

論理回路

1. 実験の目的

本実験では、デジタル回路設計の基礎として**組み合わせ回路**と**順序回路**を学ぶ。

●組み合わせ回路の実験

次の URL に本実験の設計情報が掲載されています。

<http://www.ee.secu.chukyo-u.ac.jp/LogicalCircuit/>
内容を確認してから実験を開始してください。



家庭にある炊飯器等の電化製品はどれも電気で動作している。最近の電化製品は単純なものは少なく、「マイコン炊飯器」、「センサー付きエアコン」等、どれもが複雑な機能を持つ。このような複雑な機能を実現するためにデジタル回路は欠かせない。

たとえば、LED（発光ダイオード）は2つの状態「点灯」、「消灯」を表現できるが、その形状が棒状のLEDを7つ組合せて8の字型に並べれば0から9までの数字を表すことのできる表示板になる（これを「**7セグメント表示器**」という。図1）。また、きれいに基盤の目のように並べればコンピュータディスプレイ装置のようなドットマトリクス表示器にもなる。このような表示器は、多くのLEDの「点灯」、「消灯」を制御して実現している。

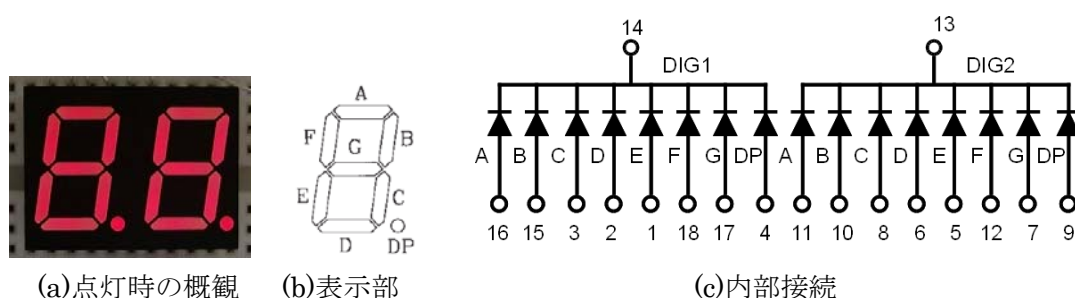


図1：2桁の数を表示できる7セグメント表示器 C-552SRD

また、コンピュータの中も同様である。数値や文字を記憶するということは2進数を覚えることと同じである。2進数は「1」と「0」の文字によって表現される単純なもので、それらを用いて加減乗除算等の演算も実現できる。

このように、「LEDの点灯・消灯」や「コンピュータ内の2進数」は2つの状態のうちどちらであるか（2者択一）を表しており、それをたくさん保持すればするほど多くの機能を実現することができる。**論理回路**は、このような状態を1つの信号線（電線）に「電圧がかかっている（H:High）」、「かかっていない（L:Low）」という2つの値で表現し、それを組み合わせることによってたくさんの論理動作を実現する回路のことをいう。また、その論理回路を実際の電子回路として実現したものを**デジタル回路**という。

現代ではパソコンのようにCPU上で動作させるソフトウェアだけではなく、ハ

ードウェアも理解したソフトウェア技術者（逆に言えば、ソフトウェアも理解したハードウェア技術者）が社会から強く求められている。それはソフトウェアではできない次のようなハードウェア（電子回路）がたくさんあるからである。

- ・高速に動作（ハードウェアはソフトウェアと比べて少なくとも数100倍以上高速）
- ・不要なものがないので単純（必要最低限のものしかない、つまり小型・軽量）
- ・集積化可能（多くの機能を1つのシステム上に実現）

たとえば、普段使っているパソコンは、さまざまな仕事を柔軟にこなすことができるが、ソフトウェアを中心にして動いているため動作が遅く、装置そのものが大きいのが難点である。一方、電子回路はトランジスタからIC、LSI、VLSIと高集積化・多機能化（1つのパッケージの中に多くの機能を実現できる）が進んでおり、一旦動きはじめたら高速に動作し、なかなか壊れない。

このように、今までにない高速・小型・多機能な製品を実現するためには、ソフトウェアだけでは限界があり、どうしてもハードウェアの力を借りなければならない。一方、ハードウェアでは設計等に時間が必要であり、柔軟できめ細かな動作をする製品を実現するためにはソフトウェアの力を必要とする。つまり、ハードウェアの知識だけではダメ、ソフトウェアの知識だけでもダメ、ハードウェアとソフトウェアの両方を知り、それらをバランスよく使いわけることができ初めてよい製品を短期間で開発することができる。現代は、ハードウェアとソフトウェアの両方を使いこなせるエンジニアが求められている。

デジタル回路（ハードウェア）の設計というと難しそうに思えるが、意外と簡単である。それはパソコンのソフトウェアを作り始める時に経験したのと同じように、機能はたくさんあるが、その一つ一つの基本機能は少なく、単純明解だからである。特に初心者には、デジタル回路の仕組みは理解しているが、作ることに自信がなく難しく感じる人が多いかも知れない。しかし、一度慣れてしまえば、たくさんの機能を実現するデジタル回路を次々と作り出すことができるようになる。この実験では、簡単なデジタル回路の製作を通して、ハードウェアに慣れることを目的とする。

2. 実験の原理

デジタル回路の基礎

まずはハードウェアを実現するためのデジタル回路の基礎を学習しよう。デジタル回路は、1つの信号線（電線）に「電圧がかかっている（H:High）」、「かかっていない（L:Low）」という2つの状態を作って動作する。以下では、H状態を「1」、L状態を「0」に割り当てる**正論理**でデジタル回路の動作を考えることにする（1と0を逆に割り当てる場合を**負論理**という）。つまり、回路上の信号線の状態と2進数の論理の間には、次の表1の関係があるものとする。

表 1 : 電線の状態と 2 進数の論理の関係 (正論理)

電線の状態	2 進数の論理
電圧がかかっている (H:High)	1
かかっていない (L:Low)	0

通常, 信号線の H 状態は約 +5 V, L 状態は約 0 V とする (**TTL レベル**という).

論理演算とゲート素子

2 進数の和や積の計算では, ビットごとに演算を行なう. デジタル回路では次のようなビット演算ができる基本素子が用意されている. これらの基本素子は, 数本の入力信号線の論理値に応じて出力信号線の論理値が決まるという動作をするため, 信号が門をくぐるというイメージから「**ゲート素子**」と呼ばれている.

それでは, 主なゲート素子を紹介しよう. 以下の各図で, 左側が各ゲート素子の回路図上のシンボルマーク, その下側が動作を表す論理式, 右側が具体的な論理入力に対する出力を示す動作表である.

AND 素子 (論理積)

入力 A, B が ともに 1 のときのみ出力 X が 1 になる (図 2).

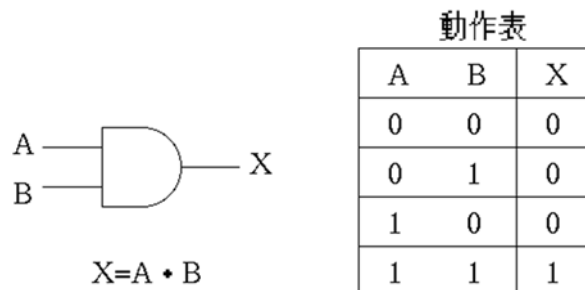


図 2 : AND 素子

OR 素子 (論理和)

入力 A, B の いずれか が 1 のときのみ出力 X が 1 になる (図 3).

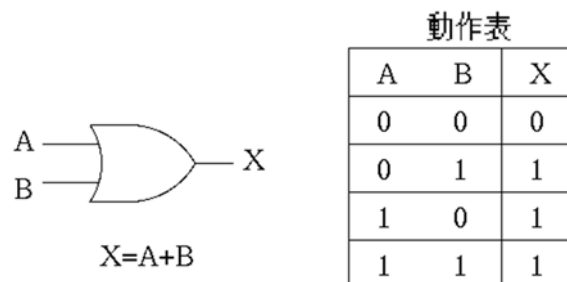


図 3 : OR 素子

NAND 素子（論理積の否定）

入力 A, B がともに 1 のときのみ出力 X が 0 になる（図 4）.

