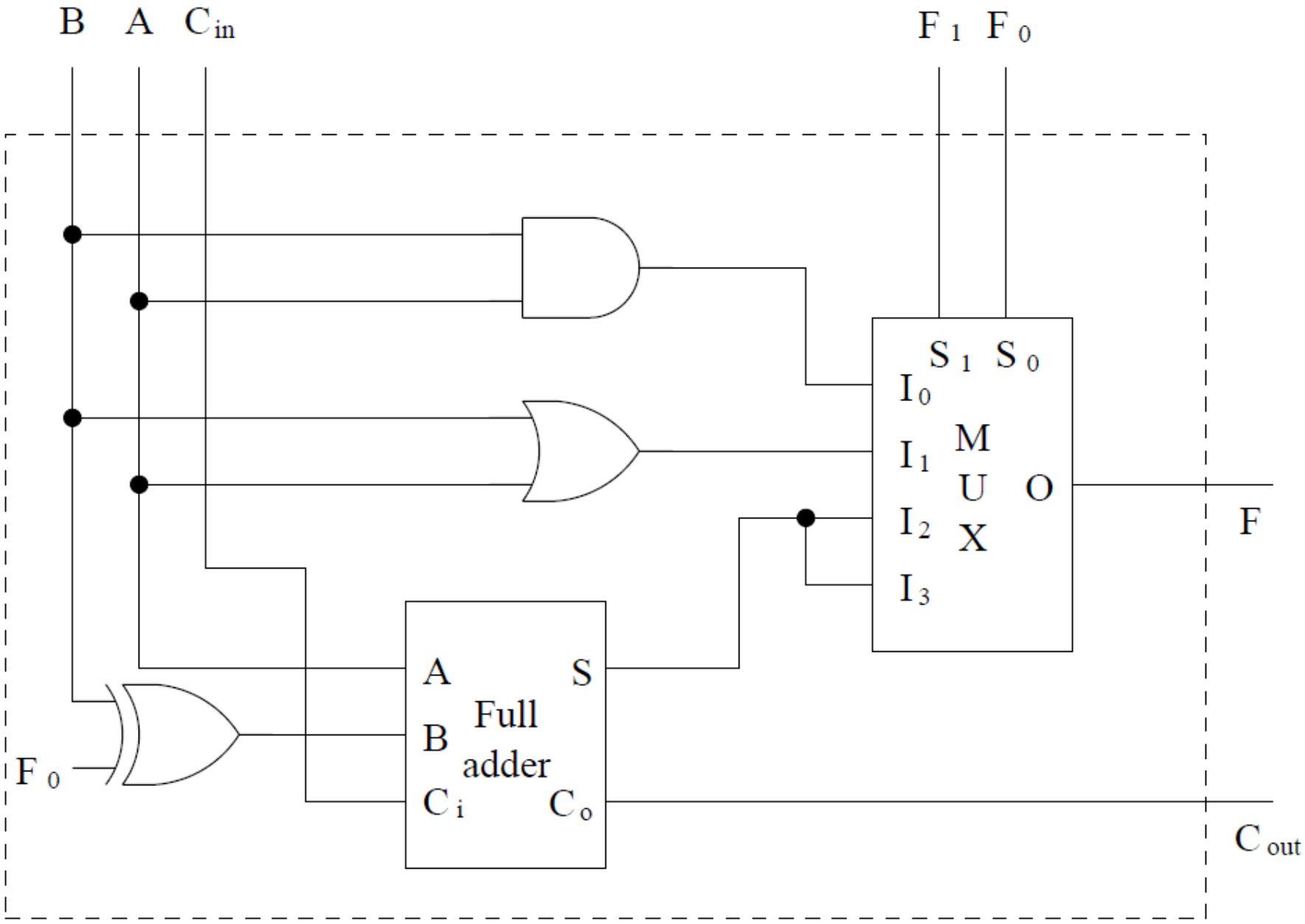
Решење (2)


B
A
Cin
F1
F0
S1
S0
I0
I1
I2
I3
M
U
X
O
F
A
B
Ci
S
Full
adder
Co
A
B
Ci
S
Full
adder
Co

Решење (3)

- Поједностављење:
  - Један сабирач може да се употребљава и за сабирање и за одузимање
  - Улаз *B* се негира у зависности од улаза F_0
    - ако је $F_0 = 1$, онда се негира
    - иначе не
  - То је еквивалентно операцији *B* XOR F_0
  - Допуна до пуног комплемента се решава као пренос

Решење (4)



Решење (5)

- Помоћу описане 1-битне АЛЈ можемо направити и сложенију, на пример, 16-битну АЛЈ

Решење (6)