memo

ディジタル電子回路

授業開始までしばらくお待ちください。

オンライン視聴できない人へ。

オンラインで受講する人も基本的に一緒です。

自宅ネットワークの事情により、授業のストリーミング配信の視聴が困難な学生は以下の対応をしてください。

1. この授業のスライドをよく読んで、不明な点は自分で調べるなどして、わかる範囲で内容を理解する。
2. このページも含め、**必要な部分がすべて理解できたと思うまで以下の 2 ステップを繰り返す**。
   - わからない部分を e-mail 等で質問する。(宛先は `hiroyuki.kobayashi@oit.ac.jp`)
   - e-mail 等による返信をよく読んで理解する。
3. この資料の末尾にある課題を行い、この資料内の方法で (Google Forms で) 提出する。

授業の受講に関して

- 講義資料（スライド等）は**COMMON**に置く。
- 講義は**Google Meet** で行い、録画した講義は**Goole Drive**に置く。

  https://stream.meet.google.com/stream/1d1866da-5bff-4881-96b2-3745413fe31a

  https://drive.google.com/drive/folders/1bT-z3ICQyMYC_5JvlL29UZYqbOhVG492
- 出席確認レポートは**Google Forms** で提出。(毎回同一 URL)

  https://forms.gle/9ruwtfJg5LQgQNpU7
- **Slack**を補助的な連絡チャネルとする。必須ではないので使いたくなければ使わなくてもいい。授業に関連したちょっとした（重要でない）追加説明をする。気楽な質問手段としても活用されたい。登録は大学の e-mail アドレスで行うこと。

  https://oitkobayashi.slack.com

R/S 科ディジタル電子回路
Digital Electronics
『Karnaugh, too』

Google Meet

小林裕之・中泉文孝

大阪工業大学 RD 学部システムデザイン工学科・ロボット工学科

OSAKA INSTITUTE OF TECHNOLOGY

4 of 14

a LaTeX + Beamer slideshow

復習兼 Karnaugh map の限界を知る練習問題

簡単化して論理関数を求めよ。

ab \ cde	000	001	011	010	110	111	101	100
00	−	1	1	−	0	1	1	−
01	1	0	0	1	1	0	0	−
11	1	0	−	1	1	0	−	−
10	0	−	−	0	0	−	1	0

Karnaugh 図の限界を超えたい人はこちら → Quine-McCluskey 簡単化 | 検索

復習兼 Karnaugh map の限界を知る練習問題

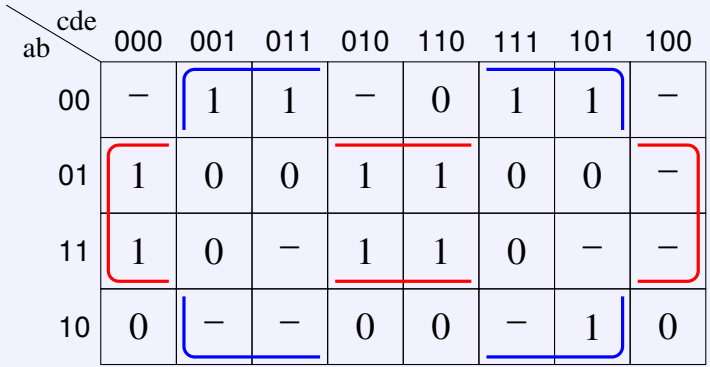
簡単化して論理関数を求めよ。

ab \ cde	000	001	011	010	110	111	101	100
00	−	1	1	−	0	1	1	−
01	1	0	0	1	1	0	0	−
11	1	0	−	1	1	0	−	−
10	0	−	−	0	0	−	1	0

Karnaugh 図の限界を超えたい人はこちら → Quine-McCluskey 簡単化 | 検索

復習兼 Karnaugh map の限界を知る練習問題

簡単化して論理関数を求めよ。



Karnaugh 図の限界を超えたい人はこちら → Quine-McCluskey 簡単化 | 検索

復習兼 Karnaugh map の限界を知る練習問題

簡単化して論理関数を求めよ。

ab \ cde	000	001	011	010	110	111	101	100
00	−	1	1	−	0	1	1	−
01	1	0	0	1	1	0	0	−
11	1	0	−	1	1	0	−	−
10	0	−	−	0	0	−	1	0