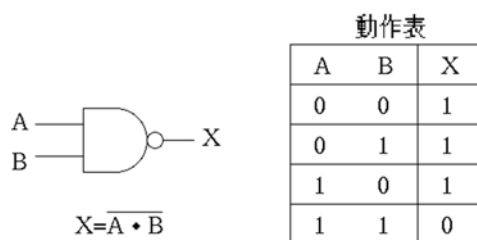


図 4 : NAND 素子

NOR 素子（論理和の否定）

入力 A, B のいずれかが 1 のときのみ出力 X が 0 になる（図 5）.

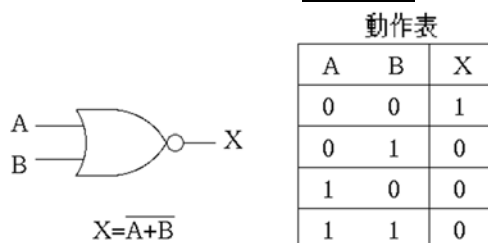


図 5 : NOR 素子

NOT 素子（否定）

入力 A の値を反転させた値が出力 X に出力される. つまり, 入力 A が 1 のとき出力 X は 0, 入力 A が 0 のとき出力 X は 1 になる（図 6）. インバータ素子とも呼ばれる.

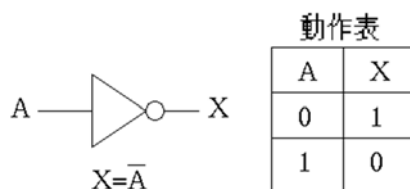


図 6 : NOT 素子

EXOR 素子（排他的論理和）

入力 A, B のいずれか一方だけ 1 のとき出力 X が 1 になる（図 7）. 入力が両方とも 1 の時は, 出力 X は 1 にはならない. つまり, 入力 A, B が一致する時のみ出力 X が 0 になる. エクスクルーシブ OR と呼ぶ.

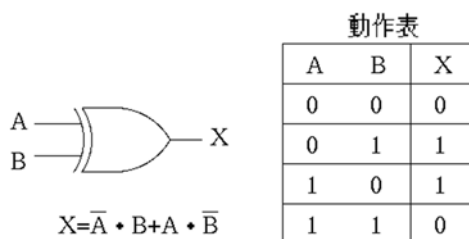


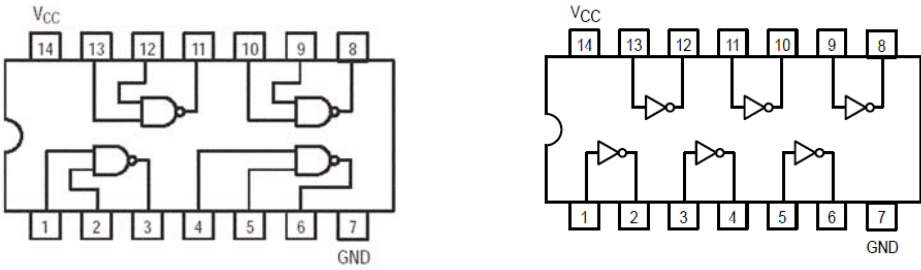
図 7 : EXOR 素子

このようなゲート素子はデジタル I C で実現され、「7 4 シリーズ」と呼ばれる代表的な I C が使われている。I C の型番が 7 4 から始まることから、この名前が付いている。7 4 シリーズの詳しい仕様は各社が W e b 等で公開している規格表（データシート）を参照すること。この実験で使用する主なゲート素子を表 2 に示す。

表 2：74 シリーズの主なゲート素子

ゲート素子	7 4 シリーズ型番
AND	7 4 L S 0 8
OR	7 4 L S 3 2
NAND	7 4 L S 0 0
NOR	7 4 L S 0 2
NOT	7 4 L S 0 4
EXOR	7 4 L S 8 6

これらの I C の多くは 1 4 本のピン（電極）を持ち、電源として 1 4 番ピンに + 5 V（VCC と書く）、7 番ピンに 0 V（GND と書く）を接続することで動作する。他のピンがどのような機能を持っているかは規格表（データシート）から調べる。例えば、7 4 L S 0 0（NAND 4 つ搭載）と、7 4 L S 0 4（NOT 6 つ搭載）は切り欠き部分を左にして上面から見ると次の図 8 のピン配置になっている。



(a) 7 4 L S 0 0 (b) 7 4 L S 0 4

図 8：7 4 シリーズ I C のピン配置

加算器（足算器）の設計（その 1）

C P U 等の内部や、一般的なデジタル回路で行なわれる演算のうち、最もよく行われるのは加算である。加算器（足算器）のデジタル回路の簡単な例として、2 進数 1 桁の加算器を設計してみよう。

2進数1桁の加算器の動作は次の通りである.

「2つの入力A, Bに0または1を入力すると, それらを加えた値をSに出力する.
もし桁上がりが発生した時には出力Cを1とする.」

次の図9は, 人間が行なう2進数の加算演算と, 2進加算器の出力の関係を示している.

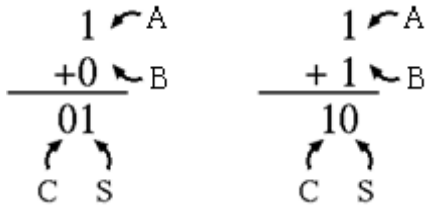


図9：2進数1桁の加算（半加算）

さて, 和信号Sと桁上がり信号Cはどのような回路で実現できるだろうか. 順を追って説明する.

加算器はAとBを入力値とし, 出力S, Cの値を一意に決める箱（ブラックボックス）と考える. このブラックボックスの動作が加算器の回路動作で, それを表で表してみる. この表を「動作表」という（図10）.

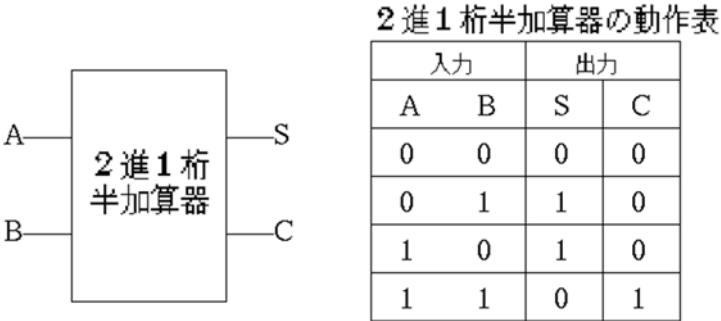


図10：半加算器の動作

この表をよく見てみると, 出力SはE X O Rと同じ動作, 出力CはA N Dと同じ動作である. ことによって, 2進1桁の加算器が実現でき, 「半加算器(half adder)」という（図11）.

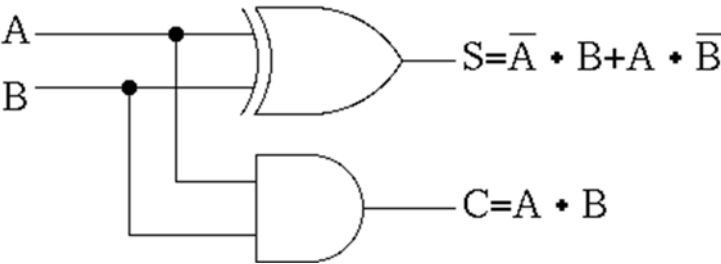


図11：半加算器の回路図

加算器（足算器）の設計（その2）

半加算器の次は「**全加算器(full adder)**」を設計しよう．全加算器は，ひとつ下の桁からの桁上げも考慮した2進1桁の加算器である．つまり，入力A，Bだけでなく，下の桁から桁上げ信号C'も考慮して設計しなければならない．同様に設計してみよう（図12）．

「下の桁から桁上げ信号C'も考慮し，2つの入力A，Bに0または1を入力とし，それらを加えた値をSに出力する．もし桁上がりが発生した時には出力Cを1とする．」

A

B

+11

101

C S C'=0

A

B

+11

110

C S C'=1

図12：2進数2桁の加算（全加算）

まず，SとCの結果を実現するための回路動作を示す動作表を書いてみよう．先ほどよりも入力の組合せが増えていることがわかるはずである．入力の選択肢が1つ増えると，組合せの数は2倍になること注意すること（図13）．

A

B

C'

