知能情報実験Ⅰ　:　第七回レポート

175751C 宮城孝明

平成 30 年 6 月 4 日

目 次

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している。また本実験では,NAND(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, 直流電源などの基本的な使用法についても学ぶ。

2　実験概要

今回の実験は, ディジタル回路の基本構造を理解するために, 実際の回路作成を行った。まず始めに,NAND だけを用いて NOT,AND,OR,NOR,XOR 回路を紙の上で回路図として表現した。回路作成する際に, 全体の見取り図として, これを役立てた。そして, 回路作成にあたり教授たちが前で基本的な説明や取り扱い際の諸注意をした。あらかじめ用意してもらった汎用ロジックとブレッドボード, 直流電源を用いて, 先ほどの回路を作成する作業に移る。そして, 出力結果を確かめるために,LED の灯が付くか付かないで確認をした。そのため, 私たち学生でも自作の回路が適切かどうか判断できる。さらに, 教授や院生, 外部の人たちが学生の質問に回って答えたり, 実際の動作を一緒に確認した。

3　実験結果

実験 (1)

ここは,NAND ゲートを用いて NOT,AND,OR,NOR,XOR ゲートを再現する。

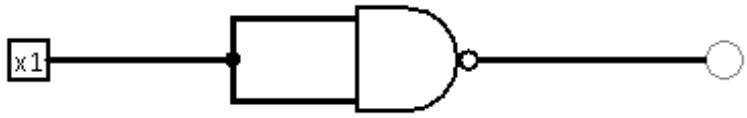

図 1: NAND ゲートで示した NOT 回路

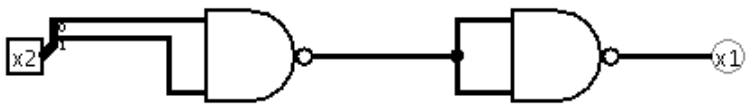

図 2: NAND ゲートで示した AND 回路

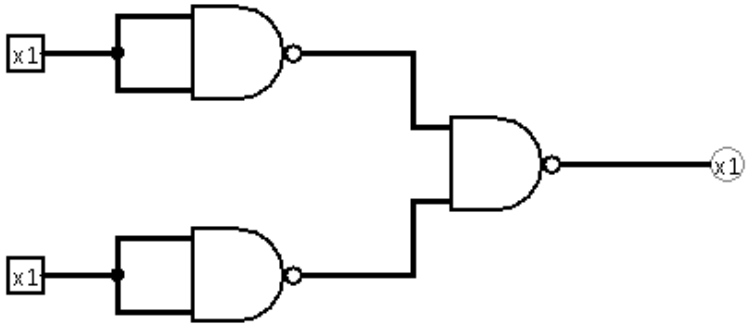

図 3: NAND ゲートで示した OR 回路

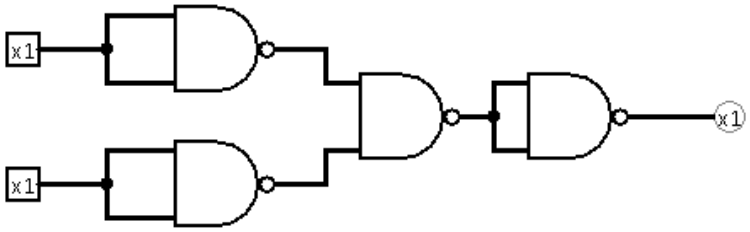

図 4: NAND ゲートで示した NOR 回路

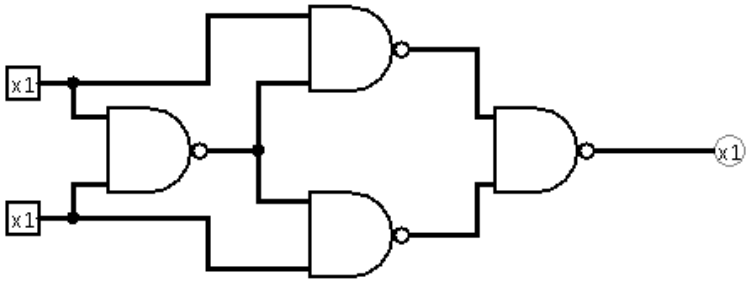

図 5: NAND ゲートで示した XOR 回路

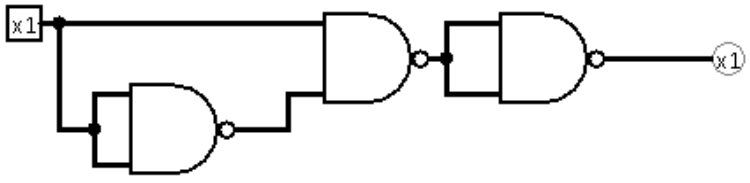

図 6: NAND ゲートで示した常に 0 を出力する回路

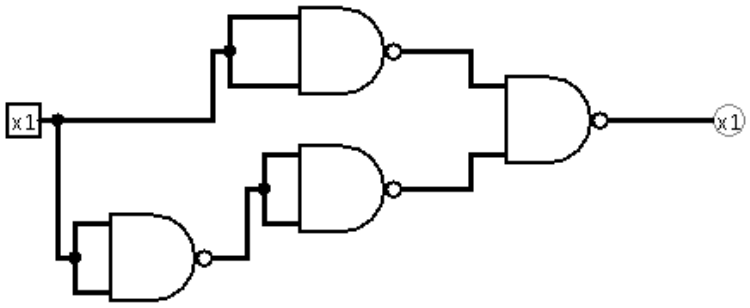

図 7: NAND ゲートで示した常に 1 を出力する回路

図 1 は,NOT 回路となっている。NOT 回路は, 出力結果が入力の値を否定になる。そして,NAND 回路は入力 (0,0) の場合は 1 を出力, 入力 (1,1) の場合は 0 を出力する。そのため, 一度入力を二つに分ける。こうして, 入力に対してその否定を出力する NOT 回路が完成する。