答: $Y = \overline{b} \cdot e + b \cdot \overline{e}$

Karnaugh 図の限界を超えたい人はこちら → Quine-McCluskey 簡単化 | 検索

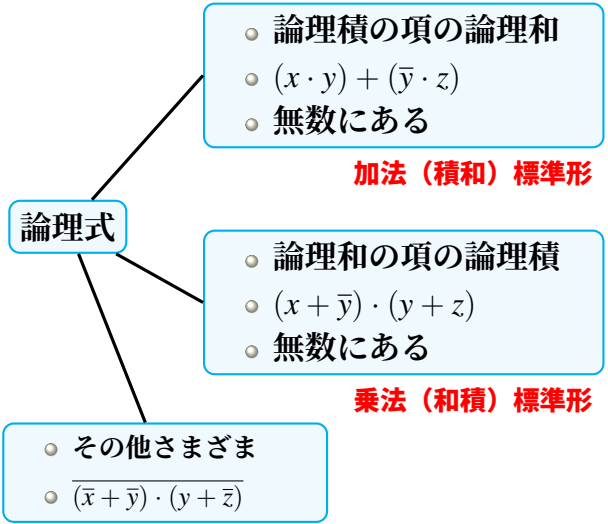
ややこしい言葉たち

ちなみに図中の論理式はすべて同じものの変形です。

論理式

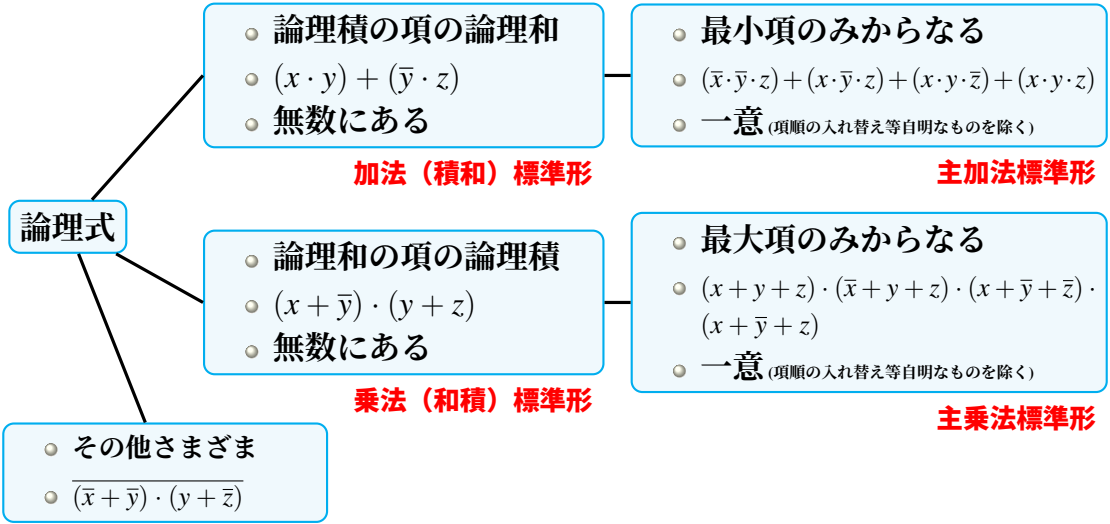
ややこしい言葉たち

ちなみに図中の論理式はすべて同じものの変形です。



ややこしい言葉たち

ちなみに図中の論理式はすべて同じものの変形です。



もうひとつの Karnaugh map による簡単化

1. 作りたいもの（論理関数）がありました。
2. **真理値表**として表現するのは簡単。
3. **論理式**として表現するのは無数にあるけれど、**主 {加, 乗} 法標準形**ならば一意で簡単。
4. 簡単化は難しいので **Karnaugh map** を使う。

→ _____ としての簡単化（既にやった）

→ _____ としての簡単化

もうひとつの Karnaugh map による簡単化

1. 作りたいもの（論理関数）がありました。
2. **真理値表**として表現するのは簡単。
3. **論理式**として表現するのは無数にあるけれど、**主 {加, 乗} 法標準形**ならば一意で簡単。
4. 簡単化は難しいので **Karnaugh map** を使う。

→ **加法 (積和) 標準形** としての簡単化（既にやった）

→ _____ としての簡単化

もうひとつの Karnaugh map による簡単化

1. 作りたいもの（論理関数）がありました。
2. **真理値表**として表現するのは簡単。
3. **論理式**として表現するのは無数にあるけれど、**主 {加, 乗} 法標準形**ならば一意で簡単。
4. 簡単化は難しいので **Karnaugh map** を使う。

   → **加法 (積和) 標準形** としての簡単化（既にやった）

   → **乗法 (和積) 標準形** としての簡単化

もうひとつの Karnaugh map による簡単化

1. 作りたいもの（論理関数）がありました。
2. **真理値表**として表現するのは簡単。
3. **論理式**として表現するのは無数にあるけれど、**主 {加, 乗} 法標準形**ならば一意で簡単。
4. 簡単化は難しいので **Karnaugh map** を使う。

→ **加法 (積和) 標準形** としての簡単化（既にやった）
→ **乗法 (和積) 標準形** としての簡単化

今回は後者に挑戦！

Karnaugh map による乗法標準形での簡単化

方法（加法標準形の双対の考えでいける。）

1. Karnaugh map を作る。
2. **“0”**をループで覆う。その際 don’t care は “0” を覆うために利用して良い。
3. 各ループが（AND 項ではなく）**OR 項**を表しているので、それらを求め、（OR ではなく）**AND**でつなげる。

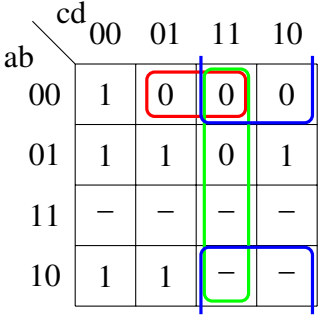
ab \ cd	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	−	−	−	−
10	1	1	−	−

$$Y =$$

Karnaugh map による乗法標準形での簡単化

方法（加法標準形の双対の考えでいける。）

1. Karnaugh map を作る。
2. **“0”**をループで覆う。その際 don’t care は “0” を覆うために利用して良い。
3. 各ループが（AND 項ではなく）**OR 項**を表しているので、それらを求め、（OR ではなく）**AND**でつなげる。



$$Y =$$

Karnaugh map による乗法標準形での簡単化

方法（加法標準形の双対の考えでいける。）

1. Karnaugh map を作る。
2. **“0”**をループで覆う。その際 don’t care は “0” を覆うために利用して良い。
3. 各ループが（AND 項ではなく）**OR 項**を表しているので、それらを求め、（OR ではなく）**AND**でつなげる。

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	−	−	−	−
10	1	1	−	−

$$Y = (\overline{c} + \overline{d}) \cdot$$

Karnaugh map による乗法標準形での簡単化

方法（加法標準形の双対の考えでいける。）

1. Karnaugh map を作る。
2. **“0”**をループで覆う。その際 don’t care は “0” を覆うために利用して良い。
3. 各ループが（AND 項ではなく）**OR 項**を表しているので、それらを求め、（OR ではなく）**AND**でつなげる。

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	−	−	−	−
10	1	1	−	−

$$Y = (\overline{c} + \overline{d}) \cdot (b + \overline{c}) \cdot$$

Karnaugh map による乗法標準形での簡単化

方法（加法標準形の双対の考えでいける。）

1. Karnaugh map を作る。
2. **“0”**をループで覆う。その際 don’t care は “0” を覆うために利用して良い。
3. 各ループが（AND 項ではなく）**OR 項**を表しているので、それらを求め、（OR ではなく）**AND**でつなげる。

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	−	−	−	−
10	1	1	−	−

$$Y = (\overline{c} + \overline{d}) \cdot (b + \overline{c}) \cdot (a + b + \overline{d})$$

Karnaugh map による乗法標準形での簡単化 (番外編)

こっちは無視してもいいです。alternative な番外編ということで。

否定に双対定理を使う方法

1. **0, 1 を入れ替えた** Karnaugh map を作る。つまり、$\overline{Y}$ を考える。
2. ふつうに簡単化して、$\overline{Y}$ を加法標準形で求める。
3. 双対定理を使って、Y を求める（と、乗法標準形になる）。

ab \ cd	00	01	11	10
00	0	1	1	1
01	0	0	1	0
11	−	−	−	−
10	0	0	−	−

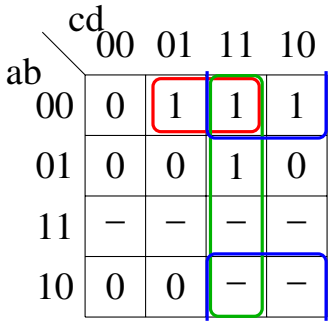
$$\overline{Y} =$$

Karnaugh map による乗法標準形での簡単化 (番外編)

こっちは無視してもいいです。alternative な番外編ということで。

否定に双対定理を使う方法

1. **0, 1 を入れ替えた** Karnaugh map を作る。つまり、$\overline{Y}$ を考える。
2. ふつうに簡単化して、$\overline{Y}$ を加法標準形で求める。
3. 双対定理を使って、Y を求める（と、乗法標準形になる）。