2進1桁全加算器

S

C

2進1桁全加算器の動作表

入力			出力	
A	B	C'	S	C
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

図13：全加算器の動作

次は，この動作表を満足する回路を設計するが，先ほどのようにゲート素子を単純に当てはめることはできない．そこで動作を表す論理式を導いた後，必要であれば回路の簡単化を行う．回路の簡単化は1年次の科目「論理回路基礎」で学んだ通りである．図14に論理式とその簡単化の過程（カルノー図を使用する等）を記述したうえで，全加算器の回路図を実験開始時までには書き込んでおくこと（回路図は図11を見本とせよ．信号が左から右に流れるように書くのが原則である．また，配線が交差する部分では，接続する場合には●で示さなければならない．一方，接続しない場合は●を記述してはいけない）

全加算器の回路図とそれを得るための途中過程も記述すること．この欄は実験時にスタッフがチェックします．

図 1 4：全加算器の回路図とその導出過程

全加算器は2進数1桁分の加算回路であるが、これと同じ回路を複数接続することで、何桁の加算器でも作ることができる．次の図 1 5 は4桁の2進加算器の例である．

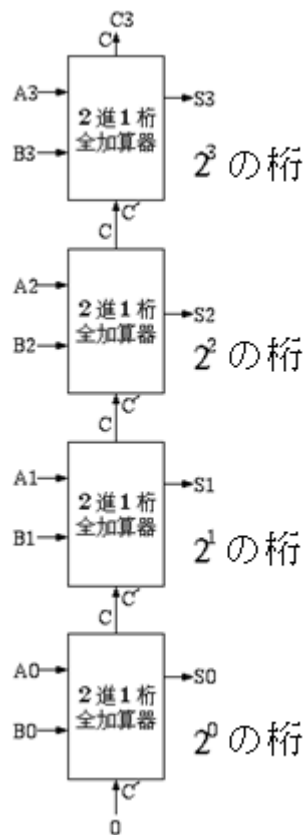


図 1 5：4桁の2進加算器

まとめ

CPU等の複雑な回路は、ここで設計した加算器のような簡単で基本的な回路が多数組合わさって複雑な動作をしている。それは、各回路への入力信号の数を増やすことで、その動作の組合せの数を爆発的に増やし、多くの動作を実現しているのである。先ほどの4桁の2進加算器が、2進1桁の全加算器を複数接続して実現できることがその例である。

どこに行っても電気製品が存在し、何を作るにしても電気を使う現代では、さらに小型で複雑な回路が必要とされ、その内部ではデジタル回路が活躍している（最近ではこのようなものを「**組み込みシステム**」と呼ぶことが多い）。現代では、これらの回路はすべて人間が設計しているのではなく、設計時間の短縮と、人間による設計誤りの混入防止のためコンピュータの力を借りて設計している。それが「**電子回路CAD(Computer Aided Design)**」や「**EDA(Electronic Design Automation)**」と呼ばれている分野である。電子回路CADを用いることで、短時間で複雑な回路を間違えずに設計することができる。本実験では、電子回路のCADの一環として、物理学実験で行った「BSch3Vを用いた回路図入力」を行い製作した回路をレポートする。電子回路の詳細な設計や、CADを用いた自動設計については、別科目「**デジタル回路とHDL**」で行うが、ここで得た回路設計・製作の体験が大きく役立つ。

本実験の前半で設計するデジタル回路は、どれも入力値の組み合わせがひとつ決まればそれに応じた値をいつも出力する回路である。このような回路を「**組合せ回路**」という。また、本実験の後半で設計するデジタル回路は、出力値がいつも同じ動作をしない回路で「**順序回路**」という。そのような回路は入力値の変化に応じて値が定まるメモリのような記憶回路である。

実際のデジタル回路の実現について

論理回路の動作についてひと通り理解したら、次は実際にデジタル回路として動かす方法について確認する。一般電化製品では、ICへの電源供給や、素子間の電氣的な接続は**プリント基板**を用いて配線する。プリント基板は、一般に緑色のガラス繊維（絶縁体）の板の表面に銅膜が貼ってあり、電線以外の不要な銅膜部分を化学反応で溶かして必要な配線（電線）を実現する（図16）。このようなプリント基板が使われているのは、多くの配線を小さい面積で実現するためだけでなく、回路の安定動作が期待できるからである。



図16：プリント基板を用いた配線例（ROM，RAMの周辺配線）

本実験ではプリント基板を製作するには時間が不足するので、簡易的にデジタル回路を実現できる「ブレッドボード（試作用基板）」を使用する（図17）。

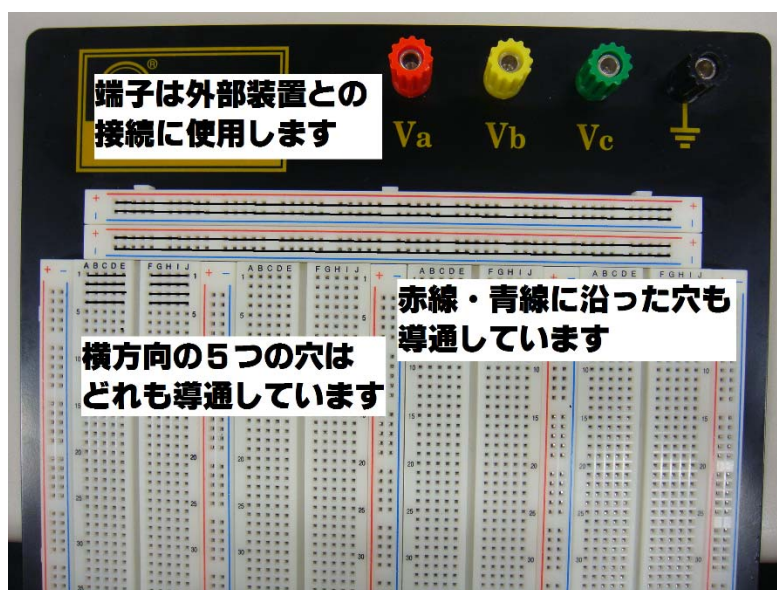


図17：ブレッドボード

ブレッドボードは電線を差し込む穴がたくさんあいている配線板である。赤色または青色の線に沿った穴は内部で導通しており、主に電源配線用に用いる。赤色は+5V、青色は0V（GND）とする。

また、何も書かれていないが、溝と垂直方向に並ぶ5個の横方向の穴も内部で導通している。ICは溝をまたぐように置き、溝の両側の穴にICのピンを挿入して使う。そして、ICのピンの横方向に並んだ残りの穴に電線を差し込み、ICのピン間を接続する。電線は、電源の+5Vに赤色、0V（GND）に黒色を使用し、それ以外の信号線にその他の色を用いること。また、可能な限り短い電線を使い、できるだけブレッドボードの縦横の穴に沿って配線するときれいに配線できる。用途に応じて色が統一されていない電線や長い電線の利用は回路の理解を阻害するばかりでなく、回路動作を不安定にさせることもあるので避けること。一般に「見た目で見やすい配線は性能がよい配線」と言われている。なお、ブレッドボード上部の端子は、安定化電源装置や波形発生器との接続に利用する。

1と0の論理値をデジタル回路に入力する方法

ゲート素子やフリップフロップ素子を動かすためには、入力信号線に1と0の値を与える必要がある。入力信号線に1と0の値を与えるには次のようにして行う。単純に、「1」は+5V、「0」は0Vだから電源の+5Vと0Vに接続すればよい、と考えたくなるが誤りである。74シリーズをはじめ、一般的なICは+5Vをかけた時に信号線に電流が流れ過ぎて壊れてしまうことがある。そこで、図18に示すスイッチと抵抗を使って1と0の値

を生成する．このとき使う抵抗（ $4.7\text{ k}\Omega$ ）を「プルアップ抵抗」という．プルアップ抵抗の抵抗値については使用する IC によって異なるが，本実験では $4.7\text{ k}\Omega$ の抵抗を使用する．

また，入力信号線をどこにも接続しないことは，その値が「1」なのか「0」なのか不明であり，デジタル回路にとって思いもよらない動作をさせてしまうことがある．よって，使用しない入力信号線は「0」（常に 0 V のところ．**グラウンド**，**GND**と書く）に接続するか， $4.7\text{ k}\Omega$ のプルアップ抵抗を通して $+5\text{ V}$ に接続し「1」にしておく．本実験で使用する TTL IC は入力信号線をどこにも接続しないと「1」と認識する IC であり，プルアップする必要はないが，値を明確に設定することで安定した動作が期待できる．

