(2) 論理和 (OR) 回路

・目的

$Y = A + B$ を理解する.

・理論

論理和は，$Y = A + B$ で表現され，入力 A と B がいずれも “0” のときのみ，出力が “0”，他の条件ではすべて “1” となるもので，この式を満足する論理回路を OR 回路という (表 0.1).

表 1: OR の真理値表

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

・実習

図 3.2 のように，パネル上の OR 回路素子を使用して結線し，$Y = A + B$ の真理値を表示器で表示して，真理値表 0.1 と一致することを確認した.

A，B の入力レベルは，設定スイッチにより設定した.

(3) 否定 (NOT) 回路

・目的

$Y = \overline{A}$ を理解する.

・理論

否定回路は，インバータとも言われ，$Y = \overline{A}$ で表現される．入力と出力の関係は常に正反対になり，この式を満足する論理回路を否定回路という (表 0.2).

表 2: NOT の真理値表

A	Y
0	1
1	0

・実習

図 3.3 のように，パネル上の NOT 回路素子を使用して結線し，$Y = \overline{A}$ の真理値を表示器で表示して，真理値表 0.2 と一致することを確認した．

A の入力レベルは，設定スイッチにより設定した．

(4) 論理積の否定 (NAND) 回路

・目的

$Y = \overline{A \cdot B}$ を理解する．

・理論

論理積の否定は，$Y = \overline{A \cdot B}$ で表現され，入力 A と B がいずれも “1” のときのみ，出力が “0”，他の条件ではすべて “1” となるもので，この式を満足する論理回路を NAND 回路という (表 0.3)．

表 3: NAND の真理値表

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

・実習

図 3.4 のように，パネル上の NAND 回路素子を使用して結線し，$Y = \overline{A \cdot B}$ の真理値を表示器で表示して，真理値表 0.3 と一致することを確認した．

A，B の入力レベルは，設定スイッチにより設定した．

(5) 論理和の否定 (NOR) 回路

・目的

$Y = \overline{A + B}$ を理解する．

・理論

論理和は，$Y = \overline{A + B}$ で表現され，入力 A と B がいずれも “0” のときのみ，出力が “1”，他の条件ではすべて “0” となるもので，この式を満足する論理回路を NOR 回路という (表 0.4)．

・実習

図 3.5 のように，パネル上の NOR 回路素子を使用して結線し，$Y = \overline{A + B}$ の真理値を表示器で表示して，真理値表 0.4 と一致することを確認した．

A，B の入力レベルは，設定スイッチにより設定した．

表 4: NOR の真理値表

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(6) ド・モルガンの定理の証明

・目的

ド・モルガンの定理

$$\overline{A \cdot B} = \overline{A} + \overline{B}, \quad \overline{A + B} = \overline{A} \cdot \overline{B}$$

の証明を実際に論理回路を構成して行う.

・理論

ド・モルガンの定理は，次の式 (0.0.1)，(0.0.2) で表される.

$$\overline{A \cdot B} = \overline{A} + \overline{B} \tag{1}$$
$$\overline{A + B} = \overline{A} \cdot \overline{B} \tag{2}$$

式 (0.0.1)，(0.0.2) を書き直すと

$$Y_1 = \overline{A \cdot B}, \quad Y_2 = \overline{A} + \overline{B}, \quad Y_1 = Y_2 \tag{3}$$
$$Y_3 = \overline{A + B}, \quad Y_4 = \overline{A} \cdot \overline{B}, \quad Y_3 = Y_4 \tag{4}$$

となる．上式 (0.0.3)，(0.0.4) が成り立つための必要十分条件は

- Y_1 と Y_2，及び Y_3 と Y_4 それぞれについて，同一の 2 入力 A，B を用いていること
- 同一 2 入力 A，B に対し，出力 Y_1 と Y_2，及び Y_3 と Y_4 それぞれが一致すること
- 2 入力 A，B の真理値の組み合わせが全て網羅されていること

を同時に満たすことである．したがって，この条件を満たすように論理回路を構成し、表 0.5, 0.6 のような真理値表を得ることができれば，式 (0.0.3)，(0.0.4) が成り立つことを証明することができる．式 (0.0.3)，(0.0.4) を論理回路で構成すると図 3.6，3.7 のようになる.

