第2章　ディジタル回路

現在我々の生活は、多種多彩なディジタル機器によって支えられている。それらは全て、基本的なディジタル回路の組合わせによって複雑な機能を実現している。本課題は、ディジタル回路設計の基本的事項について学ぶ事を目的とする。実験では、基本ゲート回路の動作および論理演算を学びつつ、ディジタル回路設計の基礎を身につける。また FPGA（Field Programmable Gate Array）の使用法と、それを利用した回路の作成法についても学ぶ。

2.1　ディジタル回路の基本

2.1.1　二値変数

ディジタル回路の入出力は、“1”または “0”に対応する 2 つの値しか許されない。すなわち、入出力の状態は**二値変数**（binary variable）によって表される。この “1”を真（truth）、“0”を偽（false）とする論理系を正論理系と呼び、その逆を負論理系と言う。一般的には、“1”に対して高電圧、“0”に対して低電圧 （∼ 0 V）を対応させ、 これをそれぞれ High 状態（H レベル）、Low 状態（L レベル）と言う。一般的なディジタル回路（論理演算回路）は、入力された二値変数の論理演算を行い、それを二値変数で出力するものである。この入力と出力の関係を 1 つの表にまとめたものを**真理値表**と呼ぶ。

2.1.2　基本ゲート回路

どのような複雑なディジタル回路であっても、いくつかの最も基本的な論理回路の組合せからできている。それらは、**NOT**（否定）、**AND**（論理積）、**OR**（論理和）であり、これらの機能をもった素子を NOT ゲート、AND ゲート、OR ゲートと言う。全ての論理演算回路は、これら 3 つのゲートを基本的な要素として作る事が出来る。一般に、回路図上の論理ゲートは MIL（ミル）論理記号で表現される。

NOT (否定)

入力が “1”の時に出力が “0”、入力が “0”の時に出力が “1”となるように、入力を反転（否定）する機能を NOT 機能と言う。これを実現する素子を NOT ゲートまたはインバーター（inverter）と言う。入力 X の反転または否定の論理式は、X にバーを付けて $\overline{\mathrm{X}}$ と表す。

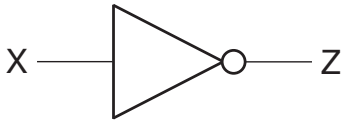

図 2.1: NOT ゲートの MIL 記号

X	$Z=\overline{X}$
0	1
1	0

表 2.1: 否定 (NOT) の真理値表

AND (論理積)

入力がともに “1”の時にだけ出力が “1”になるように、論理積をとる機能を AND 機能と言い、これを実現する素子を AND ゲートと言う。入力が n 個ある場合も同様で、入力信号の論理積が出力に現れる。n 個の入力の全部が “1” の場合のみ出力が “1”となる回路を n 入力の AND 回路という。X と Y の論理積は、論理式で $X \cdot Y$ と表す。

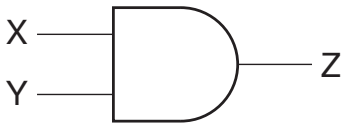

図 2.2: AND ゲートの MIL 記号

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

表 2.2: 論理積 (AND) の真理値表

OR (論理和)

入力が一つでも “1”であれば出力が “1”になるように、論理和をとる機能を OR 機能と言い、これを実現する素子を OR ゲートと言う。入力が n 個ある場合も同様で、入力信号の論理和が出力に現れる。n 個の入力のどれか一つでも “1” であれば出力が “1”となる回路を n 入力の OR 回路という。X と Y の論理和は、論理式で $X + Y$ と表す。

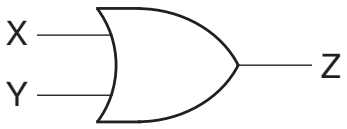

図 2.3: OR ゲートの MIL 記号

X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

表 2.3: 論理和 (OR) の真理値表

NAND (否定論理積)

X と Y の論理積の否定（否定論理積）をとる演算を論理式で表すと、$\overline{X \cdot Y}$ となる。この機能を実現する素子を NAND ゲートと言う。論理式上で否定論理積は論理積と否定の組合せから成っているが、ダイオードやドランジスタを用いてゲートを実現する際、NAND は AND や OR よりも扱いやすいという利点がある。しかも NAND の論理は完全系であり、他の全ての論理ゲートは NAND ゲートのみで構成可能である。例えば NOT 演算は $\overline{X} = \overline{X \cdot X}$、AND 演算は $X \cdot Y = \overline{\overline{X \cdot Y}}$、OR 演算は $X + Y = \overline{\overline{X} \cdot \overline{Y}}$ と書き直せる。

