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

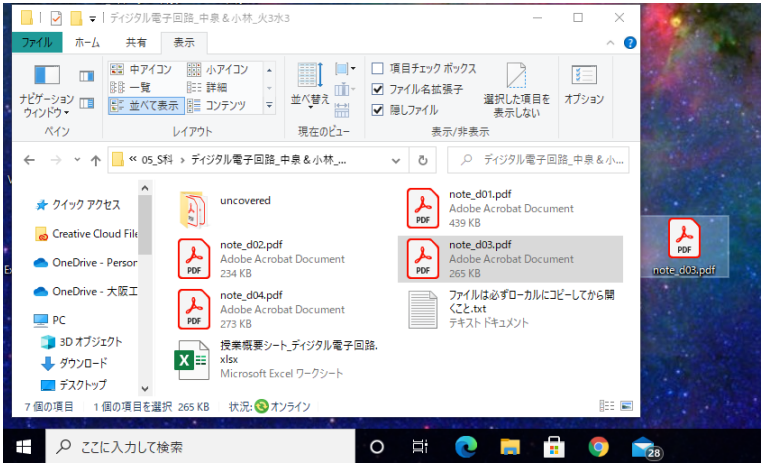
  https://drive.google.com/drive/folders/1bT-z3ICQyMYC_5JvlL29UZYqbOhVG492
- 出席確認レポートは**Google Forms** で提出。(毎回同一 URL)

  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

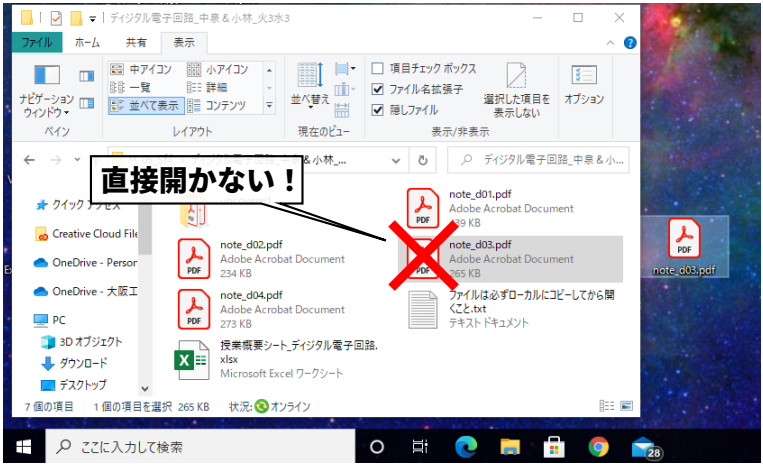
COMMON フォルダの注意事項 (全授業共通)

根源的に悪いのは Windows の仕様なのですが、ご協力ください。



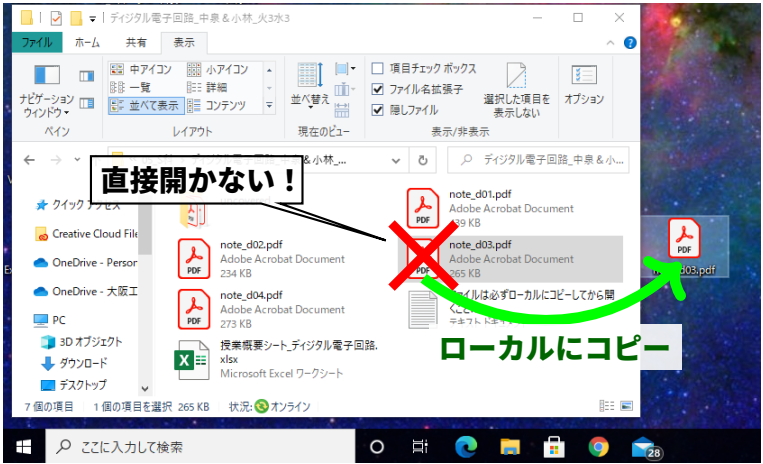
COMMON フォルダの注意事項 (全授業共通)

根源的に悪いのは Windows の仕様なのですが、ご協力ください。



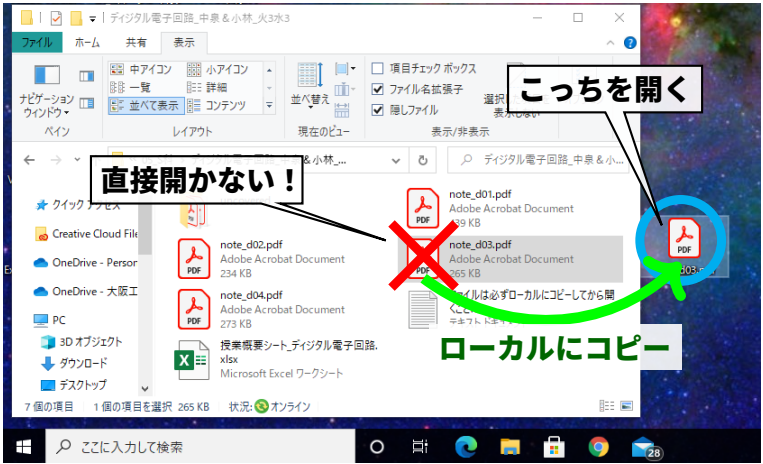
COMMON フォルダの注意事項 (全授業共通)

根源的に悪いのは Windows の仕様なのですが、ご協力ください。



COMMON フォルダの注意事項 (全授業共通)

根源的に悪いのは Windows の仕様なのですが、ご協力ください。



R/S 科ディジタル電子回路
Digital Electronics
『レジスタとバス』

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

11 of 14

a LaTeX + Beamer slideshow

レジスタ

レジスタ (← resistor じゃないよ)

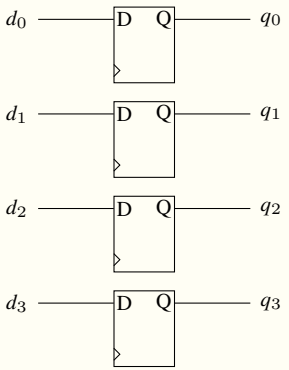
抵抗器

レジスタ

- n ビットの情報を記憶する小さなメモリ。
- CPU がプログラムを実行する際の、比較的更新頻度の高い情報を一時的に記憶するのに使う。
- n 個の D-FF を につなげると n ビットのレジスタが作れる。

cf.) 6502 の “レジスタ”

P	プロセッサステータス
A	アキュムレータ
X	インデックス X
Y	インデックス Y
PC	プログラムカウンタ
S	スタックポインタ



※ この例ではクロックが「書き込み信号」。

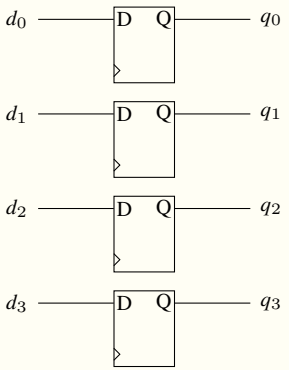
レジスタ (register ← 抵抗器 resistor じゃないよ)

レジスタ

- n ビットの情報を記憶する小さなメモリ。
- CPU がプログラムを実行する際の、比較的更新頻度の高い情報を一時的に記憶するのに使う。
- n 個の D-FF を　　につなげると n ビットのレジスタが作れる。

cf.) 6502 の “レジスタ”

P	プロセッサステータス
A	アキュムレータ
X	インデックス X
Y	インデックス Y
PC	プログラムカウンタ
S	スタックポインタ



※ この例ではクロックが「書き込み信号」。

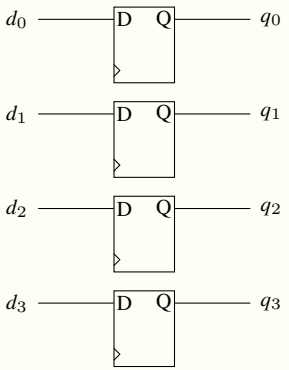
レジスタ (register ← 抵抗器 resistor じゃないよ)

レジスタ

- n ビットの情報を記憶する小さなメモリ。
- CPU がプログラムを実行する際の、比較的更新頻度の高い情報を一時的に記憶するのに使う。
- n 個の D-FF を **パラレル** につなげると n ビットのレジスタが作れる。

cf.) 6502 の “レジスタ”

P	プロセッサステータス
A	アキュムレータ
X	インデックス X
Y	インデックス Y
PC	プログラムカウンタ
S	スタックポインタ



※ この例ではクロックが「書き込み信号」。

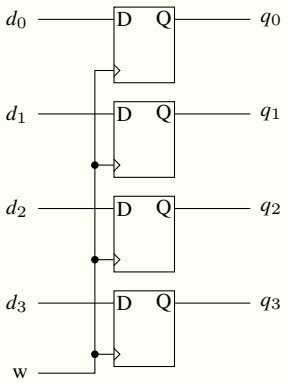
レジスタ (register ← 抵抗器 resistor じゃないよ)

レジスタ

- n ビットの情報を記憶する小さなメモリ。
- CPU がプログラムを実行する際の、比較的更新頻度の高い情報を一時的に記憶するのに使う。
- n 個の D-FF を **パラレル** につなげると n ビットのレジスタが作れる。

cf.) 6502 の “レジスタ”

P	プロセッサステータス
A	アキュムレータ
X	インデックス X
Y	インデックス Y
PC	プログラムカウンタ
S	スタックポインタ



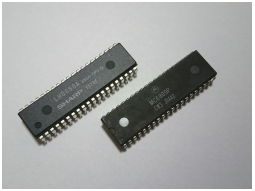
※ この例ではクロックが「書き込み信号」。

コラム: CPU 界のレジェンド MOS 6502

写真の CPU は小林の私物です、ちなみに。

MOS 6502 は米 MOS Technology が 1975 年に送り出した 8-bit CPU です。そのアーキテクチャははっきり言えば米 Motorola 製 **6800 のパクリ**と言えなくもないのですが、性能は良かったようで、なんと、あの **Apple II に採用**されました。Apple II を設計した天才魔法使い Woz こと Stephen Wozniak の完璧な機械語コードでその性能をいかんなく発揮した 6502 は大変な人気を博し (たのが理由かどうかは知りませんが、ともかく)、多くの 8-bit PC に採用されました。あの micro:bit の魂のご先祖様である BBC Micro や、Atari のパソコンは 6502 を採用しています。そして、我が国では、あの**任天堂ファミリーコンピュータ**が 6502 ベースのカスタム CPU を搭載しています。ちなみに当時の日本ではどちらかというと、米 Zilog 製の Z80 の方が人気だったようですが、こちらは基本設計を行ったのは何と日本人 (嶋正利先生) です。ついでにパクられた Motorola はその後 6800 を改良し、直交性の高い命令セットを持つ美しいアーキテクチャで **究極の 8-bit CPU**として誉れ高い MC6809 を生み出しました。

