情報工実験1

035760A：横田敏明

2004/06/7

1　実験状況

1.1　共同実験者

村山正嗣
宮城大輔

1.2　実験器具

IC :
TC4081BP
TC4071BP
TC4030BP

2　実験概要

2.1　実験目的

本実験は，簡単な組み合わせ回路の設計及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なれるとともに，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．

2.2　実験内容

- 半加算器，全加算器，2 ビット加算器の真理値表及び論理関数を求めよ．
- カルノー図を用いて論理関数を圧縮せよ
- 論理圧縮によって簡単化されたことをを確認せよ
- 各加算器の回路図を書け．
- 回路図をもとに，ブレッドボード上に実現せよ．

2.3　組み合わせ回路と順序回路

回路内部に記憶素子を含まず，入力のみによって出力を決定する回路を組み合わせ回路といい，内部に記憶素子を含む回路を順序回路という．
記憶素子は，内部にフィードバックを設けたフリップフロップ回路と呼ばれる回路を用いる．

3　結果報告

3.1　半加算器，全加算器，2 ビット加算器の真理値表と論理関数

半加算器とは，2 入力に対してキャリーとサムを返す回路である．半加算器を組み合わせて 2 入力と下位からのキャリーに対して，キャリーとサムを返す回路を全加算器という．

3.1.1　半加算器

半加算器真理値表

入力 A	入力 B	キャリー	サム
0	0	0	0
0	1	0	1
1	1	1	0
1	0	0	1

キャリーを C，サムを S とすると．

$$C = A \cdot B \tag{1}$$

$$S = (A + B) \bullet \overline{(A \bullet B)} \tag{2}$$

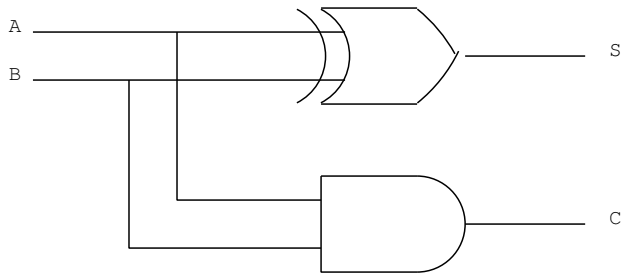

図 1: 半加算器

3.1.2 全加算器

全加算器真理値表

入力 A	入力 B	下位キャリー	C	S
0	0	0	0	0
0	0	1	0	1
0	1	1	1	0
0	1	0	0	1
1	1	0	1	0
1	1	1	1	1
1	0	1	1	0
1	0	0	0	1

全加算器の論理関数は，

$$C = (A \bullet B) + (B \bullet C) + (C \bullet A) \tag{3}$$

$$S = (A + B + C) \bullet \overline{(A \bullet B \bullet C)} \tag{4}$$

である．

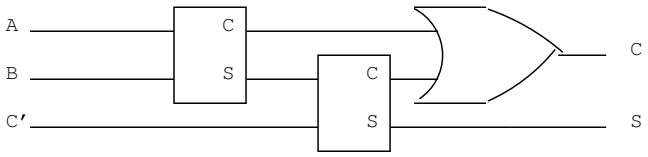

図 2: 全加算器回路図

3.1.3　2 ビット加算器

2 ビット加算器は，2 ビットと 2 ビットの加算をする 4 入力 3 出力の加算器である．

2 ビット加算器真理値表

入力 A_2	入力 A_1	入力 B_2	入力 B_1	S_3	S_2	S_1
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	1

入力された二つの 2 桁の二進数の足し算 (加算) である．出力は三桁の二進数で表すことができる．また，論理関数は以下の通りである．

$$S_1 = (A_1 + B_1) \bullet (\overline{A_1 \bullet B_1}) \tag{5}$$

$$S_2 = (A_2 + B_2) \bullet (\overline{A_2 \bullet B_2}) \bullet (\overline{A_1 \bullet B_1}) + (A_2 \bullet \overline{B_2} \bullet A_1 \bullet B_1) + (\overline{A_2} \bullet B_2 \bullet A_1 \bullet B_1) \tag{6}$$

$$S_3 = (A_2 \bullet B_2) + (A_2 \bullet \overline{B_2} \bullet (A_1 \bullet B_2)) + (\overline{A_2} \bullet B_2 \bullet (A_1 \bullet B_1)) \tag{7}$$

最大で出力は 111 を返す.