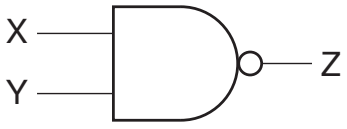

図 2.4: NAND ゲートの MIL 記号

X	Y	$Z = \overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

表 2.4: 否定論理積 (NAND) の真理値表

入力が n 個ある場合も同様で、n 個の入力のどれか一つでも “0” であれば出力が “1” となる回路を n 入力の NAND 回路と言う。

NOR (否定論理和)

X と Y の論理和の否定（否定論理和）をとる演算を論理式で表すと、$\overline{\mathrm{X}+\mathrm{Y}}$ となる。この機能を実現する素子を NOR ゲートと言う。入力が n 個ある場合も同様で、n 個の入力の全部が “0” の場合のみ出力が “1” となる回路を n 入力の NOR 回路と言う。NOR の論理も完全系である。

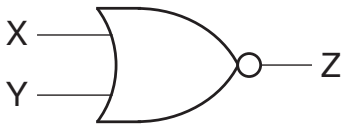

図 2.5: NOR ゲートの MIL 記号

X	Y	$\mathrm{Z}=\overline{\mathrm{X}+\mathrm{Y}}$
0	0	1
0	1	0
1	0	0
1	1	0

表 2.5: 否定論理和 (NOR) の真理値表

XOR (排他的論理和)

入力が異なる場合のみ出力が “1” となる演算を排他的論理和（exclusive OR）と言い、この機能を実現する素子を XOR ゲートと言う。X と Y の排他的論理和演算は、論理式で $\mathrm{X}\oplus\mathrm{Y}=\mathrm{X}\cdot\overline{\mathrm{Y}}+\overline{\mathrm{X}}\cdot\mathrm{Y}$ と表される。

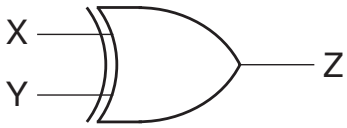

図 2.6: XOR ゲートの MIL 記号

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

表 2.6: 排他的論理和 (XOR) の真理値表

2.1.3 論理演算

二値変数を扱う代数は、ジョージ・ブール（George Boole）が考案した代数系、所謂ブール代数として完成されている。このブール代数の演算規則と諸法則を、表 2.7 と表 2.8 にまとめる。これらの法則を使えば、次節で述べるように代数的に論理式の簡略化を行うことが出来る。

2.2 回路設計法

2.2.1 真理値表による方法

ある機能を持った論理回路において、入力変数の組に対して出力が一意的に定まる場合、それを**組合せ回路**と言う。組合せ回路の論理式は、真理値表から簡単に作る事が出来る。以下にその手順を述べる。

(1) n 個の入力変数 $\mathrm{X_i}$ $(\mathrm{i}=1,2,\cdots,n)$ と出力 Z の関係を真理値表に書く。

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=1$
$0\cdot 0=0$
$0\cdot 1=0$
$1\cdot 0=0$
$1\cdot 1=1$
$\overline{1}=0,\ \overline{0}=1$

表 2.7: ブール代数の演算規則

交換則	$A\cdot B=B\cdot A$ $A+B=B+A$
吸収則	$A\cdot(A+B)=A$ $A+(A\cdot B)=A$
結合則	$(A\cdot B)\cdot C=A\cdot(B\cdot C)$ $(A+B)+C=A+(B+C)$
分配則	$A\cdot(B+C)=A\cdot B+A\cdot C$ $A+(B\cdot C)=(A+B)\cdot(A+C)$
ド・モルガンの法則	$\overline{A\cdot B}=\overline{A}+\overline{B}$ $\overline{A+B}=\overline{A}\cdot\overline{B}$
0 と 1 に関するもの	$A\cdot 1=A$　　$A\cdot 0=0$ $A+1=1$　　$A+0=A$

表 2.8: ブール代数の諸法則

(2) 出力 Z が “1” になる欄のみに注目し、$X_i=0$ の場合は $\overline{X_i}$、$X_i=1$ の場合は X_i としてこれらの論理積 (AND) を取る。X_i が “0” と “1” どちらでもよい場合は、その入力変数は無視する。

