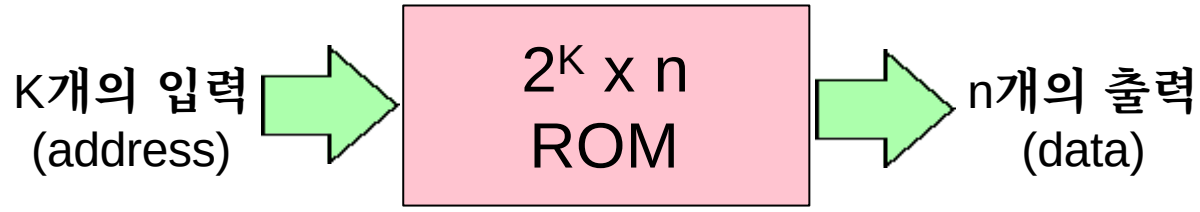
MSI & PLD

- MSI (Medium Scale Integrate Circuit)
  - 하나의 칩에 수십개의 gate를 구현한 집적회로
  - adder, subtractor, comparator, decoder, encoder, multiplexer, demultiplexer, ROM, PLA

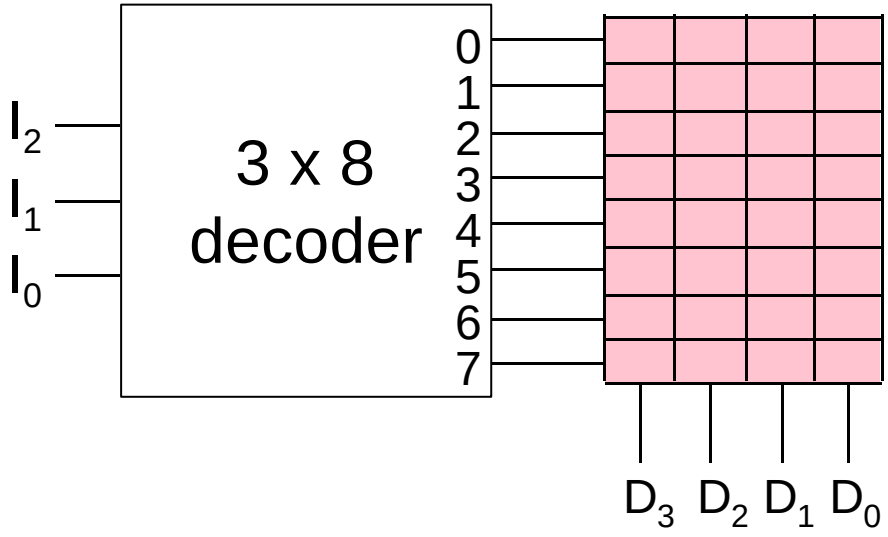
- PLD (programmable logic device)
  - 프로그램 가능한 fuse(스위치)로 연결된 게이트의 array로 구성된 IC
  - AND - OR array로 구성 → sum of product 실현
  - 초기상태는 모든 fuse가 연결되어 있어, fuse를 끊음으로써 프로그램
  - PLD는 AND-OR array에서 fuse의 배치 형태에 따라 PROM, PLA, PAL로 나뉨

ROM(Read Only Memory)

- 영구적으로 이진정보를 저장, 전원이 차단되더라도 정보를 저장(비휘발성)



- 8 x 4 ROM



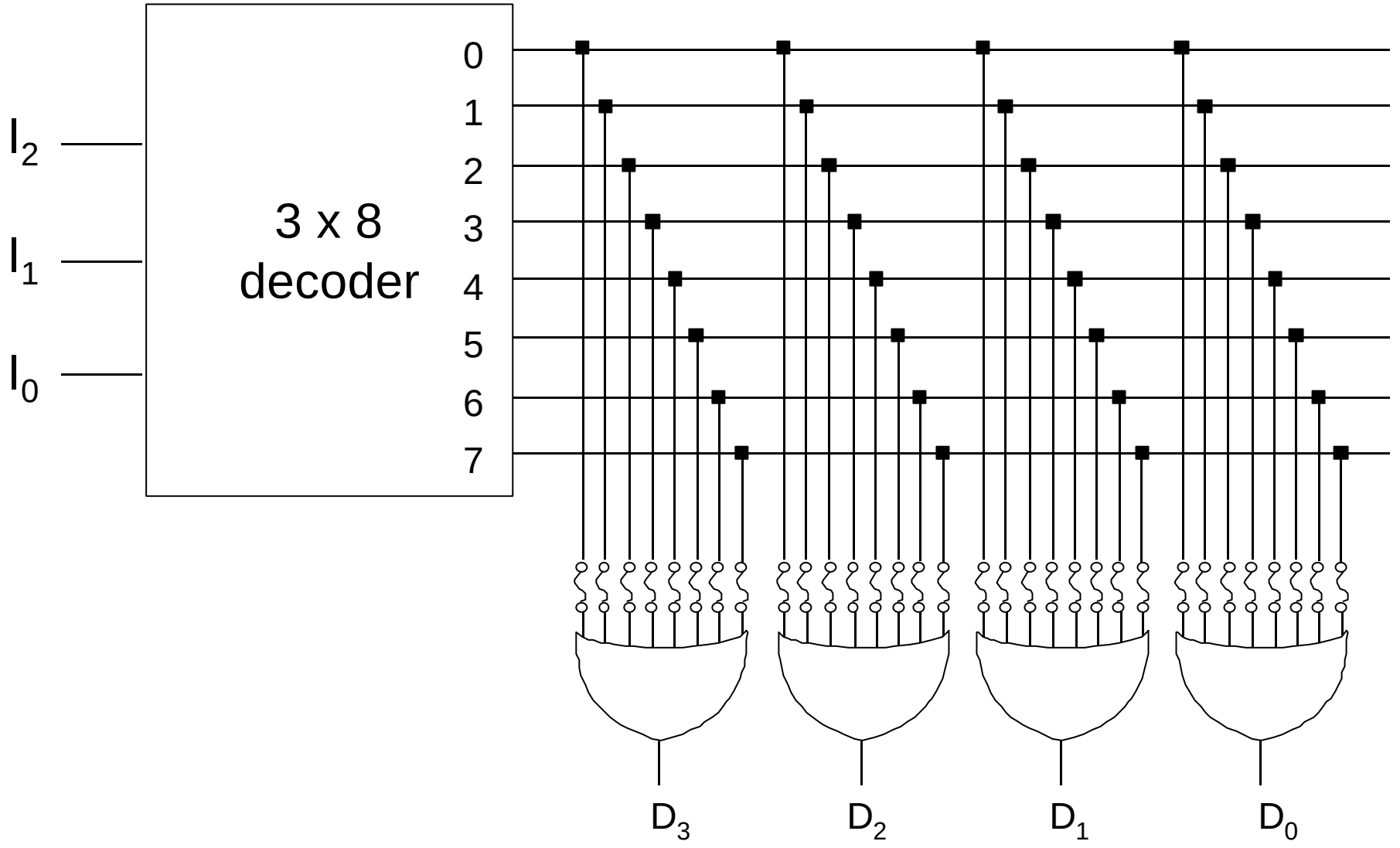
☞ 3 inputs(address line)
8 words
4bits/word

