情報工学実験I
実験2-汎用ロジックICによる組み合わせ回路の実現-

レポート作成者: 055702B 池野谷克俊

共同実験者: 055730H 新垣大志
055752J 比嘉安史
035710D 内間新祐

実験実施日: 2006年6月9日 金曜日
提出日: 2006年6月16日 金曜日

1　実験目的

本実験では, 簡単な組み合わせ回路の設計および実現を行うことによって, カルノー図などを用いた論理関数の簡単化に慣れるとともに, 実際のコンピュータに使用されている演算器の設計法について習得する.

2　実験概要

まず, 半加算器, 全加算器と 2 ビット加算器の真理値表を作成し, それを基にして論理関数, カルノー図を作成した. そしてカルノー図を用いて半加算器, 全加算器,2 ビット加算器の論理関数を簡単化できるかを確かめた. 次に半加算器, 全加算器の簡単化された論理関数を用いて回路図を作成した。そして, その回路図を基にブレッドボード上に半加算器を設計した. そのとき用いた IC は前回の NAND 回路で構成された IC だけではなく,4000 シリーズ IC を用いて設計した。我々の班は 4030B(XOR) と 4081B(AND) の IC を用いて半加算器を実現した. 最後にもう一つ半加算器を実現し,2 つの半加算器と OR ゲートを用いて全加算器を実現した.

3　実験結果

3.1　実験 (1 の結果について)

1. 半加算器, 全加算器,2 ビット加算器の真理値表を示せ

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

表 1: 半加算器

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

表 2: 全加算器

A_1	A_0	B_1	B_0	S_0	S_1	S_2
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	1	0
0	0	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	0
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	1	0	1
1	1	1	1	0	1	1

表 3: 2 ビット加算器

2. 各真理値表から, 半加算器, 全加算器,2 ビット加算器の論理関数を求めよ. なお, 各論理関数の導出課程も示すこと.

- 半加算器
  主加法標準形を用いる

$$\begin{aligned} S(A,B) &= S(0.1)+S(1,0) \\ &= A'\cdot B + A\cdot B' \end{aligned}$$

$$\begin{aligned} C(A,B) &= C(1,1) \\ &= A\cdot B \end{aligned}$$

- 全加算器
  主加法標準形を用いる

$$\begin{aligned} S(A,B,C_{in}) &= S(0,0,1)+S(0,1,0)+S(1,0,0)+S(1,1,1) \\ &= A'\cdot B'\cdot C_{in} + A'\cdot B\cdot C'_{in} + A\cdot B'\cdot C'_{in} + A\cdot B\cdot C_{in} \end{aligned}$$

$$\begin{aligned} C_{out}(A,B,C_{in}) &= C_{out}(0,1,1)+C_{out}(1,0,1)+C_{out}(1,1,0)+C_{out}(1,1,1) \\ &= A'\cdot B\cdot C_{in} + A\cdot B'\cdot C_{in} + A\cdot B\cdot C'_{in} + A\cdot B\cdot C_{in} \end{aligned}$$

- 2 ビット加算器
  主加法標準形を用いる

$$\begin{aligned} S_0(A_1,A_0,B_1,B_0) &= S(0,0,0,1)+S(0,0,1,1)+S(0,1,0,0)+S(0,1,1,0) \\ &\quad +S(1,0,0,1)+S(1,0,1,1)+S(1,1,0,0)+S(1,1,1,0) \\ &= A'_1\cdot A'_0\cdot B'_1\cdot B_0 + A'_1\cdot A'_0\cdot B_1\cdot B_0 + A'_1\cdot A_0\cdot B'_1\cdot B'_0 \\ &\quad A'_1\cdot A_0\cdot B_1\cdot B'_0 + A_1\cdot A'_0\cdot B'_1\cdot B_0 + A_1\cdot A'_0\cdot B_1\cdot B_0 \\ &\quad A_1\cdot A_0\cdot B'_1\cdot B'_0 + A_1\cdot A_0\cdot B_1\cdot B'_0 \end{aligned}$$

$$\begin{aligned} S_1(A_1,A_0,B_1,B_0) &= S(0,0,1,0)+S(0,0,1,1)+S(0,1,0,1)+S(0,1,1,0) \\ &\quad +S(1,0,0,1)+S(1,0,0,1)+S(1,1,0,0)+S(1,1,1,1) \\ &= A'_1\cdot A'_0\cdot B_1\cdot B'_0 + A'_1\cdot A'_0\cdot B_1\cdot B_0 + A'_1\cdot A_0\cdot B'_1\cdot B_0 \\ &\quad A'_1\cdot A_0\cdot B_1\cdot B'_0 + A_1\cdot A'_0\cdot B'_1\cdot B'_0 + A_1\cdot A'_0\cdot B'_1\cdot B_0 \\ &\quad A_1\cdot A_0\cdot B'_1\cdot B'_0 + A_1\cdot A_0\cdot B_1\cdot B_0 \end{aligned}$$