(3) (2) で求めた論理式の論理和 (OR) を取る。

(4) ブール代数の法則を用いて、論理式を簡略化する。

この手順に従って、次の例で真理値表から論理式を作ってみよう。

X_1	X_2	X_3	Z	
0	0	0	0	(無視)
1	0	d	1	$X_1\cdot\overline{X_2}$
0	0	1	1	$\overline{X_1}\cdot\overline{X_2}\cdot X_3$
1	1	0	0	(無視)
0	1	d	0	(無視)
1	1	1	1	$X_1\cdot X_2\cdot X_3$

表 2.9: 真理値表から論理式を作る手順

真理値表において “d” は don’t care の意味を表す記号であり、“0” と “1” どちらでもよいことを表す。表 2.9 の右側に記したように、$Z=1$ となる欄で入力（またはその否定）の論理積を作り、それらの論理和を取ると、

$$Z=X_1\cdot\overline{X_2}\ +\ \overline{X_1}\cdot\overline{X_2}\cdot X_3\ +\ X_1\cdot X_2\cdot X_3 \tag{2.1}$$

となる。これは上記 (3) までの手順であり、この回路設計法を**加法標準形**による設計と呼ぶ。

この論理式をそのままゲート回路で作成しても、所定の機能を持った回路を作る事は出来るが、実装においては出来るだけゲート数を少なくして、電力および回路作成上の手間を減らす事が望ましい。上記の例では、例えば第 2 項をド・モルガンの法則により、

$$Z=X_1\cdot\overline{X_2}\ +\ \overline{X_1+X_2}\cdot X_3\ +\ X_1\cdot X_2\cdot X_3 \tag{2.2}$$

と変形すれば、使用するゲート数を 2 つ減じる事が出来る。

2.2.2 カルノー図

論理式の簡略化において威力を発揮するのが、この**カルノー図**（Karnaugh Map）を使用する方法である。カルノー図に記される情報は、基本的には真理値表と何ら変わりはない。しかし特有の規約に沿って作られ

るカルノー図は、真理値表よりも簡潔で見易い上に、変数間における数値の関係をより容易に把握する事が出来る。ここに、カルノー図を使用した簡略化の具体的な手順を示す。

カルノー図は出力変数毎に作成する。まず複数の入力変数を 2 組に分け、それぞれ行と列に並べて表を作成する。入力変数の並べ方には規約があり、(1) 隣り合う行（列）は一変数しか変化できない。この規約に従って入力変数全ての組を並べると、端同士もまた一変数しか変化しない事になる。故に、カルノー図の (2) 端と端は隣り合っていると見ることができる。表 2.10、2.11 に 3 変数と 4 変数の場合のカルノー図の例を示す。

X_1X_2 \ X_3	0	1
00	1	1
01	0	0
11	1	0
10	1	0

表 2.10: 3 変数のカルノー図

X_1X_2 \ X_3X_4	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	0	0	0	0
10	0	0	0	0

表 2.11: 4 変数のカルノー図

4 変数以下のカルノー図による簡略化の手続きは以下の様にまとめられる。

(1) n 個の入力変数 X_i $(i = 1, 2, \cdots, n)$ と出力 Z の関係をカルノー図で表す。

(2) 出力 Z が “1” になっている部分のみに注目し、それを全て四角形で囲む。但し、四角形の一辺の長さは 2^n (n は 0 以上の整数) とし、重畳して囲んでもよい。また端では上下・左右につなげられる。d は “1” として囲めるが、囲まなくてもよい。

(3) 上で描いた四角形それぞれについて、$X_i = 0$ の場合は $\overline{X_i}$、$X_i = 1$ の場合は X_i としてこれらの論理積 (AND) を取る。X_i が “0” と “1” の両方にまたがる場合は、その入力変数は無視する。

(4) (3) で求めた論理式の論理和 (OR) を取る。

論理式を簡易化するため、四角形をなるべく多く囲むように取ると論理積の数を減らせる。また、四角形の数をなるべく減らすと、論理和の数を減らせる。どの囲み方でも、等価な論理式が導出される。