- 일반적으로 2^K x n ROM 에서

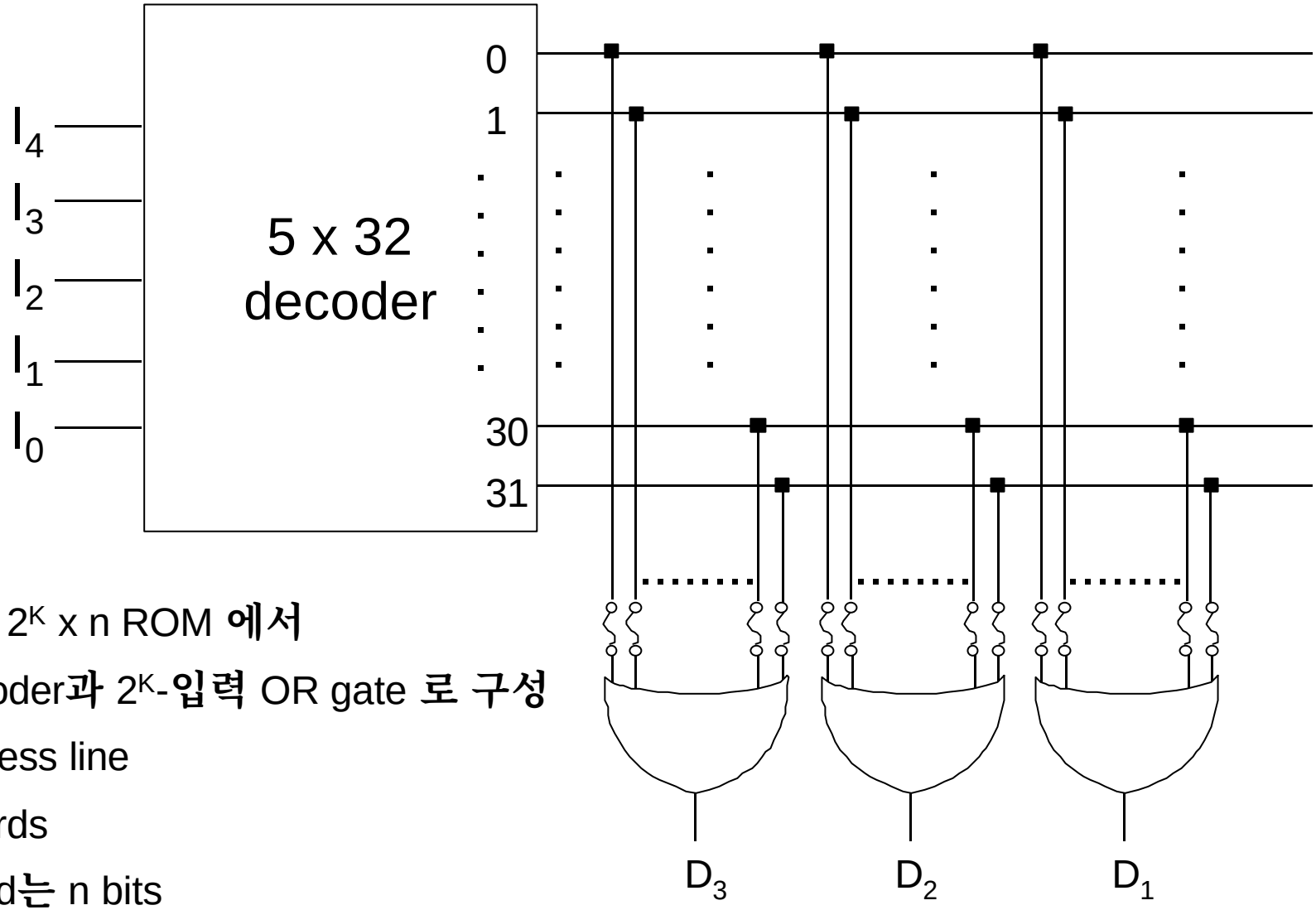
K x 2^K decoder, 2^K-입력 OR gate로 구성

K개의 address line, 2^K 개의 words, 하나의 word당 n bits

8x4 ROM Structure



32x3 ROM Structure



★ **일반적으로** 2^K x n ROM **에서**

K x 2^K decoder**과** 2^K-**입력** OR gate **로 구성**

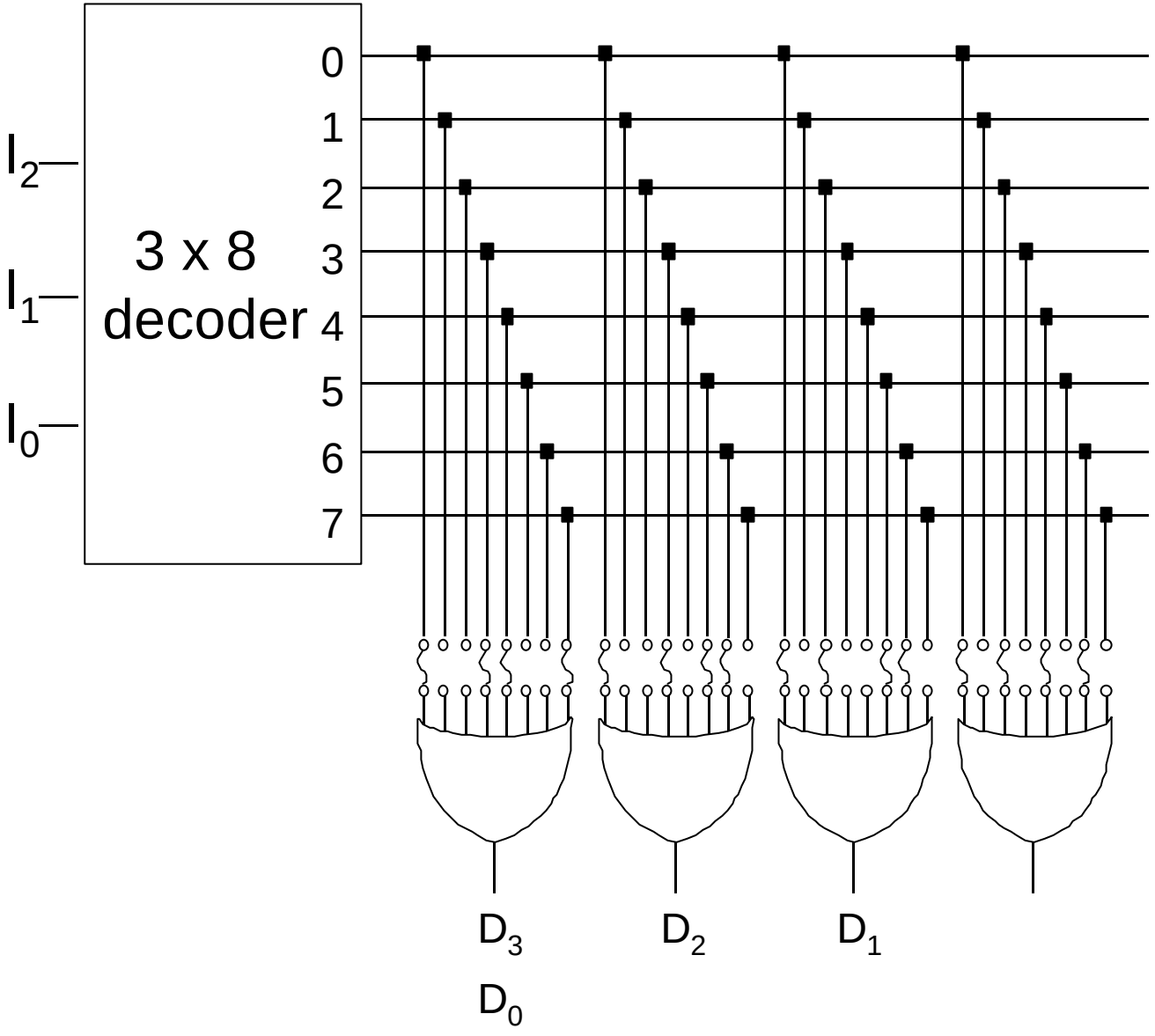
K**개의** address line

2^K **개의** words

하나의 word**는** n bits

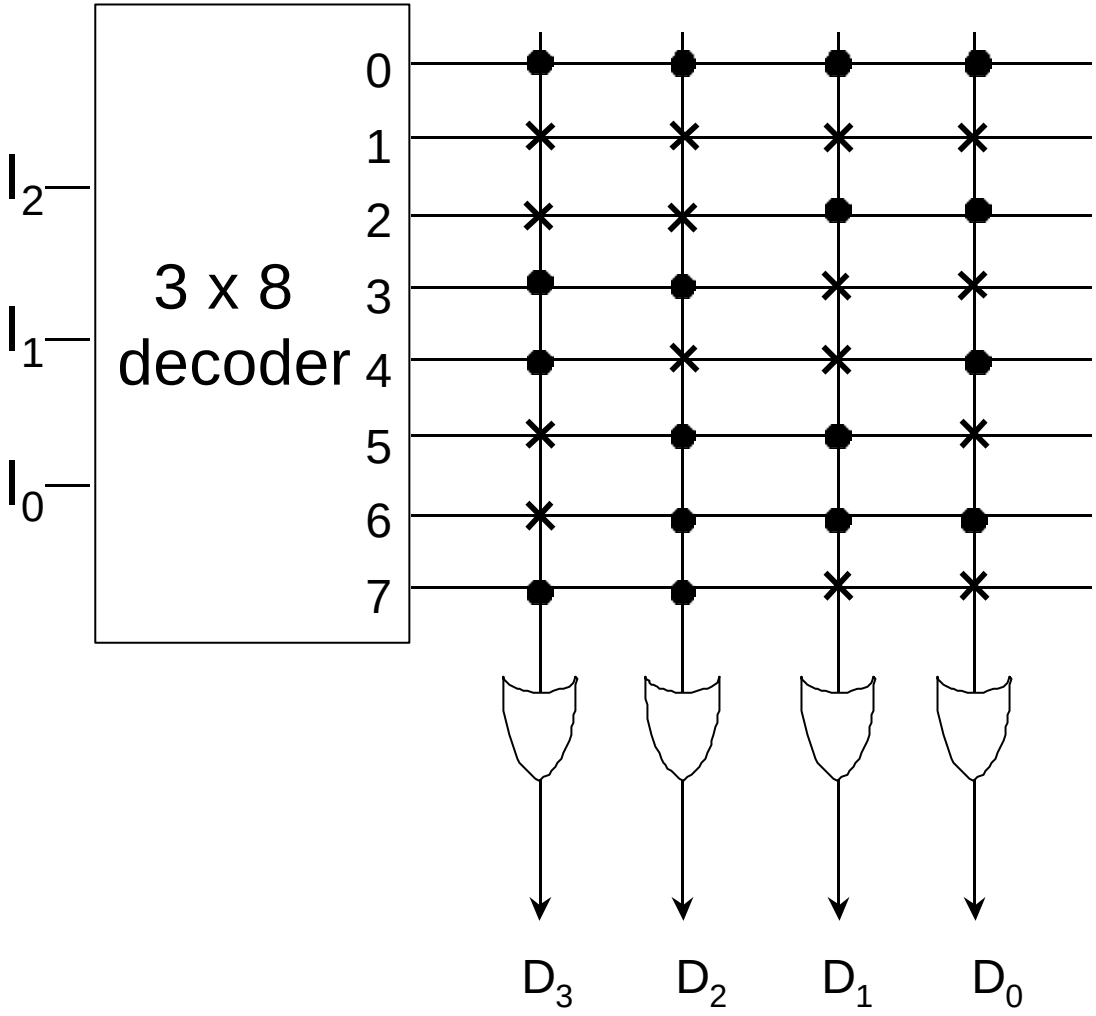