表 5: $Y_1 = \overline{A \cdot B}$, $Y_2 = \overline{A} + \overline{B}$

A	B	Y_1	Y_2
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

表 6: $Y_3 = \overline{A + B}$, $Y_4 = \overline{A} \cdot \overline{B}$

A	B	Y_3	Y_4
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

・実習

式 (0.0.3)，(0.0.4) を証明する回路をそれぞれ図 3.8，3.9 のようにパネル上で構成し，それぞれにおいて各入力パターンに対する真理値を表示器で表示して，すべての入力パターンに対し $Y_1 = Y_2$，$Y_3 = Y_4$ であることを確認し，表 0.5，0.6 のような真理値表を確かに得ることができた．これにより，式 (0.0.3)，(0.0.4) が成り立つことが証明された．

なお，A，B の入力レベルは，設定スイッチにより設定した．

・考察

図 3.6 を見ると，出力 $\overline{A \cdot B}$ には NAND 回路素子 1 つのみが用いられている一方，出力 $\overline{A} + \overline{B}$ には NOT 回路素子 2 つと OR 回路素子 1 つが用いられている．また図 3.7 を見ると，出力 $\overline{A + B}$ には NOR 回路素子 1 つのみが用いられている一方，出力 $\overline{A} \cdot \overline{B}$ には NOT 回路素子 2 つと AND 回路素子 1 つが用いられている．すなわち，いずれの場合においても，2 つの出力は等価であるが，それを構成する回路素子数には差がある．したがって，ド・モルガンの定理を用いることで論理回路を構成する回路素子数を減らし，回路をより簡単化することが可能であると分かる．

また，本実験で証明した 2 つの論理変数に対するド・モルガンの定理の式 (0.0.1)，(0.0.2) を繰り返し用いると，3 つの論理変数 A，B，C について，

$$\begin{aligned} \overline{A \cdot B \cdot C} &= \overline{(A \cdot B) \cdot C} \\ &= \overline{A \cdot B} + \overline{C} \\ &= (\overline{A} + \overline{B}) + \overline{C} \\ &= \overline{A} + \overline{B} + \overline{C} \\ \overline{A + B + C} &= \overline{(A + B) + C} \\ &= \overline{A + B} \cdot \overline{C} \\ &= (\overline{A} \cdot \overline{B}) \cdot \overline{C} \end{aligned} \tag{5}$$

$$= \overline{A} \cdot \overline{B} \cdot \overline{C} \tag{6}$$

が成り立つことが分かる．同様に，式 (0.0.1)，(0.0.2) を繰り返し用いることにより，n 個の論理変数 A_1，A_2，$\cdots$，A_n に対し

$$\overline{A_1 \cdot A_2 \cdot \cdots \cdot A_n} = \overline{A_1} + \overline{A_2} + \cdots + \overline{A_n} \tag{7}$$

$$\overline{A_1 + A_2 + \cdots + A_n} = \overline{A_1} \cdot \overline{A_2} \cdot \cdots \cdot \overline{A_n} \tag{8}$$

が成り立ち，ド・モルガンの定理を 3 つ以上の論理変数の場合に拡張することができることが確かめられる．これにより，2 入力の場合と同様に多数の入力に対してもド・モルガンの定理を適用することで，回路構成を簡単化することが可能であることが分かる．

(7) 排他的論理和 (Exclusive-OR) 回路

・目的

$Y = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$ を理解する．

・理論

排他的論理和は，入力 A，B が同じレベルのとき出力 Y が “0”，異なるレベルのとき出力 Y が “1” となるもので，この式を満足する論理回路を Excluive-OR 回路といい，図 3.10(a) のシンボルで表現される．上記のような出力を得るためには，$A = 0,\ B = 1$ 及び $A = 1,\ B = 0$ のそれぞれの場合のみ出力 “1” を得る論理積を考え，最後にその 2 つの論理積の論理和を取ればよい．したがって排他的論理和は $Y = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$ で表される (表 0.7 参照)．$Y = \overline{A} \cdot B + A \cdot \overline{B}$ を基本素子の組み合わせによって構成すると，図 3.10(b) のようになる．

表 7: $Y = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$ の真理値表

A	B	$\overline{A} \cdot B$	$A \cdot \overline{B}$	Y
0	0	0	0	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

・実習

図 3.11 のように，論理回路をパネル上で構成し，$Y = \overline{A} \cdot B + A \cdot \overline{B}$ の真理値を表示器で表示して，真理値表 0.7 と一致することを確認した．