例えば、表 2.11 で与えられた X_i と Z の関係を論理式で表すことを考える。出力 Z が “1” になっている欄は 2 カ所あり、これらは同じ行の両端に位置するため隣接していると見る。この部分を四角形で囲み、論理式を考えると直ちに $Z = \overline{X_1} \cdot X_2 \cdot \overline{X_4}$ を得る。この簡略化は、「X_i が “0” と “1” どちらでもよい場合は、その入力変数は無視できる」という原理に基づいており、この事が規約 (1) によって極めて直感的に行えるようになっている。これを全て論理式で書くならば、

$$\begin{aligned} Z &= \overline{X_1} \cdot X_2 \cdot X_3 \cdot \overline{X_4} + \overline{X_1} \cdot X_2 \cdot \overline{X_3} \cdot \overline{X_4} \\ &= \overline{X_1} \cdot X_2 \cdot (X_3 + \overline{X_3}) \cdot \overline{X_4} \\ &= \overline{X_1} \cdot X_2 \cdot \overline{X_4} \end{aligned} \tag{2.3}$$

というブール代数に基づいた式変形に裏付けられている事が分かる。

2.3 集積回路

2.3.1 ディジタル集積回路

§2.3.2 節で解説した通り、ディジタル回路の基本要素である論理ゲートは集積回路 (IC) として用意されている。論理ゲートのみならず、それを組合せて様々な機能を有する IC が製品として販売されている。ここではいくつかの代表的なディジタル IC を例に挙げ、その使用方法を学習する。

2.3.2 ディジタル回路の実際

IC (Integrated Circuit)

ディジタル回路の基本となる論理ゲートは、ダイオードやトランジスタを用いて構成することができる。しかしながら、ゲート回路を組み合わせた複雑な論理回路を一から構築する事は、膨大な手間とコストを必要とする。そのため、よく使用されるゲート回路について、その機能を持つようにダイオードやトランジスタを組み合わせた集積回路（IC：Integrated Circuit）が作られ、市販されている。一般によく使用されているのが、トランジスタを中心に作られた TTL（Transistor-Transistor Logic）形と、FET（電界効果型トランジスタ）を用いた C-MOS (Complementary Metal-Oxide-Semiconductor) 形である。両者の間で、電源電圧の使用範囲、消費電力、動作速度等に違いはあるが、多くものは互換性があるように作られている。実際に、ディジタル回路を作成する場合は、これら個々のゲートの内部構造を知らなくても、適当な IC を組み合わせる事によって、欲しい機能を有する回路を構成する事が出来る。本実験では、TTL を用いた回路作成を行う。

TTL（Transistor-Transistor Logic）

TTL には、いくつかの種類があるが、本実験では、LS-TTL を使用する。各端子は、その役割が決まっている。図 2.7 に示すように、端子の番号の付け方は、上面から見て目印を左側にしたとき、真下のピンが 1 番で反時計回りに $1, 2, \ldots$ となる。

スレッショルド・レベルと正論理/負論理

ディジタル回路では、入出力信号を、電圧の高低で表現する。TTL では、論理値の “1” と “0” を、2 つの電位状態 H (High) と L (Low) に対応させる。H と L を区別する電圧をスレッショルド・レベルといい、今回使用する TTL はグランド（基準電圧）を 0 V とした時、約 2.4 V 以上を High、0.4 V 以下を Low と判断する。信号 (1, 0) と電圧 (H, L) を対応させる場合、H → 1, L → 0 に対応させたものを「正論理 (positive logic)」、H → 0, L → 1 に対応させたものを「負論理 (negative logic)」という。2 つの対応は論理的には同等でありどちらを用いてもよいが、実際のディジタル回路では、対雑音性に優れた負論理を使用することが多い。

ファンアウト数

ディジタル回路を作成する際、論理ゲートを何段も接続する必要がある。このとき、ひとつのゲート出力に何個のゲートを接続できるか（ひとつのゲートが何個のゲートを駆動できるか）が問題となる。この数をファンアウト（fanout）数という。例えば、ファンアウト数が 10 であれば、10 個までの SN7400 を出力に接続できる。ファンアウト数以上接続すると動作が不安定になるか、出力が H/L 判別の電圧条件を満たさなくなる。なお、この数は TTL 規格表に示されている。本実験で使用する TTL（LS-TTL）のファンアウト数は 20 である。