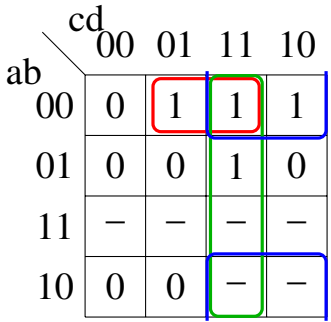
$$\overline{Y} =$$

Karnaugh map による乗法標準形での簡単化 (番外編)

こっちは無視してもいいです。alternative な番外編ということで。

否定に双対定理を使う方法

1. **0, 1 を入れ替えた** Karnaugh map を作る。つまり、$\overline{Y}$ を考える。
2. ふつうに簡単化して、$\overline{Y}$ を加法標準形で求める。
3. 双対定理を使って、Y を求める（と、乗法標準形になる）。



$$\overline{Y} = c \cdot d +$$

Karnaugh map による乗法標準形での簡単化 (番外編)

こっちは無視してもいいです。alternative な番外編ということで。

否定に双対定理を使う方法

1. **0, 1 を入れ替えた** Karnaugh map を作る。つまり、$\overline{Y}$ を考える。
2. ふつうに簡単化して、$\overline{Y}$ を加法標準形で求める。
3. 双対定理を使って、Y を求める（と、乗法標準形になる）。

ab \ cd	00	01	11	10
00	0	1	1	1
01	0	0	1	0
11	−	−	−	−
10	0	0	−	−

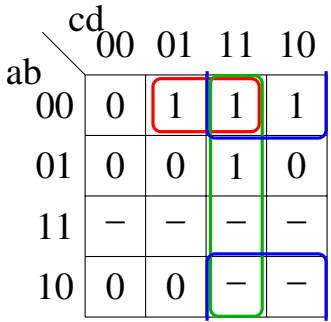
$$\overline{Y} = c \cdot d + \overline{b} \cdot c +$$

Karnaugh map による乗法標準形での簡単化 (番外編)

こっちは無視してもいいです。alternative な番外編ということで。

否定に双対定理を使う方法

1. **0, 1 を入れ替えた** Karnaugh map を作る。つまり、$\overline{Y}$ を考える。
2. ふつうに簡単化して、$\overline{Y}$ を加法標準形で求める。
3. 双対定理を使って、Y を求める（と、乗法標準形になる）。



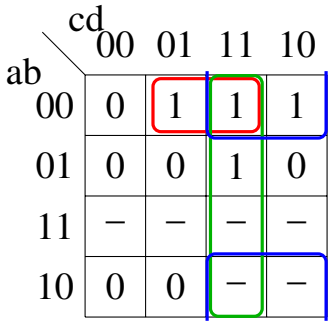
$$\overline{Y} = c \cdot d + \overline{b} \cdot c + \overline{a} \cdot \overline{b} \cdot d$$

Karnaugh map による乗法標準形での簡単化 (番外編)

こっちは無視してもいいです。alternative な番外編ということで。

否定に双対定理を使う方法

1. **0, 1 を入れ替えた** Karnaugh map を作る。つまり、$\overline{Y}$ を考える。
2. ふつうに簡単化して、$\overline{Y}$ を加法標準形で求める。
3. 双対定理を使って、Y を求める（と、乗法標準形になる）。



$$\overline{Y} = c \cdot d + \overline{b} \cdot c + \overline{a} \cdot \overline{b} \cdot d$$

双 ⬇ 対

$$Y = (\overline{c} + \overline{d}) \cdot (b + \overline{c}) \cdot (a + b + \overline{d})$$

組み合わせ回路

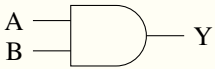
論理回路記号

名前	演算	(MIL) 記号
AND	$Y = A \cdot B$	
OR	$Y = A + B$	
NOT	$Y = \overline{A}$	

これらを用いて、論理式を回路として実現する。

組み合わせ回路

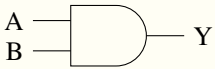
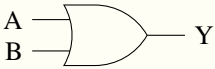
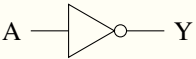
論理回路記号

名前	演算	(MIL) 記号
AND	$Y = A \cdot B$	
OR	$Y = A + B$	
NOT	$Y = \overline{A}$	

これらを用いて、論理式を回路として実現する。

組み合わせ回路

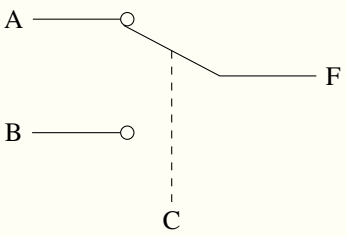
論理回路記号

名前	演算	(MIL) 記号
AND	$Y = A \cdot B$	
OR	$Y = A + B$	
NOT	$Y = \overline{A}$	

これらを用いて、論理式を回路として実現する。

例題その 1

2 to 1 データセレクタ



2 つの入力のうち 1 つを選択する回路を 2 to 1 データセレクタと言う。

$$F = \begin{cases} A & \text{if } C = 0 \\ B & \text{if } C = 1 \end{cases}$$

74157 datasheet

検索

問:

1. 2 to 1 データセレクタの論理設計をせよ ($F(A, B, C)$ を論理式で表わせ)。Karnaugh map を用いて簡単化すること。
2. 設計した 2 to 1 データセレクタの**回路図を描け**。

2 to 1 データセレクタの論理設計

1. Karnaugh map:

2. 論理式:

$$F = \\ = $$

2 to 1 データセレクタの論理設計

1. Karnaugh map:

C \ AB	00	01	11	10
0				
1				

2. 論理式:

$$F = \\ = $$

2 to 1 データセレクタの論理設計

1. Karnaugh map:

C \ AB	00	01	11	10
0	0	0	1	1
1	0	1	1	0

2. 論理式:

$$F = \\ = $$

2 to 1 データセレクタの論理設計

1. Karnaugh map:

C \ AB	00	01	11	10
0	0	0	1	1
1	0	1	1	0

2. 論理式:

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

2 to 1 データセレクタの論理設計

1. Karnaugh map:

C \ AB	00	01	11	10
0	0	0	1	1
1	0	1	1	0

2. 論理式:

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

これを回路図として書けば良い

(NOT-)AND-OR 構成

(NOT-)AND-OR 構成による 2-to-1 データセレクタの実現

F を _____ 標準形で表し (下記)、そのまま回路にしたもの。

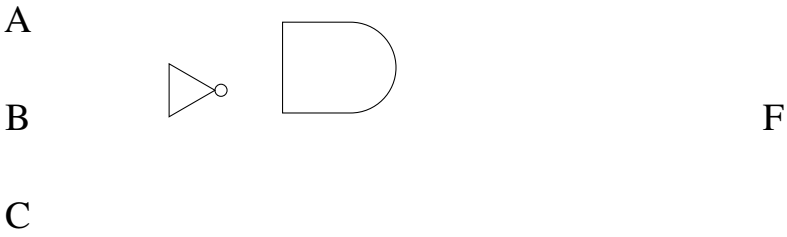
$$F = A \cdot \overline{C} + B \cdot C$$

