ディジタル電子回路

第3回

論理関数の実現に必要な基本論理回路

任意の論理関数は、・、+、$\overline{\ }$ を用いて書くことができる。
（AND回路、OR回路、NOT回路の組み合わせで実現できる。）

AND、OR、NOTすべて必要か？ ⟶ No！

ド・モルガンの法則より $\overline{A \cdot B} = A + B$

ANDとNOTがあれば、OR回路を実現できる。
⟶ ANDとNOTですべての論理関数を実現できる。

同様に、$\overline{\overline{A} + \overline{B}} = A \cdot B$　　従ってORとNOTだけでもOK。

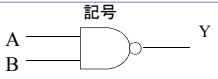
- AND、OR、NOT
- AND、NOT
- OR、NOT
- NAND
- NOR

このような回路の組み合わせでOK
NAND、NORは1つですべての論理関数を表現できる。
（万能論理関数集合とも呼ぶ）

NAND

AND演算の後に否定をとったもの

記号



論理式による表現

$Y = \overline{A \cdot B}$

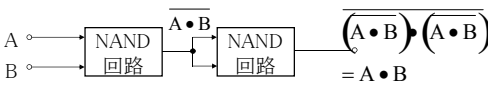
$(Y = \overline{A \wedge B} \quad Y = \overline{A\ B} \quad Y = A \mid B)$

真理値表

A	B	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

$f(A,A) = \overline{A \cdot A} = \overline{A}$ (NOT)

$\overline{(A \cdot B) \cdot (A \cdot B)} = \overline{\overline{A \cdot B}} = A \cdot B$ (AND)

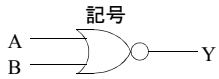


NOR

OR演算の後に否定をとったもの

論理式による表現

記号



$Y = \overline{A + B}$

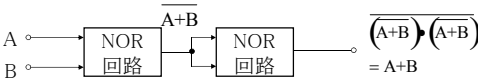
$(Y = \overline{A \vee B} \quad Y = A \downarrow B)$

真理値表

A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$f(A,A) = \overline{A + A} = A$ (NOT)

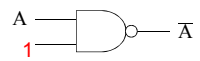
$\overline{(A+B)+(A+B)} = \overline{\overline{A+B}} = A+B$ (OR)



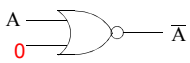
NANDとNORでNOTを表現

すでにやったように

以下のような場合でもNOTを出力する。



A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

NANDによる表現

NAND $\overline{A \cdot B}$ の形にする。　$\overline{A \cdot A} = \overline{A}$, $A \cdot B = \overline{\overline{A \cdot B}} = \overline{(A \cdot B) \cdot (A \cdot B)}$

$f(A,B,C) = (A + B) \cdot (B + C)$

$= \overline{\overline{(A \bullet B) \bullet (B \bullet C)}} = \overline{(A \bullet B) \bullet (B \bullet C)}$

$= \overline{(A \bullet B) \bullet (B \bullet C) \bullet (A \bullet B) \bullet (B \bullet C)}$

$= \overline{((A \bullet A) \bullet (B \bullet B)) \bullet ((B \bullet B) \bullet (C \bullet C)) \bullet ((A \bullet A) \bullet (B \bullet B)) \bullet ((B \bullet B) \bullet (C \bullet C))}$

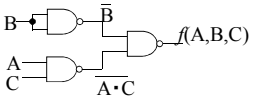
$\overline{A+B} = \overline{A} \cdot \overline{B}$
$\overline{A \cdot B} = \overline{A} + \overline{B}$

・・・と強引にやってもよいが・・・　分配則をつかうと・・

$A \cdot (B+C) = (A \cdot B) + (A \cdot C)$
$A + (B \cdot C) = (A+B) \cdot (A+C)$