・考察

排他的論理和は，2 つの入力のレベルが同じとき “0” を，異なるとき “1” を出力する演算であることを確かめた．排他的論理和は，様々な用途に応用されている．

実例として，まず，ビット値の反転に排他的論理和が用いられる．ある数値と，その数値の反転させたい桁のビット値を 1 に，それ以外の桁のビット値を 0 にした数値との排他的論理和

をとると、指定した桁のビット値が反転した数値が得られる．例えば，$0011_{(2)}$ の下から 2 ビット目と 3 ビット目を反転させたい場合，$0110_{(2)}$ との排他的論理和をとることで

$$0011_{(2)} \oplus 0110_{(2)} = 0101_{(2)}$$

となり，2 ビット目と 3 ビット目が反転した値が得られる．

また，上記の性質により，排他的論理和は，2 回同じ値で演算を行うと元の値が得られるという性質がある．すなわち，2 つの論理変数 A，B に対して

$$(A \oplus B) \oplus B = A \tag{9}$$

が成り立つ．このことは次に示す真理値表 0.14 で確かめることができる．

表 8: $Y = (A \oplus B) \oplus B$ の真理値表

A	B	$A \oplus B$	Y
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

上の真理値表で，A と Y は真理値が一致しており，確かに $Y = A$ となっている．この性質は非常に重要であり，多くの応用例がある．

例えば，プログラミングにおいて排他的論理和を用いると，一時変数を用いずに変数の値の交換することができる．すなわち，$X \oplus Y = Z$ とすると $X \oplus Z = Y$，$Y \oplus Z = X$ であることを利用すると，

```
{
    ...
    x = x ^ y;
    y = x ^ y;
    x = x ^ y;
    ...
}
```

のようにプログラムを組むことで，値の交換が可能となる．ここで ^ は排他的論理和を表す演算子である．この方法は，一時変数を用いないため，メモリの節約につながる．

また，式 (0.0.9) の性質を用いると，暗号鍵と復号鍵が同一な暗号を作成することができる．平文を P，鍵を K とすると，

$$(P \oplus K) \oplus K = P$$

となることから，平文と同じ長さのビット列を鍵として用意することで，暗号化と復号に同一の鍵を用いることができる．このような暗号はバーナム暗号と呼ばれる．

(8) 加算器 (ADDER)

・目的

1. 加算器の S の部分が排他的論理和 (Exclusive-OR) であることを理解する．
2. 半加算器の動作を理解する．

・理論

加算器には，下位からの桁上げを考慮しない半加算器 (Half-ADDER) と，下位からの桁上げを考慮する全加算器 (Full-ADDER) とがある．全加算器は，半加算器を 2 個，OR 回路を 1 個直列に接続した形になる．

半加算器の論理式は，次の式で与えられる．

$$S = \overline{A} \cdot B + A \cdot \overline{B} \ (= A \oplus B) \tag{10}$$
$$C = A \cdot B \tag{11}$$

この論理式を論理回路に置き換えると図 3.12 になる．また，式 (0.0.10) から分かるように，回路の和 (Sum) を構成している部分は，Exclusive-OR になる．

全加算器の論理式は，次の式で与えられる．

$$S = \overline{A} \cdot \overline{B} \cdot C_i + \overline{A} \cdot B \cdot \overline{C_i} + A \cdot \overline{B} \cdot \overline{C_i} + A \cdot B \cdot C_i \tag{12}$$
$$\begin{aligned} C_O &= \overline{A} \cdot B \cdot C_i + A \cdot \overline{B} \cdot C_i + A \cdot B \cdot \overline{C_i} + A \cdot B \cdot C_i \\ &= A \cdot B + B \cdot C_i + A \cdot C_i \end{aligned} \tag{13}$$

いま，$S_1 = \overline{A} \cdot B + A \cdot \overline{B}$ とすれば，

$$S = \overline{S_1} \cdot C_i + S_1 \cdot \overline{C_i} \tag{14}$$

また，$C_1 = A \cdot B$，$C_2 = S_1 \cdot C_i$ とすれば，

$$\begin{aligned} C_O &= C_1 + C_2 \\ &= A \cdot B + S_1 \cdot C_i \end{aligned} \tag{15}$$

となり，さらに，Exclusive-OR を用いて表せば，

$$S = A \oplus B \oplus C \tag{16}$$
$$C_O = A \cdot B + (A \oplus B) \cdot C_i \tag{17}$$

