情報工学実験 I

基本ゲート回路

担当教員：吉田たけお

実験実施日：６月２２日

氏名：大城健太 (065712D)

共同実験者

祝三志郎 (065706K)

仲栄真言祈 (065744B)

山内遼史 (065760D)

実験提出締切日６月２９日

1 実験目的

・現代社会に欠かすことのできないコンピュータは、大規模なディジタル回路として構成されている。本実験では、ディジタル回路の構成要素である基本ゲート回路と論理演算の基礎を習得することを目的とする。また本実験では、NAND ゲートを用いて他のゲート回路 (NOT,AND,OR,NOR,XOR) を構成することによって、汎用ロジック IC およびブレッドボード、直流電流などの基本的な使用法についても学ぶ。

2 実験概要

・4 人 1 組のグループに分かれて実験を行う。まず紙に NAND ゲートだけを用いて他の基本ゲート (NOT,AND,OR,NOR,XOR) を設計する。このとき、論理式やド・モルガンの定理を使って設計する。

・設計ができたら、設計した各ゲート回路をブレッドボード上に IC や配線、直流電源などを用いて実現する。ここで、IC は NAND ゲートだけで構成されている 4000B シリーズ IC の 4011B を用いる。

・ブレッドボードに設計ができたら回路が正しく設計できているかを確かめる。確かめ方は、直流電流を流して電球の明かりがつくかつかないかで判断する。出力が'1' の場合は電球の明かりがつき、出力が'0' の場合は電球の明かりがつかないのでそれを利用する。回路が上手く設計できていなければ、できるように設計をし直す。確認が正しければ、片付けをして実験を終了する。