(NOT-)AND-OR 構成

(NOT-)AND-OR 構成による 2-to-1 データセレクタの実現

F を **加法 (積和)** 標準形で表し (下記)、そのまま回路にしたもの。

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

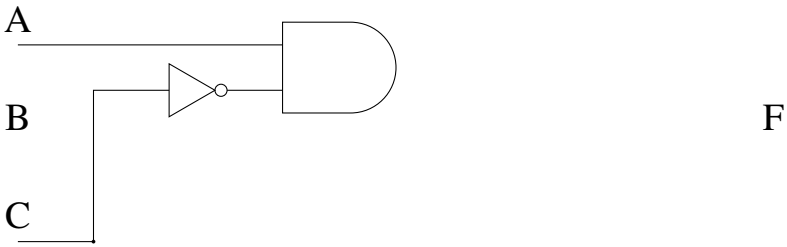


(NOT-)AND-OR 構成

(NOT-)AND-OR 構成による 2-to-1 データセレクタの実現

F を **加法 (積和)** 標準形で表し (下記)、そのまま回路にしたもの。

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

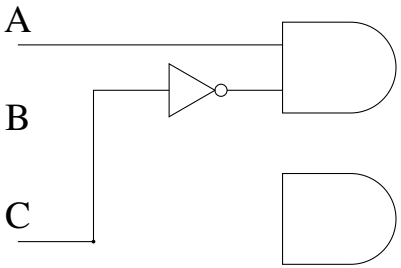


(NOT-)AND-OR 構成

(NOT-)AND-OR 構成による 2-to-1 データセレクタの実現

F を **加法 (積和)** 標準形で表し (下記)、そのまま回路にしたもの。

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

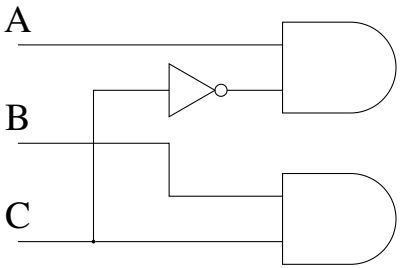


(NOT-)AND-OR 構成

(NOT-)AND-OR 構成による 2-to-1 データセレクタの実現

F を **加法 (積和)** 標準形で表し (下記)、そのまま回路にしたもの。

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

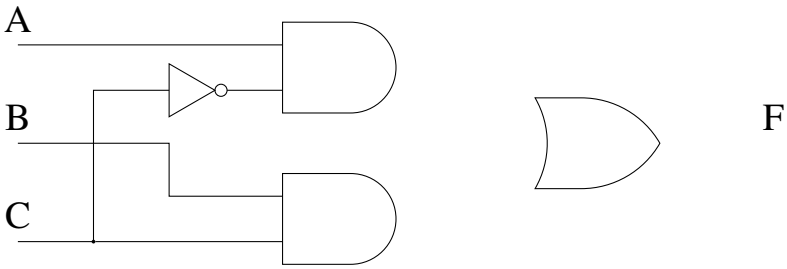


(NOT-)AND-OR 構成

(NOT-)AND-OR 構成による 2-to-1 データセレクタの実現

F を **加法 (積和)** 標準形で表し (下記)、そのまま回路にしたもの。

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

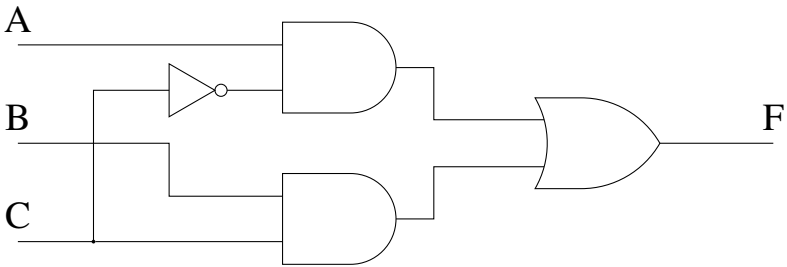


(NOT-)AND-OR 構成

(NOT-)AND-OR 構成による 2-to-1 データセレクタの実現

F を **加法 (積和)** 標準形で表し (下記)、そのまま回路にしたもの。

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



(NOT-)OR-AND 構成

(NOT-)OR-AND 構成による 2-to-1 データセレクタの実現

F を _____ 標準形で表し (下記)、そのまま回路にしたもの。

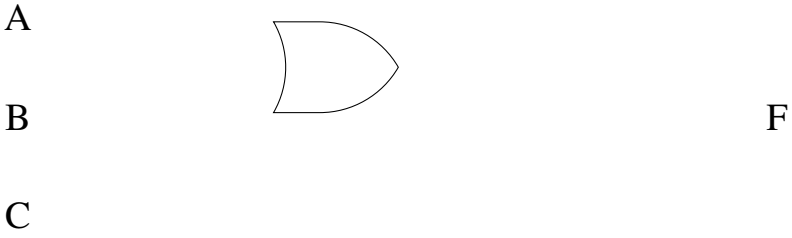
$$F = (A + C) \cdot (B + \overline{C})$$