Apple II 互換機に入っていた 6502。本家 MOS Technology のオリジナルではなく、米 Synertek 社製の SY6502。

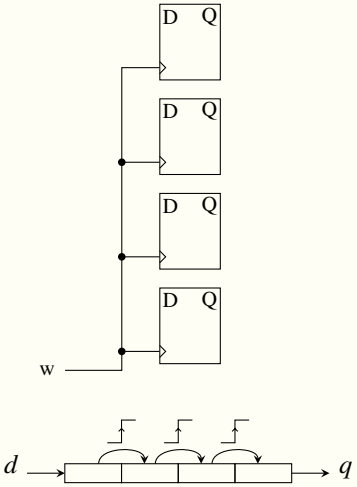


Z80(シャープ製セカンドソース) と MC6809

シフトレジスタ

シフトレジスタ（SISO (　　) タイプ）

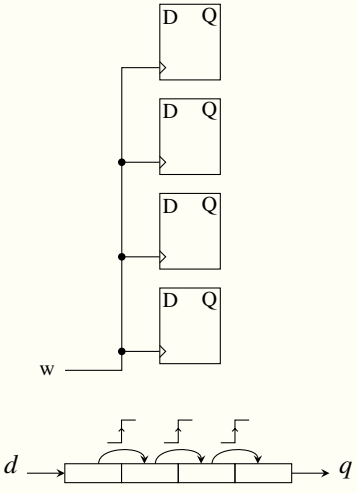
- n ビットの情報を記憶するメモリ。
- 入力は MSB（もしくは LSB）から 1 ビットずつ行い、クロック毎にシフトレジスタ内で 1 ビットずつ「ずれて（シフトして）」行く。
- 出力からは LSB（もしくは MSB）から「あふれる」ものが 1 ビットずつ出てくる。
- n 個の D-FF を　　につなげると n ビットの SISO レジスタが作れる。
- 発展形として、入力や出力を parallel にした、　　，　　，　　がある。マイコン内部の UART などで使われている。



シフトレジスタ

シフトレジスタ（SISO (Serial In Serial Out) タイプ）

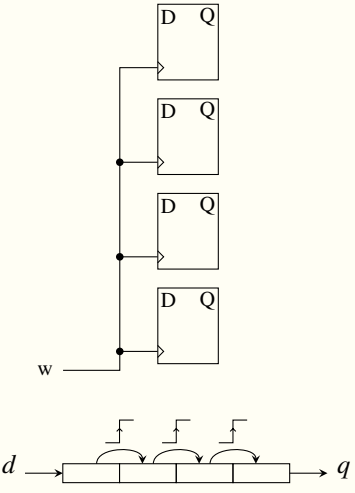
- n ビットの情報を記憶するメモリ。
- 入力は MSB（もしくは LSB）から 1 ビットずつ行い、クロック毎にシフトレジスタ内で 1 ビットずつ「ずれて（シフトして）」行く。
- 出力からは LSB（もしくは MSB）から「あふれる」ものが 1 ビットずつ出てくる。
- n 個の D-FF を　　につなげると n ビットの SISO レジスタが作れる。
- 発展形として、入力や出力を parallel にした、　　，　　，　　がある。マイコン内部の UART などで使われている。



シフトレジスタ

シフトレジスタ（SISO (Serial In Serial Out) タイプ）

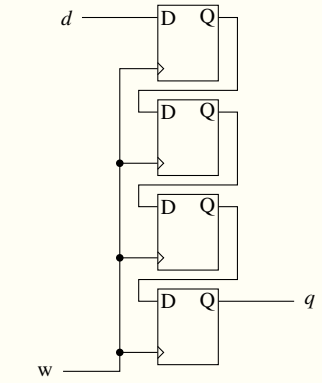
- n ビットの情報を記憶するメモリ。
- 入力は MSB（もしくは LSB）から 1 ビットずつ行い、クロック毎にシフトレジスタ内で 1 ビットずつ「ずれて（シフトして）」行く。
- 出力からは LSB（もしくは MSB）から「あふれる」ものが 1 ビットずつ出てくる。
- n 個の D-FF を **シリアル** につなげると n ビットの SISO レジスタが作れる。
- 発展形として、入力や出力を parallel にした、　　，　　，　　がある。マイコン内部の UART などで使われている。



シフトレジスタ

シフトレジスタ（SISO (Serial In Serial Out) タイプ）

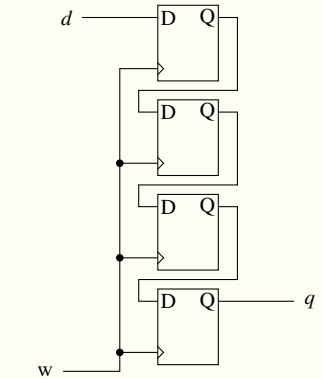
- n ビットの情報を記憶するメモリ。
- 入力は MSB（もしくは LSB）から 1 ビットずつ行い、クロック毎にシフトレジスタ内で 1 ビットずつ「ずれて（シフトして）」行く。
- 出力からは LSB（もしくは MSB）から「あふれる」ものが 1 ビットずつ出てくる。
- n 個の D-FF を **シリアル** につなげると n ビットの SISO レジスタが作れる。
- 発展形として、入力や出力を parallel にした、　　，　　，　　がある。マイコン内部の UART などで使われている。



シフトレジスタ

シフトレジスタ（SISO (Serial In Serial Out) タイプ）

- n ビットの情報を記憶するメモリ。
- 入力は MSB（もしくは LSB）から 1 ビットずつ行い、クロック毎にシフトレジスタ内で 1 ビットずつ「ずれて（シフトして）」行く。
- 出力からは LSB（もしくは MSB）から「あふれる」ものが 1 ビットずつ出てくる。
- n 個の D-FF を **シリアル** につなげると n ビットの SISO レジスタが作れる。
- 発展形として、入力や出力を parallel にした、PISO, SIPO, PIPO がある。マイコン内部の UART などで使われている。



CPU 内部でのシフトレジスタ

シフト演算とシフトジスタ

```
1    int a = 10;
2    a = a << 3;
```

こんなとき、PIPO のシフトレジスタが使える。

1. 10（$= (00001010)_2$）をシフトレジスタに入れる。
2. シフトレジスタにクロックを入れる。
3.
4.
5. 結果を取り出す。

CPU 内部でのシフトレジスタ

シフト演算とシフトジスタ

```
1    int a = 10;
2    a = a << 3;
```

こんなとき、PIPO のシフトレジスタが使える。

1. 10（$= (00001010)_2$）をシフトレジスタに入れる。
2. シフトレジスタにクロックを入れる。
3. 〃
4.
5. 結果を取り出す。

CPU 内部でのシフトレジスタ

シフト演算とシフトジスタ

```
1    int a = 10;
2    a = a << 3;
```

こんなとき、PIPO のシフトレジスタが使える。

1. 10（$= (00001010)_2$）をシフトレジスタに入れる。
2. シフトレジスタにクロックを入れる。
3. 〃
4. 〃
5. 結果を取り出す。

CPU 内部でのシフトレジスタ

シフト演算とシフトジスタ

```
1    int a = 10;
2    a = a << 3;
```

こんなとき、PIPO のシフトレジスタが使える。

1. 10（$= (00001010)_2$）をシフトレジスタに入れる。
2. シフトレジスタにクロックを入れる。
3. 〃
4. 〃
5. 結果を取り出す。

遅い!!

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) =$ 　　。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) =$ 　　。
- アルゴリズム等では…
  $\mathcal{O}($ 　　$) \rightarrow$ めちゃすごい, $\mathcal{O}($ 　$) \rightarrow$ とてもすごい,
  $\mathcal{O}($ 　　$) \rightarrow$ まずまずすごい, $\mathcal{O}($ 　$) \rightarrow$ ふつう以下, $\mathcal{O}($ 　$) \rightarrow$ だめ過ぎ,

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) =$ 　　。
- アルゴリズム等では…
  $\mathcal{O}($ 　　$) \rightarrow$ めちゃすごい, $\mathcal{O}($ 　$) \rightarrow$ とてもすごい,
  $\mathcal{O}($ 　　$) \rightarrow$ まずまずすごい, $\mathcal{O}($ 　$) \rightarrow$ ふつう以下, $\mathcal{O}($ 　$) \rightarrow$ だめ過ぎ,

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) = \mathcal{O}(n^2)$。
- アルゴリズム等では…
  $\mathcal{O}(\quad) \rightarrow$ めちゃすごい, $\mathcal{O}(\ ) \rightarrow$ とてもすごい,
  $\mathcal{O}(\quad) \rightarrow$ まずまずすごい, $\mathcal{O}(\ \ ) \rightarrow$ ふつう以下, $\mathcal{O}(\ \ ) \rightarrow$ だめ過ぎ,

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) = \mathcal{O}(n^2)$。
- アルゴリズム等では…
  $\mathcal{O}(\log n)$ → めちゃすごい, $\mathcal{O}(\ )$ → とてもすごい,
  $\mathcal{O}(\qquad)$ → まずまずすごい, $\mathcal{O}(\quad)$ → ふつう以下, $\mathcal{O}(\quad)$ → だめ過ぎ,

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) = \mathcal{O}(n^2)$。
- アルゴリズム等では…
  $\mathcal{O}(\log n) \rightarrow$ めちゃすごい, $\mathcal{O}(n) \rightarrow$ とてもすごい,
  $\mathcal{O}(\quad) \rightarrow$ まずまずすごい, $\mathcal{O}(\quad) \rightarrow$ ふつう以下, $\mathcal{O}(\quad) \rightarrow$ だめ過ぎ,

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) = \mathcal{O}(n^2)$。
- アルゴリズム等では…
  $\mathcal{O}(\log n) \rightarrow$ めちゃすごい, $\mathcal{O}(n) \rightarrow$ とてもすごい,
  $\mathcal{O}(n \log n) \rightarrow$ まずまずすごい, $\mathcal{O}(\quad) \rightarrow$ ふつう以下, $\mathcal{O}(\quad) \rightarrow$ だめ過ぎ,

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) = \mathcal{O}(n^2)$。
- アルゴリズム等では…
  $\mathcal{O}(\log n) \to$ めちゃすごい, $\mathcal{O}(n) \to$ とてもすごい,
  $\mathcal{O}(n \log n) \to$ まずまずすごい, $\mathcal{O}(n^2) \to$ ふつう以下, $\mathcal{O}(\quad) \to$ だめ過ぎ,

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) = \mathcal{O}(n^2)$。
- アルゴリズム等では…
  $\mathcal{O}(\log n) \rightarrow$ めちゃすごい, $\mathcal{O}(n) \rightarrow$ とてもすごい,
  $\mathcal{O}(n \log n) \rightarrow$ まずまずすごい, $\mathcal{O}(n^2) \rightarrow$ ふつう以下, $\mathcal{O}(2^n) \rightarrow$ だめ過ぎ,

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) = \mathcal{O}(n^2)$。
- アルゴリズム等では…
  $\mathcal{O}(\log n) \rightarrow$ めちゃすごい, $\mathcal{O}(n) \rightarrow$ とてもすごい,
  $\mathcal{O}(n \log n) \rightarrow$ まずまずすごい, $\mathcal{O}(n^2) \rightarrow$ ふつう以下, $\mathcal{O}(2^n) \rightarrow$ だめ過ぎ,
  $\mathcal{O}(1) \rightarrow$

※ あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。

オーダーの概念

Landau 記号: $\mathcal{O}()$

注目対象の数 n に対し、計算量や大きさなど（の最悪値）がどの程度の勢いで増加するかを表す記号。

- 定数係数は無視。
  $f_1(n) = n^2$ **も** $f_2(n) = 3n^2$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_2) = \mathcal{O}(n^2)$。
- n が大きいときに無視できる項も無視。
  $f_1(n) = n^2$ **も** $f_3(n) = n^2 + 3n$ **もオーダーは同じで** $\mathcal{O}(f_1) = \mathcal{O}(f_3) = \mathcal{O}(n^2)$。
- アルゴリズム等では…
  $\mathcal{O}(\log n) \rightarrow$ めちゃすごい, $\mathcal{O}(n) \rightarrow$ とてもすごい,
  $\mathcal{O}(n \log n) \rightarrow$ まずまずすごい, $\mathcal{O}(n^2) \rightarrow$ ふつう以下, $\mathcal{O}(2^n) \rightarrow$ だめ過ぎ,
  $\mathcal{O}(1) \rightarrow$ **神**

※ **あくまでも個人の感想であり、アルゴリズム等の優劣を保証するものであはありません。**

PIPO を繰り返し使うシフタの速さ

問: PIPO を繰り返し使う方式の n ビットシフタの速度のオーダはいくらか?
ヒント:

- 1 ビット左右にシフトするのにかかる時間は _____。
- 高々（最悪でも）_ ビットシフトすれば十分。

答:

問: 同様に複雑さはいくらか?