となる．この論理式を論理回路に置き換えると図 3.13 になる．

表 9: Half-ADDER の真理値表

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

表 10: Full-ADDER の真理値表

A	B	C_i	C_O	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

・実習

図 3.12，3.13 の論理回路を図 3.14，3.15，3.16，3.17 のようにパネル上で構成し，各真理値を表示器で表示して，真理値表 0.9，0.10 と一致することを確認した．

・考察 半加算器は，下位からの桁上げを考慮せずに 1 ビット値の加算を行う回路であるが，これを 2 つ組み合わせることにより，下位からの桁上げも考慮して 1 ビット値の加算を行う全加算器を構成することができることが分かった．1 ビットにつき 1 つの全加算器が必要であるから，n ビット同士の加算を行うためには，n 個分の全加算器を用意すればよいことが分かる．なお，整数値の範囲では，LSB については下位からの桁上げを考慮する必要がないため，LSB の演算にのみ，半加算器を用いてもよい．このとき，回路は，1 個の半加算器と $n-1$ 個の全加算器で構成されることになる．一方，LSB の加算にも全加算器を用いる場合は，その全加算器への入力 C_O が常に “0” となるようアースすることになる．

加算器は，他の四則演算を行う回路，すなわち，減算回路，乗算回路，除算回路においても用いられるなど，非常に重要な演算回路である．

(9) デコーダ (DECODER)

・目的

2 進数を 10 進数に変換する動作を理解する．

・理論

4 ビットの 2 進数コードを、もとの 10 進数に戻すようなコード翻訳動作をする論理回路を、デコーダといい、2 進数と 10 進数の関係性は次のようになる.

2 進数コード: $A = 2^0$ ビット，$B = 2^1$ ビット
$C = 2^2$ ビット，$D = 2^3$ ビット

10 進数: “0” ～ “9” として

$$\text{“0”} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$$
$$\text{“1”} = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$$
$$\text{“2”} = \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D}$$
$$\text{“3”} = A \cdot B \cdot \overline{C} \cdot \overline{D}$$
$$\text{“4”} = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D}$$
$$\text{“5”} = A \cdot \overline{B} \cdot C \cdot \overline{D}$$
$$\text{“6”} = \overline{A} \cdot B \cdot C \cdot \overline{D}$$
$$\text{“7”} = A \cdot B \cdot C \cdot \overline{D}$$
$$\text{“8”} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D$$
$$\text{“9”} = A \cdot \overline{B} \cdot \overline{C} \cdot D$$

この式を満足する論理回路，及び真理値表を図 3.18，及び表 0.11 に示す.

表 11: デコーダ (2 進数と 10 進数) の真理値表

	2 進数				10 進数									
	A	B	C	D	“0”	“1”	“2”	“3”	“4”	“5”	“6”	“7”	“8”	“9”
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	1	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	1	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0	1	0	0	0	0
6	0	1	1	0	0	0	0	0	0	0	1	0	0	0
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0
8	1	0	0	0	0	0	0	0	0	0	0	0	1	0
9	1	0	0	1	0	0	0	0	0	0	0	0	0	1

・実習

パネル上のデコーダ部分を使用し，図 3.19 のように論理回路を構成して，各真理値を表示器で表示し，真理値表 0.11 と一致することを確認した.

・考察

10 進数 0 ～ 9 それぞれを 4 ビットの 2 進数で表すことにより，特定の 1 つの 4 ビット入力に対してのみ “1” を出力し，0 ～ 9 のうちただ 1 つだけを表示することができる．このような表示法は 2 進化 10 進法 (Binary Coded Decimal:BCD) と呼ばれる.

デコーダは復号器とも呼ばれ，エンコーダ (符号器) とともに用いられることが多い．また，10 進数だけでなく，8 進数や 16 進数と 2 進数との相互変換を行うエンコーダやデコーダを構成することも可能であると考えられる．8 進数の場合，デコーダは 3 入力 8 出力となる.