$$
\begin{aligned}
S_2(A_1, A_0, B_1, B_0) &= S(0,1,1,1) + S(1,0,1,0) + S(1,0,1,1) + S(1,1,0,1) \\
&\quad + S(1,1,1,0) + S(1,1,1,1) \\
&= A_1' \cdot A_0 \cdot B_1 \cdot B_0 + A_1 \cdot A_0' \cdot B_1 \cdot B_0' + A_1 \cdot A_0' \cdot B_1 \cdot B_0 \\
&\quad A_1 \cdot A_0 \cdot B_1' \cdot B_0 + A_1 \cdot A_0 \cdot B_1 \cdot B_0' + A_1 \cdot A_0 \cdot B_1 \cdot B_0
\end{aligned}
$$

3.2 実験 (2) の結果について

1. 半加算器, 全加算器,2 ビット加算器のカルノー図を示せ.

   - 半加算器

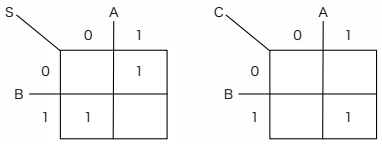

図 1: 半加算器

   - 全加算器

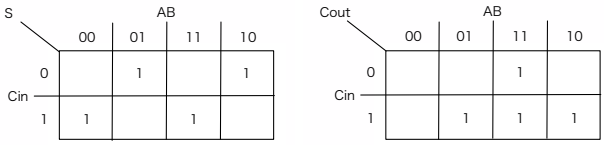

図 2: 全加算器

- 2 ビット加算器

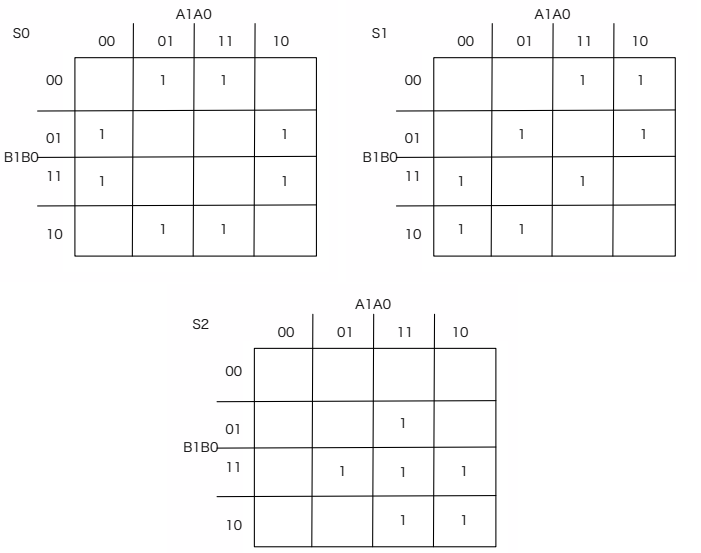

図 3: 2 ビット加算器

2. 各カルノー図から半加算器, 全加算器,2 ビット加算器の簡単化された論理関数を求めよ. なお, 簡単化の課程も示すこと.

- 半加算器
  カルノー図より, 簡単化できないことが分かる

- 全加算器
  カルノー図より,
  $C_{out} = B \cdot C_{in} + A \cdot B + A \cdot C_{in}$
  S は簡単化できない.

- 2 ビット加算器
  カルノー図より,
  $S_0 = A_0 \cdot B_0' + A_0' \cdot B_0$
  $S_1 = A_1 \cdot B_1' \cdot B_0' + A_1 \cdot A_0' \cdot B_1' + A_1' \cdot A_0' \cdot B_1 + A_1' \cdot B_1 \cdot B_0' + A_1' \cdot A_0 \cdot B_1' \cdot B_0 + A_1 \cdot A_0 \cdot B_1 \cdot B_0$
  $S_2 = A_0 \cdot B_1 \cdot B_0 + A_1 \cdot A_0 \cdot B_0 + A_1 \cdot B_1$

3.3 実験 (3) の結果について

1. 簡単化後に, 各論理関数がどのように変形されたかを説明せよ.