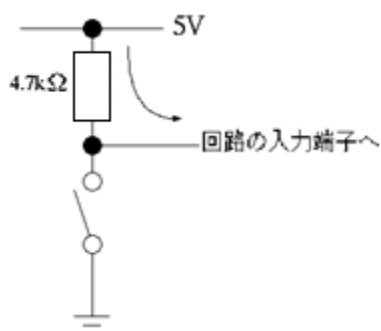


図 18：回路入力とプルアップ抵抗

出力の 1 と 0 の論理値を調べる方法

デジタル回路では，各信号線の論理値が「1」なのか「0」なのかを人間が見てわかるように LED（発光ダイオード）を使用して表示することが多い．LED はダイオードの一種で，一方向に電圧を加えたときのみ電流が流れ発光する半導体素子である．電球と違って，電流が流れる時は LED に無制限に電流が流れてしまい，LED を壊してしまう．そこで，そのようなことが起こらないように図 19 のように抵抗（ 330Ω ）を直列に挿入して流れる電流を制限する．このような抵抗を「電流制限抵抗」という．

このように回路出力に LED を接続することで，消灯のとき「1」，点灯の時「0」の論理値であることがわかる（直感的に負論理）．もし点灯イメージとあわせて点灯のとき「1」，消灯の時「0」としたいなら（直観的に正論理），NOT 素子で出力端子の信号を論理反転すればよい．

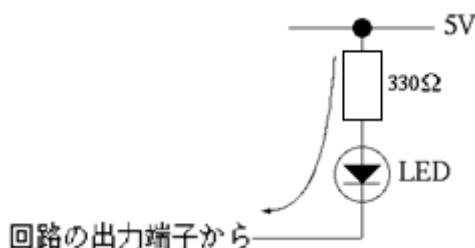


図 19：LED による信号線の状態表示

電源，クロックについて

7 4 シリーズの TTLIC を動かすには，+ 5 V の電源が必要である．一般的な電池 1 本の端子電圧は 1. 5 V なので，それを 3 本直列に接続して 4. 5 V の電源を作り代用することもできるが，実験では本格的な「**実験用安定化電源装置**」を使用する．実験用安定化電源装置は，出力電圧や最大電流を自由に設定することができ，万一回路がショートした時でもやけどや火災がおきないように保護装置が入っている（図 2 0 (a)）．

また，クロック信号のような + 5 V と 0 V の方形波形は「**波形発生器（シグナルジェネレータ）**」を使用する（図 2 0 (b)）．波形発生器はさまざまな波形の信号を発生させることができるが，デジタル回路では，「T T L / C M O S」と書かれた + 5 V と 0 V のいずれかの方形波形を出力する端子から作製した回路に信号を加える．これらの機器の使用方法は 1 年次の「物理学実験指導書」の冒頭に記載されているので参照すること．



(a)安定化電源 PR18-3A



(b)波形発生器 FG-274

図 2 0：安定化電源装置（直流電源）と波形発生器（シグナルジェネレータ）

3. 実験機材

組み合わせ回路の実験に必要な部品を表 3 に示す．

表 3：組み合わせ回路の実験に必要な部品

品名	規格・型番	個数
7 4 シリーズ I C	適切な品番	適切な数
抵抗	3 3 0 Ω (1 / 8 W)	適切な数
抵抗	4. 7 k Ω (1 / 8 W)	適切な数
L E D	赤	適切な数
電線	各色	適切な数

注意：抵抗は，帯色によって抵抗値が示されている．使用する前に必ずその値を確認してから使用すること．返却するときも必ず帯色によって抵抗値を確認し，違う値の場所に返却しないこと．

4. 実験方法

実験 1～5 を開始する前に、予備実験として、スイッチ入力回路 (図 1 8) と LED 出力回路 (図 1 9) を接続した「スイッチ入力 LED 出力回路」を製作し、動作を確認せよ。その後、実験 1～5 のデジタル回路 (組み合わせ回路) を製作せよ。製作は次の手順に従うこと。特に初心者は「**急がず焦らず落ち着いて**」この手順に従うこと。この手順に従わないとプロでもブレッドボード上に回路を作るのは難しい。うまく動作せず、どこが間違っているのかわからず、完成するまで時間がかかりとても効率が悪いので絶対に避けること。なお、事前に「<http://www.ee.secu.chukyo-u.ac.jp/LogicalCircuit/>」の閲覧を勧める。

1. ブレッドボード上に IC を回路図中の番号 (U1, U2, ...) の順に並べて配置する
2. 各 IC の電源ピン (主に 1 4 番ピン) を赤色の電線で +5 V (VCC) に接続する
3. 各 IC の GND ピン (主に 7 番ピン) を黒色の電線で 0 V (GND) に接続する
4. その他の部品 (抵抗, LED, スイッチ等) をブレッドボード上に適当な間隔をあけて配置する
5. +5 V (VCC) に接続された残りの配線を赤色の電線ですべて接続する
6. 0 V (GND) に接続された残りの配線を黒色の電線ですべて接続する
7. 回路図に記載された各 IC のピン番号を参考にして、他の配線を赤色・黒色以外の電線ですべて接続する (同電位の配線は同一色の電線で結ぶこと)
8. 他の部品間も赤色・黒色以外の電線ですべて接続し、完成させる
9. 回路図上のすべての配線が電線で接続されているかどうかをチェックする

注意

- 本実験では、理論値 (真理値表または動作表の値) と実際に測定した値 (LED の点灯または消灯) が正しいか否かを確認し、実験ノートに表として記録すること。
- 回路の正しい動作が確認できたら、各実験終了時にスタッフからその動作のチェックを受けること。チェックを受けずに次の実験に進むことのないようにすること。
- 所望の動作が確認できないときは、机上のデジタルマルチテスタを使用して各信号線の値を調べるとよい。テスタのマイナス (黒) 端子を GND に接続し、プラス (赤) 端子を調べたい信号線に接続すると、その信号線の値が 0 V または 5 V で値が 0 なのか 1 なのか判断できる。
- 連続する実験では共通部分が多く存在している。実験ごとにすべての回路を作り直すのではなく、流用できる部分は残し、新規部分のみ追加して実験時間を短縮せよ。

レポートは、各実験でブレッドボード上に作製した回路を、回路図入力 CAD 「BSch3V」で入力した回路図として直接印刷して報告するのみとする (2017 年度から)。BSch3V は、「ファイル」メニュー→「印刷オプション」→「ページフィット」にチェックを入れて A4 用紙 (横) に印刷すること。