(NOT-)OR-AND 構成

(NOT-)OR-AND 構成による 2-to-1 データセレクタの実現

F を **乗法 (和積)** 標準形で表し (下記)、そのまま回路にしたもの。

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

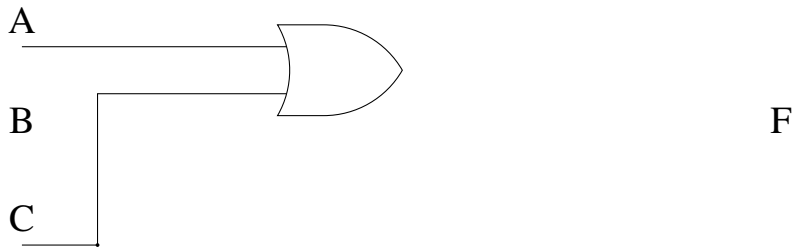


(NOT-)OR-AND 構成

(NOT-)OR-AND 構成による 2-to-1 データセレクタの実現

F を **乗法 (和積)** 標準形で表し (下記)、そのまま回路にしたもの。

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

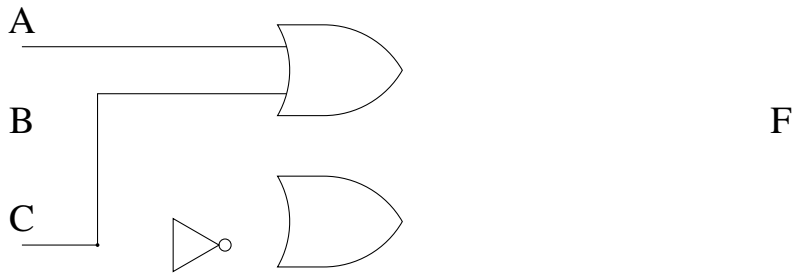


(NOT-)OR-AND 構成

(NOT-)OR-AND 構成による 2-to-1 データセレクタの実現

F を **乗法 (和積)** 標準形で表し (下記)、そのまま回路にしたもの。

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

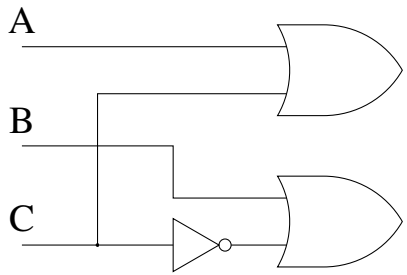


(NOT-)OR-AND 構成

(NOT-)OR-AND 構成による 2-to-1 データセレクタの実現

F を **乗法 (和積)** 標準形で表し (下記)、そのまま回路にしたもの。

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

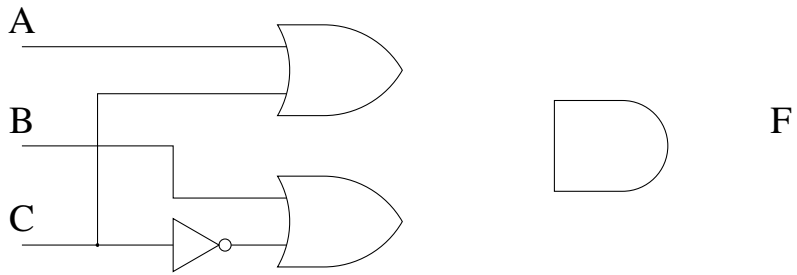


(NOT-)OR-AND 構成

(NOT-)OR-AND 構成による 2-to-1 データセレクタの実現

F を **乗法 (和積)** 標準形で表し (下記)、そのまま回路にしたもの。

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

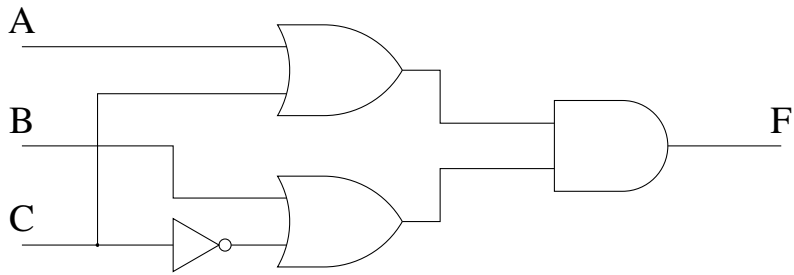


(NOT-)OR-AND 構成

(NOT-)OR-AND 構成による 2-to-1 データセレクタの実現

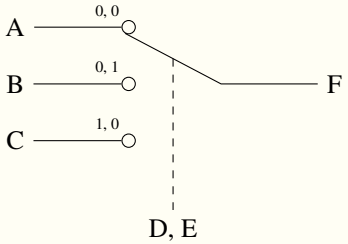
F を **乗法 (和積)** 標準形で表し (下記)、そのまま回路にしたもの。

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



例題その 2

3 to 1 データセレクタ



3 つの入力のうち 1 つを選択する回路を 3 to 1 データセレクタと言う。

$$F = \begin{cases} A & \text{if } (D, E) = (0, 0) \\ B & \text{if } (D, E) = (0, 1) \\ C & \text{if } (D, E) = (1, 0) \\ \text{don't care.} & \text{otherwise} \end{cases}$$

問:

1. 3 to 1 データセレクタを (NOT-)AND-OR 構成で論理設計し、回路図を描け。
2. 3 to 1 データセレクタを (NOT-)OR-AND 構成で論理設計し、回路図を描け。

正論理と負論理

- **真理値は “1” と “0”**。これを実現するための物理量は（例えば）電圧の “H”(高) と “L”(低)。
- ただし、“1” を “H” にしなければならない決まりはない。逆も頻繁に使われる[1]。この『**逆**』を　　という。
- 回路全体を負論理とすることはあまりなく、部分的に負論理を採用する。

	真理値	電圧
正論理 (positive logic)	1	H
	0	L
負論理 (negative logic)	1	
	0	

注意: 負論理とは 0, 1 を入れ替えること＿＿＿＿、『{0, 1} と {L, H} の＿＿＿＿』を入れ替えること。

[1]例えば、『電源入』の真理値は当然 “1” としたい、でも異常時に物理的な電圧が高くなる傾向がある回路の場合、$L \triangleq 1$ と定義した方がより安全そうな気がするでしょ?

正論理と負論理

- **真理値は “1” と “0”**。これを実現するための物理量は（例えば）電圧の “H”(高) と “L”(低)。
- ただし、“1” を “H” にしなければならない決まりはない。逆も頻繁に使われる[1]。この**『逆』を負論理という**。
- 回路全体を負論理とすることはあまりなく、部分的に負論理を採用する。

	真理値	電圧
正論理 (positive logic)	1	H
	0	L
負論理 (negative logic)	1	
	0	

注意: 負論理とは 0, 1 を入れ替えること＿＿＿＿＿、『{0, 1} と {L, H} の＿＿＿＿＿』を入れ替えること。

[1]例えば、『電源入』の真理値は当然 “1” としたい、でも異常時に物理的な電圧が高くなる傾向がある回路の場合、$L \triangleq 1$ と定義した方がより安全そうな気がするでしょ?

正論理と負論理

- **真理値は “1” と “0”**。これを実現するための物理量は（例えば）電圧の “H”(高) と “L”(低)。
- ただし、“1” を “H” にしなければならない決まりはない。逆も頻繁に使われる[1]。この**『逆』を負論理という**。
- 回路全体を負論理とすることはあまりなく、部分的に負論理を採用する。

	真理値	電圧
正論理 (positive logic)	1	H
	0	L
負論理 (negative logic)	1	L
	0	H

注意: 負論理とは 0, 1 を入れ替えること＿＿＿＿＿、『{0, 1} と {L, H} の＿＿＿＿＿』を入れ替えること。

[1]例えば、『電源入』の真理値は当然 “1” としたい、でも異常時に物理的な電圧が高くなる傾向がある回路の場合、$L \triangleq 1$ と定義した方がより安全そうな気がするでしょ?