LED（Light Emitting Diode）

回路によって演算された結果（出力電圧）を、人間にわかる形で表示させる必要がある。本実験では発光ダイオード（LED：Light Emitting Diode）を使用する。一般的な砲弾型 LED 外形と電極の関係は、図 2.8 の様になっている。LED は普通のダイオードと同じように電流の流れる向きが決まっており、赤色発光ダイオードでは順方向に 1.8 V 以上の電圧で電流が流れたときに発光する。

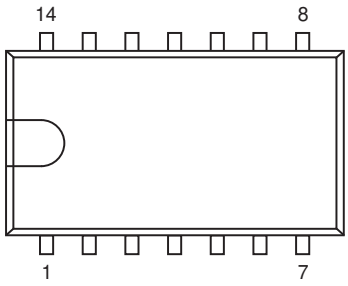

図 2.7: 14 ピン IC のピン配列

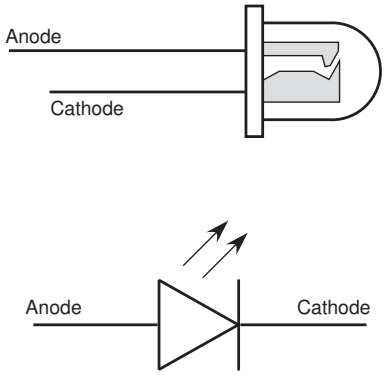

図 2.8: 発光ダイオード (LED) の概形と回路記号

基本ゲート IC

NAND、NOR、NOT、AND ゲート IC の型番とピン配置を以下に示す。これらは全て 14 ピンの IC であり、それぞれ複数個の論理ゲートが入っている。V_{CC} 端子に電源電圧 (∼ +5 V) を印加し、GND 端子の間を接地する事で動作する。