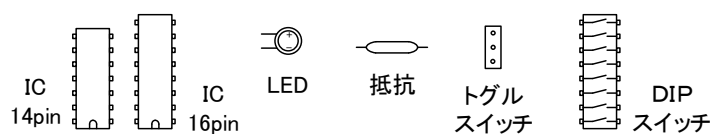
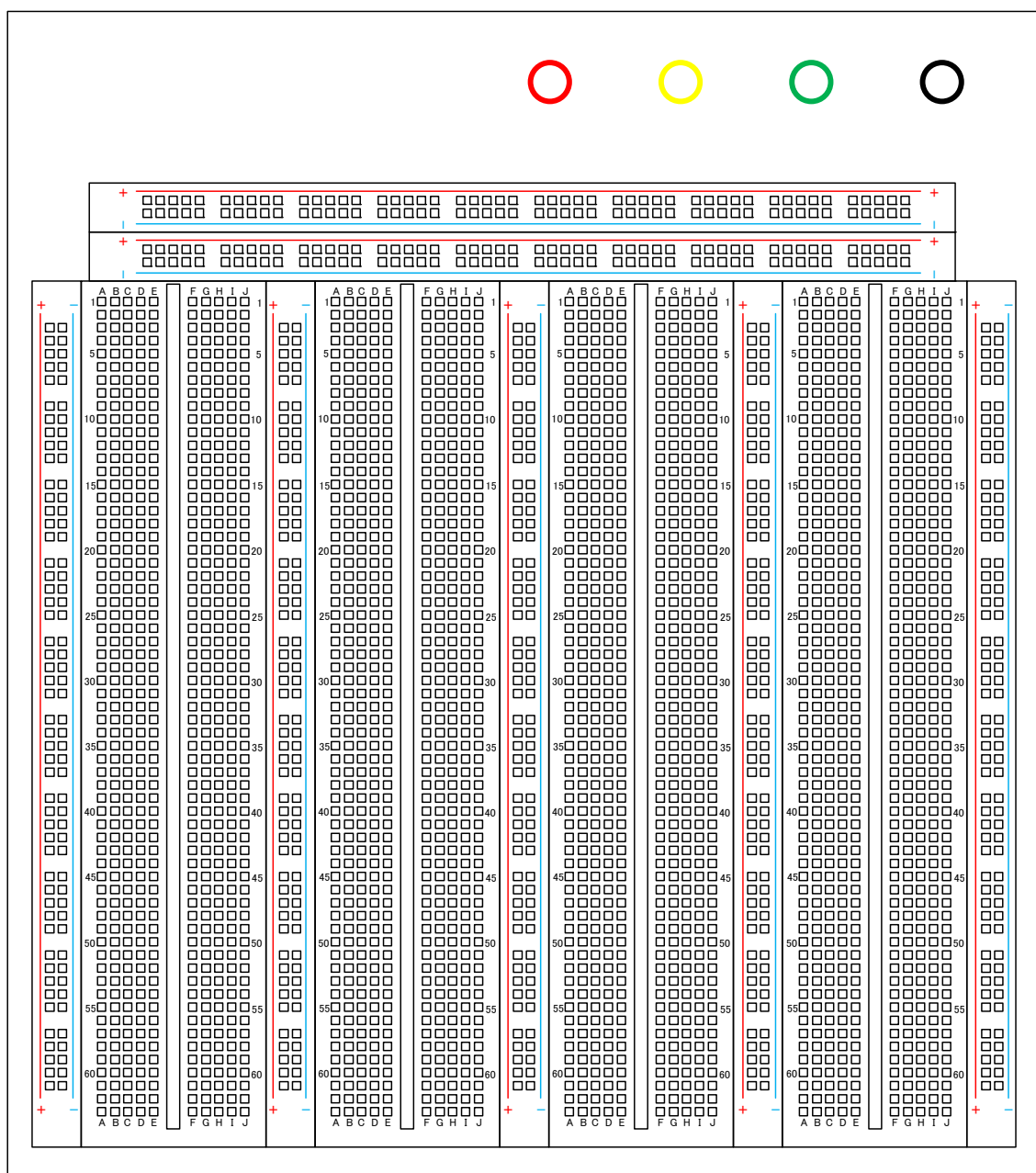
「BSch3V」(水魚堂) <http://www.suigyodo.com/online/schsoft.htm>

(参考)「BSch3V」はフリーソフトである。上記アドレスにアクセスして各自のパソコンにインストールすることができる。また、電気電子工学実験室のパソコンにはすでに「BSch3V」がインストールされている。

参考

製作した回路を記録するためのブレッドボードの図を次に用意した。コピーするか、次のところからダウンロードして使用するとよい。

<http://www.ee.secu.chukyo-u.ac.jp/LogicalCircuit/>



実験 1（まず、前述の予備実験で練習してから実験 1 を行うこと）

NAND（74LS00）の動作を確認する回路を作製し、その動作を確認せよ。

（回路の作製手順）

1. NAND 素子 1 つに、2 つの入力用スイッチおよび 1 つの出力確認用 LED を接続した回路図が次の場所に保存されているので、実験開始前に入手せよ。



<http://www.ee.secu.chukyo-u.ac.jp/LogicalCircuit/>

ここに入手した回路図を貼り付けるか、印刷してきた回路図を提示すること。

図 2 1 : 74LS00 動作確認回路の回路図

2. 入手した回路図を参考にして、前述の実験方法の手順に従ってブレッドボード上に回路を作製せよ（図 2 2）。このとき、ブレッドボード上部の赤色・黒色端子を +5 V（赤）と GND（黒）として安定化電源装置との接続に使用すること。このとき、配線作業を集中して行なわないと人間は配線ミスを起こすことに注意すること。
3. 安定化電源装置とブレッドボードの上部端子を接続する前に、安定化電源装置単独で +5 V が出力されるように電圧計を見ながら **VOLTAGE** つまみと **FINE** つまみを回して調整する。一旦安定化電源装置のスイッチを切り、安定化電源装置の + 出力端子と GND 端子の間をバナナジャック付き電線等で接続（短絡）し、安定化電源装置のスイッチを入れ、**CURRENT** つまみで最大電流を 1 A 程度に調整する。調整が終わったら、安定化電源装置のスイッチを切り、短絡していた電線を取り除く。これで作製した回路内で短絡（ショート）していても大電流が流れず IC や回路が壊れにくくなる。
4. ブレッドボード上部の +5 V 端子に赤色の電線で、GND 端子に黒色の電線で安定化電源と接続する。

5. 安定化電源のスイッチを入れる．（もしC Cと書かれた赤ランプが点灯したり，電流計が1 A程度振れたりした場合は，作製した回路内で短絡（ショート）しているのですので安定化電源装置のスイッチを切って回路を確認すること）
6. 作製した回路の入力信号線に0または1の状態を与えるために2つのスイッチを動かし，そのときのNAND回路の出力信号線の状態をLEDにより観測し，スイッチの位置とLEDの点灯・消灯の関係をすべて表に記録する（表4）．（スイッチはレバーを倒した方と逆側の2端子間が接続（短絡）されることに注意すること）
7. 記録した表から，動作が正しいかどうか確認する．

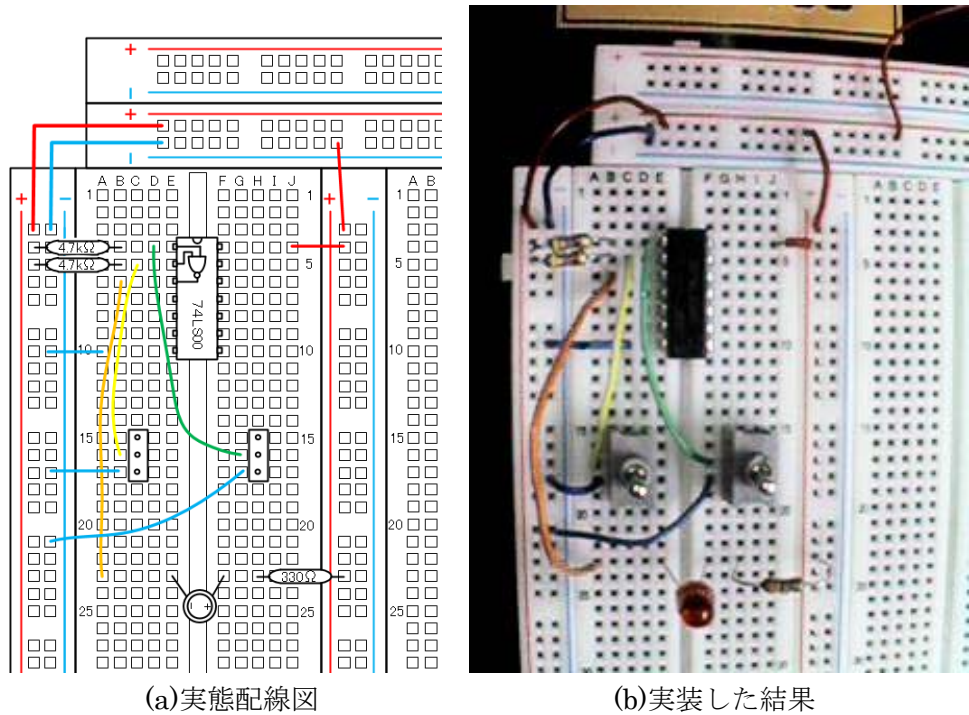


図 2 2 : ブレッドボードへの実装例

表 4 : NAND回路の動作結果

スイッチの位置 (上下左右等を記入)		スイッチの位置に 相当する入力論理値 (0または1を記入)		LEDの状態 (点灯または 消灯を記入)	LEDの状態に 相当する 出力論理値 (0または1を記入)	NANDの 動作確認 (○または×を 記入)
A	B	A	B			

この動作結果表が NAND の動作と合っているかを確認すること．確認ができれば，スタッフに提示してチェックを受けること．

(以下の実験も、実験 1 に習って、同様の手順で回路製作と報告を行うこと)