この回路図に初めに 0 を入力する。入力を 2 分にして,NAND に入れる。そのため,NAND 回路には入力 (0,0) が入力されたことになる。よって、出力結果は,0 となる。同様に,1 を入力したら出力は 0 になる。

図 2 は,AND 回路となっている。AND 回路は, 入力 (1,1) の時だけに出力 1 を返す。つまり,NOND 回路を NOT 回路で否定した回路になることが予測できる。よって, 初めに NAND 回路に値を入力し, その出力を 1 番目に作成した NOT 回路で否定することで AND 回路は完成する。

この回路に (0,0),(1,0),(0,1) を入力したら最初の NAND 回路で 1 の値になる。それを次の NOT 回路で否定することで出力結果は,0 となる。(1,1) という入力では最初の NAND 回路で 0 の値になり、NOT 回路で 1 という出力になる。よって、AND 回路の完成になる。

図 3 は,OR 回路になる。OR 回路は入力値 (0,0) 以外は 1 を返すという回路である。つまり, 初めの入力値を否定しそれを NAND 回路で出力することで OR 回路は完成する。

この回路に,(1,1),(1,0),(0,1) を入力する。始めの否定回路により, 全ての値は反対になる。その際, 値に（1,1）はない。そのため, この 3 入力は出力結果は必ず 1 が返ってくる。しかし、入力値を (0,0) した場合, 否定回路で (1,1) となる。これにより, 出力結果は 0 が返る。

図 4 は,NOR 回路である。この回路は,OR 回路の否定であるため, 先ほど作成した OR 回路に否定の NOT 回路を付け加えれば完成する。

OR 回路の否定ため,2 つ目の NAND 回路で値が 1 になったり 0 になったりたのを否定する。この回路に,(1,1),(1,0),(0,1) を入力したら 0 を返し,(0,0) の入力値に対しては 1 に返す。

図 5 は,XOR 回路である。入力値を NAND 回路を取り入れ, その値を初

めの値たちで NAND を取り, さらにその値を NAND を取ることによって, 完成する。

この値に (1,0),(0,1) を入力すれば, 始めの NAND 回路で 1 を出力する。そして, 次の NAND 回路で片方は 0 をもう片方は 1 を出力する。これにより, 最後の NAND 回路で値は 1 を出力する。しかし,(1,1),(0,0) を入力したら, 始めの NAND 回路で入力値の値は反対になる。それにより,2 つ目の NAND 回路両方とも 1 を出力する。そして, 最後の NAND 回路で 0 が出力される。