・順序回路 [実験 (10)，(11)]

順序回路 (Sequential Circuit) は，出力が入力だけでなく回路そのものの状態によって左右される論理回路で，R-S フリップ・フロップ回路 (R-S Flip·Flop Circuit)，J-K フリップ・フロップ回路 (J-K Flip·Flop Circuit)，シフトレジスタ (Shift Register) などがあり，その応用として n 進カウンタや，n ビットのシフトレジスタがある．

(10) R-S フリップ・フロップ回路

・目的

1. R-S フリップ・フロップ回路が 1 ビットの記憶素子であることを理解する．
2. R-S フリップ・フロップ回路において禁止とされる入力が存在することを理解する．

・理論

R-S フリップ・フロップ回路は，“0”，または “1” の論理レベルを記録する機能を持った回路である．論理式，シンボル，回路，真理値表は各々式 (0.0.18)，(0.0.19)，図 3.20(a)，図 3.20(b)，表 0.12 で表現される．

$$Q^{(n+1)} = \overline{\overline{S^{(n)}} \cdot \overline{Q}^{(n)}} \tag{18}$$

$$\overline{Q}^{(n+1)} = \overline{\overline{R^{(n)}} \cdot Q^{(n)}} \tag{19}$$

ただし時刻 n に関して，$S^{(n)}$，$R^{(n)}$，及び $Q^{(n)}$，$\overline{Q}^{(n)}$ は図 3.20 における入力 S，R と出力 Q，$\overline{Q}$ を表すものとする．

動作は，Q と $\overline{Q}$ が相補の関係 (Q と $\overline{Q}$ が互いに異なる値をもつ) にあるとき，R と S が “0” レベルの時，出力は元の状態を保持し，S が “1”，R が “0” なら出力 Q は “1”，S が “0”，R が “1” なら出力 Q は “0” にそれぞれ落ち着く．しかしながら S と R がともに “1” の場合は Q と $\overline{Q}$ が “0” と “1” のレベルを相互に繰り返すこととなる (ただし，実際には図 3.20(b) の各 NAND 素子の応答速度の差異や配線の長さによって，Q と $\overline{Q}$ が相補になるように落ち着く)．このため，R-S フリップ・フロップ回路では，S と R を共に “1” として入力することを “禁止” としている場合が多い．

表 12: R-S フリップ・フロップ回路の真理値表

$S^{(n)}$	$R(n)$	$Q^{(n+1)}$	$\overline{Q}^{(n+1)}$	
0	0	$Q^{(n)}$	$\overline{Q}^{(n)}$	
0	1	0	1	
1	0	1	1	
1	1	1	0	禁止

・実習

図 3.21 のように，パネル上の R-S フリップ・フロップの素子を使用し，S，R の入力レベルに対する出力レベル Q，$\overline{Q}$ のレベルを表示器で確認した．

・考察

禁止入力を行ったときの各ビットの状態の遷移について考察する．まず，$(S,\ R) = (0,\ 0),\ (0,\ 1), (1,\ 0)$ の順に入力すると，図 3.A(a)(b)(c) のように，NAND ゲート素子 G1，G2 のどちらが先に開いても，Q と $\overline{Q}$ は 1 つの相補状態で安定する．その後 $(S,\ R) = (1,\ 1)$ を入力すると，図 3.A(d) のように $Q = 1,\ \overline{Q} = 1$ となり相補関係が満たされなくなるが，出力は 1 種類で安定している．しかし次に同時に $(S,\ R) = (0,\ 0)$ を入力すると，G1 が先に開いた場合は $Q = 0,\ \overline{Q} = 1$，G2 が先に開いた場合は $Q = 1,\ \overline{Q} = 0$ となり，どちらのゲートが先に開くかによって Q と $\overline{Q}$ の値が変わって，出力が不定となってしまう (図 3.A(e)(f))．すなわち，本来値を保持するはずの入力 $(S,\ R) = (0,\ 0)$ を行っているにも関わらず，値が変わってしまう．そこで，$(S,\ R) = (1,\ 1)$ を禁止入力とすることで，値の記憶機能を保っているのである．

(11) J-K フリップ・フロップ回路

・目的

1. J-K フリップ・フロップがトリガ型フリップ・フロップ回路であることを理解する．
2. J，K をともに “1” としたときには，T フリップ・フロップにもなることを理解する．

・理論

R-S フリップ・フロップ回路は，R，S のレベルが直接出力を決定するのに対して，J-K フリップ・フロップ回路は，J と K のレベルの他に，トリガが加えられないと，出力が決定されない，トリガ型のフリップ・フロップ回路の一種である．