実験 2

EXOR (74LS86) の動作を確認する回路を作製し、その動作を確認せよ。

実験 3

2 進数 1 桁の全加算器回路を作製し、その動作を確認せよ。

実験 4

「2 つのスイッチで 2 桁の 2 進数を表わす回路」を 2 つ使用することで、両者の 2 進数が一致したとき LED が点灯する回路を作製し、その動作を確認せよ。

実験 5

3 個のスイッチのうち 2 個だけが同時に H の時、LED が発光する回路を作製し、その動作を確認せよ。

●順序回路の実験

ここまでに扱った「組み合わせ回路」は入力信号の値が決まるとそれに応じて出力信号の値も一意に決まる回路であった。それに対して「順序回路」は過去の状態を覚えておき (メモリ)、その状態と入力値をもとに値を出力する回路のことである。つまり、同じ入力値に対して出力値がいつも同じ値とは限らない回路のことである。

フリップフロップ

メモリは「フリップフロップ (FF)」と呼ばれる 1 ビットを記憶することのできる素子の集合である。フリップフロップには多くの種類があるが、ここでは、「RS 型フリップフロップ」、「D 型フリップフロップ」、「JK 型フリップフロップ」そして「T 型フリップフロップ」と呼ばれる記憶素子を紹介する。

RS 型フリップフロップ

RS 型フリップフロップ (RS-FF) は、出力の値 Q を H レベル (1 レベル) に設定する $\overline{S}$ 端子と、L レベル (0 レベル) に設定する $\overline{R}$ 端子の 2 つ入力端子からなる基本的なフリップフロップである。 $\overline{S}$ 端子を 0 レベルにすると出力が 1 になり、1 に戻してもそのまま出力は 1 を保持し続ける。また $\overline{R}$ 端子を 0 レベルにすると出力が 0 になり、1 に戻してもそのまま出力は 0 を保持し続ける。 (「 $\overline{}$ 」は、否定を表す。記号が使えない時は記号の後に「 $\neg$ 」をつけて表すこともある。S は「セット」、R は「リセット」の意味である。)

R S型フリップフロップ回路の内部はNAND素子を2つ組合せ、お互いの出力を他方の入力とするような回路で構成されている。出力端子 $\bar{Q}$ は Q の逆の値を出力する（図23）。

74シリーズのTTLでは74279がRS-FFを4つ搭載している。

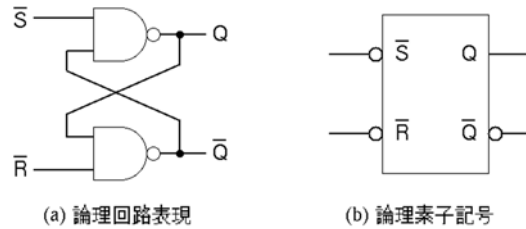


図23：RS-FF素子（正論理表現）

また、RS-FFを負論理表現すると、図24のようになる。

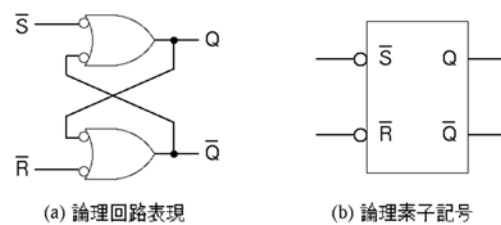


図24：RS-FF素子（負論理表現）

RS-FFは表5のように $\bar{S}$ 端子と $\bar{R}$ 端子の信号を変化させた瞬間に値を記憶する。

表5：RS-FF素子の機能表（ Q_0 は直前に覚えていた値を表す）

$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$
0	0	—	—
0	1	1	0
1	0	0	1
1	1	Q_0	$\bar{Q}_0$

D型フリップフロップ

D-FFは、クロック信号入力端子（CLK）が0から1に変化する瞬間のD端子の信号状態を記憶して出力端子 Q に出力するフリップフロップ素子である。出力端子 $\bar{Q}$ は Q の逆の値を出力する。

このようにクロック信号が0から1に変化した瞬間の値を記憶するFF素子を「**ポジティブエッジトリガFF**」という（逆に、クロック信号が1から0に変化した瞬間の値を記憶するFF素子を「**ネガティブエッジトリガFF**」という）。

また、CLR端子がある場合は、出力端子 Q の値を強制的に0に設定（クリア）することができる。その逆にPR端子は、出力端子 Q の値を強制的に1に設定（プリセット）することができる（図25）。

74シリーズのTTLでは7474が2つ、74273が8つのD-FFを搭載している。

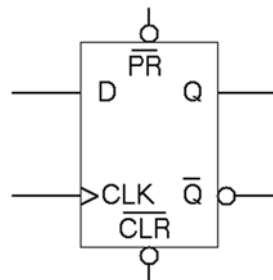


図25：D-FF素子

D-FFの動作は表6に従う。

表6：D-FF素子の機能表

(Q₀は直前に覚えていた値を表す。*は動作を保証していない)

$\overline{PR}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
0	1	x	x	1	0
1	0	x	x	0	1
0	0	x	x	1*	1*
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	x	Q ₀	$\overline{Q_0}$

J K型フリップフロップ

クロック信号の立ち下がり（負エッジ）の瞬間に、端子JおよびKの値によって定まる値を記憶し、端子Qに出力する。出力端子 $\overline{Q}$ はQの逆の値を出力する。また、CLR端子は、次のクロックの立ち下がりで強制的に出力の値をLレベル（0レベル）に設定する。PR端子は、次のクロックの立ち下がりで強制的に出力の値をHレベル（1レベル）に設定する（図26）。

74シリーズのTTLでは74112がJ K-FFを2つの搭載している。

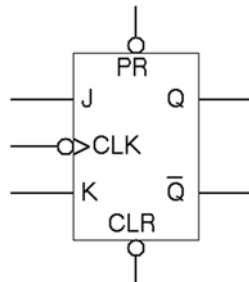


図26：J K-FF素子

J K－F F が記憶する値は表 7 に従う。

表 7：J K－F F 素子の機能表（ Q_0 は直前に覚えていた値を表す）

PR	CLR	CLK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\overline{Q_0}$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\overline{Q_0}$

T 型フリップフロップ

T はトグル(toggle)の意味で，トリガ（クロック）パルスが入力されるたびに出力 Q が反転する F F である．T－F F は入力端子として T 端子と CLK 端子を持ち，T 端子が H レベルの状態 CLK 端子に負エッジトリガが入力されると出力が反転する．T 端子が L レベルの時，出力は反転しない（図 2 7）。

T－F F は D－F F や J K－F F を使用して簡単に作製できるので，7 4 シリーズには用意されていない。

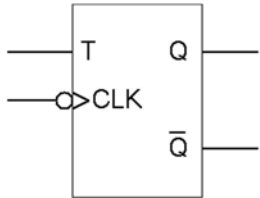


図 2 7：T－F F 素子

T－F F が記憶する値は表 8 に従う。

表 8：T－F F 素子の機能表
（ Q_0 は直前に覚えていた値を表す）

CLK	T	Q	$\overline{Q}$
↓	L	Q_0	$\overline{Q_0}$
↓	H	$\overline{Q_0}$	Q_0

ここまで説明したフリップフロップ素子を使うことで，過去に設定した値（状態）によって，出力が多彩に変化する回路を実現することができる。

シフトレジスタ

フリップフロップを応用した回路の一つに、「シフトレジスタ」がある。シフトレジスタは、RS-FF回路の出力を次段のFF回路の入力とすることにより、記憶している値を次段のFFへ次々と受け渡すことができる回路である。

74シリーズのTTLでは74LS164が8つのFFでシフトレジスタを構成しており、「8ビットシフトレジスタ」と呼ばれている。

74LS164には、通常のRS-FFと違って次段に値を受け渡すタイミングを与えるためのクロック端子があり、クロック信号が入るたびに次のFFに値を移動させる。またFFのR、S端子が正論理（1のとき有効）になっている。つまり、前段の出力端子Qと次段の入力端子S、前段の出力端子 $\overline{Q}$ と次段の入力端子Rを接続するだけでFF間のデータ受渡しが行われている（図28、29）。