8x4 ROM Programming

주소	D_3	D_2	D_1	D_0
0	1	1	1	1
1	0	0	0	0
2	0	0	1	1
3	1	1	0	0
4	1	0	0	1
5	0	1	1	0
6	0	1	1	1
7	1	0	0	0



8x4 ROM Programming

주소	D_3	D_2	D_1	D_0
0	1	1	1	1
1	0	0	0	0
2	0	0	1	1
3	1	1	0	0
4	1	0	0	1
5	0	1	1	0
6	0	1	1	1
7	1	0	0	0



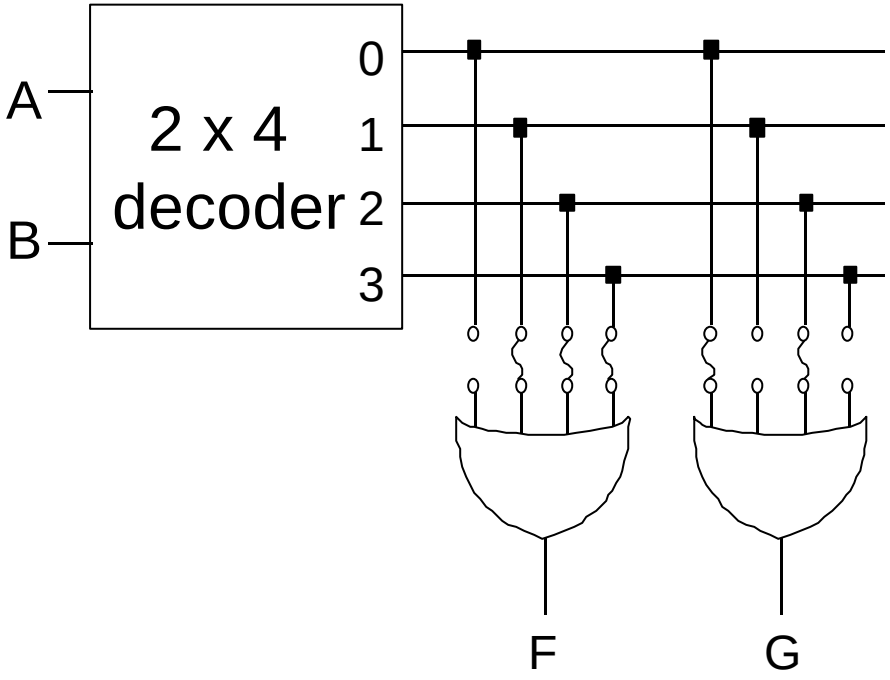
ROM을 이용한 조합회로의 구현

(ex) F(A,B) = ∑(1,2,3)

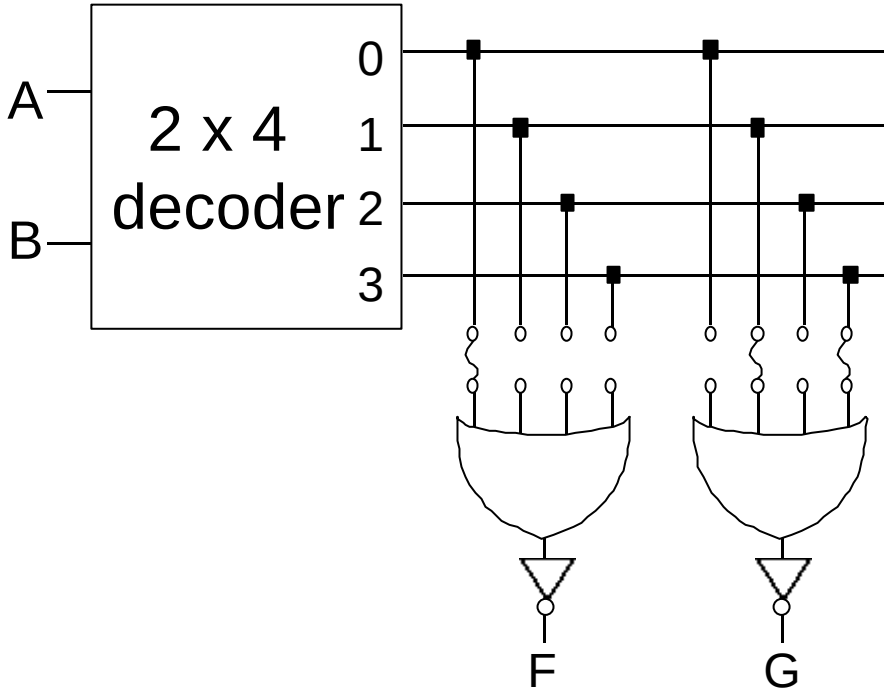
G(A,B) = ∑(0,2)