論理式，回路，真理値表は，各々式 (0.0.20)，(0.0.21)，図 3.22(a)，表 0.13 で表される．

$$Q^{(n+1)} = \overline{K^{(n)}} \cdot Q^{(n)} + J^{(n)} \cdot \overline{Q}^{(n)} \tag{20}$$

$$\overline{Q}^{(n+1)} = \overline{Q^{(n+1)}} \tag{21}$$

動作は次のようになる．

1. J と K が “0” レベルのときの出力 Q は，トリガパルス T が加えられても元の状態を保持する．
2. J が “1”，K が “0” のときの出力 Q は，トリガパルス T が加えられると “1” になり，この状態でさらにトリガパルスが加えられても、元の状態 (“1”) を保持する．
3. J が “0”，K が “1” のときの出力 Q は，トリガパルス T が加えられると “0” となり，この状態でさらにトリガパルスが加えられても、元の状態 (“0”) を保持する．
4. J と K が “0” レベルのときの出力 Q はトリガパルス T が加えられるごとにレベルが反転する．(T フリップ・フロップ:トグルフリップ・フロップ)

5. PC を “0” にすると，J，K，T のレベルに関係なく，出力 Q は “0” になる．(Pre-Clear)

なお，トリガパルス入力端子 (図 3.22(a) における T と書かれた端子) の先端は否定回路同様に “◯” と書かれるが，これはトリガパルスの電圧下降部でトリガされることを意味する．

表 13: J-K フリップ・フロップ回路の真理値表

$J^{(n)}$	$K^{(n)}$	T	$Q^{(n+1)}$	$\overline{Q}^{(n+1)}$	動作
0	0	↓	$Q^{(n)}$	$\overline{Q}^{(n)}$	ホールド
0	1	↓	0	1	リセット
1	0	↓	1	0	セット
1	1	↓	$\overline{Q}^{(n)}$	$Q^{(n)}$	トグル

PC	$Q^{(n+1)}$
0	0
1	$Q^{(n)}$

・実習

図 3.23 のように，パネル上の J-K フリップ・フロップの素子を使用し，J，K の入力に対してトリガパルスを加えたときの出力 Q，$\overline{Q}$ のレベルを表示器で確認した．

トリガパルスには，パネル上のパルス発生器から手動による単発パルスを用いた．また，PC 端子には，同じパルス発生器のクロックパルスを使用した．

・考察

J-K フリップ・フロップは，R-S フリップ・フロップとは異なり，禁止入力がないため，扱いやすい記憶素子であると考えられる．

また，J-K フリップ・フロップ回路は，トリガパルスを用いることで，出力のタイミングを自由に決めることができるという利点がある．この性質を用いると，0 から $N-1$ までカウントし，カウントが N になった時点でカウントを 0 にリセットする N 進アップカウンタをつくることができる．J-K フリップ・フロップ回路ではトリガパルスの電圧下降時にトリガされることから，下位からの桁上げを下位ビットの $1 \to 0$ の反転 (すなわち，下位ビットの電圧降下) により検知し，その電圧降下をトリガとして用いることによって自分のビットを反転させる．これにより，10 進で 1 カウントアップすることができる．また，各ビットの出力を並べた値が $N_{(10)}$ に対応する 2 進数に達した瞬間にだけ出力が “1” となる回路を構成し，その出力をすべての J-K フリップ・フロップの PC に用いることで値を “0” にリセットすることができる．このようにして N 進カウンタをつくることができる．

0.1 NAND 回路による組み合わせ回路の構成

組み合わせ回路は全て NAND 回路で構成できることを示す．基本素子 AND，OR，NOT，NOR が NAND のみで構成できることが分かれば十分であるから，これを例を挙げながら示していくことにする．なお，NAND の真理値表は表 0.3 の通りである．

(1) NOT 回路

図 3.B(a) のように，NAND 素子に同一の “A” 2 つを入力することにより，次の真理値表 0.14 が得られ，$\overline{A} = \overline{A \cdot A}$ であることが分かる．したがって，NOT 回路は NAND 素子 1 つで実現することができる．実際，$X = X + X$ が成り立つこととド・モルガンの定理を用いれば，