3 実験結果

- 実施した実験項目

（1）NAND ゲートのみを用いて、NOT,AND,OR,NOR,XOR ゲートを設計せよ（NAND ゲートのみで構成された回路図を描け）

（2）実験（1）で設計した各ゲート回路を実際に NAND ゲート IC を用いてブレッドボード上に実現し，それらの動作を確認せよ。

3.1 (1) の実験結果

3.1.1 NAND ゲートのみを用いて実現された、NOT,AND,OR,NOR,XOR ゲートのそれぞれの回路図を示せ。

図3.1 NOTの回路図

図3.2 ANDの回路図

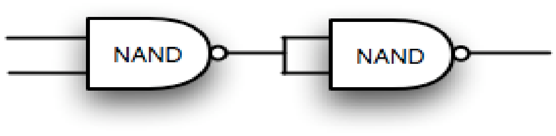


図3.3 ORの回路図

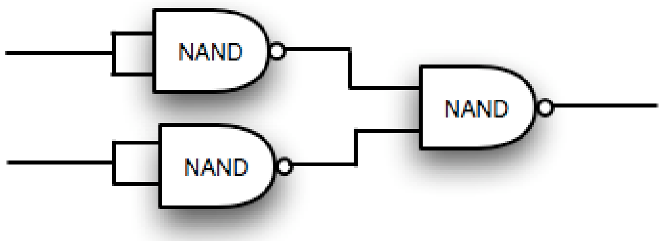


図3.4 NORの回路図

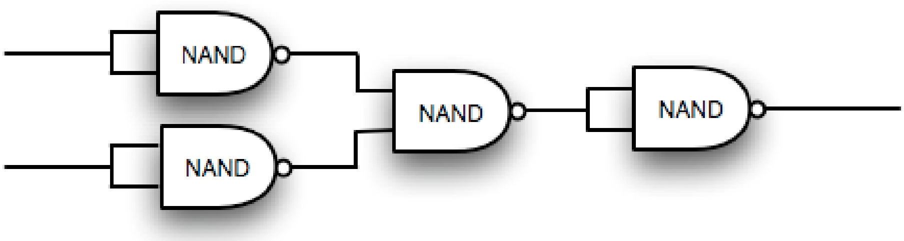
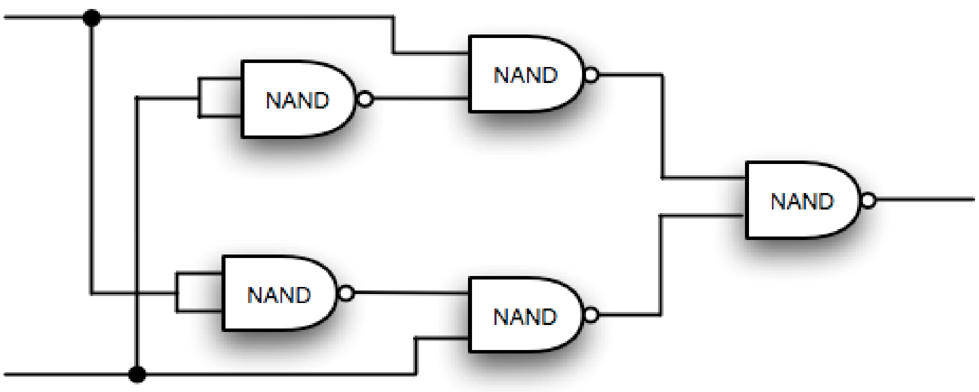


図3.5 XORの回路図



3.1.2 上記の各回路図が、それぞれ、NOT,AND,OR,NOR,XOR ゲートの機能を実現していることを説明せよ。

・上記の各回路図がそれぞれ各ゲートの機能を実現しているか真理値表を用いて説明する。

表3.1

A	f
0	0
1	1

・表 3.1 は上記の図 3.1 の回路図の真理値表を表している。入力を A, 出力を f とすると、この真理値表は、NOT ゲートの真理値表と同じである。これより、上記の図 3.1 は NOT ゲートの機能を実現していることが分かる。

表3.2

A	B	f
0	0	0
0	1	0
1	0	0
1	1	1

・表 3.2 は上記の図 3.2 の回路図の真理値表を表している。入力を A,B, 出力を f とすると、この真理値表は、AND ゲートの真理値表と同じである。これより、上記の図 3.2 は AND ゲートの機能を実現していることが分かる。

表3.3

A	B	f
0	0	0
0	1	1
1	0	1
1	1	1

・表 3.3 は上記の図 3.3 の回路図の真理値表を表している。入力を A,B, 出力を f とすると、この真理値表は、OR ゲートの真理値表と同じである。これより、上記の図 3.3 は OR ゲートの機能を実現していることが分かる。

表3.4

A	B	f
0	0	1
0	1	0
1	0	0
1	1	0

・表 3.4 は上記の図 3.4 の回路図の真理値表を表している。入力を A,B, 出力を f とすると、この真理値表は、NOR ゲートの真理値表と同じである。これより、上記の図 3.4 は NOR ゲートの機能を実現していることが分かる。

表3.5

A	B	f
0	0	0
0	1	1
1	0	1
1	1	0

・表 3.5 は上記の図 3.5 の回路図の真理値表を表している。入力を A,B, 出力を f とすると、この真理値表は、XOR ゲートの真理値表と同じである。これより、上記の図 3.5 は XOR ゲートの機能を実現していることが分かる。

3.1.3 上記の各回路図が導かれた経緯を説明せよ。

・上記の各回路図が導かれた経緯を論理式などを用いて説明する。

・NOT の論理関数を NAND のみで表すと次のようになる。このとき、入力を A、出力を f とする。

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

これを回路図で表すと、図 3.1 が導かれる。

・AND の論理関数を NAND のみで表すと次のようになる。このとき、入力を A,B、出力を f とする。

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

これを回路図で表すと、図 3.2 が導かれる。

・OR の論理関数を NAND のみで表すと次のようになる。このとき、入力を A,B、出力を f とする。

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

これを回路図で表すと、図 3.3 が導かれる。

・NOR の論理関数を NAND のみで表すと次のようになる。このとき、入力を A,B、出力を f とする。

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

これを回路図で表すと、図 3.4 が導かれる。

・XOR の論理関数を NAND のみで表すと次のようになる。このとき、入力を A,B、出力を f とする。

$$f = A \oplus B = A \boldsymbol{\cdot}\overline{B} + \overline{A}\boldsymbol{\cdot}\, B = \overline{\overline{A \boldsymbol{\cdot}\overline{B} + \overline{A}\boldsymbol{\cdot}\, B}} = \overline{\overline{A \boldsymbol{\cdot}\overline{B}\boldsymbol{\cdot}\overline{A}\boldsymbol{\cdot}\, B}}$$