A	B	F	G
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

ROM with AND-OR



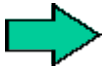
ROM with AND-OR-NOT



ROM을 이용한 조합회로의 구현

❑ 입력: 3bit 이진수, 출력: 제곱에 해당되는 이진수

입 력			출 력						십진
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

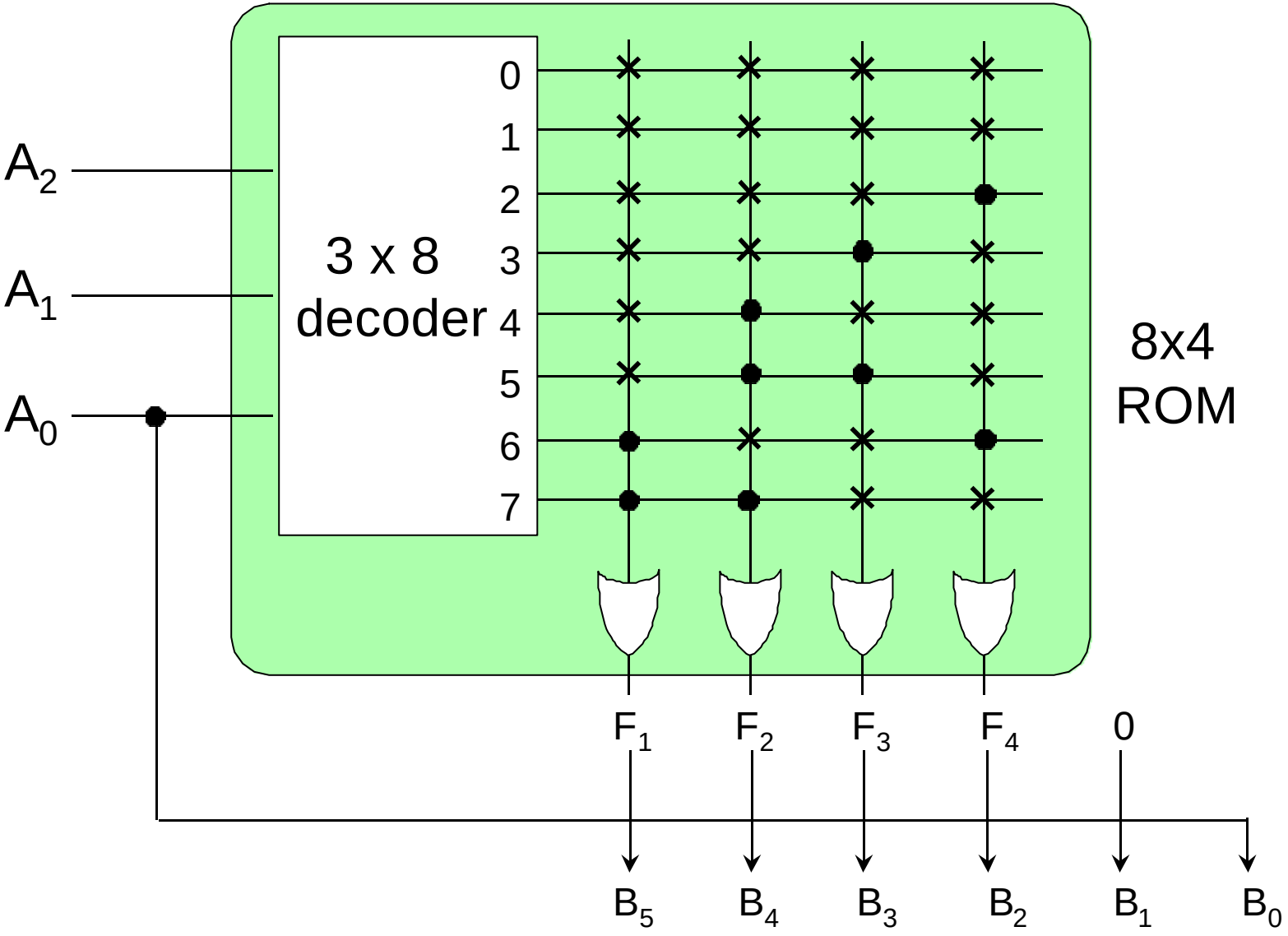


입 력			출 력			
A_2	A_1	A_0	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

- $B_1 = 0$, $B_0 = A_0$
- 필요한 최소 ROM : 3입력, 4출력
  (8x4 ROM)