図 6 は, 常に出力値 0 になる回路図である。そのためには, 入力値とその値の補集合の論理積を取れば常に 0 を取る。

この回路図にある値を入力すれば, 始めの否定の回路で反対の値はとなる。あとは, その値と始めの入力値の AND 回路を取ることで, 値は必ず 0 を取る。

図 7 は, 常に出力値 1 になる回路図である。そのためには, 始めの入力値とその補集合の論理和を取れば常に 1 を取る。

この回路図にある値を入力し, その否定を取る。そして, 始めの入力値とその否定値を OR 回路に入力する。これにより, 値は常に 1 を出力する。

実験 (2)

正常に動作した。

実験 (3)

正常に動作した。

実験 (4)

図 8: NOR ゲートで示した NOT 回路

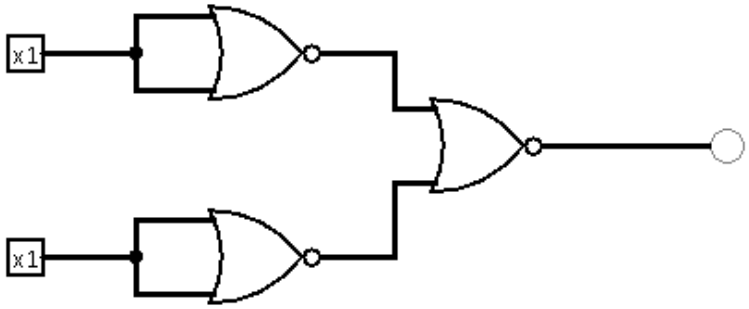

図 9: NOR ゲートで示した AND 回路

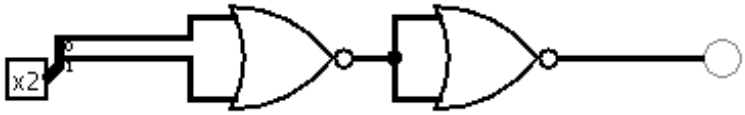

図 10: NOR ゲートで示した OR 回路

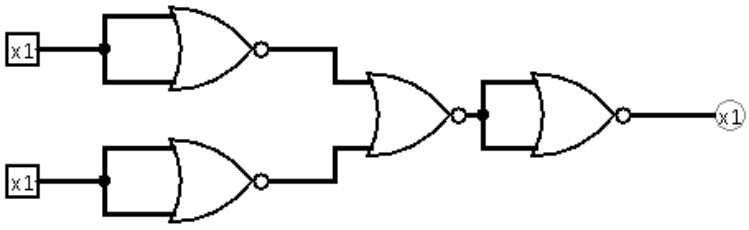

図 11: NOR ゲートで示した NOND 回路

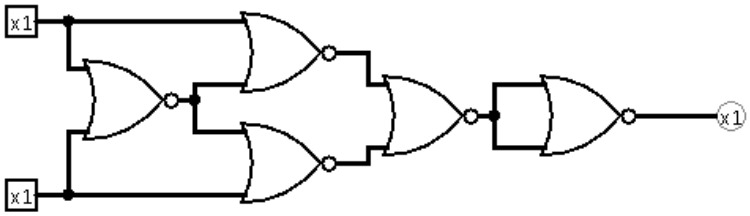

図 12: NOR ゲートで示した XOR 回路

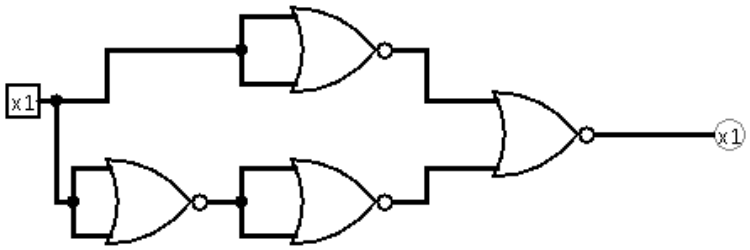

図 13: NOR ゲートで示した常に 0 を出力する回路

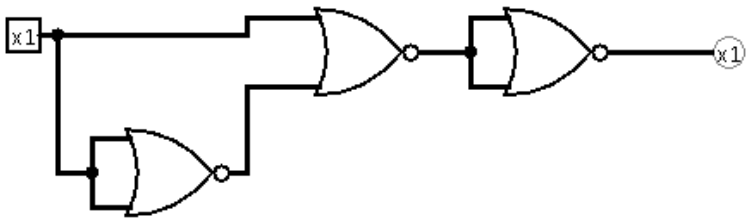

図 14: NOR ゲートで示した常に 1 を出力する回路

図 8 は,NOR を用いて NOT 回路を再現した。これも図 1 と同様にある入力値の否定を取るため, ある入力値を 2 つに分け NOR 回路に入れる。これにより,NOR 回路には同じ値の入力値が入り, そしてその値の否定を取る。

この回路に 1 を入れる。そうすれば,(1,1) が入力され,OR 回路の否定を取ることにより, 結果として出力は 0 となる。また,0 を入力すると結果は 1 となる。

図 9 は,NOR を用いて AND 回路を表現した。AND を表現するためには,2 つの入力値をお互いに否定を取る。それらをさらに,NOR に入力されることで AND 回路を表現する。2 つの入力値 (1,1) を入力したら初めの NOR によって、否定され出力結果は 0 となる。これにより, 最後の NOR に入力される入力値は (0,0) となり, これらを NOR に入力されたら出力結果は 0 となる。