正論理と負論理

- **真理値は “1” と “0”**。これを実現するための物理量は（例えば）電圧の “H”(高) と “L”(低)。
- ただし、“1” を “H” にしなければならない決まりはない。逆も頻繁に使われる[1]。この**『逆』を負論理という**。
- 回路全体を負論理とすることはあまりなく、部分的に負論理を採用する。

	真理値	電圧
正論理 (positive logic)	1	H
	0	L
負論理 (negative logic)	1	L
	0	H

注意: 負論理とは 0, 1 を入れ替えることではなくて、『{0, 1} と {L, H} の _____』を入れ替えること。

[1]例えば、『電源入』の真理値は当然 “1” としたい、でも異常時に物理的な電圧が高くなる傾向がある回路の場合、$L \triangleq 1$ と定義した方がより安全そうな気がするでしょ?

正論理と負論理

- **真理値は “1” と “0”**。これを実現するための物理量は（例えば）電圧の “H”(高) と “L”(低)。
- ただし、“1” を “H” にしなければならない決まりはない。逆も頻繁に使われる[1]。この**『逆』を負論理という**。
- 回路全体を負論理とすることはあまりなく、部分的に負論理を採用する。

	真理値	電圧
正論理 (positive logic)	1	H
	0	L
負論理 (negative logic)	1	L
	0	H

注意: 負論理とは 0, 1 を入れ替えることではなくて、『{0, 1} と {L, H} の対応づけ』を入れ替えること。

[1]例えば、『電源入』の真理値は当然 “1” としたい、でも異常時に物理的な電圧が高くなる傾向がある回路の場合、$L \triangleq 1$ と定義した方がより安全そうな気がするでしょ?

負論理を回路で実現する

注意点

- **論理式は正論理とか負論理とか関係ない**。あくまでも本来の論理で考える。
- 回路記号は (真理値ではなく) あくまでも電圧での動作を表す。  は、必ず “H” と “H” を入れたときのみ “H” が出る。

問: 負論理で $Y = A \cdot B$ を実現せよ。(A, B, Y すべて負論理で。)

真理値表は…

A	B	Y

負論理で考えると…

A	B	Y

これを実現する回路は…

負論理を回路で実現する

注意点

- **論理式は正論理とか負論理とか関係ない**。あくまでも本来の論理で考える。
- 回路記号は (真理値ではなく) あくまでも電圧での動作を表す。  は、必ず “H” と “H” を入れたときのみ “H” が出る。

問: 負論理で $Y = A \cdot B$ を実現せよ。(A, B, Y すべて負論理で。)

真理値表は…

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

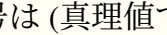
負論理で考えると…

A	B	Y

これを実現する回路は…

負論理を回路で実現する

注意点

- **論理式は正論理とか負論理とか関係ない**。あくまでも本来の論理で考える。
- 回路記号は (真理値ではなく) あくまでも電圧での動作を表す。  は、必ず “H” と “H” を入れたときのみ “H” が出る。

問: 負論理で $Y = A \cdot B$ を実現せよ。(A, B, Y すべて負論理で。)

真理値表は…

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

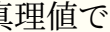
負論理で考えると…

A	B	Y
H	H	H
H	L	H
L	H	H
L	L	L

これを実現する回路は…

負論理を回路で実現する

注意点

- **論理式は正論理とか負論理とか関係ない**。あくまでも本来の論理で考える。
- 回路記号は (真理値ではなく) あくまでも電圧での動作を表す。  は、必ず “H” と “H” を入れたときのみ “H” が出る。

問: 負論理で $Y = A \cdot B$ を実現せよ。(A, B, Y すべて負論理で。)

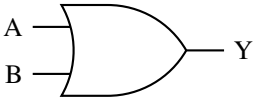
真理値表は…

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

負論理で考えると…

A	B	Y
H	H	H
H	L	H
L	H	H
L	L	L

これを実現する回路は…



負論理を回路で実現する

注意点

- **論理式は正論理とか負論理とか関係ない。**あくまでも本来の論理で考える。
- 回路記号は (真理値ではなく) あくまでも電圧での動作を表す。  は、必ず “H” と “H” を入れたときのみ “H” が出る。

問: 負論理で $Y = A \cdot B$ を実現せよ。(A, B, Y すべて負論理で。)

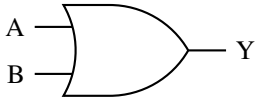
真理値表は…

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

負論理で考えると…

A	B	Y
H	H	H
H	L	H
L	H	H
L	L	L

これを実現する回路は…



なんと、負論理の AND は正論理の OR と同じ回路でした!

負論理を回路で実現する

注意点

- **論理式は正論理とか負論理とか関係ない。**あくまでも本来の論理で考える。
- 回路記号は (真理値ではなく) あくまでも電圧での動作を表す。  は、必ず “H” と “H” を入れたときのみ “H” が出る。

問: 負論理で $Y = A \cdot B$ を実現せよ。(A, B, Y すべて負論理で。)

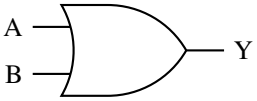
真理値表は…

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

負論理で考えると…

A	B	Y
H	H	H
H	L	H
L	H	H
L	L	L

これを実現する回路は…



なんと、負論理の AND は正論理の OR と同じ回路でした!

→ でもちょっとわかりにくい…。

わかりやすい (というか、本来の) 負論理の表現

○ = 負論理の信号

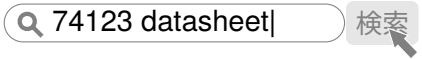
- その信号が負論理であることを表すために、入力や出力に ○ 印をつける。
- ○ 印は MIL 記号で負論理 (H と L の入れ替え) を表す。
- 回路図や CAD 等では信号名を $\overline{X}$, $/X$, $\#X$, $*X$ などと表すことが多い。

例:

- (入出力とも) 負論理の AND:

参考: 前ページでの表現 

- 入力は正論理、出力だけ負論理の AND:

- 負論理の使用例: 

わかりやすい (というか、本来の) 負論理の表現

○ = 負論理の信号

- その信号が負論理であることを表すために、入力や出力に ○ 印をつける。
- ○ 印は MIL 記号で負論理 (H と L の入れ替え) を表す。
- 回路図や CAD 等では信号名を $\overline{X}$, $/X$, $\#X$, $*X$ などと表すことが多い。

例:

- (入出力とも) 負論理の AND:

参考: 前ページでの表現

- 入力は正論理、出力だけ負論理の AND:

- 負論理の使用例: 74123 datasheet 検索

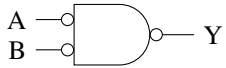
わかりやすい (というか、本来の) 負論理の表現

○ = 負論理の信号

- その信号が負論理であることを表すために、入力や出力に ○ 印をつける。
- ○ 印は MIL 記号で負論理 (H と L の入れ替え) を表す。
- 回路図や CAD 等では信号名を $\overline{X}$, $/X$, $\#X$, $*X$ などと表すことが多い。