8x4 ROM

ROM을 이용한 조합회로의 구현

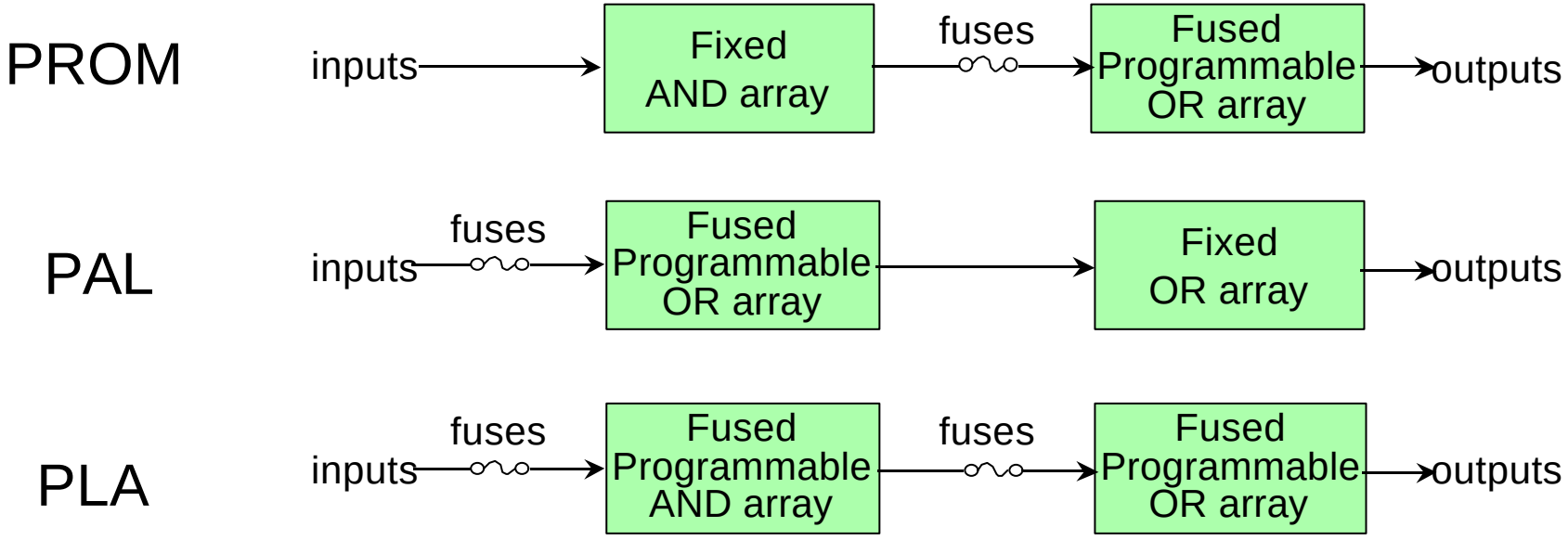


ROM의 종류

- ROM이 프로그램되는 방식에 따라
  - ROM
    - 반도체 공장에서 진리표에 의해 fuse를 프로그램 (mask programming)
    - 대량생산에 적합
    - 일단 프로그램되면 변경 불가능
  - PROM (Programmable Read-Only Memory)
    - 초기에는 모든 fuse가 연결된 상태로 출하
    - PROM의 fuse에 전류 펄스를 인가함으로써 끊어서 사용
    - 일단 프로그램되면 변경 불가능
  - EPROM (Erasable Programmable Read-Only Memory)
    - 자외선을 일정 시간 이상 쬐면 원래의 상태로 복원되므로
    - 초기상태로 만든후 다시 프로그램하여 여러번 사용 가능
  - EEPROM(Electrically Erasable Programmable Read-Only Memory)
    - 전기적 신호를 가함으로써 초기상태로 복원
    - 초기상태로 만든후 다시 프로그램하여 여러번 사용 가능

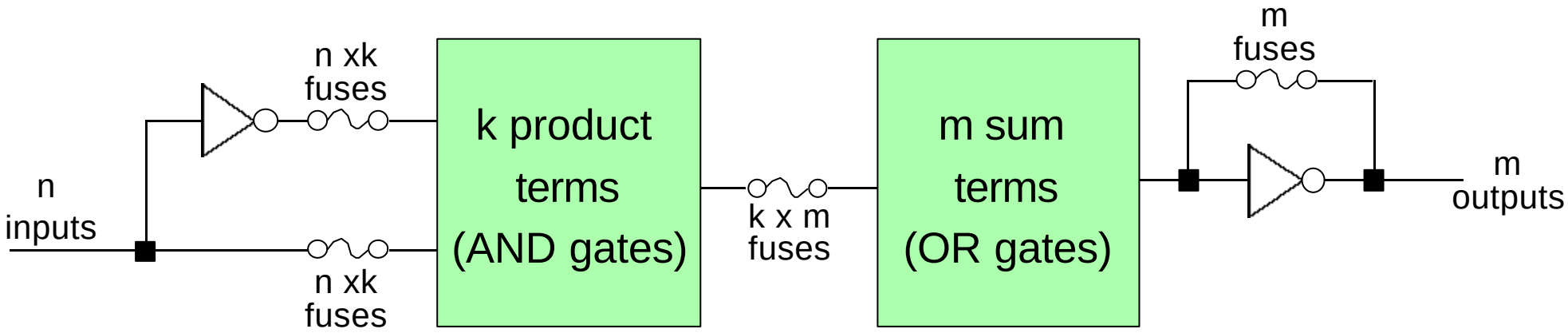
PLD(Programmable Logic Device)

- PLD (programmable logic device)
  - 프로그램 가능한 fuse(스위치)로 연결된 게이트의 array로 구성된 IC
  - AND - OR array로 구성 → sum of product 실현
  - PLD의 초기상태는 모든 fuse가 연결되어 있으므로 fuse를 끊음으로써 프로그램
  - PLD는 AND-OR array에서 fuse의 배치 형태에 따라서 PROM, PLA, PAL



Programmable Logic Array(PLA)

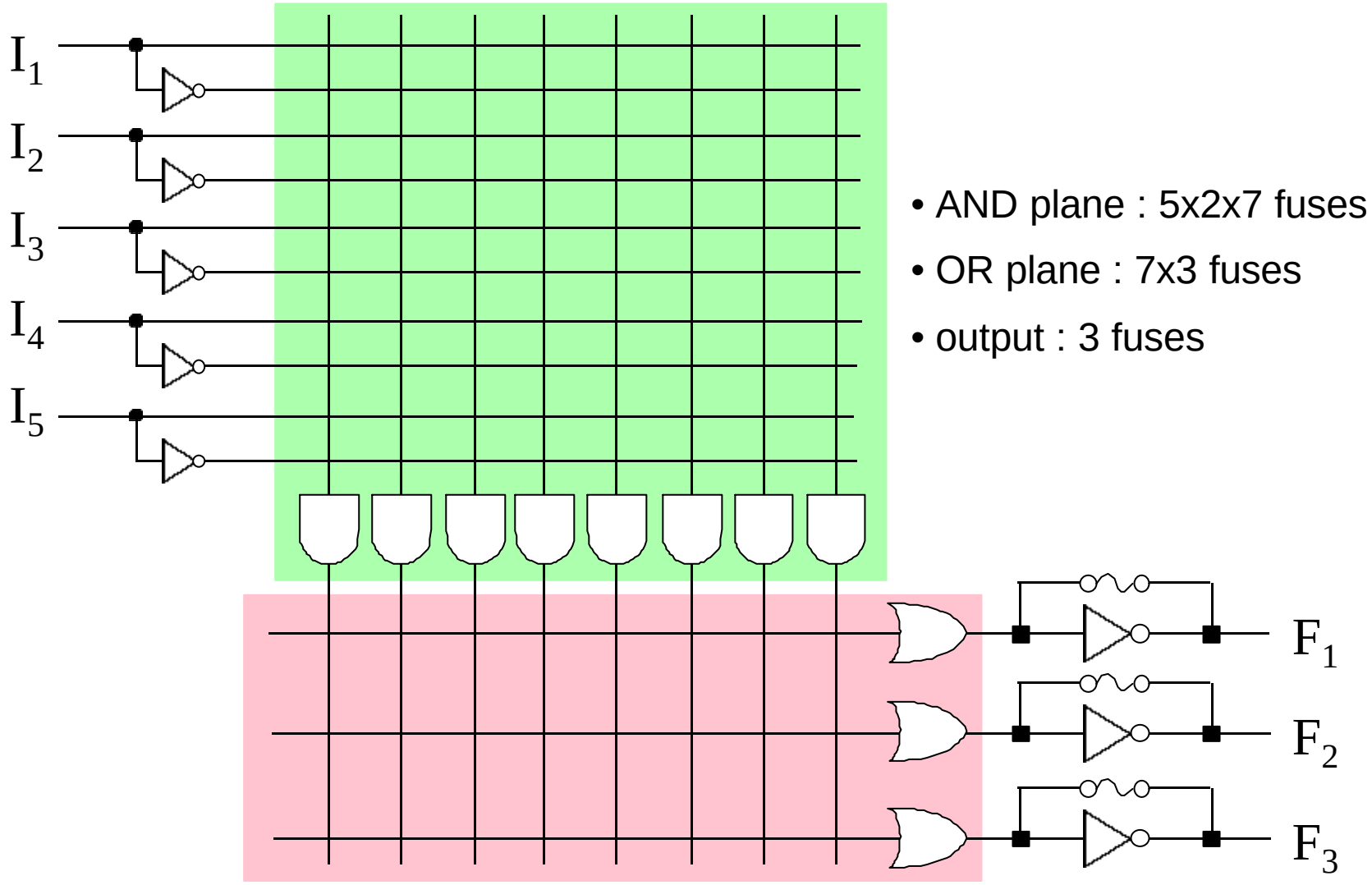
- PLA내부에 AND, OR gate가 fuse를 사용하여 이미 구성
  → 필요에 따라서 sum of product 형태의 부울함수를 fuse를 끊음으로써 프로그램
- PLA는 PROM과 달리 decoder가 불필요
  → 입력의 조합으로 만들수 있는 모든 최소항을 생성하지 않는다
  → 필요한 product term만 생성함으로 PROM에 비해서 효율적
- PLA의 크기 : product term의 갯수, 입력 수, 출력 수에 따라 결정
  - 일반적인 PLA IC : 16 inputs, 48 product terms, 8 outputs