$f(A,B,C) = (A+B) \cdot (B+C)$
$= B + A \cdot C = \overline{\overline{B + A \cdot C}}$
$= \overline{\overline{B} \cdot \overline{(A \cdot C)}} = \overline{(\overline{B} \cdot \overline{B}) \cdot (\overline{A \cdot C})}$

と簡単になる。



真理値表で確かめてみよう。

$f(A,B,C) = (A+B)\cdot(B+C) = \overline{(\overline{B\cdot B})\cdot(\overline{A\cdot C})}$

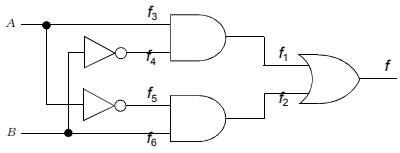
A	B	C	A+B	B+C	$f(A,B,C)$	$\overline{B\cdot B}$	$\overline{A\cdot C}$	$\overline{(\overline{B\cdot B})\cdot(\overline{A\cdot C})}$
0	0	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1	0
0	1	0	1	1	1	0	1	1
0	1	1	1	1	1	0	1	1
1	0	0	1	0	0	1	1	0
1	0	1	1	1	1	1	0	1
1	1	0	1	1	1	0	1	1
1	1	1	1	1	1	0	0	1

AND, OR, NOTゲートより構成される論理回路の解析

出力から順次各論理ゲートの出力状態を入力で表していく。(すでにやっている)

＜例＞

$f = f_1 + f_2$
$f_1 = f_3 + f_4$
$f_2 = f_5 + f_6$
$f_3 = A$
$f_4 = \overline{B}$
$f_5 = \overline{A}$
$f_6 = B$

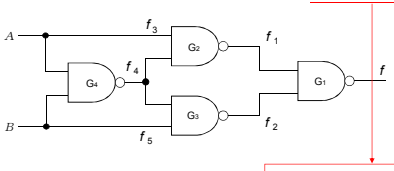


$f = A\cdot\overline{B} + \overline{A}\cdot B$

A	B	f(A,B)
0	0	0
0	1	1
1	0	1
1	1	0

NAND、NORゲートより構成される論理回路の解析

NAND、NORで構成される場合も同様であるが・・・多少見通しが悪い。



AND, OR, NOTによる表現に変換

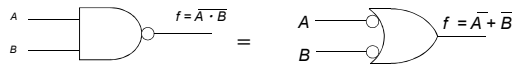
$f = \overline{(A\cdot\overline{A\cdot B})\cdot(\overline{A\cdot B}\cdot B)}$

$= \overline{(\overline{A}+(A\cdot B))\cdot((A\cdot B)+\overline{B})}$
$= \overline{\overline{A}\cdot B + A\cdot B} = \overline{A+B} + \overline{A+\overline{B}}$
$= \overline{(A+B)\cdot(\overline{A}+\overline{B})} = (A+B)\cdot(\overline{A}+\overline{B})$
$= A\cdot\overline{A} + A\cdot\overline{B} + \overline{A}\cdot B + B\cdot\overline{B}$
$= A\cdot\overline{B} + \overline{A}\cdot B$

A	B	f(A,B)
0	0	0
0	1	1
1	0	1
1	1	0

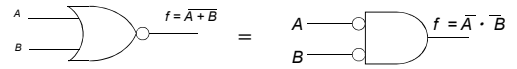
NAND、NORゲートの等価変換

NAND　$\overline{A\cdot B} = \overline{A} + \overline{B}$　(AとBの否定をとってからOR演算)



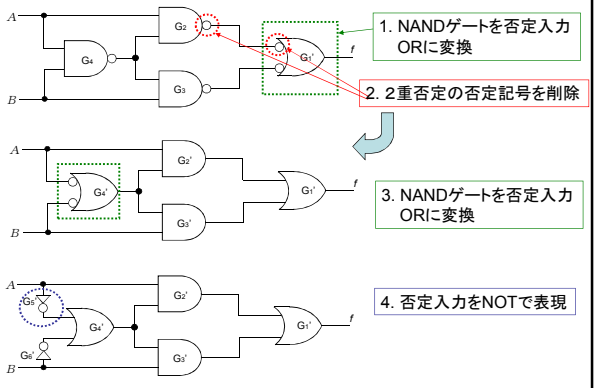
(a) NAND　　(b) 否定入力のOR

NOR　$\overline{A+B} = \overline{A}\cdot\overline{B}$　(AとBの否定をとってからAND演算)