これを回路図で表すと、図 3.5 が導かれる。

3.2 (2) の実験結果

3.2.1 NAND ゲート IC を用いてブレッドボード上に実現した各回路が、それぞれ、NOT,AND,OR,NOR,XOR ゲートと等価な動作をしたか否かを報告せよ。

・最初の回路をブレッドボード上に実現するのは、配線ミスなどがあったりして手こずったが、最初の回路を実現した後はミスも少なくなり、実験を円滑に進めることができた。最後に実現した XOR 回路は配線の数が多くなったが、なんとか実現することができた。これより、ブレッドボード上に実現した各回路がそれぞれ、NOT,AND,OR,NOR,XOR ゲートと等価な動作をすることができた。

4 考察

- 上記の実験結果に基づいて、以下の点について詳しく述べよ。

4.1 実験（1）の考察について

4.1.1 上記の実験結果で報告した回路よりも、使用する NAND ゲートの数を減らすことが可能かどうかを、NOT,AND, OR,NOR,XOR ゲートのそれぞれについて考察せよ。

- NOT ゲートについて

・NOT ゲートは、一つの NAND ゲートで構成されているのでこれ以上 NAND ゲートの数を減らすことは不可能である。

- AND ゲートについて

・AND ゲートは、NAND ゲートの数が 2 つで実現できる。これは、NAND ゲートを NOT で否定することによって AND ゲートの機能を実現しているためであるため、これ以上 NAND ゲートを減らすことは不可能である。また、上記で表した論理式を簡単化しようとしてもできないことからも NAND ゲートを減らすことが不可能であることがわかる。

- OR ゲートについて

・OR ゲートは、NAND ゲートの数が 3 つで実現できる。これは、NOT で入力の値を否定して NAND ゲートに入力することによって、OR ゲートの機能を実現しているためであるため、これ以上 NAND ゲートを減らすことは不可能である。また、上記で表した論理式を簡単化しようとしてもできないことからも NAND ゲートを減らすことが不可能であることわかる。

- NOR ゲートについて

・NOR ゲートは、NAND ゲートの数が 4 つで実現できる。これは、OR ゲートを NOT で否定することによって NOR ゲートの機能を実現しているため、これ以上 NAND ゲートを減らすことは不可能である。また、上記で表した論理式を簡単化しようとしてもできないことからも NAND ゲートを減らすことが不可能であることがわかる。

- NOR ゲートについて

・XOR ゲートは、NAND ゲートの数が 5 つで実現できる。しかし、下のように論理式を簡単化することによって、NAND ゲートの数が 4 つで実現できる。

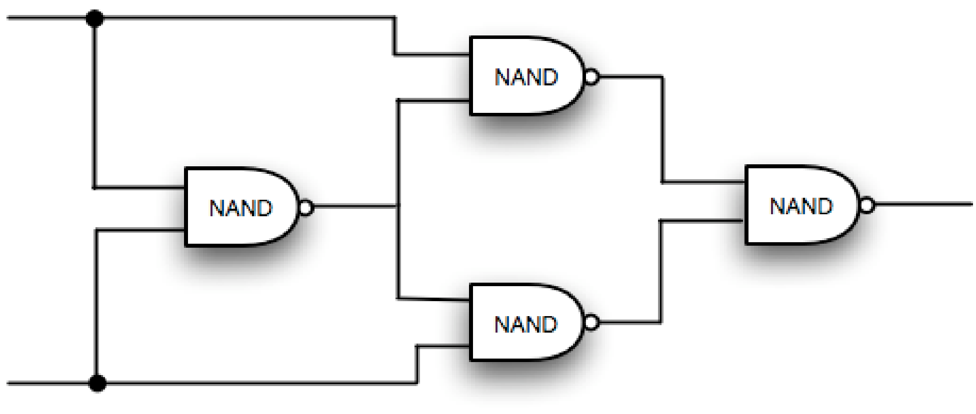
$$f = A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B = \overline{A} \cdot A + A \cdot \overline{B} + \overline{A} \cdot B + \overline{B} \cdot B = (\overline{A} + \overline{B}) \cdot (A + B)$$