図 10 は,NOR を用いて OR 回路をを再現した。OR 回路は,NOR 回路を否定することでできる。2 つの入力を NOR に入れ, その出力値を否定することで,OR 回路は完成する。

この入力値 (1,0),(0,1),(1,1) を入力することで始めの NOR 回路によって, よって,0 の出力になる。そして, その値を否定することで, 出力結果は 1 になる。(0,0) の場合は,NOR 回路によって 1 になり, 否定することで 0 と出力する。

図 11 は,NOR を NAND 回路に表現している。これは, 図 9 で作成した AND 回路を図 8 で作成した否定回路を繋げることで完成する。

入力値 (1,1) を入れると,2 つ目の NOR 回路で 1 と出力するそれを最後の NOR で否定し,0 と返す。(0,0),(1,0),(0,1) を入力値にすると,0 を返す。

図 12 は,NOR で表現した XOR 回路になる。これは,AND 回路と始めの入力値の NOR で出力した値をさらに,NOR に入れることで,XOR 回路ができる。

入力値 (1,0),(0,1) を入れると AND 回路では 0 と出力し, 始めの入力値を NOR に入れた値は,0 を返す。そして, 最後の NOR に入れることによって, 値は 1 を出力する。次に, 入力値を (0,0),(1,1) にすることで,(0,0) では 0 を (1,1) では 1 を出力する。そして, 始めの入力値を NOR 回路に入れて,(0,0) は 1 を (1,1) は 0 を出力する。そして, 最後の NOR で出力結果が 0 になる。

図 13 は, 常に 0 を出力する回路である。この回路を導くために, 論理式で入力値とその否定値の論理積を書くことによって, この回路は完成する。

この回路にある値を入れ, その否定値を出します。始めの入力値と否定値を AND 回路に入れることで, この回路は常に 0 を出力する回路となる。

図 14 は, 常に 1 を出力する回路である。この回路を導くために, 論理式で入力値とその否定値の論理和を書くことによってこの回路は完成する。

ある値を入力すると, まず始めの NOR によって入力値を否定し, その 2 つの値の OR 回路に入れる。これにより, 常に出力値を 1 にすることができる。

4　考察

実験 (1)