PIPO を繰り返し使うシフタの速さ

問: PIPO を繰り返し使う方式の n ビットシフタの速度のオーダはいくらか?
ヒント:

- 1 ビット左右にシフトするのにかかる時間は 一定。
- 高々（最悪でも）_ ビットシフトすれば十分。

答:

問: 同様に複雑さはいくらか?

PIPO を繰り返し使うシフタの速さ

問: PIPO を繰り返し使う方式の n ビットシフタの速度のオーダはいくらか?
ヒント:

- 1 ビット左右にシフトするのにかかる時間は 一定。
- 高々（最悪でも）n ビットシフトすれば十分。

答:

問: 同様に複雑さはいくらか?

PIPO を繰り返し使うシフタの速さ

問: PIPO を繰り返し使う方式の n ビットシフタの速度のオーダはいくらか?
ヒント:

- 1 ビット左右にシフトするのにかかる時間は一定。
- 高々（最悪でも）n ビットシフトすれば十分。

答: $\mathcal{O}(n)$

問: 同様に複雑さはいくらか?

PIPO を繰り返し使うシフタの速さ

問: PIPO を繰り返し使う方式の n ビットシフタの速度のオーダはいくらか?
ヒント:

- 1 ビット左右にシフトするのにかかる時間は一定。
- 高々（最悪でも）n ビットシフトすれば十分。

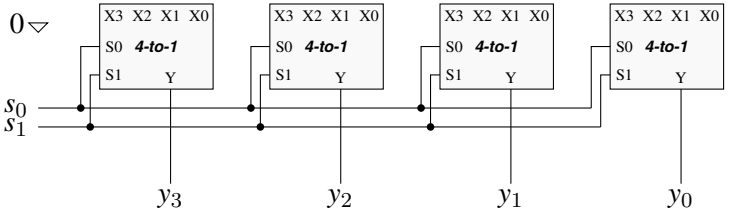
答: $\mathcal{O}(n)$

問: 同様に複雑さはいくらか?

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

で一発解決か?

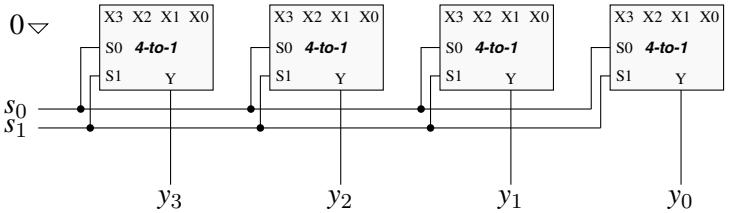


4 ビット高速シフタの構成例

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?

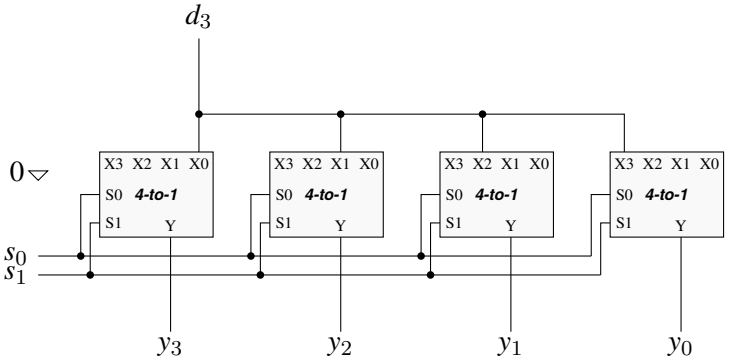


4 ビット高速シフタの構成例

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?

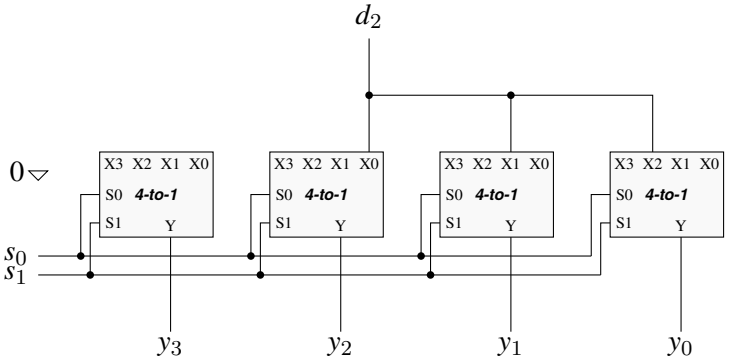


4 ビット高速シフタの構成例

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?

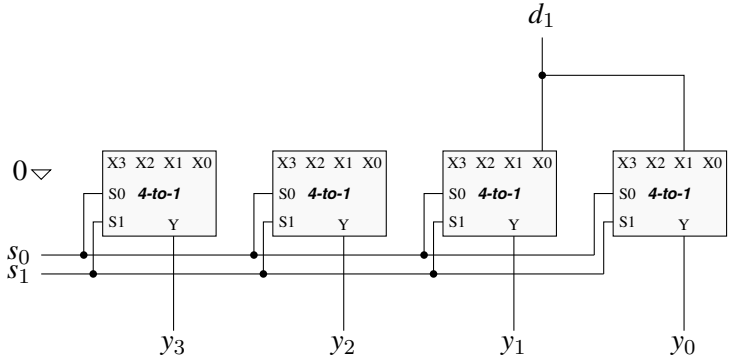


4 ビット高速シフタの構成例

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?

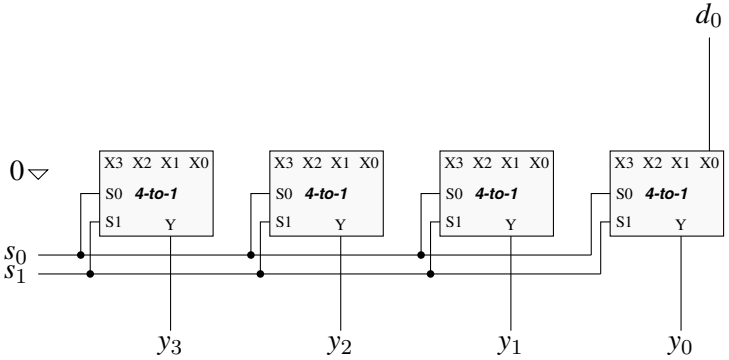


4 ビット高速シフタの構成例

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?

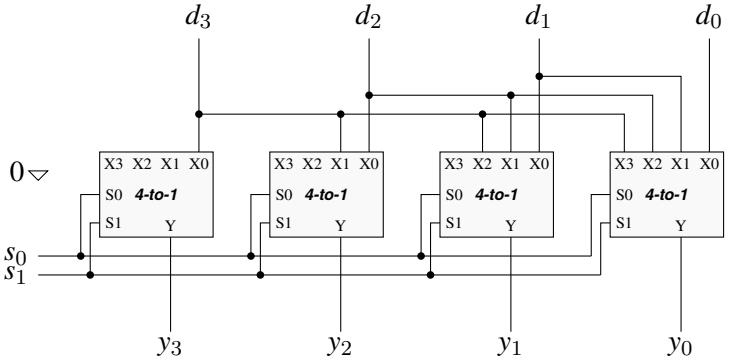


4 ビット高速シフタの構成例

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?

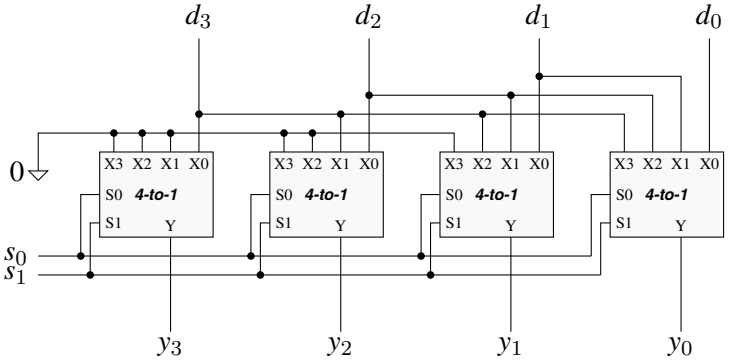


4 ビット高速シフタの構成例

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?

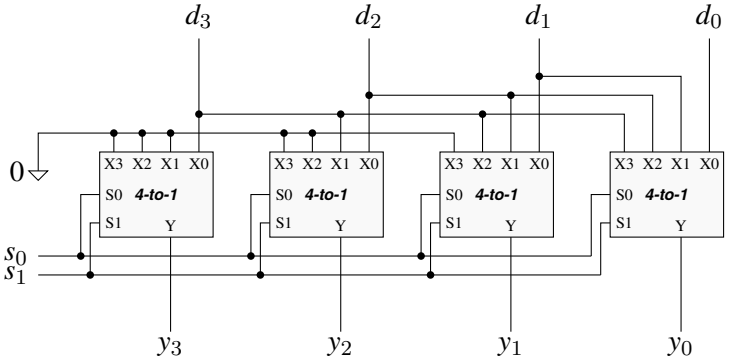


4 ビット高速シフタの構成例

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?



4 ビット高速シフタの構成例

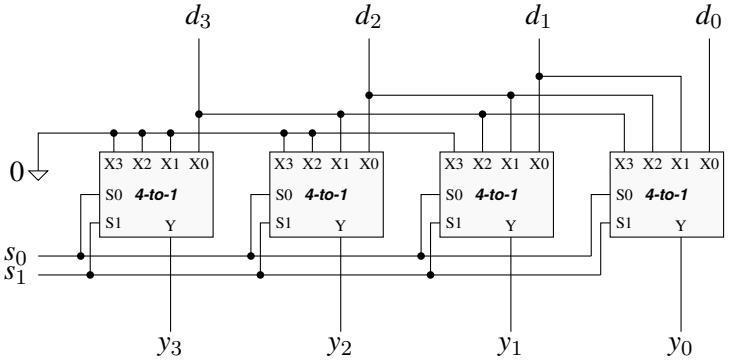
○ 速い

× 複雑

高速シフト案

『シフト演算が遅い…』そんなお悩みは、

n-to-1 データセレクタで一発解決か?