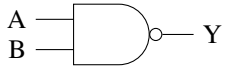
例:

- (入出力とも) 負論理の AND:



参考: 前ページでの表現

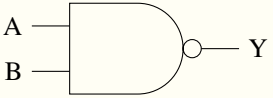
- 入力は正論理、出力だけ負論理の AND:



- 負論理の使用例: 74123 datasheet 検索

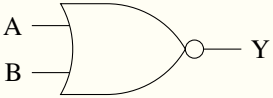
NAND と NOR と EXOR

NAND



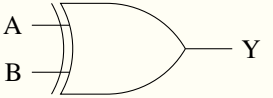
$Y =$

NOR



$Y =$

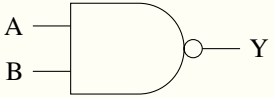
EXOR （排他的論理和）



$Y =$

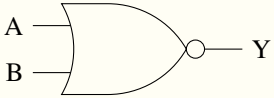
NAND と NOR と EXOR

NAND



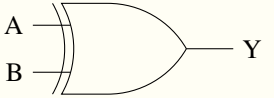
$$Y = \overline{A \cdot B}$$

NOR



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

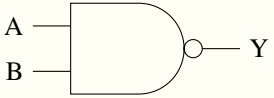
EXOR （排他的論理和）問. 真理値表を作成せよ。



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

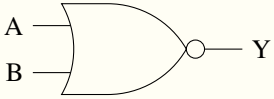
NAND と NOR と EXOR

NAND



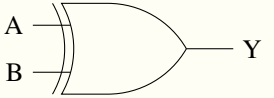
$$Y = \overline{A \cdot B}$$

NOR



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

EXOR （排他的論理和）問. 真理値表を作成せよ。



$$Y = \left(A \cdot \overline{B}\right) + \left(\overline{A} \cdot B\right) = A \oplus B$$

ブール代数の完全系

完全系とは

どのような論理式でも作れる論理演算の組。例: {NOT, AND, OR}
要するに、ある論理演算の組だけを組み合わせて NOT, AND, OR を作ることができたら、それは完全系。

Q: 以下はいずれも完全系である。証明せよ。

1. {AND, NOT}
2. {OR, NOT}
3. {NAND}
4. {NOR}
5. {EXOR, AND, 1}

ブール代数の完全系

完全系とは

どのような論理式でも作れる論理演算の組。例: {NOT, AND, OR}
要するに、ある論理演算の組だけを組み合わせて NOT, AND, OR を作ることができたら、それは完全系。

Q: 以下はいずれも完全系である。証明せよ。

1. {AND, NOT}

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

2. {OR, NOT}
3. {NAND}
4. {NOR}
5. {EXOR, AND, 1}

NAND と NOR

NAND と NOR は

なぜならば…

1. 作り易い。
2. それだけで　　系である。

NAND と NOR

NAND と NOR は **エラい！**

なぜならば…

1. 作り易い。
2. それだけで　　系である。

NAND と NOR

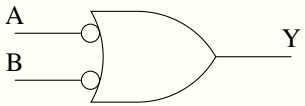
NAND と NOR は **エラい！**

なぜならば…

1. 作り易い。
2. それだけで完全系である。

(NOT-)NAND-NAND 回路

負論理入力の論理和は…



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

=

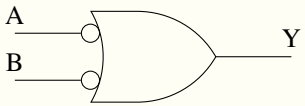
双対定理を使うと…

す・な・わ・ち

1. _____ で論理式が得られると…
2. _____ 回路で簡単に実現でき、
3. さらに _____ **回路** に変換できる!

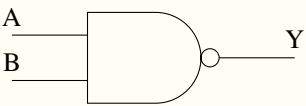
(NOT-)NAND-NAND 回路

負論理入力の論理和は…



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

=



$\overline{A \cdot B}$

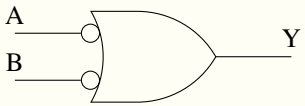
双対定理を使うと…

す・な・わ・ち

1. _____ で論理式が得られると…
2. _____ 回路で簡単に実現でき、
3. さらに _____ **回路** に変換できる!

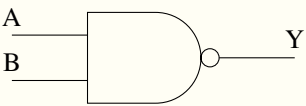
(NOT-)NAND-NAND 回路

負論理入力の論理和は…



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

=



$\overline{A \cdot B}$

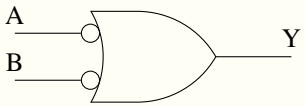
双対定理を使うと…

す・な・わ・ち

1. 加法 (積和) 標準形 で論理式が得られると…
2. _____ 回路で簡単に実現でき、
3. さらに _____ **回路** に変換できる!

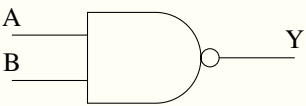
(NOT-)NAND-NAND 回路

負論理入力の論理和は…



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

=



$\overline{A \cdot B}$

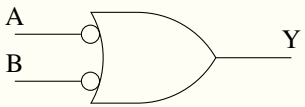
双対定理を使うと…

す・な・わ・ち

1. 加法 (積和) 標準形 で論理式が得られると…
2. (NOT-)AND-OR 回路で簡単に実現でき、
3. さらに **回路** に変換できる!

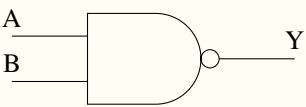
(NOT-)NAND-NAND 回路

負論理入力の論理和は…



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

=



$\overline{A \cdot B}$

双対定理を使うと…

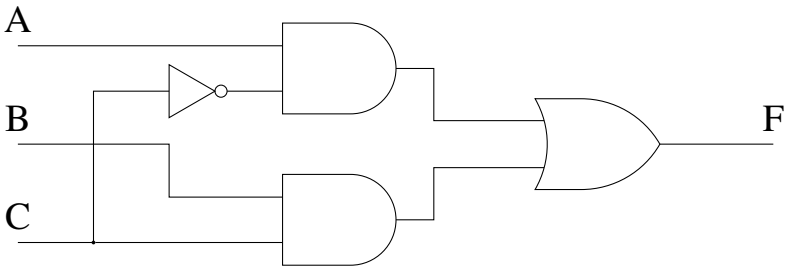
す・な・わ・ち

1. 加法 (積和) 標準形 で論理式が得られると…
2. (NOT-)AND-OR 回路で簡単に実現でき、
3. さらに**(NOT-)NAND-NAND回路** に変換できる!