※ fuse의 갯수 = 2nxk + kxm + m, ROM의 fuse = 2^nxm

PLA의 구조

- PLA with 5 inputs, 7 product terms, 3 outputs



- AND plane : 5x2x7 fuses
- OR plane : 7x3 fuses
- output : 3 fuses

PLA를 이용한 조합회로의 설계

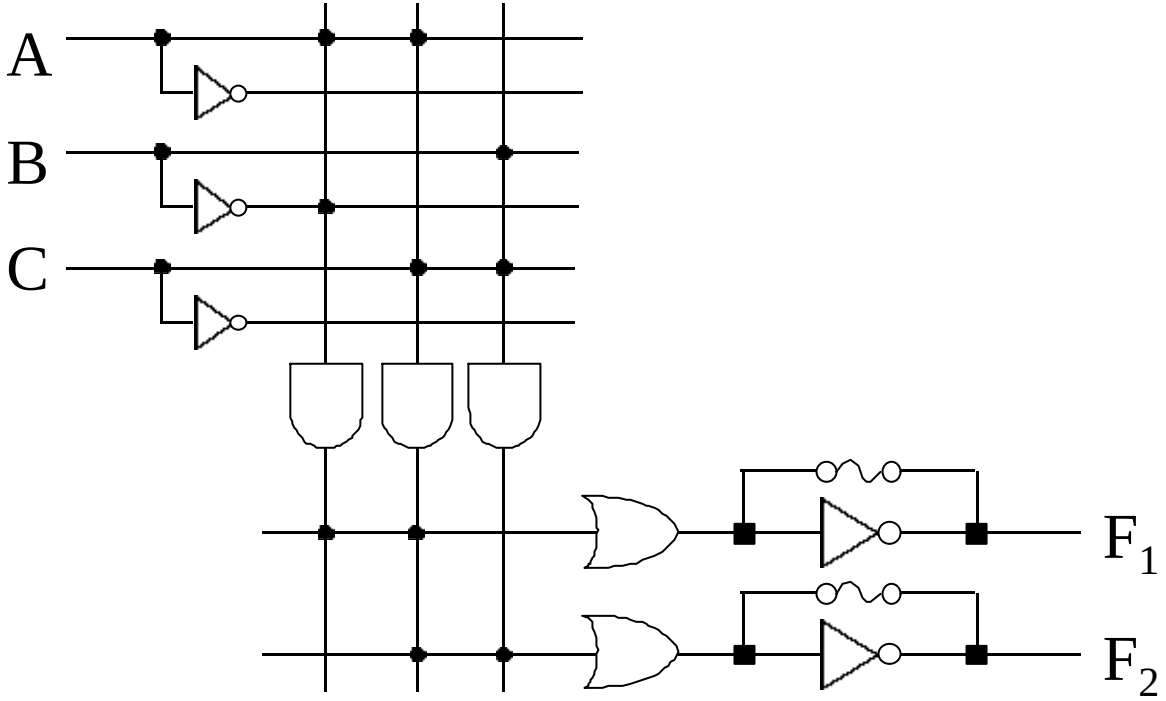
(ex) $F_1(A,B,C) = \sum(4,5,7)$, $F_2(A,B,C) = \sum(3,5,7)$

A \ BC	00	01	11	10
0				
1	**1**	**1**	**1**	

$F_1 = AB' + AC$

A \ BC	00	01	11	10
0			**1**	
1		**1**	**1**	

$F_2 = AC + BC$



PLA를 이용한 조합회로의 설계

(ex) $F_1(A,B,C) = \sum(0,1,2,4)$

$F_2(A,B,C) = \sum(0,5,6,7)$

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$F_1 = A'B' + A'C' + B'C'$

$F_1' = AB + AC + BC$

A \ BC	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$F_2 = AB + AC + A'B'C'$

$F_2' = A'C + A'B + AB'C'$

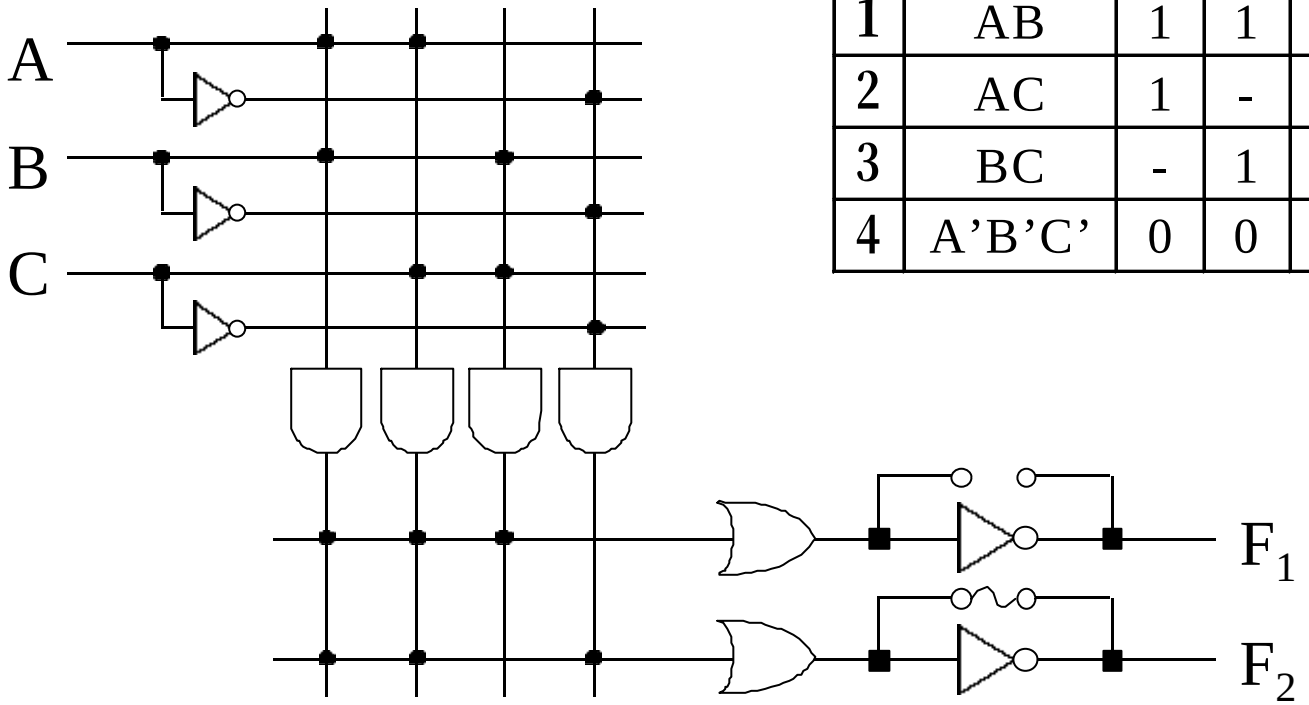
- F_1, F_1' 과 F_2, F_2' 중에서 공유하는 product term이 많은 쌍을 선택