4 ビット高速シフタの構成例

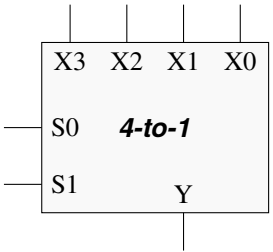
○ 速い

× 複雑

定量的に！

データセレクタ 1 個分の回路の複雑さ（規模）のオーダ考

単純化して、**真理値表の行数が回路の規模を表す**と考えることにする。



4-to-1 データセレクタの場合

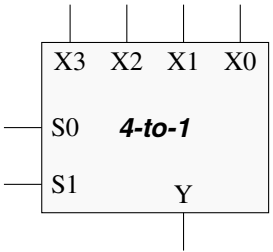
- X0〜X3(以下『**X 入力**』) は_ 個。
- セレクト入力 (以下『**S 入力**』) は_ 個。
- 真理値表の行数は__。

n-to-1 データセレクタに一般化

- n-to-1 データセレクタの **X 入力**は_ 個。
- n-to-1 データセレクタの **S 入力**は_____ 個。
- 真理値表の行数は_____。

データセレクタ 1 個分の回路の複雑さ（規模）のオーダ考

単純化して、**真理値表の行数が回路の規模を表す**と考えることにする。



4-to-1 データセレクタの場合

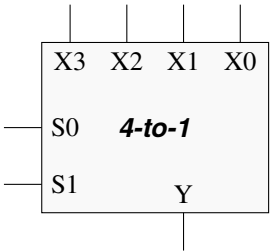
- X0〜X3(以下『**X 入力**』) は4 個。
- セレクト入力 (以下『**S 入力**』) は_ 個。
- 真理値表の行数は__。

n-to-1 データセレクタに一般化

- n-to-1 データセレクタの **X 入力**は_ 個。
- n-to-1 データセレクタの **S 入力**は_____ 個。
- 真理値表の行数は_____。

データセレクタ 1 個分の回路の複雑さ（規模）のオーダ考

単純化して、**真理値表の行数が回路の規模を表す**と考えることにする。



4-to-1 データセレクタの場合

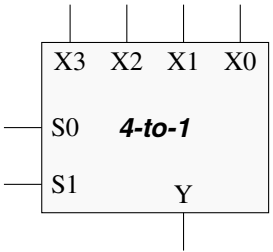
- X0〜X3(以下『**X 入力**』) は4 個。
- セレクト入力 (以下『**S 入力**』) は2 個。
- 真理値表の行数は__。

n-to-1 データセレクタに一般化

- n-to-1 データセレクタの **X 入力**は_ 個。
- n-to-1 データセレクタの **S 入力**は_____ 個。
- 真理値表の行数は_____。

データセレクタ 1 個分の回路の複雑さ（規模）のオーダ考

単純化して、**真理値表の行数が回路の規模を表す**と考えることにする。



4-to-1 データセレクタの場合

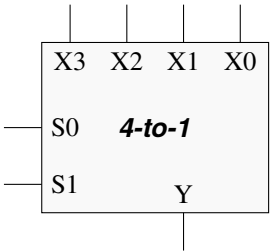
- X0〜X3(以下『**X 入力**』) は4 個。
- セレクト入力 (以下『**S 入力**』) は2 個。
- 真理値表の行数は2^6。

n-to-1 データセレクタに一般化

- n-to-1 データセレクタの **X 入力**は_ 個。
- n-to-1 データセレクタの **S 入力**は_____ 個。
- 真理値表の行数は_____。

データセレクタ 1 個分の回路の複雑さ（規模）のオーダ考

単純化して、**真理値表の行数が回路の規模を表す**と考えることにする。



4-to-1 データセレクタの場合

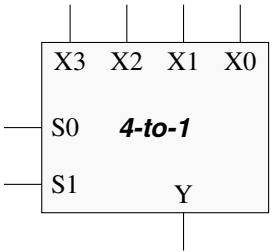
- X0〜X3(以下『**X 入力**』) は4 個。
- セレクト入力 (以下『**S 入力**』) は2 個。
- 真理値表の行数は2^6。

n-to-1 データセレクタに一般化

- n-to-1 データセレクタの **X 入力**はn 個。
- n-to-1 データセレクタの **S 入力**は＿＿＿ 個。
- 真理値表の行数は＿＿＿＿＿＿。

データセレクタ 1 個分の回路の複雑さ（規模）のオーダ考

単純化して、**真理値表の行数が回路の規模を表す**と考えることにする。



4-to-1 データセレクタの場合

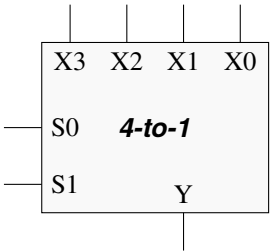
- X0〜X3(以下『**X 入力**』) は4 個。
- セレクト入力 (以下『**S 入力**』) は2 個。
- 真理値表の行数は2^6。

n-to-1 データセレクタに一般化

- n-to-1 データセレクタの **X 入力**はn 個。
- n-to-1 データセレクタの **S 入力**は$\log_2 n$ 個。
- 真理値表の行数は＿＿＿＿＿＿。

データセレクタ 1 個分の回路の複雑さ（規模）のオーダ考

単純化して、**真理値表の行数が回路の規模を表す**と考えることにする。



4-to-1 データセレクタの場合

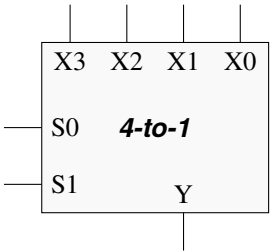
- X0〜X3(以下『**X 入力**』) は4 個。
- セレクト入力 (以下『**S 入力**』) は2 個。
- 真理値表の行数は2^6。

n-to-1 データセレクタに一般化

- n-to-1 データセレクタの **X 入力**はn 個。
- n-to-1 データセレクタの **S 入力**は$\log_2 n$ 個。
- 真理値表の行数は$2^{n+\log_2 n} =$ ＿＿＿。

データセレクタ 1 個分の回路の複雑さ（規模）のオーダ考

単純化して、**真理値表の行数が回路の規模を表す**と考えることにする。



4-to-1 データセレクタの場合

- X0〜X3(以下『**X 入力**』) は4 個。
- セレクト入力 (以下『**S 入力**』) は2 個。
- 真理値表の行数は2^6。

n-to-1 データセレクタに一般化

- n-to-1 データセレクタの **X 入力**はn 個。
- n-to-1 データセレクタの **S 入力**は$\log_2 n$ 個。
- 真理値表の行数は$2^{n+\log_2 n} = n2^n$。

データセレクタを使ったシフタの速さ・複雑さ

問: n-to-1 データセレクタを n 個使う方式の n ビットシフタの速度のオーダはいくらか?

答:

問: 同様に複雑さはいくらか?

答:

データセレクタを使ったシフタの速さ・複雑さ

問: n-to-1 データセレクタを n 個使う方式の n ビットシフタの速度のオーダはいくらか?

答: $\mathcal{O}(1)$

問: 同様に複雑さはいくらか?

答:

データセレクタを使ったシフタの速さ・複雑さ

問: n-to-1 データセレクタを n 個使う方式の n ビットシフタの速度のオーダはいくらか?

答: $\mathcal{O}(1)$

問: 同様に複雑さはいくらか?

答: $\mathcal{O}(n^2 2^n)$

データセレクタを使ったシフタの速さ・複雑さ

問: n-to-1 データセレクタを n 個使う方式の n ビットシフタの速度のオーダはいくらか?

答: $\mathcal{O}(1)$

問: 同様に複雑さはいくらか?

答: $\mathcal{O}(n^2 2^n)$

ほど良い落としどころが欲しい。

データセレクタを使ったシフタの速さ・複雑さ

問: n-to-1 データセレクタを n 個使う方式の n ビットシフタの速度のオーダはいくらか?

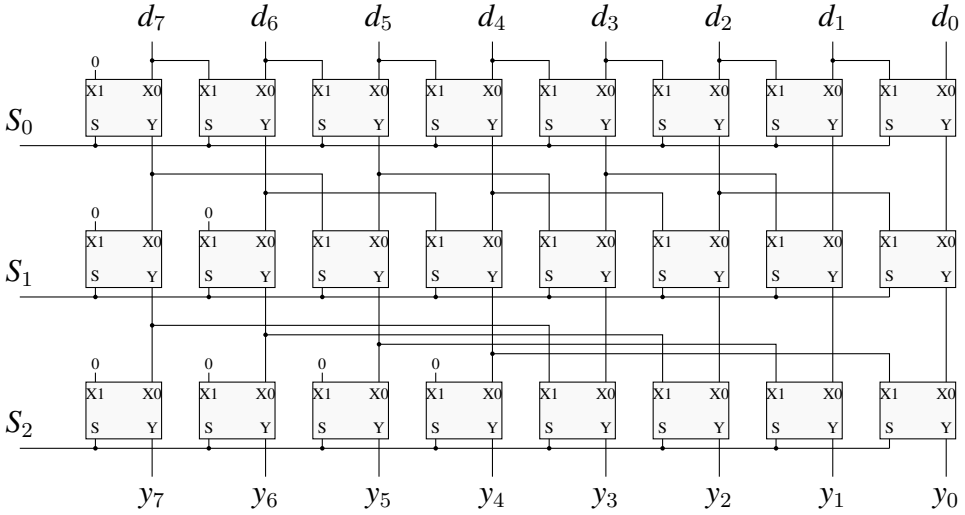
答: $\mathcal{O}(1)$

問: 同様に複雑さはいくらか?

答: $\mathcal{O}(n^2 2^n)$

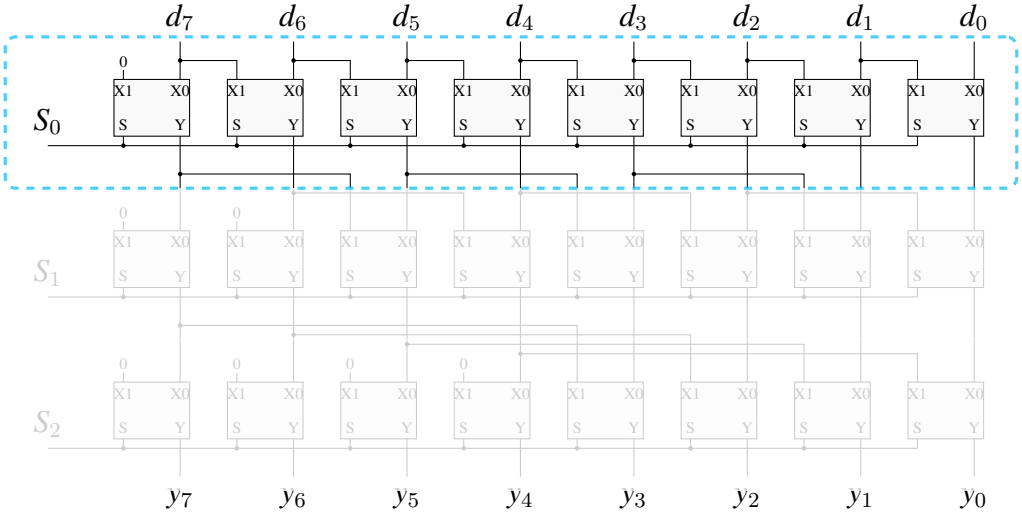
ほど良い落としどころが欲しい。 → **バレルシフタ**

barrel shifter



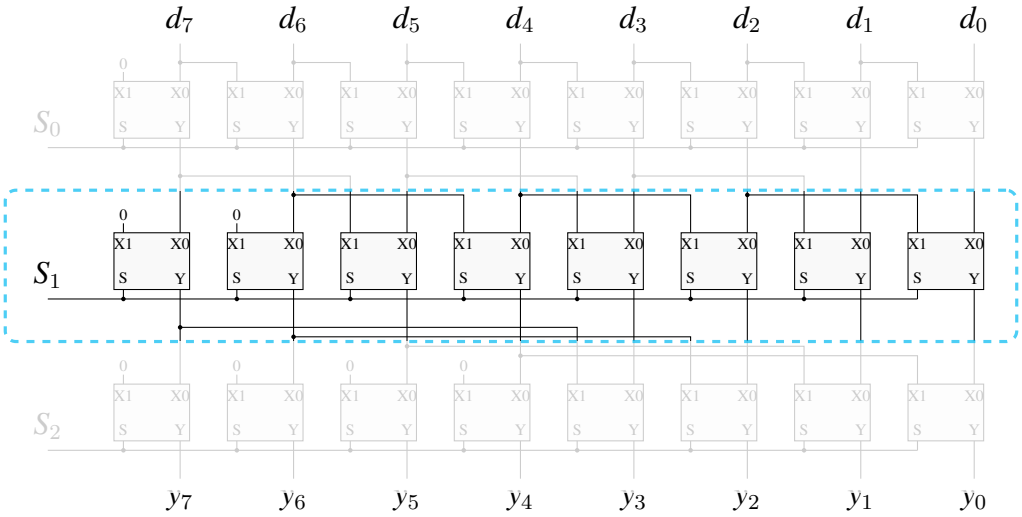
右シフト量 $n = 4S_2 + 2S_1 + S_0$, 速度: $\mathcal{O}($ $)$, 複雑さ: $\mathcal{O}($ $)$