(NOT-)NAND-NAND 回路の例

Q: 次式の 2 to 1 データセレクタを**NAND だけ**で作れ。

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

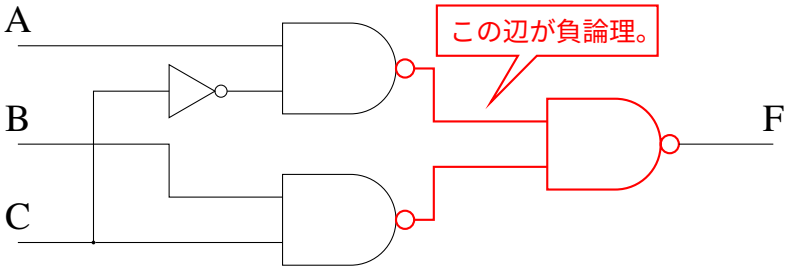


し・か・も、

(NOT-)NAND-NAND 回路の例

Q: 次式の 2 to 1 データセレクタを**NAND だけ**で作れ。

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

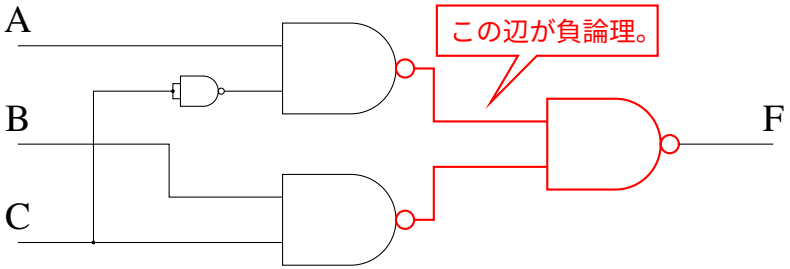


し・か・も、7400 一個でできる！

(NOT-)NAND-NAND 回路の例

Q: 次式の 2 to 1 データセレクタを**NAND だけ**で作れ。

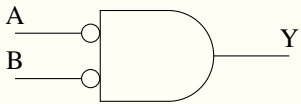
$$F = A \cdot \overline{C} + B \cdot C$$



し・か・も、7400 一個でできる！

(NOT-)NOR-NOR 回路

負論理入力の論理和は…



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

=

双対定理を使うと…

す・な・わ・ち

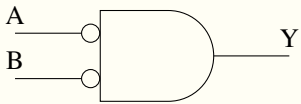
で論理式が得られると…

回路で簡単に実現でき、

3. さらに **回路**にできる!

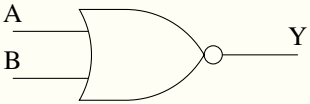
(NOT-)NOR-NOR 回路

負論理入力の論理和は…



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

=



$\overline{A + B}$

双対定理を使うと…

す・な・わ・ち

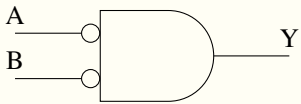
で論理式が得られると…

回路で簡単に実現でき、

3. さらに **回路**にできる!

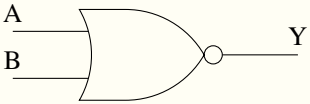
(NOT-)NOR-NOR 回路

負論理入力の論理和は…



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

=



$\overline{A + B}$

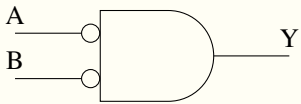
双対定理を使うと…

す・な・わ・ち

1. 乗法標準形で論理式が得られると…
   回路で簡単に実現でき、
3. さらに **回路**にできる!

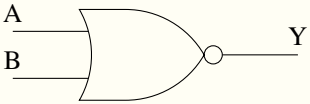
(NOT-)NOR-NOR 回路

負論理入力の論理和は…



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

=



$\overline{A + B}$

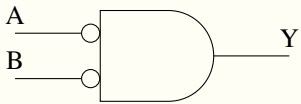
双対定理を使うと…

す・な・わ・ち

1. 乗法標準形で論理式が得られると…
2. (NOT-)OR-AND 回路で簡単に実現でき、
3. さらに **回路**にできる!

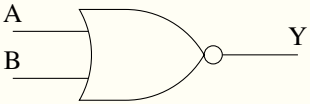
(NOT-)NOR-NOR 回路

負論理入力の論理和は…



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

=



$\overline{A + B}$

双対定理を使うと…

す・な・わ・ち

1. 乗法標準形で論理式が得られると…
2. (NOT-)OR-AND 回路で簡単に実現でき、
3. さらに**(NOT-)NOR-NOR 回路**にできる!

(NOT-)NOR-NOR 回路の練習

Q: 乗法標準形で簡単化した 2 to 1 データセレクタを
NOR だけで実現せよ。

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

し・か・も、

(NOT-)NOR-NOR 回路の練習

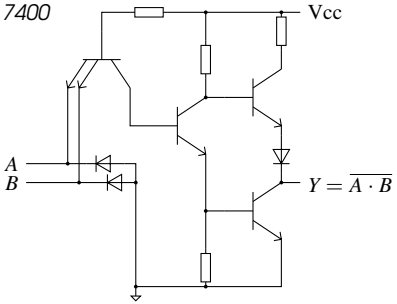
Q: 乗法標準形で簡単化した 2 to 1 データセレクタを
NOR だけで実現せよ。

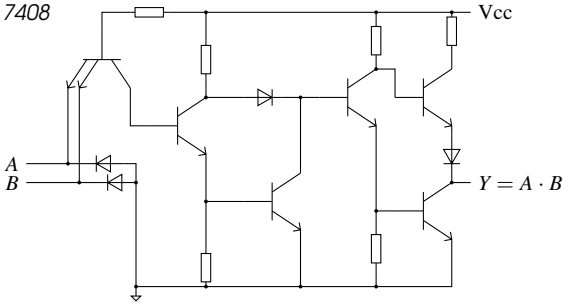
$$F = (A + C) \cdot (B + \overline{C})$$

し・か・も、7402 一個でできる！

コラム NAND と NOR はやっぱりエラい件

NAND, NOR はどちらもブール代数に置ける完全系を構成できるので数学的に (AND, OR) よりエラい、という話は数ページ前に書きました。さてここで論理 IC の最大手の規格、74 シリーズを見てみましょう。シリーズ**筆頭の 7400, 7401, 7402, 7403 はそれぞれ NAND, NAND, NOR, NOR** なのです。それに対して**AND は 7408、OR は 7432**です。これには数学的な有用さの他に、**モノとしてのシンプルさ**という背景もあるような気がします。そういう意味でも NAND, NOR はエラいのです。





出席確認レポート課題 (次の月曜の 12 時締め切り)

2 ページ前の問題を解け。ただし、復習の意味も込めて Karnaugh map による F の式の導出も示すこと。

提出は下記 URL の Google Forms。歪んでいない、開いた時に横倒しになっていない、コントラストが読むに耐えうる PDF で提出すること。

https://forms.gle/9ruwtfJg5LQgQNpU7