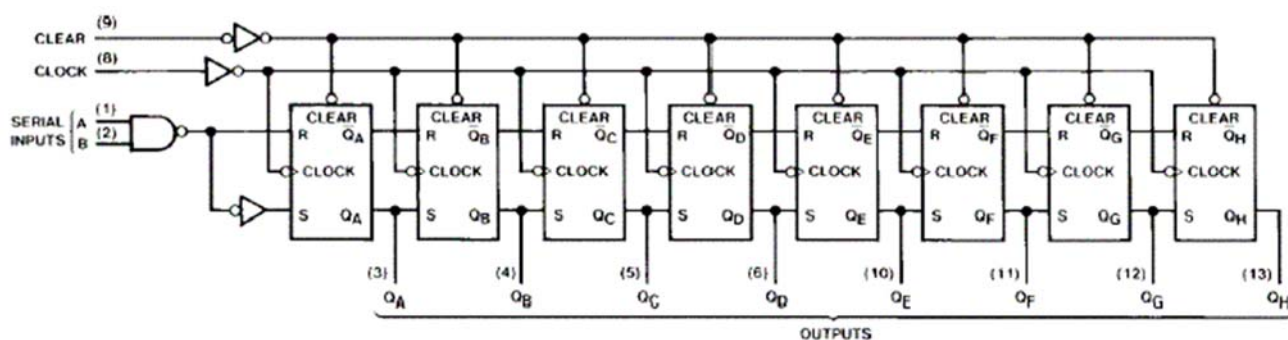


図28：74LS164の内部構造

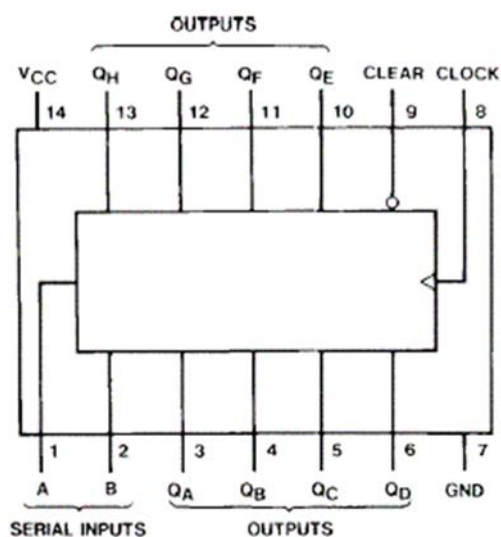


図29：74LS164のピン配置

非同期 2 進カウンタ

F F の出力値が H レベル（1 レベル）から L レベル（0 レベル）になるとき、次段の F F の値が反転するように F F を接続すると、F F に 2 進数の各桁の値（ビット）を記憶させることができる。クロックを負エッジとすることで、2 進カウンタ（計数器）が構成できる。ここで「非同期」とあるのは、各 F F への信号伝搬には時間がかかり、クロックの瞬間に同時にすべての F F の値が定まらない（同期しない）ためである。

この非同期カウンタは、0 から（2 のビット乗 - 1）までの値を計数することができ、最大値までくると次は再び 0 に戻る（実際には桁あふれが起こっている）。任意の値で 0 に戻るようにしたい時は、クリアしたい値になったら L レベル（0 レベル）を出力する組合せ回路を作製し、この出力信号をすべての F F のクリア端子に入力すればよい。このようにすると、目的の値になった瞬間にすべての F F はクリアされ、出力値はすべて L レベル（0 レベル）になる。この結果、あたかも目的の値の 1 つ前の値の次にリセットがかかったように動作する。桁が複数あるときには、この出力信号を桁上げ信号としても利用できる。

7 セグメント数字表示器

F F に記憶された 2 進数は LED 等で出力を確認できるが、人間にとっては非常にわかりにくいものである。そこで、人間にも容易にわかるように、数字で表示させることを考える。数字は 8 の字に配置した 7 つの線分（セグメント）状の LED を点灯させることで表示可能である。このような LED を 7 セグメント数字表示器という。7 セグメント表示器は LED のカソード側が 1 本にまとめられたカソードコモンタイプと、アノード側がまとめられたアノードコモンタイプがある。

本実験で使用する C-552SRD はカソードコモンタイプの 7 セグメント表示器である。概観とピン配置は図 1 を参照すること。

BCD 7 セグメントデコーダ

入力として 2 進 10 進数（BCD）の 4 つの信号 A(2⁰), B(2¹), C(2²), D(2³) を入力し、7 セグメント表示器の 7 つのセグメント LED を制御する機能をもつ回路を BCD 7 セグメントデコーダという。

74HC4511 はカソードコモンタイプの 7 セグメント表示器をドライブすることができる組合せ回路が構成されている IC である。74HC4511 を使用するときは、それぞれのセグメントの LED に対して電流制限抵抗を接続することを忘れないこと。

表 9 はよく使用するフリップフロップとそれを応用した回路を搭載した IC をまとめたものである。IC のピン配置や内部構成は規格表（データシート）を Web 等で参照すること。

表 9 : フリップフロップとその応用回路としてよく使用する主な 74 シリーズ I C

素子名	74 シリーズ型番
R S 型 F F (4 個入)	74 L S 2 7 9
D 型 F F (2 個入)	74 L S 7 4
J K 型 F F (2 個入)	74 L S 1 1 2
D 型 F F (8 個入)	74 L S 2 7 3
8 ビットシフトレジスタ	74 L S 1 6 4
2 進 + 5 進 (1 0 進), 2 進 + 8 進 (1 6 進) カウンタ	74 L S 9 0 / 9 3
2 進 + 8 進 (1 6 進) カウンタ	74 L S 2 9 3
B C D 7 セグメントデコーダ	74 H C 4 5 1 1

チャタリング

有接点スイッチをオンからオフに切替えると、スイッチレバーのわずかな振動等のため、接点の断接がわずかな間に何度も切り替わる現象が発生する。この現象を**チャタリング**といい、スイッチのオン・オフによって1つのパルスを入力したつもりでも実際には不特定多数のパルスが入力され誤動作の原因になる。特に順序回路のクロック信号では、チャタリングは絶対に避けなければならない。このようなチャタリング現象を防止して確実に1つのパルスを発生させるために F F を用いる方法がある。つまり、一度でも H レベルになったらどのような入力が入っても出力を変化しないように過去の状態を F F で記憶すればよい。J K - F F を用いたチャタリング防止回路の例を図 3 0 に示す。

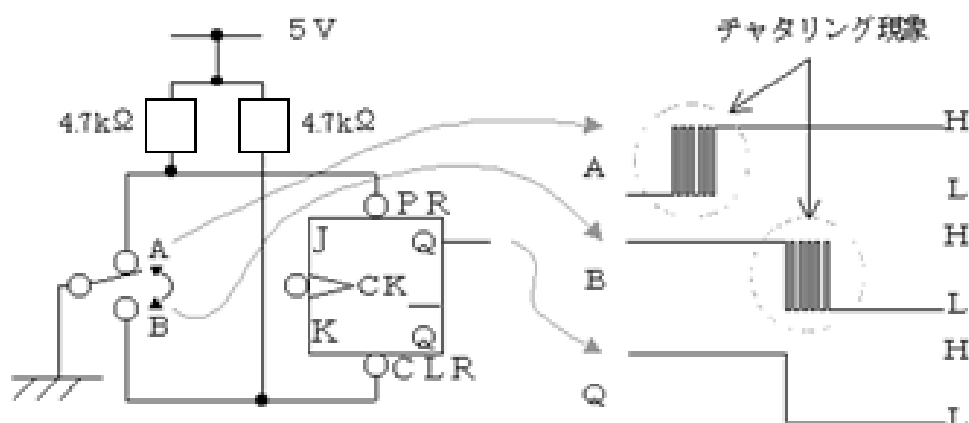


図 3 0 : チャタリング防止回路

5. 実験機材

実験に必要な部品を表 10 に示す.

表 10 : 後半の実験に必要な部品

品名	規格・型番	個数
74 シリーズ IC	適切な品番	適切な数
抵抗	LED 電流制限用 330Ω ($1/8W$)	適切な数
抵抗	プルアップ用 $4.7k\Omega$ ($1/8W$)	適切な数
LED	赤	適切な数
7 セグメント表示器	赤	適切な数
スイッチ	トグル型, DIP 型	適切な数
電線	各色	適切な数

6. 実験方法

これまでと同様に, 実験 6 ~ 12 のデジタル回路 (順序回路) をこの順で設計せよ.