barrel shifter



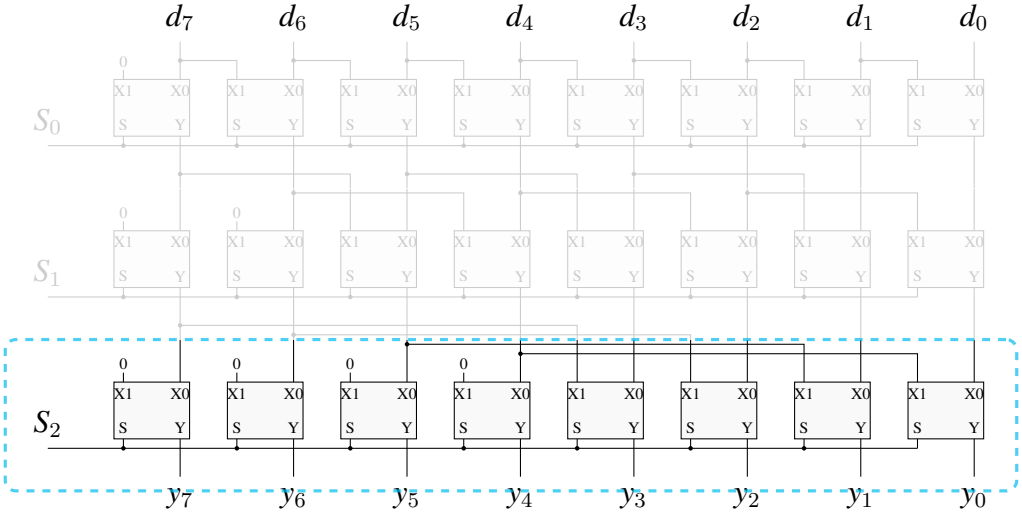
右シフト量 $n = 4S_2 + 2S_1 + S_0$, 速度: $\mathcal{O}($ $)$, 複雑さ: $\mathcal{O}($ $)$

barrel shifter



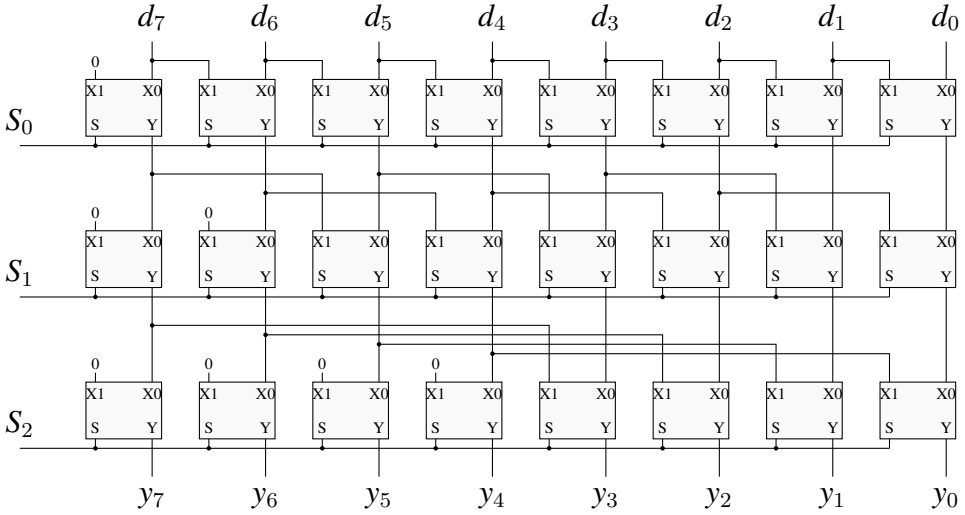
右シフト量 $n = 4S_2 + 2S_1 + S_0$, 速度: $\mathcal{O}($ $)$, 複雑さ: $\mathcal{O}($ $)$

barrel shifter



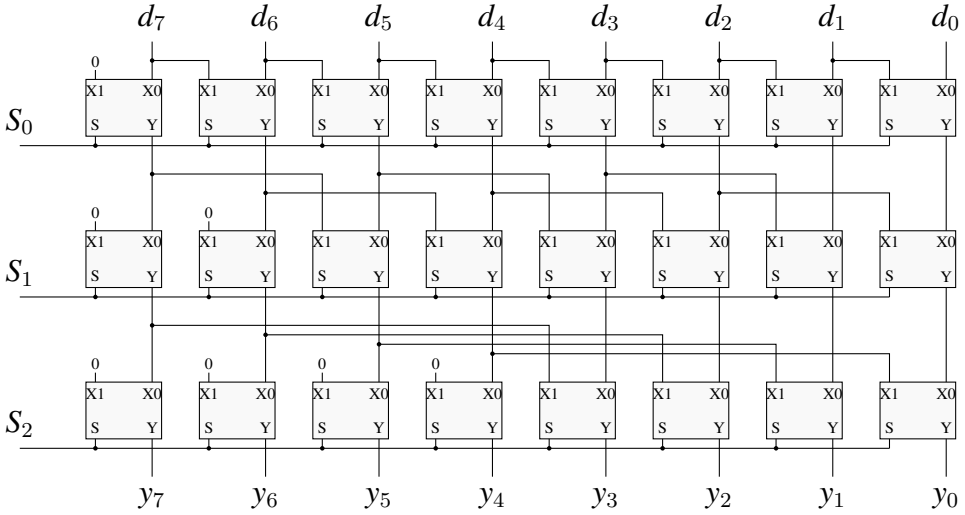
右シフト量 $n = 4S_2 + 2S_1 + S_0$, 速度: $\mathcal{O}($ $)$, 複雑さ: $\mathcal{O}($ $)$

barrel shifter



右シフト量 $n = 4S_2 + 2S_1 + S_0$, 速度: $\mathcal{O}(\log n)$, 複雑さ: $\mathcal{O}($ $)$

barrel shifter

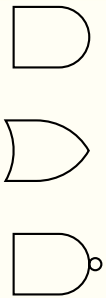


右シフト量 $n = 4S_2 + 2S_1 + S_0$, 速度: $\mathcal{O}(\log n)$, 複雑さ: $\mathcal{O}(n \log n)$

論理的でない話

入出力の接続〜やっていいことと悪いこと〜

論理的におかしくなければ (基本的には) 、いい。

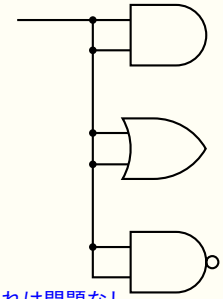


- 一つの節点に出力は一つだけ。
- 一つの節点に入力はいくつあっても良い。
  現実: _____ を考える必要がある。

- fan-in …論理ゲートの入力数
- fan-out …一本の出力に繋げることのできる（後段の）入力数

入出力の接続〜やっていいことと悪いこと〜

論理的におかしくなければ (基本的には) 、いい。



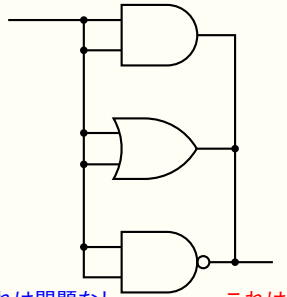
これは問題なし

- 一つの節点に出力は一つだけ。
- 一つの節点に入力はいくつあっても良い。
  現実: _____ を考える必要がある。

- fan-in …論理ゲートの入力数
- fan-out …一本の出力に繋げることのできる（後段の）入力数

入出力の接続〜やっていいことと悪いこと〜

論理的におかしくなければ (基本的には) 、いい。

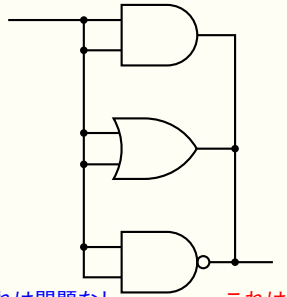


- 一つの節点に出力は一つだけ。
- 一つの節点に入力はいくつあっても良い。
  現実: _____ を考える必要がある。

- fan-in …論理ゲートの入力数
- fan-out …一本の出力に繋げることのできる（後段の）入力数

入出力の接続〜やっていいことと悪いこと〜

論理的におかしくなければ (基本的には) 、いい。



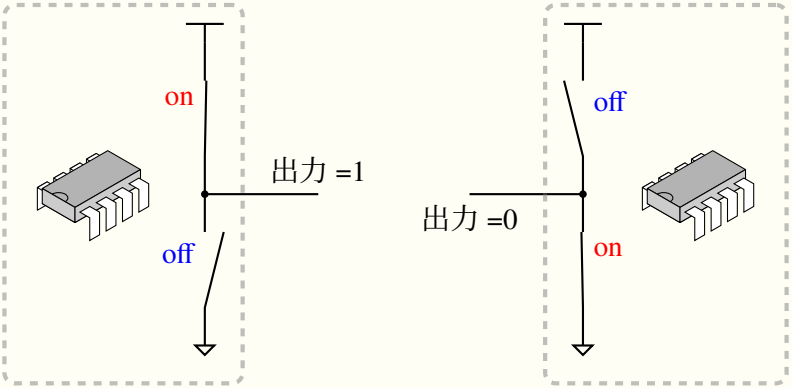
- 一つの節点に出力は一つだけ。
- 一つの節点に入力はいくつあっても良い。
  現実: ファンアウト を考える必要がある。

- fan-in …論理ゲートの入力数
- fan-out …一本の出力に繋げることのできる（後段の）入力数

トーテムポール出力

トーテムポール出力による “H” と “L” の出力

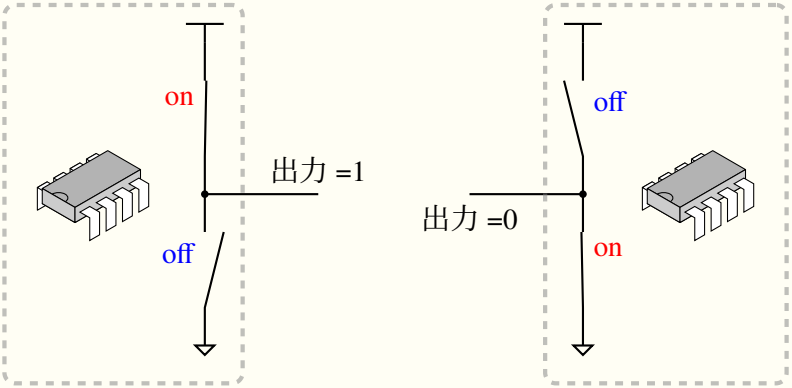
IC の出力段 (トーテムポール出力) のイメージ。実際はこんなスイッチではなくトランジスタを使っているが、基本的な理解としてはこの描像で ok。