3.2 カルノー図

論理関数を求めるとき，真理値表より理論圧縮し簡単に関数を求める方法が，カルノー図による論理関数の簡単化である．カルノー図は，一つの軸が 2 変数以上になったとき，互いに隣接するマスは一つずつずらされた (差異は一桁) 並び方をしている．論理圧縮は隣接するマスが同じく 1 である場合をグループ化し，その差異のある入力要素は消去することができるというもの．

半加算器のカルノー図

S	0	1
0	0	1
1	1	0

C	0	1
0	0	0
1	0	1

全加算器のカルノー図

S	00	01	11	10
0	0	1	0	1
1	1	0	1	0

C	00	01	11	10
0	0	0	1	0
1	0	1	1	1

2 ビット加算器のカルノー図

S_3	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	1

S_2	00	01	11	10
00	0	0	1	1
01	0	1	0	1
11	1	0	1	1
10	1	1	1	0

S_1	00	01	11	10
00	0	1	0	0
01	1	0	0	1
11	1	0	1	0
10	0	1	0	0

なお，簡略化された論理関数は既に記述しているため，省略する．

4　加算器以外の実用的な組み合わせ回路

組み合わせ回路とは，内部に記憶素子を組み込まず，以前の操作や記憶状態に依存せずに入力に対してのみに依存する回路のことをいう．組み合わせ回路に対して，内部に記憶素子が組み込まれ，以前の操作に依存し，入力に対しても依存性をもつ回路を順序回路という．組み合わせ回路の実用例として，補数器，減算回路，積回路，を示す．

4.1　補数器

先頭ビットを正負を区別するビットとし，減算を実現するために，補数器を用いる．ただし，入力されるそれぞれの数値は同じビット数でなければならない．

$$\text{入力 } A - \text{入力 } B = \text{出力 } C \tag{8}$$

とすると，入力 B を 2 の補数表現で表し，加算すれば良い．

入力ビット数が 3 である場合の補数器の実現法と，回路を示す．
まず，減算する入力を 1 だけインクリメントし，0 と 1 を反転させる．インクリメントの回路を省略し，反転方法だけを示す．

補数器の真理値表

B	B	B	C	C	C
0	0	0	1	1	0
0	0	1	1	0	1
0	1	0	1	0	0
0	1	1	0	1	1

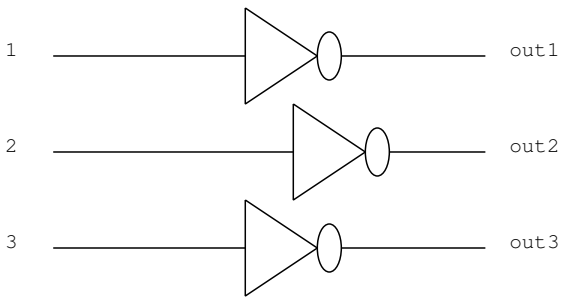

図 3: 補数器回路図

4.2 減算回路

減算回路を実現するためには，補数器と加算器を組み合わせる．まず，補数器によって 2 の補数に変換し，それぞれの桁ごとに加算器を通して結果を返す．出力の先頭ビットは正負を表すための出力である．以下に簡略化した回路を示す．