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

となる．

表 14: $\overline{A}$, $\overline{A \cdot A}$ の真理値表

A	A	$\overline{A}$	$\overline{A \cdot A}$
0	0	1	1
1	1	0	0

(2) AND 回路

$A \cdot B = \overline{\overline{A \cdot B}}$ であることを用いれば，NAND 素子の出力 $Y_1 = \overline{A \cdot B}$ を NOT 素子に入力することで，AND 回路を構成することができる．したがって，(1) を考慮すると，図 3.B(b) のように NAND 回路を 2 つ用いて論理回路を構成すればよい．

表 15: $A \cdot B$, $\overline{\overline{A \cdot B}}$ の真理値表

A	B	$A \cdot B$	$Y_1 = \overline{A \cdot B}$	$\overline{Y_1}$
0	0	0	1	0
0	1	0	1	0
1	0	0	1	0
1	1	1	0	1

(3) OR 回路

ド・モルガンの定理より，$A + B = \overline{\overline{A + B}} = \overline{\overline{A} \cdot \overline{B}}$ であることを用いれば，入力 A，B を NOT 素子に入力し，その出力を NAND 素子に入力することで OR 回路を構成することができる．具体的には，(1) を考慮して，図 3.B(c) のように NAND 回路を 3 つ用いて論理回路を構成すればよい．

表 16: $A+B$, $\overline{\overline{A}\cdot\overline{B}}$ の真理値表

A	B	$A+B$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$\overline{\overline{A}\cdot\overline{B}}$
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	0	1	0	1
1	1	1	0	0	0	1

(4) NOR 回路

(3) で構成した OR 回路の出力を (1) で構成した NOT 素子に入力することで，NOR 回路を構成することができる．具体的には，図 3.B(d) のように NAND 素子を 4 つ用いて論理回路を構成すればよい．

以上により，基本素子がすべて NAND 素子のみで構成可能であることが分かった．これにより，基本素子で構成されるすべての組み合わせ回路についても，NAND 素子のみで構成可能であることが示された．

0.2　2 値論理における減算の理論

減算を行う回路には，加算と同様に，半減算器と全減算器がある．2 値論理における減算には図 3.C(a) に示すような 4 つの基本演算がある．$0-1$ のときはその桁から引けないので，1 つ上の桁から 1 を借りるという操作が必要である．$D=A-B$，及び桁借り B_O として，半減算器のシンボルと真理値表はそれぞれ，図 3.C(a)，表 0.17 のようになる．

表 17: 半減算器の真理値表

A	B	B_O	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

真理値表から図 3.C(c)(d) のような半減算回路が構成できる．また，加算と同様，半減算回路を 2 つ用いることで，桁借りを考慮した全減算回路を作ることができる (図 3.D)．

また，$A-B=A+(-B)$ であることを利用して，補数表示した 2 つの数値を加算するという方法で減算を実現することも可能である．

0.3　同期・非同期カウンタの回路と動作原理

非同期式 4 進カウンタと同期式 4 進カウンタの回路を図 3.E に示した．これらのカウンタは，$Q_1Q_0 =\in \{00, 01, 10, 11\}$ の値をクロックパルスが与えられるごとにカウントアップする回路である．非同期カウンタでは，Q_0 の値が $1 \to 0$ に下がったときに Q_1 の値が反転することを利用し，Q_0 を 2 桁目の加算のクロックに用いている．一方同期カウンタでは，Q_0, Q_1 ともに同一のクロックパルスを用い，また J_1，K_1 に Q_0 を入力することにより Q_0，Q_1 に対して演算を行っている．

1　結論

本実験では，論理回路の基本的な素子の動作とその応用を理解した．とりわけ，加減算回路などは計算機設計などでは欠かせない分野であり，さらに理解を深める必要があると思われる．

参考文献

[1] 東京理科大学工学部情報工学科 (2017) 『情報工学実験 1:平成 29 年度』

[2] 大類重範 (2010-2017) 『ディジタル電子回路』 日本理工出版会

[3] 松本洋平 『情報処理基礎論』 [online]http://www2.kaiyodai.ac.jp/ matumoto/lecture02/ 2017 年 5 月 28 日最終閲覧

[4] “Camp Network” [online]http://capm-network.com/ 2017 年 5 月 28 日最終閲覧

[5] 『エンコーダとデコーダ』 [online]http://home.a00.itscom.net/hatada/dc2/chap15/decoder.html 2017 年 5 月 28 日最終閲覧

[6] 『RS フリップフロップの禁止 (不定) について』 [online]http://shusaku721-bibou6.seesaa.net/article/441811548.html 2017 年 5 月 28 日最終閲覧