   - 半加算器
     半加算器は簡単化できなかった.
   - 全加算器
     S は簡単化できなかった.
     C_{out} はカルノー図を用いて $A'BC_{in} + AB'C_{in} + ABC'_{in} + ABC_{in}$ から, $BC_{in} + AB + AC_{in}$ に簡単化できた.
   - 2 ビット加算器
     $S_0S_1S_2$ のいずれもカルノー図によって簡単化できた.

3.4 実験 (4) の結果について

1. 半加算器, 全加算器の簡単化された論理関数をもとに, それぞれの論理関数を実現する回路図を示せ.

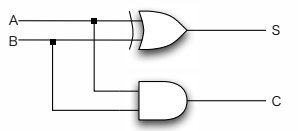

図 4: 半加算器

全加算器の S の論理式は簡単化できなかったので C_{out} についての回路図を示す.

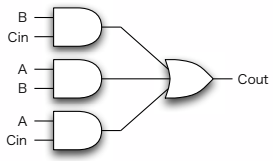

図 5: 全加算器 C_{out}

3.5 実験 (5) の結果について

ブレッドボード上に実現した回路が, 半加算器と同等な動作をしました. どのような回路を設計したかは, 省略します.

3.6 実験 (6) の結果について

ブレッドボード上に実現した回路が, 全加算器と同等な動作をしました. どのような回路を設計したかは, 省略します.

4 考察

4.1 実験 (3) の考察について

- 半加算器
  S,C のいずれも,XOR,AND ゲート 1 つで表されているのでこれ以上減らすことはできない.

- 全加算器
  S についてはカルノー図を見て分かるようにすべてのハミング距離が 2 なので, これ以上簡単化できない. よって論理演算子の数を減らすことはできない.
  C_{out} については, カルノー図によって簡単化された論理関数の論理演算子を減らすことはできないが,NAND ゲート 4 つで表すことによってトランジスタの数は減らすことができる.

- 2 ビット加算器
  S_0 はカルノー図によって簡単化された論理関数が XOR ゲート 1 つで表せるため論理演算子を減らすことはできない. S_1,S_2 については論理演算子の数は減らせないが,S_1 は NAND ゲート 7 つで, S_2 は NAND ゲート 4 つで表すことで, トランンジスタの数は減らすことができる.

4.2 実験 (4) の考察について

- 半加算器
  論理関数の簡単化をすることができなかったので, ゲート数や配線数は変化しない.

- 全加算器
  C_{out} が簡単化できたため,C_{out} の入力数が 6 個から 3 個に減ったので, 入力に使う配線数が減る. さらに簡単化する前は NOT ゲート 3 つ,NAND ゲート 5 個で表せたが, 簡単化後は,NAND ゲート 4 つで表せるのでゲート数も減ることになる.

4.3 その他の考察

今回の実験で, ブレッドボード上に実現した回路が正常な動作をするかどうかを判断するのに, オシロスコープではなく豆電球を使って確かめた. 豆電球が光れば 1, 光らなければ 0 が出力されたことになるので, オシロスコープを使わなくても判断できることが判明した. オシロスコープを使うよりも簡単に判断することができるので, 便利だった.

5 調査課題

(a) エンコーダ, デコーダ, マルチプレクサ, デマルチプレクサ, コンパレータ

- エンコーダ
  エンコーダは 10 進数を 2 進数に変換するものである.10 進数を 4 ビットの 2 進数コードにコード化するような論理回路をエンコーダを呼ぶ.

10 進入力	2 進数出力			
A	F_3	F_2	F_1	F_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

- デコーダ
  デコーダは２進数を１０進数に変換するものである.

2 進入力				10 進出力
A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

- マルチプレクサ
  複数の入力から１つをえらぶ回路, 選択制御信号 S が０で A を出力, １で B を出力する.

選択制御信号 S	入力 A	入力 B	出力 F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- デマルチプレクサ
  １つの入力を複数の出力へ振り分ける回路. 選択制御信号 S が０で入力を X に出力, １で入力を Y に出力する.

選択制御信号 S	入力 A	出力 X	出力 Y
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

- コンパレータ
  1 ビットの 2 進数 A と B の大小を比較する回路

入力 A	入力 B	出力 A < B	出力 A = B	出力 A > B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

(b) クワイン-マクラスキー法

クワイン-マクラスキー法とは, 補元の性質 $xP + x'P = P$ を利用した簡単化を全て行い, 最終的に必要な項のみを集める手法である.