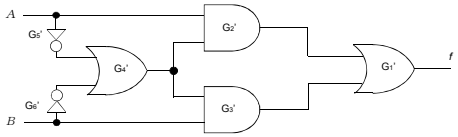


(a) NOR　　(b) 否定入力のAND

NANDゲート構成からAND、OR、NOT構成への変換



論理関数の確認

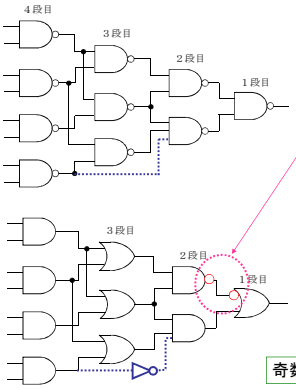


変形した上図から論理関数を求めてみると・・・

$f = A\cdot(\overline{A}+\overline{B}) + B\cdot(\overline{A}+\overline{B})$

$= A\cdot\overline{A} + A\cdot\overline{B} + \overline{A}\cdot B + B\cdot\overline{B}$
$= A\cdot\overline{B} + \overline{A}\cdot B$

NANDゲート構成からAND、OR、NOT構成への変換（一般論）



(i) 出力側より格段に番号をつける。

(ii) 奇数段目のNANDゲートを否定入力をORで置き換える。

(iii) 2重否定の否定記号を削除する。

ただし、打ち消すべき否定が片方しかない場合はNOTを挿入する。

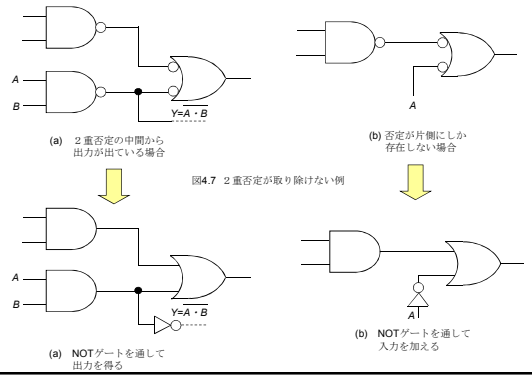
点線のようなパスがはいると・・・

NOTを１個挿入しないといけない

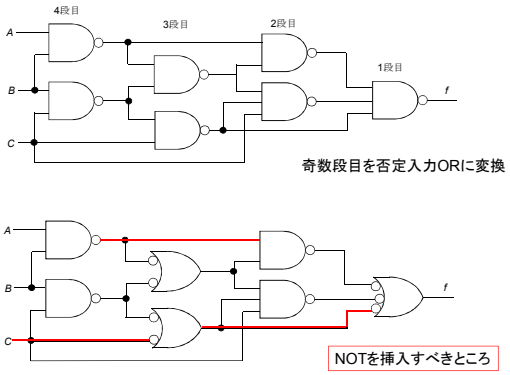
奇数段目がOR、偶数段目がANDになる

NANDゲート構成からAND、OR、NOT構成への変換（一般論）

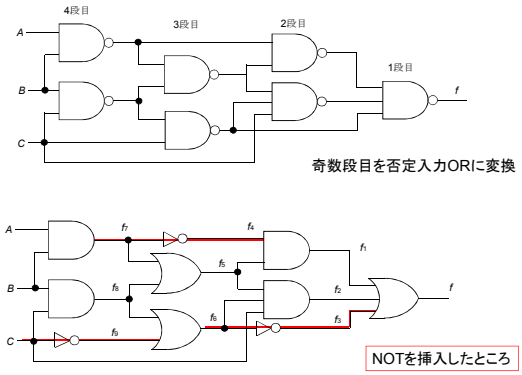
打ち消すべき否定が片方しかない場合の例



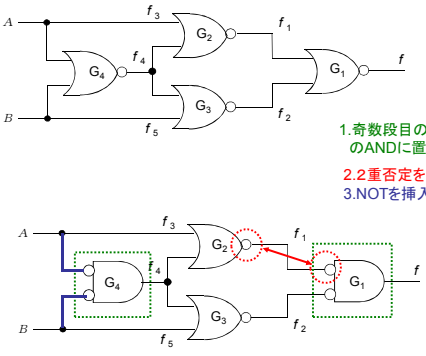
NANDゲート構成からAND、OR、NOT構成への変換例