∴ F_1' 와 F_2 는 공유하는 항이 가장 많으므로 선택

- PLA 프로그램표

Product term		입 력			출 력	
		A	B	C	F_1	F_2
1	AB	1	1	-	1	1
2	AC	1	-	1	1	1
3	BC	-	1	1	1	-
4	A'B'C'	0	0	0	-	1
					C	T

PLA를 이용한 조합회로의 설계



Product term		입 력			출 력	
		A	B	C	F_1	F_2
1	AB	1	1	-	1	1
2	AC	1	-	1	1	1
3	BC	-	1	1	1	-
4	A'B'C'	0	0	0	-	1
					C	T

ROM versus PLA

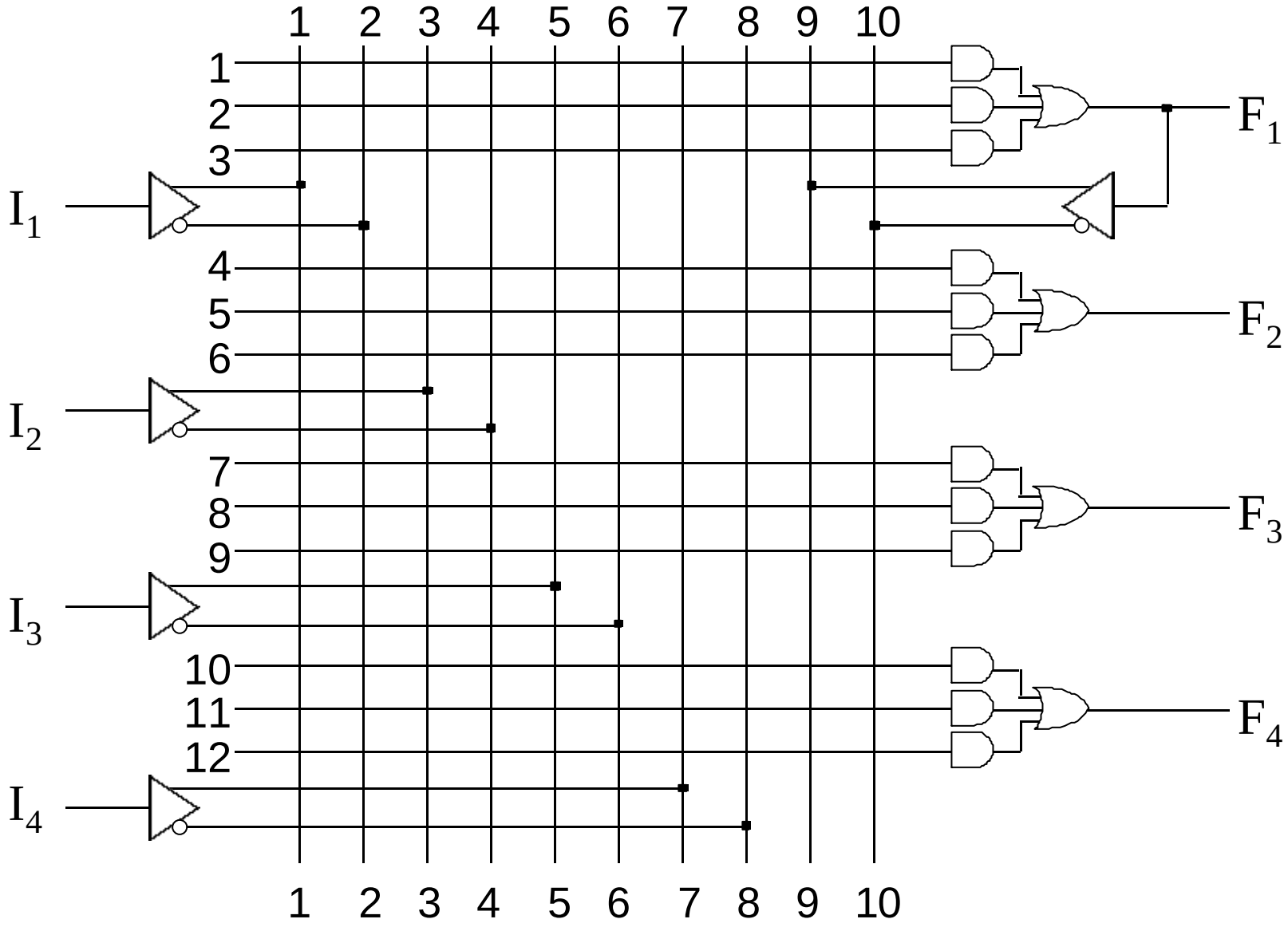
- ROMs are advantageous when
  - Design time is short
  - Need most or all input combinations (ex, code converter)
  - Little sharing of product terms
  - ROM problems
    - Size doubles for each additional input
    - Can' t exploit don' t care
- PLAs are advantageous when
  - Design tools allow logic minimization
  - Relatively unique minterms
  - Minterms are shared among output functions
  - PLA problems
    - Hard-wired fanins on OP plane

Programmable Array Logic(PAL)

- 고정된 OR gate, 프로그램 가능한 AND array
- 출력함수중 하나가 buffer 또는 inverter를 통하여 AND gate로 feedback
- PLA와 달리 product term을 공유하지 않으므로, 출력함수 자체를 간략화해야 한다
- Cheaper and faster than PLA
- PLA and PAL

PLA	PAL
Programmable AND array	Programmable AND array
Programmable OR array	Fixed OR array
Sharing of AND term	No sharing of AND term