具体的な手順

1. 主加法標準形の n 変数論理関数 f の各最小項をハミング重みによって分類する.
2. ハミング重みが j($j = 0, 1, ..., n-1$) のグループ内の各最小項に対してハミング距離が 1 となる最小項を, ハミング重みが j+1 のグループから探す. ハミング距離が 1 となる最小項の対が存在する場合は, 両者の論理和に対して補元則を適用し,1 変数少なくなった新しい最小項を別に記録する. また, この新しい最小項の導出に使用した最小項に印を付ける. すでに導出されている最小項と同じものが導出された場合も, その導出に印を付ける.
3. m 次圧縮によって導出された各最小項に対して,2 と同様の圧縮操作を行う.
4. 新しい最小項が導出されなくなるまで 3 を繰り返し, 最終的に印のついてない最小項を主項とする.

- 半加算器

  S について

グループ	最小項
G_1	$A' \cdot B$
	$A \cdot B'$

ハミング距離が 1 の最小項の組み合わせが無いのでこれ以上変形できない. よって S は変わらない.

C について

グループ	最小項
G_1	$A \cdot B$

ハミング距離が 1 の最小項の組み合わせが無いのでこれ以上変形できない. よって C は変わらない.

以上より, 半加算器はクワイン-マクラスキー法を使ってもカルノー図を用いても簡単化できないので. 同じ論理関数のままである.

- 全加算器

S について

グループ	最小項
G_1	$A' \cdot B' \cdot C_{in}$
	$A' \cdot B \cdot C'_{in}$
	$A \cdot B' \cdot C'_{in}$
G_2	$A \cdot B \cdot C_{in}$

ハミング距離が 1 の最小項の組み合わせが無いのでこれ以上変形できない. よって S は変わらない.

C_{out} について

グループ	最小項
G_1	$A' \cdot B \cdot C_{in}$
	$A \cdot B' \cdot C_{in}$
	$A \cdot B \cdot C'_{in}$
G_2	$A \cdot B \cdot C_{in}$

ハミング距離が 1 の最小項を補元則を用いて 1 次圧縮を行うと次のようになる.

グループ	最小項
G_1,G_2	$B \cdot C_{in}$
	$A \cdot C_{in}$
	$A \cdot B$

ハミング距離が 1 の最小項がないので, これ以上変形できない.
よって C_{out} は $A \cdot B + B \cdot C_{in} + A \cdot C_{in}$ となる.

以上より, 全加算器はクワイン-マクラスキー法を用いてもカルノー図を用いても同じ結果になる.

(c) 同期式と非同期式

同期式

- 動作時点を指定するクロックパルスを用いる.
- クロックパルスに依存するが, 現在の状態が十分安定したと思われる時間間隔で動作されるので安定性がよい.

非同期式

- 伝搬遅延時間を利用する.
- 高速だが, 回路が複雑になると遅延時間の推定が困難になる.

(d) D フリップフロップ, カウンタ

- D フリップフロップ

クロック入力ＣＬＫの立ち上がりタイミングの時の入力Ｄの信号状態が, 出力Ｑに出力され保持される. クロックの立ち上がりの瞬間に入力Ｄが１ならば, 出力Ｑも１に, 入力Ｄが０だったら出力Ｑも０になる. 次のクロックの立ち上がりが入力されるまで, 入力Ｄの状態に関係なく出力は保持される.

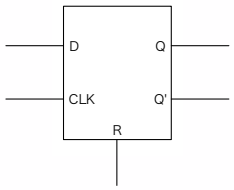

図 6: D フリップフロップ

R	CLK	D	Q	Q’
0	-	-	0	1
1	無し	0	変化無し	変化無し
1	無し	1	変化無し	変化無し
1	↑	0	0	1
1	↑	1	1	0

- カウンタ

  カウンタとは入力が入るごとに出力の値が $+1$ または -1 するものです.
  同期式カウンタは非同期式と比べて, 伝搬遅延時間がたまりにくく, 予期しない状態になりにくい.

6 感想

今回の課題はカルノー図の作成や真理値表の作成に手間がかかった. 今回の実験でオシロスコープを使わずに豆電球で回路ができているかを確かめたのだが, オシロスコープよりも楽に確かめられたのでなかなかよかった (TAの人に教えてもらいました). オシロスコープを使えるようになるのも大事だと思うが, 豆電球を使ってみるのもいいかもしれないと思った.

参考文献

[1] 論理設計 ~ 順序回路 1
http://jadore.jp/~kpcsite/tech/logic05.html

[2] 色々なフリップフロップ回路
http://homepage1.nifty.com/rikiya/software/413FF.html