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

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

これを回路図に表すと次のようになる。

図4.1 NANDゲート４つのXORゲート



4.2　実験（2）の考察について

4.2.1　ブレッドボード上で回路を構成する際に、配線ミスを減らすためには、配線数そのものを減らすことが望ましい。配線数を減らすための工夫について考察せよ。

・今回の実験だと回路を実現する順番を考えると良い。例えば、NOT ゲートを実現した後に AND ゲートを実現したり、OR ゲートの後に NOR ゲートを実現したりする。これは NOT ゲートをそのまま次の AND ゲートに利用することによって、配線の数をなくすこともできるし、配線を始めからし直す必要もなくなるなるので、実験を円滑に進めることもできる。OR ゲートのときも同様である。

4.2.2　配線数を減らす以外の配線ミスを減らすための工夫について考察せよ。

・配線ミスを減らすようにするために、入力や出力などの抜き差しの回数が多くなるような場所は、作業しやすいようにスペースを空けておくと良い。また、配線の長さが余らないように適した長さの配線を使うことによって複雑にならずに配線が扱いやすくなる。

4.3　その他の実験について

4.3.1　本実験を通して得られた新たな知見について詳しく説明せよ。

・和田先生のディジタル回路の授業を受けているので、ある程度回路の知識を持っていました。ですが、ディジタル回路では、実験などは行わずに知識の方だけを習っていました。したがって、今回の実験のように実際に基本ゲートなどを実現することは、今まで習ってきた知識を活かすこともできるし、実現することによって新しい知識を得ることもできました。例えば、ブレッドボードや IC の使い方、配線ミスが起こらないようにする工夫など、実際に作業しなければ得ることのできない知識を知ることができました。

5　調査課題

- 下記の各項目について調査し、その結果を報告せよ。

5.1　（a）2 変数の論理関数は全部で 16 種類ある。何故 16 種類になるか説明せよ。また、2 変数の論理関数を 16 種類すべて列挙し、否定 (NOT), 論理積 (AND), 論理和 (OR) のみを用いて表現せよ。

・2 変数論理関数は、'0' と'1' の 2 つの値をとる変数を 2 つもった関数である。これらの変数がとり得る値の組み合わせは、全部で 2^2=4 通りである。また、2 変数論理関数は、この 4 通り全ての場合に対して、'0' または'1' の出力が出る。これより、2 変数論理関数は全部で 2^4=16 種類あることになる。この 16 種類の論理関数を表 5.1 の真理値表に示し列挙する。

表5.1　2変数論理関数の種類

A	B	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

f0(A,B) ＝ 0　・入力 A,B の値に関わらず, 常に 0 を出力する。
f1(A,B) ＝ A ・ B　・論理積 (AND)
f2(A,B) ＝ $A \cdot \overline{B}$
f3(A,B) ＝ A　・入力 B の値に関わらず, 常に A の値をそのまま出力する。
f4(A,B) ＝ $\overline{A} \cdot B$
f5(A,B) ＝ B　・入力 A の値に関わらず, 常に B の値をそのまま出力する。
f6(A,B) ＝ $A \cdot \overline{B} + \overline{A} \cdot B = A \oplus B$　・排他的論理和 (XOR)
f7(A,B) ＝ A ＋ B　・論理和 (OR)
f8(A,B) ＝ $\overline{A + B}$　・論理和否定 (NOR)
f9(A,B) ＝ $A \cdot B + \overline{A} + \overline{B} = \overline{A \oplus B}$　・排他的論理和否定 (XNOR)
f10(A,B) ＝ $\overline{B}$　・入力 A の値に関わらず, 常に B の値の否定を出力する。
f11(A,B) ＝ $A + \overline{B}$
f12(A,B) ＝ $\overline{A}$　・入力 B の値に関わらず, 常に A の値の否定を出力する。