トーテムポール出力

トーテムポール出力による “H” と “L” の出力

IC の出力段 (トーテムポール出力) のイメージ。実際はこんなスイッチではなくトランジスタを使っているが、基本的な理解としてはこの描像で ok。

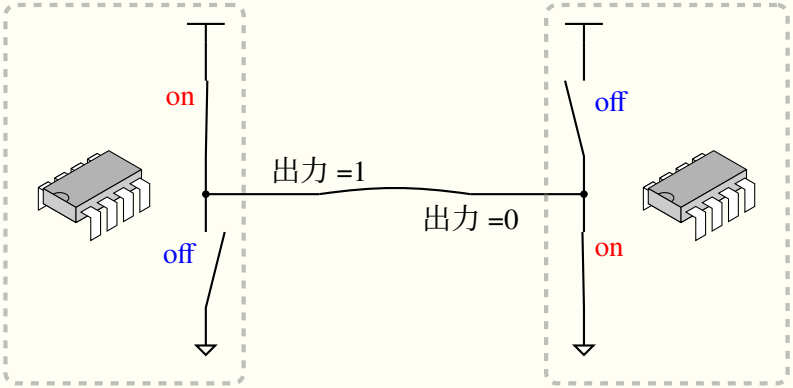


- “1” と “0” の出力を直結すると

トーテムポール出力

トーテムポール出力による “H” と “L” の出力

IC の出力段 (トーテムポール出力) のイメージ。実際はこんなスイッチではなくトランジスタを使っているが、基本的な理解としてはこの描像で ok。

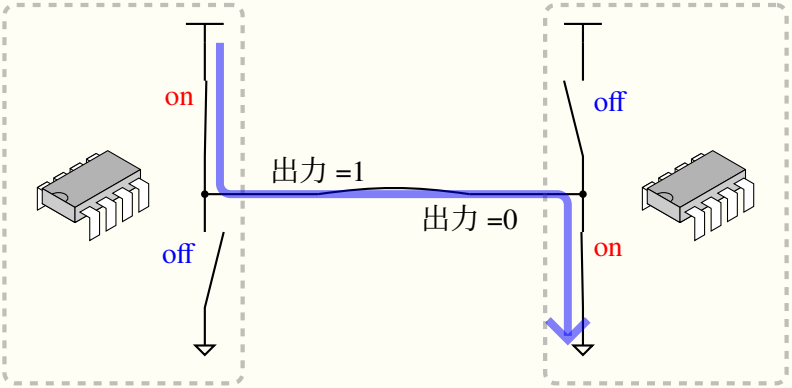


- “1” と “0” の出力を直結すると

トーテムポール出力

トーテムポール出力による “H” と “L” の出力

IC の出力段 (トーテムポール出力) のイメージ。実際はこんなスイッチではなくトランジスタを使っているが、基本的な理解としてはこの描像で ok。

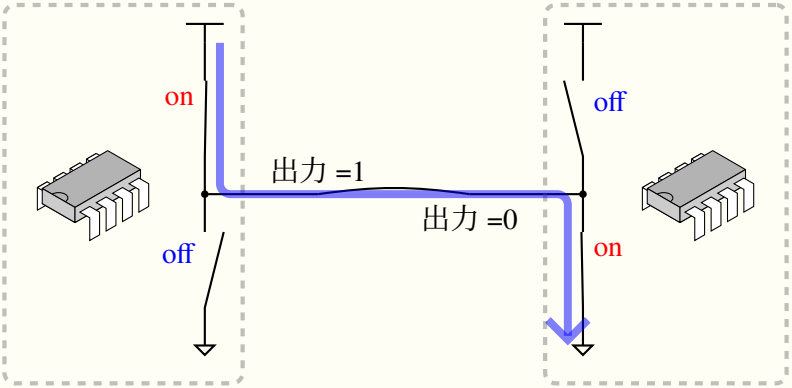


- “1” と “0” の出力を直結すると
- **大電流**が流れて、

トーテムポール出力

トーテムポール出力による “H” と “L” の出力

IC の出力段 (トーテムポール出力) のイメージ。実際はこんなスイッチではなくトランジスタを使っているが、基本的な理解としてはこの描像で ok。

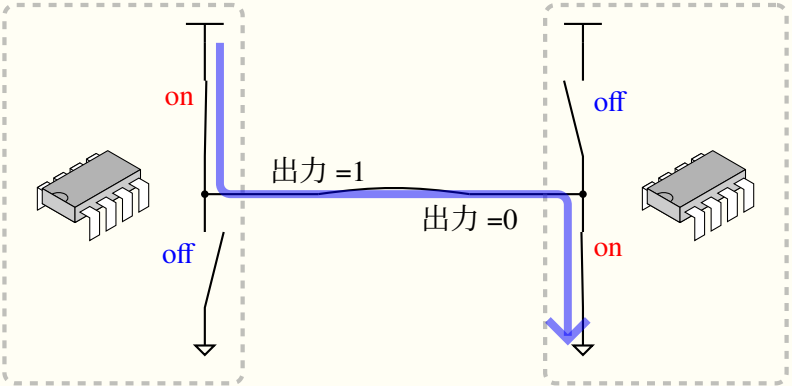


- “1” と “0” の出力を直結すると
- **大電流**が流れて、
- IC を傷める。

トーテムポール出力

トーテムポール出力による “H” と “L” の出力

IC の出力段 (トーテムポール出力) のイメージ。実際はこんなスイッチではなくトランジスタを使っているが、基本的な理解としてはこの描像で ok。

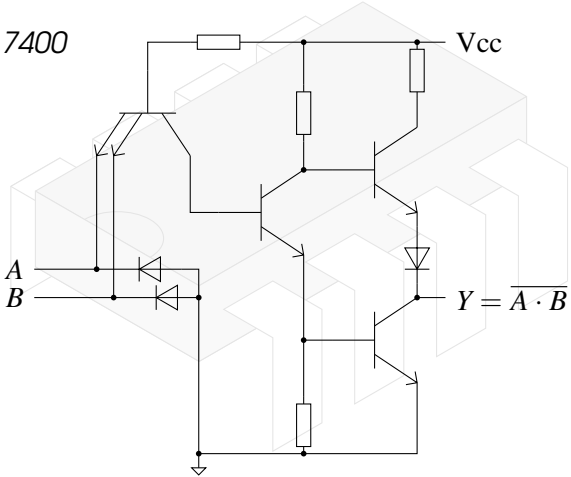


- “1” と “0” の出力を直結すると
- **大電流**が流れて、
- IC を傷める。

…で、それが何か?

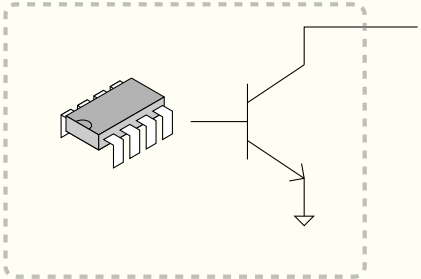
(参考) モンモノ (7400) の例

どこがトーテムポールだかわかるかな？



オープンコレクタ出力

そのままでは (特に “H” が) きちんと出力できない特殊な出力

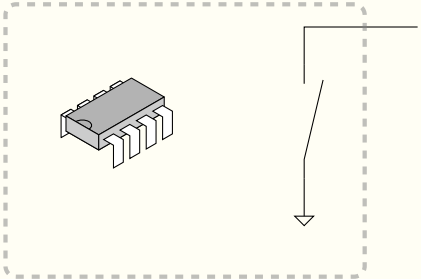


- このままでは電圧出力はできない。
- 外部に　　をつければ電圧出力ができる。
- (出力が “L” のとき) 比較的大きな電流の　　ができるタイプが多い。
- 複数の出力を　　すれば、　　が実現できる!

※ 内部がトランジスタではなく FET 構成の場合は「　　」と言う。

オープンコレクタ出力

そのままでは (特に “H” が) きちんと出力できない特殊な出力

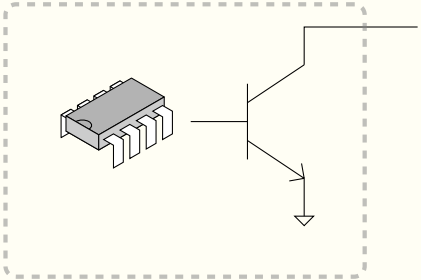


- このままでは電圧出力はできない。
- 外部に**プルアップ抵抗**をつければ電圧出力ができる。
- (出力が “L” のとき) 比較的大きな電流の　　ができるタイプが多い。
- 複数の出力を　　すれば、　　が実現できる!

※ 内部がトランジスタではなく FET 構成の場合は「　　」と言う。

オープンコレクタ出力

そのままでは (特に “H” が) きちんと出力できない特殊な出力

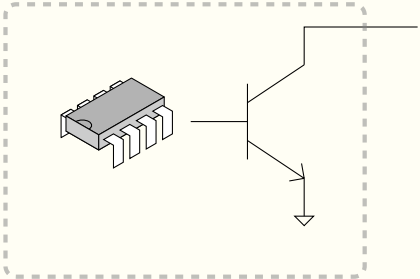


- このままでは電圧出力はできない。
- 外部に**プルアップ抵抗**をつければ電圧出力ができる。
- (出力が “L” のとき) 比較的大きな電流の 流し込みができるタイプが多い。
- 複数の出力を　　すれば、
  　　が実現できる!

※ 内部がトランジスタではなく FET 構成の場合は「　　」と言う。

オープンコレクタ出力

そのままでは (特に “H” が) きちんと出力できない特殊な出力

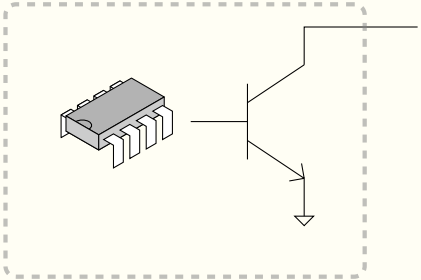


- このままでは電圧出力はできない。
- 外部に**プルアップ抵抗**をつければ電圧出力ができる。
- (出力が “L” のとき) 比較的大きな電流の 流し込みができるタイプが多い。
- 複数の出力を**直結**すれば、**ワイヤード AND/OR**が実現できる!

※ 内部がトランジスタではなく FET 構成の場合は「　　」と言う。

オープンコレクタ出力

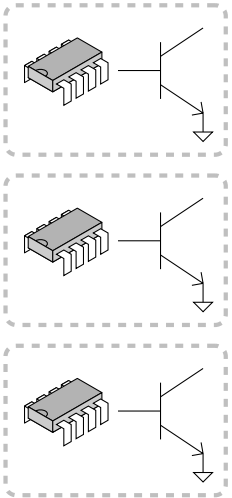
そのままでは (特に “H” が) きちんと出力できない特殊な出力



- このままでは電圧出力はできない。
- 外部に**プルアップ抵抗**をつければ電圧出力ができる。
- (出力が “L” のとき) 比較的大きな電流の 流し込みができるタイプが多い。
- 複数の出力を**直結**すれば、**ワイヤード AND/OR**が実現できる!