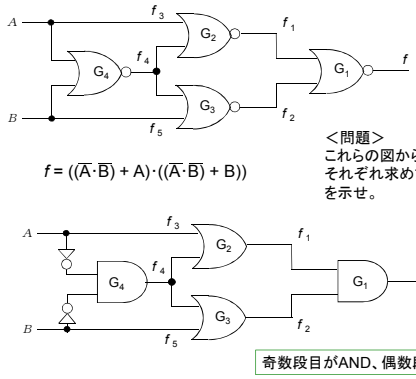
NANDゲート構成からAND、OR、NOT構成への変換例



NORゲート構成からAND、OR、NOT構成への変換例



NORゲート構成からAND、OR、NOT構成への変換例

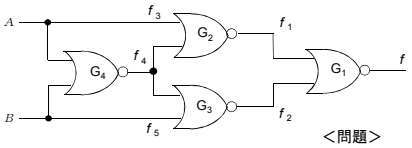


$f = ((\overline{A \cdot B}) + A) \cdot ((\overline{A \cdot B}) + B))$

＜問題＞
これらの図から論理関数をそれぞれ求めて、一致することを示せ。

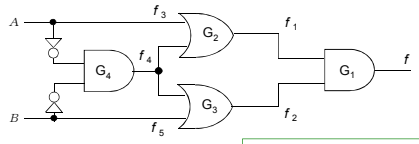
奇数段目がAND、偶数段目がORになる

NORゲート構成からAND、OR、NOT構成への変換例



$f = \overline{(\overline{A+B}) + A + (\overline{A+B}) + B}$

＜問題＞
これらの図から論理関数を
それぞれ求めて、一致すること
を示せ。



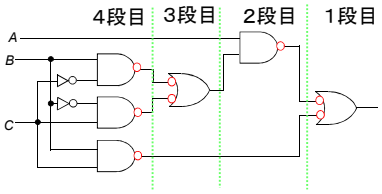
$f = ((\overline{A}\cdot\overline{B}) + A)\cdot((\overline{A}\cdot\overline{B}) + B))$

奇数段目がAND、偶数段目がORになる

AND,OR,NOT積和形からNAND構成への変換

AND、OR、NOTで実現した積和形の回路

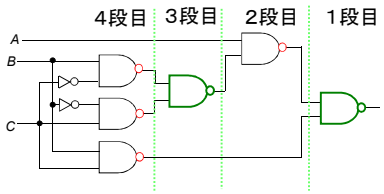
(i) 出力から数えて各段に番号をつける。
(ii) 奇数段の論理ゲートの入力側と偶数段の論理ゲートの出力側に否定記号（○）をつけ、両端に○がある場合はそのまま、片側のみの場合はNOTを挿入する。
(iii) 否定入力の論理ゲート（OR）をNANDに書き換える。
(iv) 必要ならNOTもNANDで表現する。