f13(A,B) $= \overline{A} + B$

f14(A,B) $= \overline{A \cdot B}$ ・論理積否定 (NAND)

f15(A,B) $= 1$ ・入力 A,B の値に関わらず, 常に 1 を出力する。

5.2 (b）今回の実験で学んだように、論理関数の表現は一意ではなく複数（厳密には無数に）ある。実際に回路を実現する場合は、それらの関数の中で, できるだけ簡単な表現を採用すべきである。そのような簡単な論理関数表現を求める際、カルノー図と呼ばれる図表が用いられる事がある。カルノー図とはどのような図表か調査し報告せよ。また、カルノー図の使用方法を具体例を用いて説明せよ。

・カルノー図とは、真理値表を升目を用いて表したものであり、論理回路などにおいて論理式を簡単化するための表である。カルノー図は、図 5.1 のような図のことである。

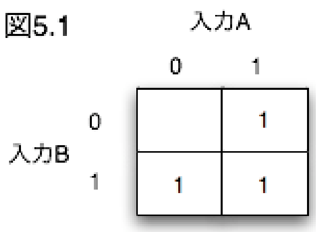


図5.1

・具体的な例として下の論理関数をカルノー図を用いて簡単化する。

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

この論理関数をカルノー図を用いて表すと図 5.2 のようになる。

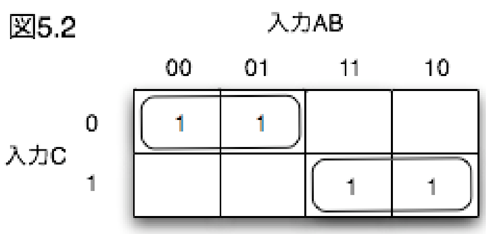


図5.2

これは、ハミング距離が 1 となるように図が組まれている（図の入力欄の隣同士の真偽値が 1 つだけ違うように組まれている)。ハミング距離が 1 のとき、その論理式は簡単化できる。図 5.2 では、$\overline{A} \cdot \overline{B} \cdot \overline{C}$ と $\overline{A} \cdot B \cdot \overline{C}$、$A \cdot B \cdot C$ と $A \cdot \overline{B} \cdot C$ のハミング距離が 1 となっているのでそれぞれ簡単化できる。したがって、下のような論理関数となる。

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

カルノー図はこのようにして使い、論理関数を簡単化する。

5.3 (c) 実用的なディジタル回路として半加算器および全加算器がある。これらの回路はどのような回路か調査し、それぞれの真理値表を示せ。また、各真理値表を用いて、それらの機能や特徴を説明せよ。さらに、それぞれの回路図の一例と、各回路図が実現している論理関数を示せ。

5.3.1 半加算器

・半加算器とは 1 ビットと 1 ビットの加算を行う回路である。入力は 2 つ、出力はそのビット出力（Sum) と桁上り（Carry）の 2 つとなる。真理値表とその回路図は、次のようになる。このとき、入力を A,B、出力を S, 桁上げ出力を C とする。