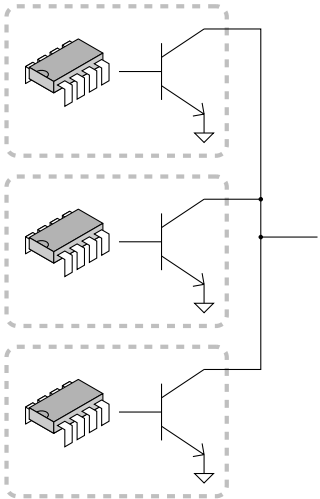
※ 内部がトランジスタではなく FET 構成の場合は「**オープンドレイン**」と言う。

出力同士の直結: ワイヤードロジック



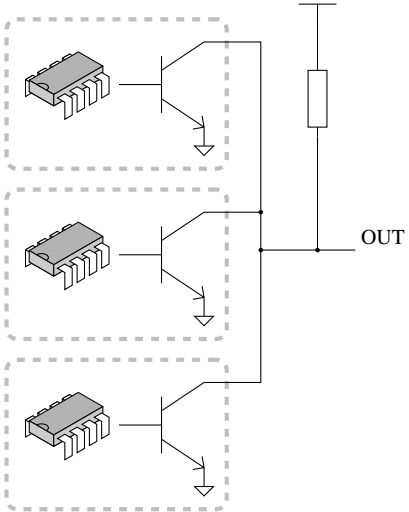
- すべてのトランジスタが off のとき (すなわち、**すべての出力が “1” のとき**)、プルアップ抵抗には電流が流れ＿＿＿。
  → OUT の電位は ＿**レベル**＿。
- いずれかのトランジスタが on のとき (すなわち、**少なくとも一つの出力が “0” のとき**)、プルアップ抵抗には電流が 流れ＿＿＿。
  → OUT の電位は ＿**レベル**＿。

出力同士の直結: ワイヤードロジック



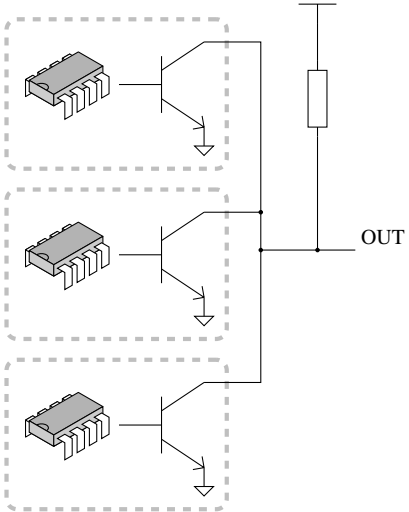
- すべてのトランジスタが off のとき (すなわち、**すべての出力が “1” のとき**)、プルアップ抵抗には電流が流れ＿＿＿。
  → OUT の電位は ＿**レベル**＿。
- いずれかのトランジスタが on のとき (すなわち、**少なくとも一つの出力が “0” のとき**)、プルアップ抵抗には電流が 流れ＿＿＿。
  → OUT の電位は ＿**レベル**＿。

出力同士の直結: ワイヤードロジック



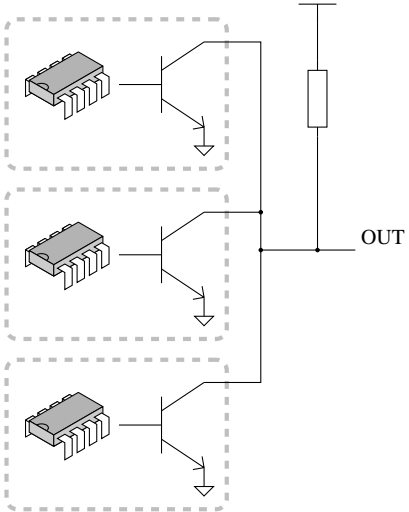
- すべてのトランジスタが off のとき (すなわち、**すべての出力が “1” のとき**)、プルアップ抵抗には電流が流れ ない。
  → OUT の電位は **レベル**。
- いずれかのトランジスタが on のとき (すなわち、**少なくとも一つの出力が “0” のとき**)、プルアップ抵抗には電流が 流れ 。
  → OUT の電位は **レベル**。

出力同士の直結: ワイヤードロジック



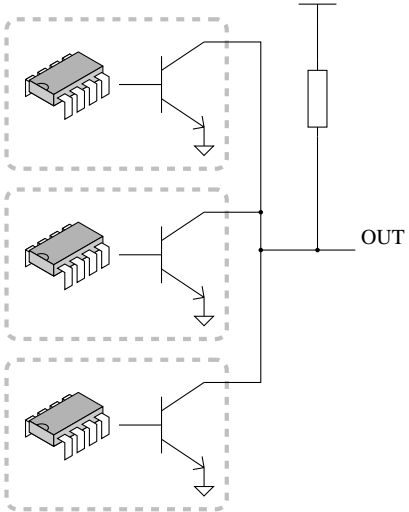
- すべてのトランジスタが off のとき (すなわち、**すべての出力が “1” のとき**)、プルアップ抵抗には電流が流れ ない。
  → OUT の電位は **Hレベル**。
- いずれかのトランジスタが on のとき (すなわち、**少なくとも一つの出力が “0” のとき**)、プルアップ抵抗には電流が 流れ 。
  → OUT の電位は **レベル**。

出力同士の直結: ワイヤードロジック



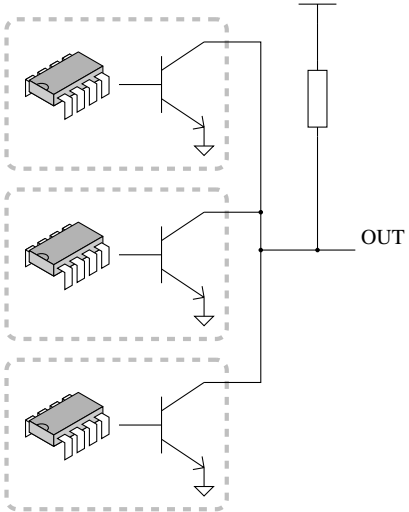
- すべてのトランジスタが off のとき (すなわち、**すべての出力が “1” のとき**)、プルアップ抵抗には電流が流れ ない。
  → OUT の電位は **Hレベル**。
- いずれかのトランジスタが on のとき (すなわち、**少なくとも一つの出力が “0” のとき**)、プルアップ抵抗には電流が 流れ る 。
  → OUT の電位は **レベル**。

出力同士の直結: ワイヤードロジック



- すべてのトランジスタが off のとき (すなわち、**すべての出力が “1” のとき**)、プルアップ抵抗には電流が流れ ない。
  → OUT の電位は **Hレベル**。
- いずれかのトランジスタが on のとき (すなわち、**少なくとも一つの出力が “0” のとき**)、プルアップ抵抗には電流が 流れ る 。
  → OUT の電位は **Lレベル**。

出力同士の直結: ワイヤードロジック



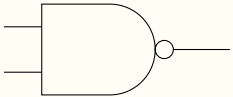
- すべてのトランジスタが off のとき (すなわち、**すべての出力が “1” のとき**)、プルアップ抵抗には電流が流れ ない。
  → OUT の電位は **Hレベル**。
- いずれかのトランジスタが on のとき (すなわち、**少なくとも一つの出力が “0” のとき**)、プルアップ抵抗には電流が 流れ る 。
  → OUT の電位は **Lレベル**。

AND が結線 (wired) だけでできた!

→**wired AND**

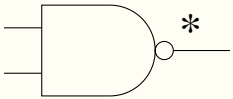
オープンコレクタは実在する

7400: Quad 2 Input NAND



トーテムポール

7403: Quad 2 Input O.C. NAND



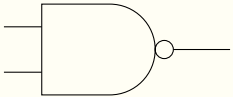
オープンコレクタ

※ オープンコレクタを強調したいときには何か記号 (＊とか) を書く場合もある。

問: オープンコレクタを活用して 3 入力多数決回路を設計せよ。

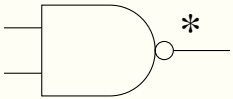
オープンコレクタは実在する

7400: Quad 2 Input NAND



トーテムポール

7403: Quad 2 Input O.C. NAND



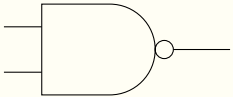
オープンコレクタ

※ オープンコレクタを強調したいときには何か記号 (＊とか) を書く場合もある。

問: オープンコレクタを活用して 3 入力多数決回路を設計せよ。

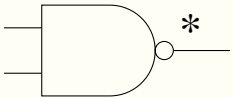
オープンコレクタは実在する

7400: Quad 2 Input NAND



トーテムポール

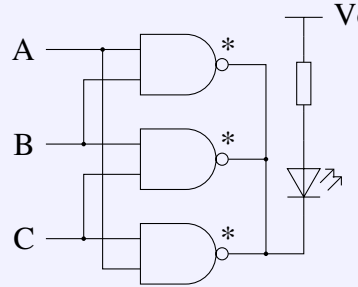
7403: Quad 2 Input O.C. NAND



オープンコレクタ

※ オープンコレクタを強調したいときには何か記号 (＊とか) を書く場合もある。

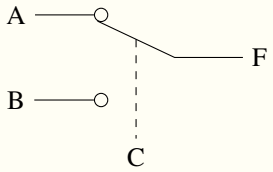
問: オープンコレクタを活用して 3 入力多数決回路を設計せよ。



マルチプレクサ (データセレクタ)

- **multiplexer**: n 個の中から一つを選ぶ回路。

2 to 1 データセレクタの例



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

1. Karnaugh map:

C \ AB	00	01	11	10
0				
1				

2. 論理式:

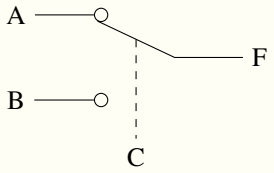
$$F =$$

$$=$$

マルチプレクサ (データセレクタ)

- **multiplexer**: n 個の中から一つを選ぶ回路。

2 to 1 データセレクタの例



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

1. Karnaugh map:

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

2. 論理式:

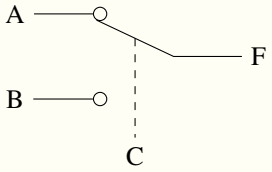
$$F =$$

$$=$$

マルチプレクサ (データセレクタ)

- **multiplexer**: n 個の中から一つを選ぶ回路。

2 to 1 データセレクタの例



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

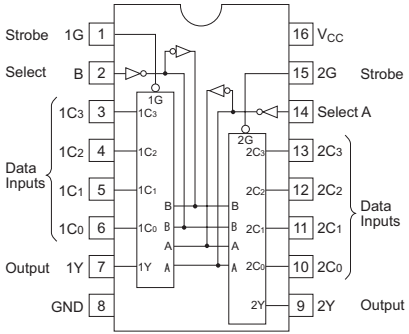
1. Karnaugh map:

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

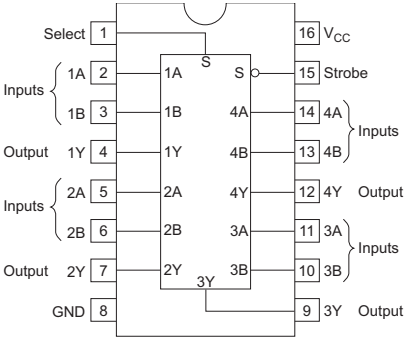
2. 論理式:

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

実際のデータセレクタの例[1]