AND,OR,NOT積和形からNAND構成への変換

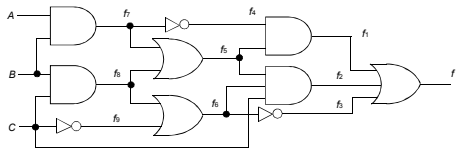
AND、OR、NOTで実現した積和形の回路

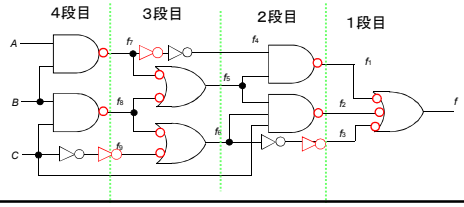
(i) 出力から数えて各段に番号をつける。
(ii) 奇数段の論理ゲートの入力側と偶数段の論理ゲートの出力側に否定記号（○）をつけ、両端に○がある場合はそのまま、片側のみの場合はNOTを挿入する。
(iii) **否定入力の論理ゲート（OR）をNANDに書き換える。**
(iv) 必要ならNOTもNANDで表現する。



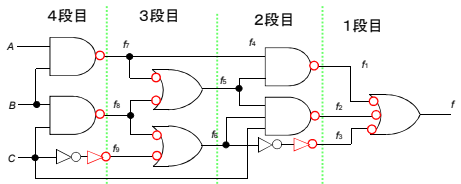
例題

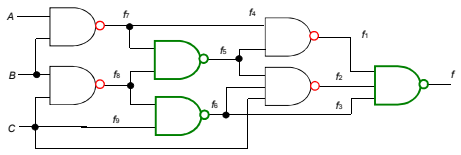
前にやった以下の回路をNAND構成に戻してみよう。





NAND構成への変換プロセス





組み合わせ論理回路の例

半加算器：1ビットの2進数を2つ加算し、和と桁上がりを出力

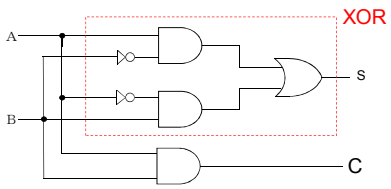
真理値表

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

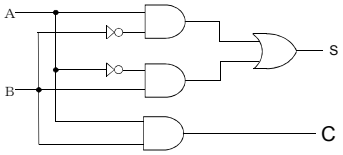
S：和（Sum）、C：桁上がり（Carry）

$S = \overline{A}\cdot B + A\cdot\overline{B}$　実はXOR

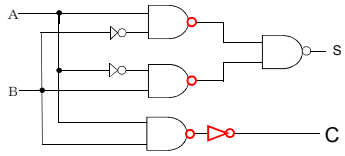
$C = A\cdot B$　桁上がりはAND



半加算器(NAND構成)



NANDで構成すると・・・



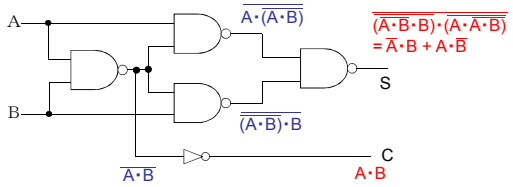
半加算器(NAND構成)

$S = \overline{\overline{A\cdot B} + A\cdot B} = \overline{(A\cdot B)}\cdot\overline{(A\cdot B)}$

$= \overline{(A\cdot B + B\cdot B)}\cdot\overline{(A\cdot B + A\cdot A)}$

$= \overline{((A + B)\cdot B)}\cdot\overline{(A\cdot(A + B))}$

$= \overline{(A\cdot B\cdot B)}\cdot\overline{(A\cdot A\cdot B)}$　　と変形すれば、以下でもよい。