実験 6

トグルスイッチ 1 つと JK-FF (74LS112 の中の 1 つ) を使ってチャタリング防止回路を作製し, その動作を LED の点灯・消灯によって確認せよ. このとき, スイッチをどちらに倒したらチャタリング防止回路の出力が H レベル, または L レベルになる時の点灯・消灯を記録せよ (表 11).

表 11 : 実験 6 の動作結果

スイッチの位置	スイッチの位置に対する入力論理値	LED の点灯/消灯	LED に対応する出力論理値

実験 7

74LS273 (8 回路ポジティブエッジトリガ D-FF, 図 31) の 8 つの D-FF を 2 進数における各桁の記憶素子と考え, 教員から指定された 0 から 127 までの 10 進数を入力し, 記憶させる回路を作製し, その動作を確認せよ. (表 12).

表 1 2 : 教員から指定された値 (予定 : 学籍番号の下 3 桁)

1 0 進数	2 進数

1. 各 D-FF の D 端子への 2 進数入力信号を DIP スイッチ (1 列に端子が並んだ小さなスイッチ群) とプルアップ抵抗を使用せよ.
2. 入力するクロック信号はトグルスイッチとプルアップ抵抗による信号に実験 6 で作製したチャタリング防止回路を接続した回路として構成せよ.
3. まず, $\overline{\text{CLR}}$ 端子に接続されたリセットスイッチを操作し, 8 つの D-F F に一旦 0 を記憶させよ. (LED がすべて点灯)
4. 次に, チャタリング防止回路のトグルスイッチを動かすことでクロック信号を入力し, 各 FF に記憶されている値を LED の点灯・消灯で確認せよ.

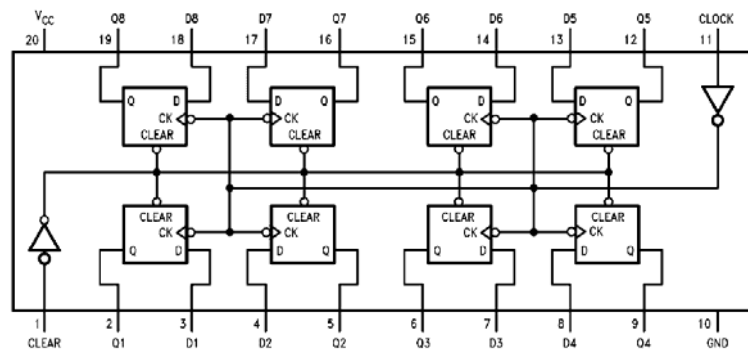


図 3 1 : 74LS273 のピン配置

実験 8

ネガティブエッジトリガ JK-FF 74LS112 を使って T-FF を実現する回路を作製し, その動作を確認せよ. クロックには実験 6 で作製したチャタリング防止回路の出力を用いよ.

実験 9

非同期式カウンタ 74LS293 を使用して 16 進カウンタを作製し, その動作を 7 セグメント表示器 C-552SRD で値を表示させて確認せよ (74LS293 は 74LS93 のピン配置を変えたものである. 図 3 2).

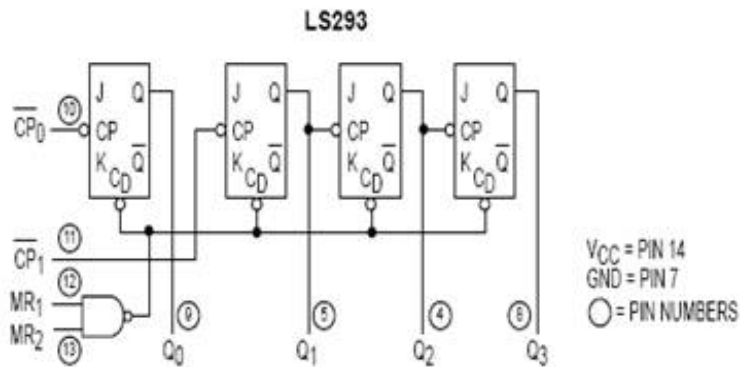
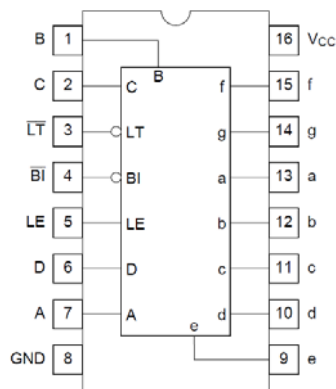


図 3 2 : 74LS293 のピン配置

- カウンタの 4 ビット BCD 出力に 7 セグメントデコーダ 7 4 H C 4 5 1 1 (図 3 3) を接続し, 7 セグメント表示器 C-552SRD の右側 1 桁に数字 1 文字を表示する回路を作製せよ. (7 4 H C 4 5 1 1 の入力端子 LE は 0, $\overline{\text{BI}}$ は 1, $\overline{\text{LT}}$ は 1 に固定せよ)
- カウンタのクロックに実験 6 で作製したチャタリング防止回路の出力を接続し, 1 6 進カウンタの動作を確認せよ.
(特に, 1 6 進数 A から F まではどのような表示になるかも記録せよ)
- チャタリング防止回路の代わりに直接トグルスイッチを使ったオンオフパルスを与えると回路にチャタリング現象が発生する. トグルスイッチの出力を直接クロック信号に接続し, チャタリングがどのように起こったか記録せよ. (チャタリング防止回路の PR 端子または CLR 端子は直接トグルスイッチから出力されるオンオフパルスなので, チャタリングが含まれた入力信号としてそのまま流用できる)
- また, チャタリング防止回路の代わりに, ファンクションジェネレータの T T L 出力を接続し, 高速に動作させた場合でも回路が安定して動作することを確認せよ.



入力							出力							
LE	$\overline{BI}$	$\overline{LT}$	D	C	B	A	a	b	c	d	e	f	g	表示
X	X	L	X	X	X	X	H	H	H	H	H	H	H	8
X	L	H	X	X	X	X	L	L	L	L	L	L	L	blank
L	H	H	L	L	L	L	H	H	H	H	H	H	L	0
L	H	H	L	L	L	H	L	H	H	L	L	L	L	1
L	H	H	L	L	H	L	H	H	L	H	H	L	H	2
L	H	H	L	L	H	H	H	H	H	H	L	L	H	3
L	H	H	L	H	L	L	L	H	H	L	L	H	H	4
L	H	H	L	H	L	H	H	L	H	H	L	H	H	5
L	H	H	L	H	H	L	L	L	H	H	H	H	H	6
L	H	H	L	H	H	H	H	H	H	L	L	L	L	7
L	H	H	H	L	L	L	H	H	H	H	H	H	H	8
L	H	H	H	L	L	H	H	H	H	L	L	H	H	9
L	H	H	H	L	H	L	L	L	L	L	L	L	L	blank
L	H	H	H	L	H	H	L	L	L	L	L	L	L	blank
L	H	H	H	H	L	L	L	L	L	L	L	L	L	blank
L	H	H	H	H	H	L	L	L	L	L	L	L	L	blank
L	H	H	H	H	H	H	L	L	L	L	L	L	L	blank
H	H	H	X	X	X	X	(注)							(注)

【注】 LE = L のときに印加された BCD コードによって決まる。

図 3 3 : 7 4 H C 4 5 1 1 のピン配置と機能表

実験 1 0

実験 9 で作製した 1 6 進カウンタに回路を追加して、1 0 進カウンタを作製し、その動作を確認せよ。(1 0 までクロックパルスを数えたらすぐに全ての記憶を 0 にクリアする回路を追加する)

実験 1 1

実験 1 0 で作製した 1 0 進カウンタを 2 つ作製し、非同期 B C D 6 0 進カウンタを作製せよ。また、同様にして非同期 2 4 進カウンタを作製し、その動作を確認せよ。

実験 1 2

実験 1 0 で作製した非同期 B C D 6 0 進カウンタで「分」を、同様に作製した非同期 B C D 2 4 進カウンタで「時」を表すことで、非同期 2 4 時間時計（時分表示）を作製し、その動作を確認せよ。

7. 応用課題（時間がある場合に挑戦するとよい）

1 2 時間時計を実現する回路の回路図を示し，実現せよ．（24 時間時計と同様の設計では動作しない．ヒント：12 時の次は 0 時ではなく 1 時である．）