PAL의 구조


1 2 3 4 5 6 7 8 9 10
1
2
3
I1
4
5
6
I2
7
8
9
I3
10
11
12
I4
F1
F2
F3
F4
1 2 3 4 5 6 7 8 9 10

PAL을 이용한 조합회로의 설계

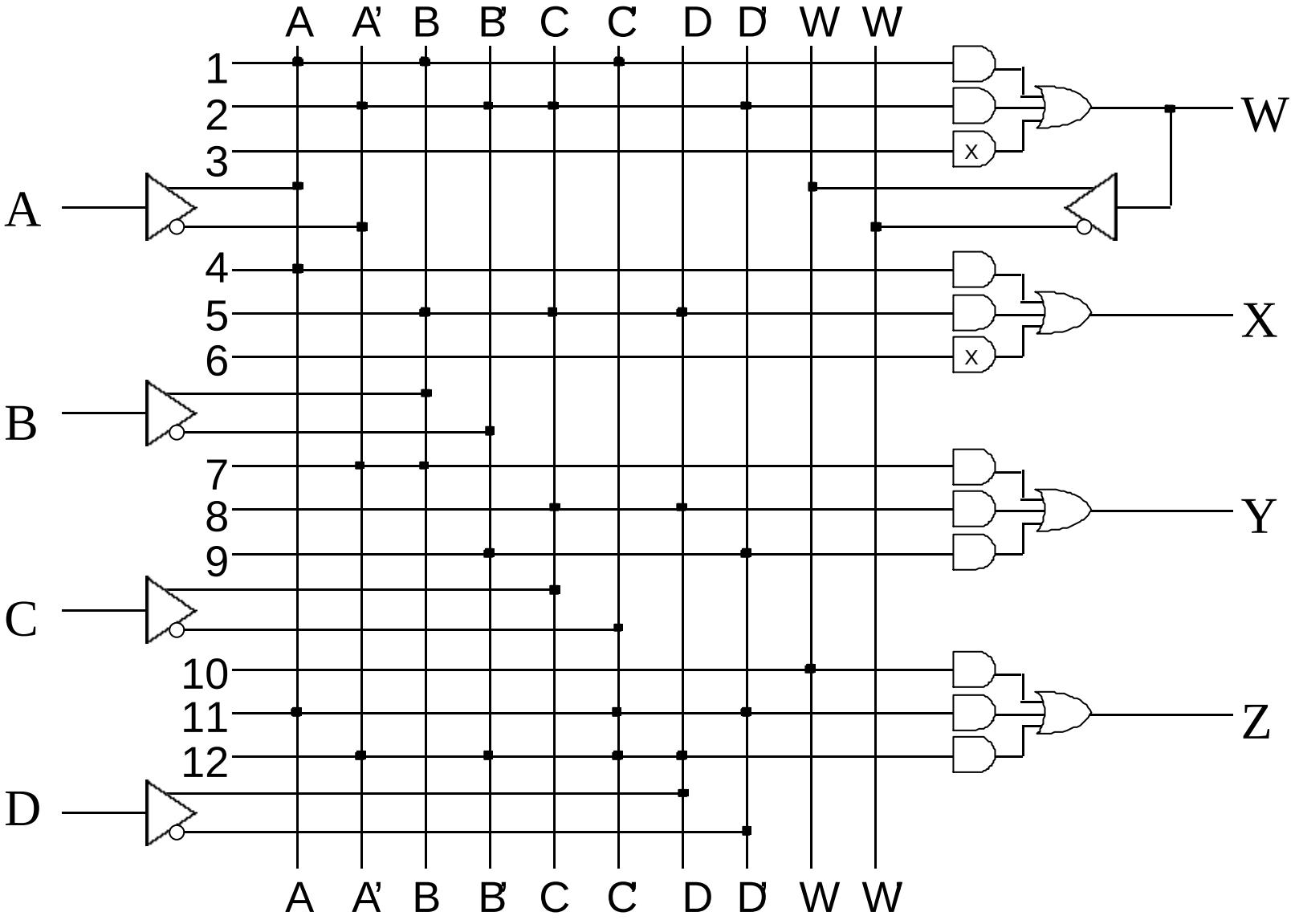
W(A,B,C,D) = ∑(2,12,13)
X(A,B,C,D) = ∑(7,8,9,10,11,12,13,14,15)
Y(A,B,C,D) = ∑(0,2,3,4,5,6,7,8,10,11,15)
Z(A,B,C,D) = ∑(1,2,8,12,13)

부울함수 간소화에 의해

W = ABC' + A' B' CD'
X = A + BCD
Y = A' B + CD + B' D'
Z = ABC' + A' B' CD' + AC' D' + A' B' C D
= W + AC' D' + A' B' C D

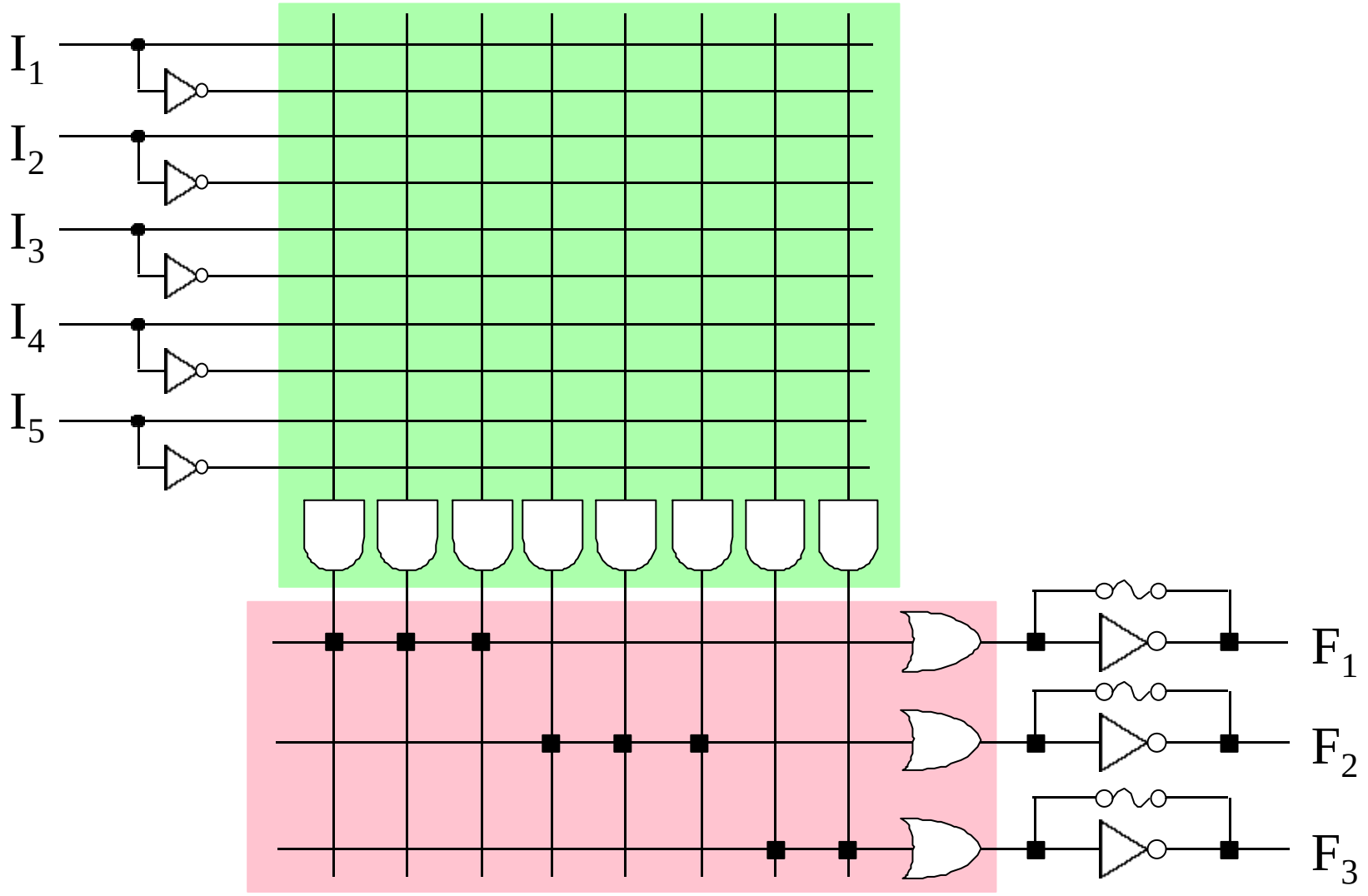
곱항	AND 입력					출력
	A	B	C	D	W	
ABC' A'B'CD'	1 0 -	1 0 -	0 1 -	- 0 -	- - -	W= ABC'+ A'B'CD'
A BCD	1 - -	- 1 -	- 1 -	- 1 -	- - -	X= A + BCD
A'B CD B'D'	0 - -	1 - 0	- 1 -	- 1 0	- - -	Y= A'B + CD + B'D'
W AC'D' A'B'C'D	- 1 0	- - 0	- 0 0	- 0 1	1 - -	Z= W + AC'D'+ A'B'C'D

PAL을 이용한 조합회로의 설계



ROM/PLA/PAL

- ROM
  - Full AND plane, general OR plane
  - Simple to design
  - Can implement any function of n inputs
- PLA
  - Programmable AND and OR plane
  - Complex to design
  - Slow because of two programmable planes
  - Can implement any function up to the number of product terms
- PAL
  - Programmable AND plane and fixed OR plane
  - Moderate to design
  - Fast because of one programmable plane that is smaller than ROM decoder
  - Can implement any function limited by the number of AND or OR terms


I1
I2
I3
I4
I5
F1
F2
F3