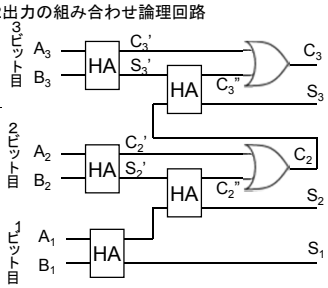


全加算器

1ビット分をみると・・・3入力2出力の組み合わせ論理回路

真理値表

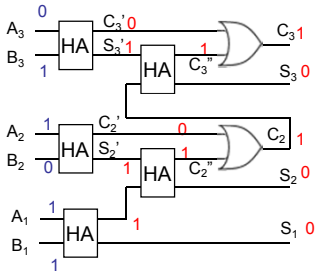
A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$S_i = \overline{A_i}\cdot\overline{B_i}\cdot C_{i-1} + \overline{A_i}\cdot B_i\cdot\overline{C_{i-1}} + A_i\cdot\overline{B_i}\cdot\overline{C_{i-1}} + A_i\cdot B_i\cdot C_{i-1}$

$C_i = \overline{A_i}\cdot B_i\cdot C_{i-1} + A_i\cdot\overline{B_i}\cdot C_{i-1} + A_i\cdot B_i\cdot\overline{C_{i-1}} + A_i\cdot B_i\cdot C_{i-1}$

全加算器



$(011)_2+(101)_2$= 3+5 = 8 = $(1000)_2$

$A = (011)_2$ $B = (101)_2$
として足し算をやってみよう。

1ビット目

1+1=10なので、S_1=0, C_1=1

2ビット目

1+0=1なのでS_2'=1, C_2'=0

1ビット目からの繰り上がりとS_2'を足して　S_2=0, C_2''=1

よって2ビット目の桁上がりは

C_2=1

3ビット目

0+1=1なのでS_3'=1, C_3'=0

2ビット目からの繰り上がりとS_3'を足して　S_3=0, C_3''=1

よって3ビット目の桁上がりは

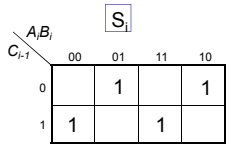
C_3=1

全加算器

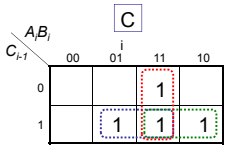
$S_i = \overline{A_i}\cdot\overline{B_i}\cdot C_{i-1} + \overline{A_i}\cdot B_i\cdot\overline{C_{i-1}} + A_i\cdot\overline{B_i}\cdot\overline{C_{i-1}} + A_i\cdot B_i\cdot C_{i-1}$

$C_i = \overline{A_i}\cdot B_i\cdot C_{i-1} + A_i\cdot\overline{B_i}\cdot C_{i-1} + A_i\cdot B_i\cdot\overline{C_{i-1}} + A_i\cdot B_i\cdot C_{i-1}$

カルノー図を書く



S_iは簡単化できない

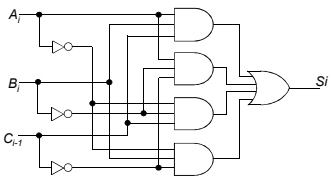


$C_i = A_i\cdot B_i + B_i\cdot C_{i-1} + C_{i-1}\cdot A_i$

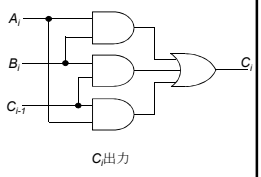
全加算器

$S_i = \overline{A_i}\cdot\overline{B_i}\cdot C_{i-1} + \overline{A_i}\cdot B_i\cdot\overline{C_{i-1}} + A_i\cdot\overline{B_i}\cdot\overline{C_{i-1}} + A_i\cdot B_i\cdot C_{i-1}$