NOT 回路について。この回路はすでに 1 個で作成されているためこれ以上少なくすることはできない。

AND 回路について。AND 回路は NAND 回路の否定により, 構成することができる。つまり, 初めの入力値を NAND 回路に入れる。そして, 次

の NAND を否定の回路にすることで,AND 回路が成り立つ。よって, この回路では最低でも 2 つの NAND が必要だとわかる。

OR 回路について。NAND は,(1,1) 以外の入力を 1 として返す。このことを前提にして, 入力値を (0,0) 以外の値を入れある値に変換させれた時に, その値を 1 として出力するためには, 初めに否定を取り, そして最後に 2 つ入力を NAND としてうけとる必要がある。そのため,2 つの入力を否定するための NAND と最後に受け取るための NAND の 3 つ必要である。

NAND 回路について。NAND 回路は、その一つを置いておけば回路として成立するため 1 つ十分である。

NOR 回路について。NOR 回路は,OR 回路の否定である。つまり, 一番最短な回路の構成としては OR 回路に NOT 回路を繋げた形である。よって,OR 回路の 3 つと NOT 回路の 1 つの計 4 つ必要である。

XOR 回路について。この回路を導くために論理式を用いる。

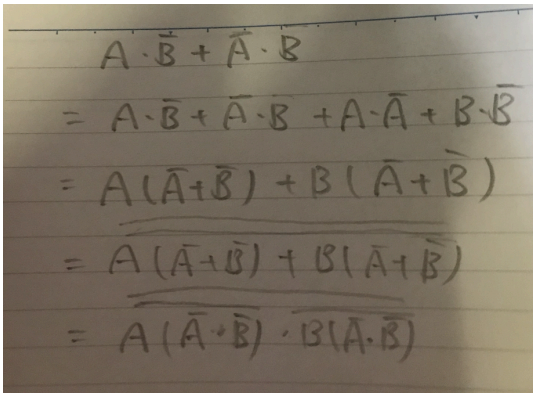

図 15: XOR を求めた論理式

図 15 からわかるように, これ以上小さくできないため NAND の数は最低 4 つは必要である。

常に 1 と 0 を出力するための回路は, 論理式により, これ以上小さくできないため,0 の場合は 3 つ。1 の場合は 4 つ必要である。

実験 (3)

配線を減らすためには, まずは使う量を決めることが大事である。使えるだけ使うでは, 手当たり次第使ってしまう恐れがある。仮に, 使う量を制限しておけば, その範囲だけでどうやって回路図を作り上げるか考える思考力を上げることができる。配線ミスをなくすためには, 頭の中でシミュレーションをすることが大事である。例えば, 頭の中で実際の完成図を作り上げそれの通りに配線を繋いでいく。それか, 図や絵にして, 見てわかるようにすれば配線ミスは減らすことができる。

実験 (4)

NOT 回路について。この回路はすでに 1 個で作成されているためこれ以上少なくすることはできない。

AND 回路について。NOR は (0,0) の時にしか値 1 を返さないから, まず先の NOR を否定にして, その否定した 2 つの値を次の NOR に入れることによって完成する。よって, この回路は, 最低 3 つの NOR が必要である。

OR 回路について。OR 回路は,NOR の否定であるため, まず始めに NOR に値を入れそれを図 8 で求めた否定を繋げる。これにより, これにより回路は完成する。よって, この回路には最低 2 つの NOR が必要である。

NAND 回路について。NAND 回路は AND 回路の否定であるため, 図 9 で作成した AND 回路を図 8 の否定を繋げることで完成する。これにより, 最低でも 4 つは必要である。