表5.2 半加算器の真理値表

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

図5.3 半加算器

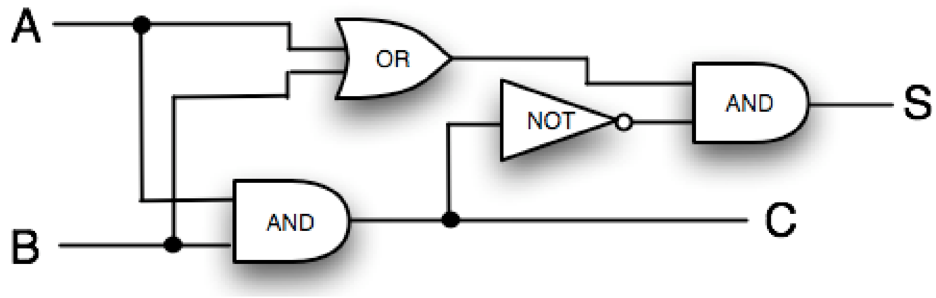


表 5.2 の真理値表を見てみる。S の真理値は XOR に等しく、C の真理値は AND に等しいことがわかる。よって、これらの論理関数は下のようになる。

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

$$C = A \cdot B$$

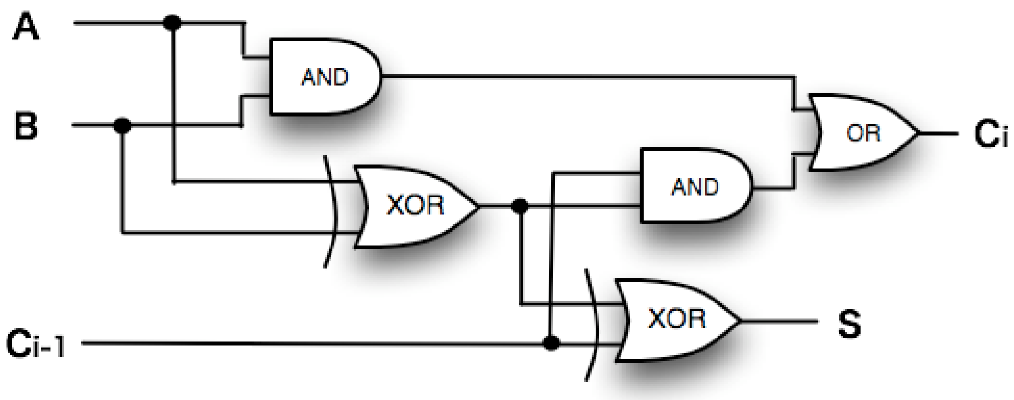
5.3.2 全加算器

・全加算器は、2 進数の 1 つの桁を演算し、下位からの繰り上がりを含めて出力する。下位からの桁上がりを上位の桁上げ入力に接続することによって、任意の桁数の 2 進数の加算ができる。真理値表とその回路は、次のようになる。このとき、入力を A,B、出力を S, 桁上げ入力を C_{i-1}, 桁上げ出力を C_i とする。

表 5.3 全加算器の真理値表

A	B	Ci-1	S	Ci
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5.4 全加算器



S と C_i の論理関数は次のようになる。

$$S = \overline{A}\cdot B \cdot\overline{C_{i-1}} + A \cdot\overline{B}\cdot\overline{C_{i-1}} + \overline{A}\cdot\overline{B}\cdot C_{i-1} + A \cdot B \cdot C_{i-1}$$

$$C_i = A \cdot B + A \cdot C_{i-1} + B \cdot C_{i-1}$$

6　感想

・今回の実験は、回路を実現するのに配線ミスなどがあったりして、試行錯誤の連続でした。しかし、その分配線が正しくできていて電球が光ったときは達成感がありました。この実験で、実際に回路を実現することによって紙上ではわからないことを体験できたと思います。今回の実験の意見としては、ブレッドボードに回路を実現する前に、先生方が見本を見せるとよりわかりやすかったと思います。私たちはブレッドボードや IC を使用するのが初めての人が多いと思うので、口頭の説明の後にすぐ実行に移るのは少し厳しいかなと思いました。実際に口頭の説明の後、実行したけど、どうして良いかわからずに戸惑っているグループが多かったように見えました。簡単な例としては、NAND 回路を実現することでブレッドボードや IC、直流電源の使用方法を見せると良いと思います。

7　参考文献・参考ページ

・VHDL で学ぶディジタル回路 (吉田たけお・尾知博　共書)CQ 出版社、2002 年 4 月 20 日初版発行

・wikipedia
http://ja.wikipedia.org/wiki/メインページ

・琉球大学　和田先生のディジタル回路のページ
http://www.ie.u-ryukyu.ac.jp/~wada/lecture.html