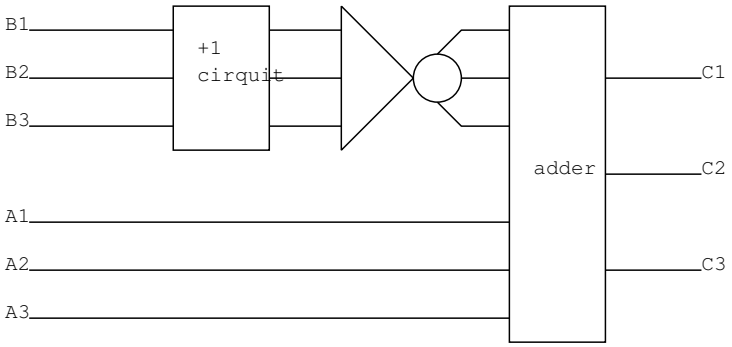

図 4: 減算回路図

4.3 積回路

積を求めるには，論理積回路と加算器を組み合わせることで実現できる．各ビットごとに論理積を求め，それを全ての組み合わせで行い，加算器でそれらの和を表現する．

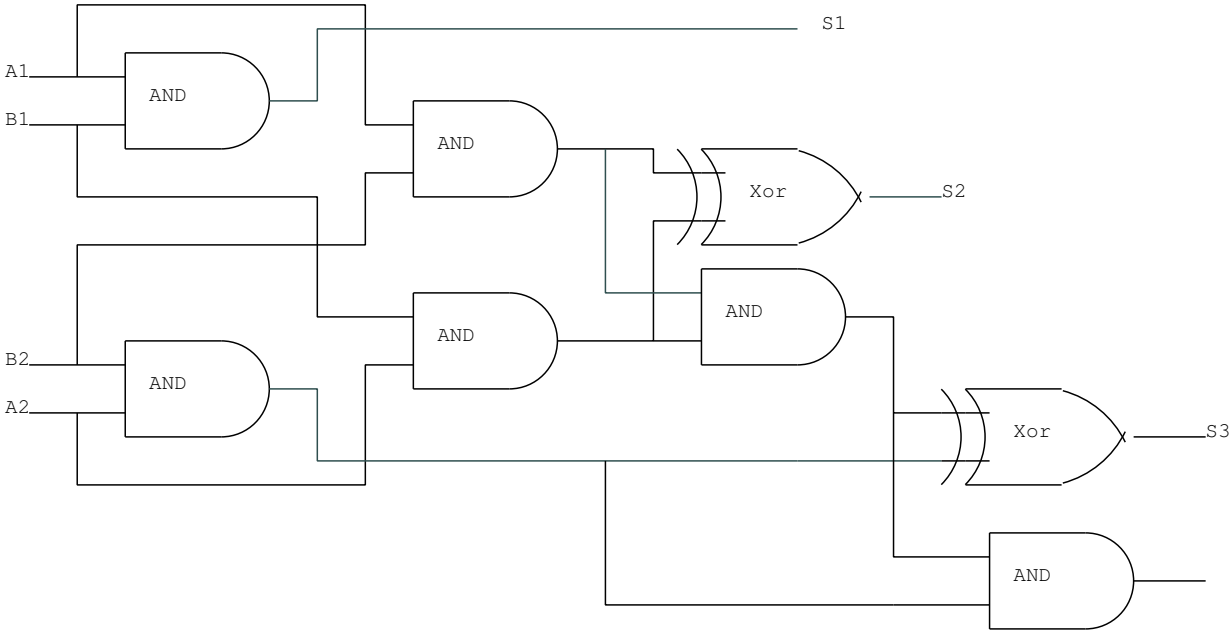

図 5: 積回路

5　キャリールックアヘッド方式

桁上げ先見加算器ともいう．桁上げ後の加算結果を用いて，処理の高速化を実現する．

6　順序回路の概要

順序回路とは，記憶素子を用いることでメモリを実現し，前回の状態と入力によって出力を決定する回路のことをいう．

順序回路には，データを蓄えるフリップフロップ回路などがある．フリップフロップ回路は回路内にフィードバックを設置することで，1 ビットのデータを蓄える機能をもつ回路である．また，順序回路には同期式と非同期式がある．前者はクロック入力によってタイミングを合わせるため，回路の設計が容易なかわりに処理速度が非同期式よりも遅い．後者はクロック入力をもたず，処理がはやい．その半面，タイミングを計算し，注意して設計しなければならないため，設計が非常に困難である．

7　D フリップフロップとカウンタ回路

D フリップフロップには，入力端子と出力端子があり，コントロールするクロック入力が存在する．

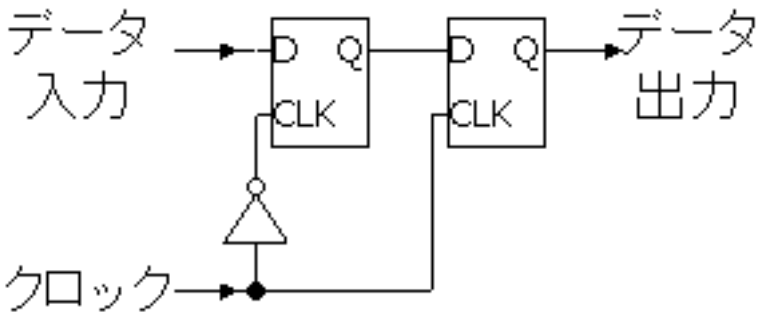

図 6: D フリップフロップの回路図

カウンタ回路は，クロック入力の回数を数え，その数を二進数として出力する回路である．16 進数カウンタでは，15 の次は 0 になり，ループする．

カウンタは $\frac{1}{2^n}$ の周期をとることができる．

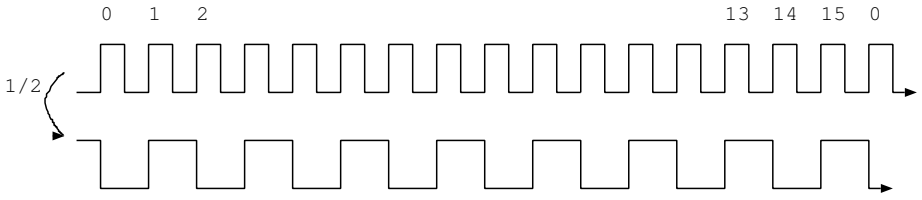

図 7: カウンタのタイミングチャート

8　考察

8.1　カルノー図

今回の実験で，カルノー図によって論理関数をもとめた．カルノー図は隣接するマスの差異は，1 ビットだけであるという制約をもつことによって，論理圧縮を容易にするアイディアであった．カルノー図を作成せずとも論理関数を求めることはできるが，論理圧縮が容易にできるのはカルノー図を用いた場合が多い．

8.2　論理関数の圧縮

回路を設計する場合はできる限り圧縮された論理関数を用いると，最も効率的に回路設計が実現できる．真理値表のみを頼りに回路を設計することは，大変難しい．今回はカルノー図の重要性を認識できた．

8.3　組み合わせ回路と順序回路

組み合わせ回路は記憶素子を持たず，データを蓄えることができないため，入力に対して一意に出力を決定する (一意とは限らない場合もある)．次回の実験は記憶素子を含む順序回路についての実験を実施するため，あらかじめ D フリップフロップ等の調査を行った．順序回路は，組み合わせ回路に記憶素子を組み込んで応用したものといえる．

9　参考

http://www.ie.u-ryukyu.ac.jp/wada/digicir04/latch.html