74153

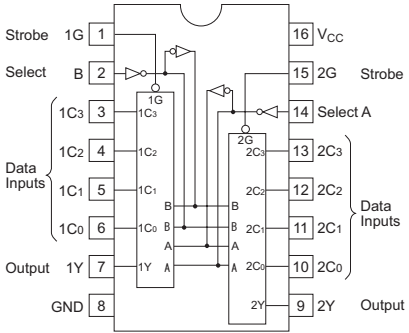


74157

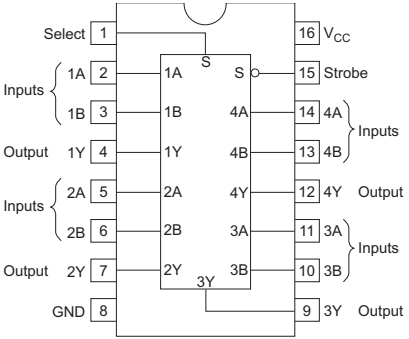
- 74153 は『　　-to-　データセレクタ』
- 74157 は『　　-to-　データセレクタ』

[1]図は日立製 HD74HC153, HD74HC157 のデータシートより拝借

実際のデータセレクタの例[1]



74153

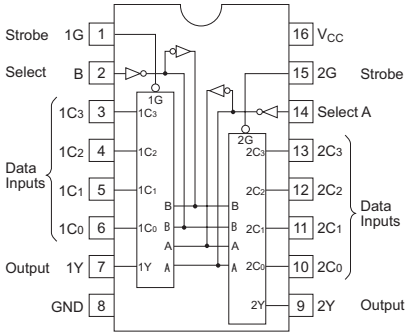


74157

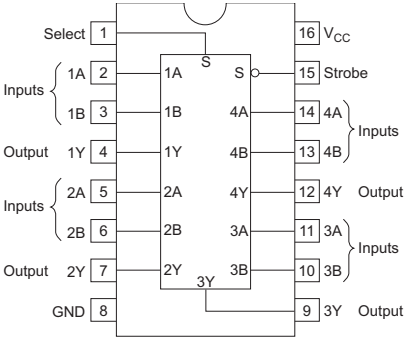
- 74153 は『　　4-to-1 データセレクタ』
- 74157 は『　　-to-　データセレクタ』

[1]図は日立製 HD74HC153, HD74HC157 のデータシートより拝借

実際のデータセレクタの例[1]



74153

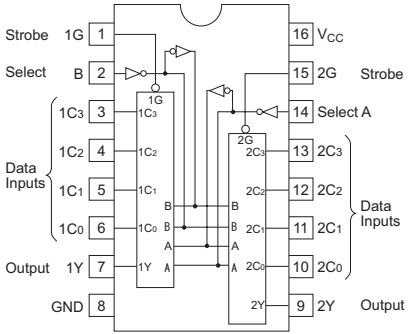


74157

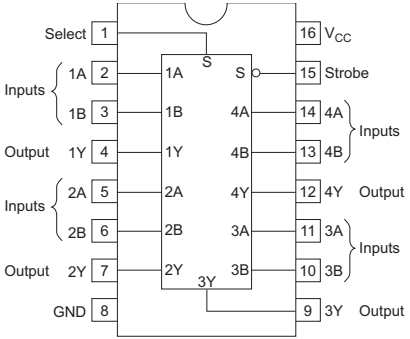
- 74153 は『　　4-to-1 データセレクタ』
- 74157 は『　　2-to-1データセレクタ』

[1]図は日立製 HD74HC153, HD74HC157 のデータシートより拝借

実際のデータセレクタの例[1]



74153

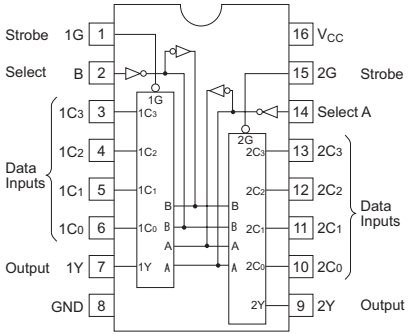


74157

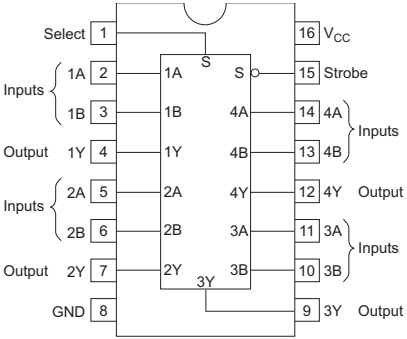
- 74153 は『dual 4-to-1 データセレクタ』
- 74157 は『　　2-to-1データセレクタ』

[1]図は日立製 HD74HC153, HD74HC157 のデータシートより拝借

実際のデータセレクタの例[1]



74153



74157

- 74153 は『dual 4-to-1 データセレクタ』
- 74157 は『quad 2-to-1データセレクタ』

[1]図は日立製 HD74HC153, HD74HC157 のデータシートより拝借

H でも L でもない第三の状態。その名は『　　』。

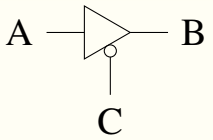
H 電位が高く、電流を流し出せる。

L 電位が 0 で、電流を流し込める。

回路から切り離され、自らは電位を定めず電流の流入出もない。

➡ 状態

状態の出力は _____。



この場合 C=1 で Hi-Z 状態。

H でも L でもない第三の状態。その名は『Z (Hi-Z)』。

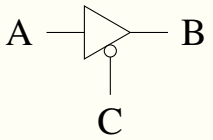
H 電位が高く、電流を流し出せる。

L 電位が 0 で、電流を流し込める。

Z 回路から切り離され、自らは電位を定めず電流の流入出もない。

➡ 状態

状態の出力は _____。



この場合 C=1 で Hi-Z 状態。

H でも L でもない第三の状態。その名は『Z (Hi-Z)』。

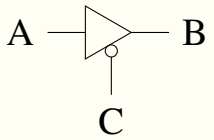
H 電位が高く、電流を流し出せる。

L 電位が 0 で、電流を流し込める。

Z 回路から切り離され、自らは電位を定めず電流の流入出もない。

➡ **ハイ・インピーダンス (high Z)**状態

Hi-Z状態の出力は _____。



この場合 C=1 で Hi-Z 状態。

H でも L でもない第三の状態。その名は『Z (Hi-Z)』。

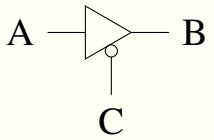
H 電位が高く、電流を流し出せる。

L 電位が 0 で、電流を流し込める。

Z 回路から切り離され、自らは電位を定めず電流の流入出もない。

➡ **ハイ・インピーダンス (high Z)**状態

Hi-Z状態の出力は 何もしない。



この場合 C=1 で Hi-Z 状態。

H でも L でもない第三の状態。その名は『Z (Hi-Z)』。

H 電位が高く、電流を流し出せる。

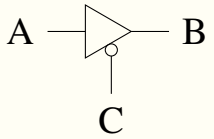
L 電位が 0 で、電流を流し込める。

Z 回路から切り離され、自らは電位を定めず電流の流入出もない。

➡ **ハイ・インピーダンス (high Z)**状態

Hi-Z状態の出力は 何もしない。

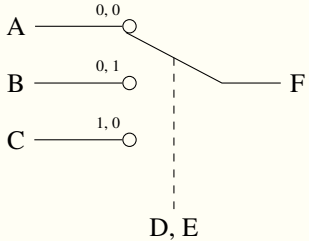
3 state buffer



この場合 C=1 で Hi-Z 状態。

もう一つの n 択: バス

復習: n 択を実現する

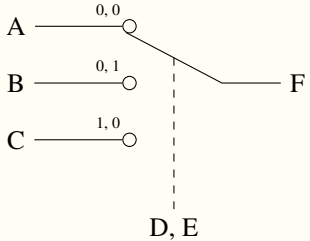


$$F = \begin{cases} A & \text{if } DE = 00 \\ B & \text{if } DE = 01 \\ C & \text{if } DE = 10 \end{cases}$$

欠点: ,

もう一つの n 択: バス

復習: n 択を実現する

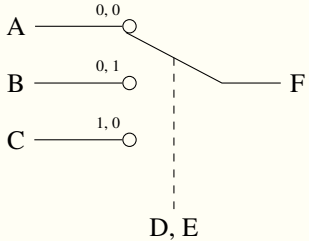


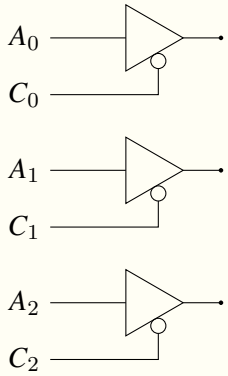
$$F = \begin{cases} A & \text{if } DE = 00 \\ B & \text{if } DE = 01 \\ C & \text{if } DE = 10 \end{cases}$$

欠点: n が大きくなると大変,

もう一つの n 択: バス

復習: n 択を実現する



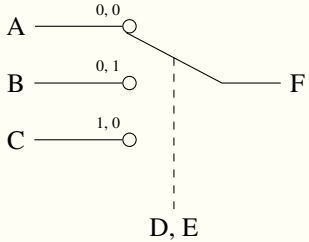


$$F = \begin{cases} A & \text{if } DE = 00 \\ B & \text{if } DE = 01 \\ C & \text{if } DE = 10 \end{cases}$$

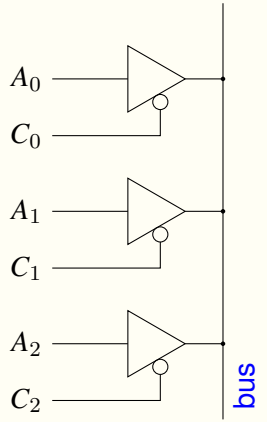
欠点: n が大きくなると大変,

もう一つの n 択: バス

復習: n 択を実現する



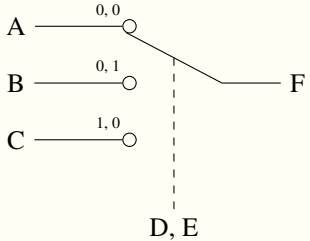
$$F = \begin{cases} A & \text{if } DE = 00 \\ B & \text{if } DE = 01 \\ C & \text{if } DE = 10 \end{cases}$$

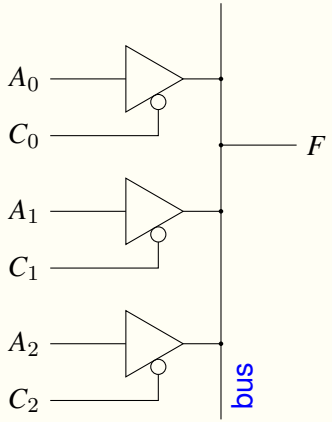


欠点: n が大きくなると大変,

もう一つの n 択: バス

復習: n 択を実現する



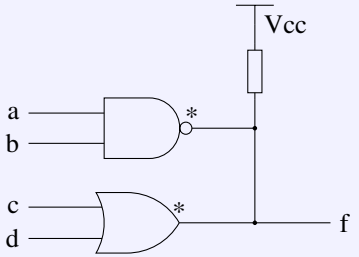


$$F = \begin{cases} A & \text{if } DE = 00 \\ B & \text{if } DE = 01 \\ C & \text{if } DE = 10 \end{cases}$$

欠点: n が大きくなると大変,

出席確認レポート課題 (次の月曜の 12 時締め切り)

問: 右の回路の真理値表を作成し、カルノー図等で簡単化した $f(a, b, c, d)$ を求めよ。加法標準形・乗法標準形いずれでも構わない。なお、図のとおり 2 つの論理素子はいずれもオープンコレクタ出力である。



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