XOR 回路について。この回路を導くために論理式を用いる。

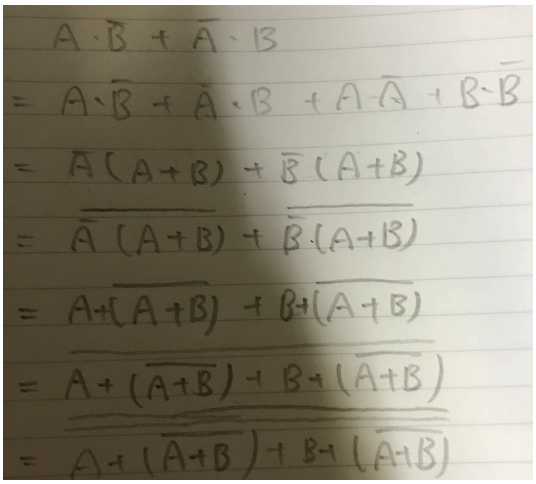

図 16: XOR を求めた論理式

常に 1 と 0 を出力するための回路は, 論理式により, これ以上小さくできないため,0 の場合は 4 つ。1 の場合は 3 つ必要である。

5 調査課題

(a) 簡単な論理関数を求める際に必要な技法として, カルノー図を用いる。図 17 は, カルノー図の例である。カルノー図とは, 主加法標準形の論理関数を最小項別に分け, それを最小項の真理値に当てはまるように図に書いていく。真理値表と似ているが, 論理変数の順番が違っていたり, 使用可能な変数が限られている。6 変数以上は事実上扱うことができない。そして, 論理関数を簡単にする方法をグループ化という。グループ化の対象となるのは, マスに 1 と記されたところだけである。4×4 や 2×2,1×4

のようなまとまったグループを作っていく。そして, 論理関数に戻りまとまったグループで, 関数を簡単化する。これにより, 論理関数は短くなり簡単になる。

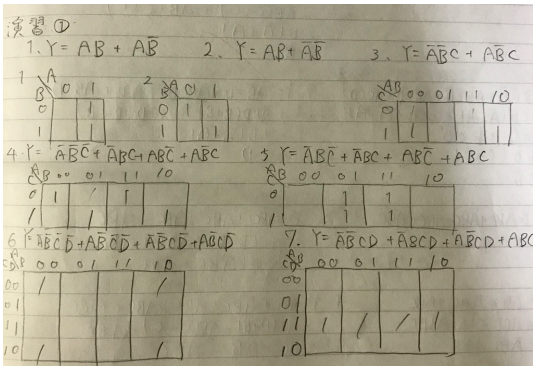

図 17: カルノー図の例

(b) 半加算器と全加算器。半加算器とは, 足し算に使われる回路の一部である。XOR と AND 回路を用いる。とても, シンプルな回路である。しかし, この回路は, 桁の繰り上げを考慮してないために加算器としてはまだ完全とは言えない。全加算器は, 前者の半加算器ができなかった桁の繰り上げができる。この加算器は, 実は半加算器と半加算器を繋げただけである。初めの半加算器の計算結果で繰り上げになった出力値を次の半加算器に入力することで, 加算器として成り立っている。図 18 は, 半加算器と全加算器の回路図, 真理値表の例である。

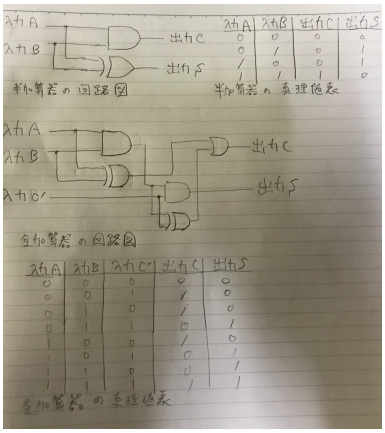

図 18: 半加算器と全加算器の回路図, 真理値表

6　感想

今回の実験を通し, 回路の複雑さを理解した。コンピュータには, 多くの論理回路が使われており, 今回私たちが作成したのはそれのほんの一部に過ぎないことを体感した。それにより, コンピュータ内部の論理回路の仕組みに非常に興味深く感じた。

意見としては, 教授や院生方々は常に生徒の作業状況を確認しておく必要にあると感じた。理由としては, 私の隣に座ってた学生は最初の作業さえままならない状況だと思ったからである。手が止まり, 周りを見渡すことしかできていません。そのため, 周りの人たちよりも作業状況が悪く見えました。仮に, 院生の方々がそれに気づき声かけをすれば, 彼の理解も作業状況も上がっていたのだろうと思った。

7　参考文献・引用文献

参考文献

[1] ryukyu: http://www.ie.u-ryukyu.ac.jp/ e035740/jikn/rep6.pdf

[2] classroom: https://classroom.google.com/o/MTE0OTM1MTY3OTla