$C_i = A_i\cdot B_i + B_i\cdot C_{i-1} + C_{i-1}\cdot A_i$



Si

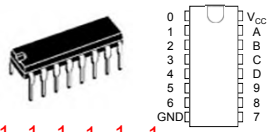


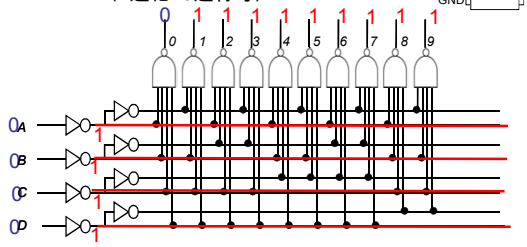
C_i出力

デコーダとエンコーダ

デコーダ(e.g. SN74145)
2進数 → 10進数、16進数など

エンコーダ
10進数 → 2進数
（2進化10進符号）

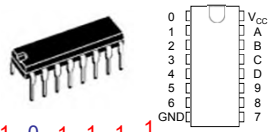


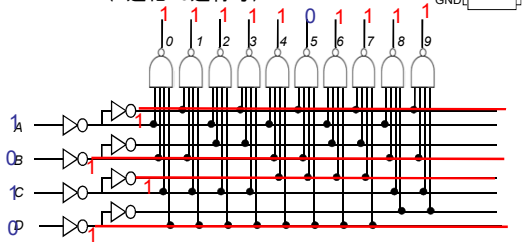


デコーダとエンコーダ

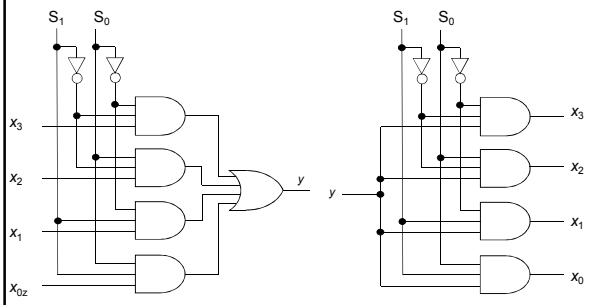
デコーダ(e.g. SN74145)
2進数 → 10進数、16進数など

エンコーダ
10進数 → 2進数
（2進化10進符号）





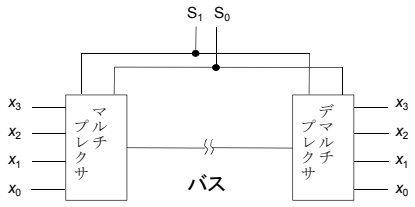
マルチプレクサとデマルチプレクサ



マルチプレクサ
（多重化装置）

デマルチプレクサ
（多重化複号装置）

マルチプレクサとデマルチプレクサ

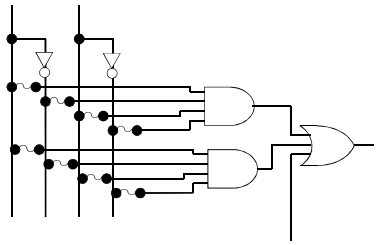


複数個のデータを1本の伝送路で送る。

cf. 通信方式でのTDMA、FDMA等

プログラマブル・ロジック・デバイス（PLD）

PAL(Programmable Array Logic):
ANDアレイが書き換え可能で、ORアレイが固定のもの。
ロジックの書き込みには微細なヒューズを用いる（書き込み一回きり）



GAL(Generic Array Logic):
ANDアレイが書き換え可能、ORアレイが固定で、ORアレイの出力をANDアレイに入力できるもの。
ロジックの書き込みにはEEPROMを用いる（再書き込み可能）

FPGA (Field Programmable Gate Array)

ロジックゲート、論理ブロックなどが多数配置された集積回路が市販されており、ユーザがプログラムして、独自の論理回路を実現できる。

・少量から所望の論理回路ICが実現できる。
・パソコンでプログラムして、短時間で論理回路ICが実現できる。
・変更が容易。
・カスタマイズが容易。

×ある程度の無駄が生じる。
×大量生産されたASICにはコストで劣る。
×フルカスタムの高性能LSIには性能で劣る。

しかし最近では、高速化、高集積化、低消費電力化、低コスト化へ

最近では液晶TV等の製品にも使用されているらしい。

主要メーカ：アルテラ（ALTERA）、ザイリンクス（XILINX）、その他多数