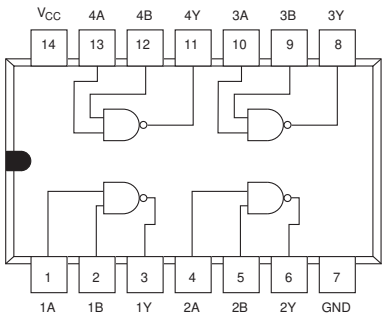

図 2.9: SN7400 (2 入力 NAND×4) のピン配置

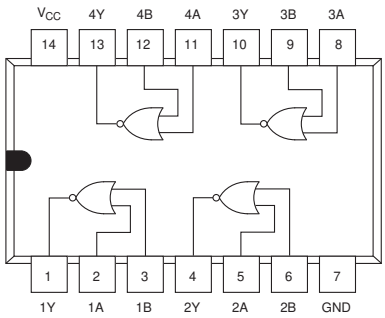

図 2.10: SN7402 (2 入力 NOR×4) のピン配置

ディジタル回路の作成手順

以上のことを踏まえ、ディジタル回路は以下の手順で設計・作成される。

1. 必要な機能から入力と出力を整理し、入出力関係を考える。
2. それらを “1” と “0” で表す（真理値表）。
3. 出力信号を入力信号の論理式で表す。

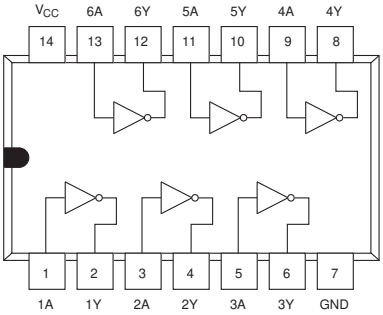

図 2.11: SN7404 (NOT×6) のピン配置

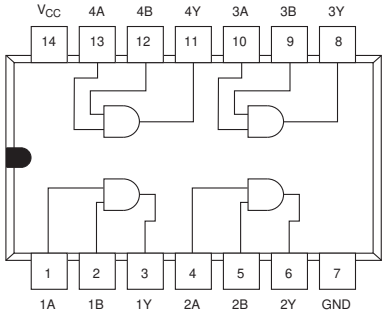

図 2.12: SN7408 (2 入力 AND×4) のピン配置

4. 得られた論理式を、ブール代数またはカルノー図を用いて簡略化する。

5. “1”を H に、“0”を L 読み替え (正論理)、得られた論理式を論理ゲートを使って構成する。

6. 動作チェック（正常に動かない場合は、1.〜5. に戻る）。

ディジタル回路は、論理設計と回路作成が正しく行われてさえいれば、初心者でも正常に動作するものを作成する事が出来る。

2.3.3 練習機「DIGITAL IC TRAINER」

「DIGITAL IC TRAINER」を用いてディジタル IC の使用方法を学習する。

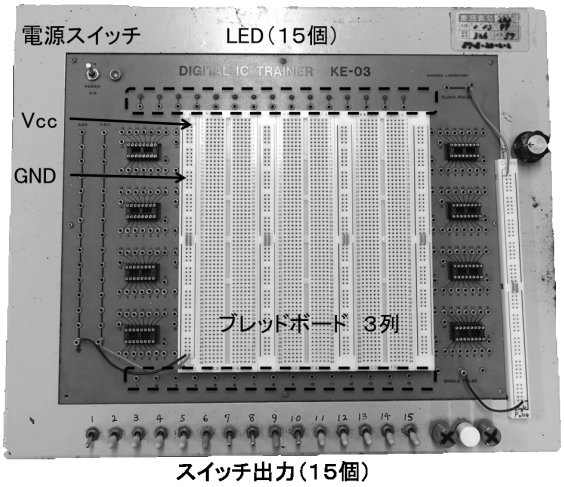

図 2.13: 練習機「DIGITAL IC TRAINER」の外観

概観

この練習機では、IC ソケットに入った DIP(Dual Inline Package)IC と配線をブレッドボードに挿して使用する。ブレッドボードは 3 つ用意されており、それぞれの中心に E と F をまたぐよう IC を差し込む。同じ横列の A から E と F から J はそれぞれ導通しており、配線のために用いる。使用するには、まず電源プラ

グを実験机横の電源タップに挿し込み、電源タップのスイッチと本体左上の電源スイッチ (POWER SW) を入れる。電源が投入されると、電源スイッチ右の電源 LED が緑色に点灯する。

スイッチ出力

本体下部に 15 個のスイッチが並んでおり、スイッチの状態に応じてそのすぐ上側の 1 ～ 15 端子より L（“0”）または H（“1”）が出力される。

VCC 端子、GND 端子

白色のブレッドーボード左端の縦 1 列は 64 個の “GND”端子として使用でき、常に L（“0”）を出力する。またそのすぐ右側 1 列には “V_{CC} (+5V)”端子が用意されており、常に H（“1”）を出力できる。

LED

本体上部に、出力状態を可視化するための LED が 15 個装備されている。出力信号をそれぞれ真下の端子に接続すれば、出力が H（“1”）の場合に LED が点灯する。

2.3.4 実験 I

練習機でゲート IC とブレッドボードを用いて、以下の実験を行う。

(a) NOT, AND, NAND, NOR ゲート回路の動作を確かめよ。結果を、真理値表にまとめ、正しいゲート動作と言えるかどうかを報告する。

(b) $n\,(1 \leq n \leq 3)$ 番目のスイッチを入れたとき、2 個の LED により n を二進表示する回路を作れ。真理値表に入力 (3 つのスイッチ) と出力 (2 つの LED）の値をまとめ、論理式を構築し、ゲート IC により作成した回路での動作を報告する。

(c) 二進表示された 0 ～ 3 の 2 つの数 X_1X_2 と Y_1Y_2 で $X_1X_2 > Y_1Y_2$ を判定する回路を作れ。但し、0＝00, 1＝01, 2＝10, 3＝11 である。表示には 1 個の LED を使用する。真理値表に入力と出力の値をまとめ、カルノー図を用いて論理式を構築し、実際に作成した回路の動作を報告する。

2.4 加算器

2.4.1 算術演算

これまで二値変数の論理演算を考えてきたが、ここでは整数の算術演算を論理演算で実現する。整数の加算は、足し算の筆算で繰り上がりを考えながら、下の位から桁ごとに計算する要領で考えればよい。2 進数で表される 2 つの数を、桁ごとに分解して加算するための論理回路として、半加算器と全加算器を考える。

2.4.2 半加算器

半加算器 (Half adder) は、2 進数の 1 ビット (1 桁) どうしの和を考えるが、和は 1 ビットを越える (2 桁になる) こともある。このために、大きな桁が生じた場合の桁上げ出力を考える。この演算で生じる桁上りのみ考えるため、最下位の桁の演算に用いられる。

入力 A、入力 B、出力 S (Sum)、桁上げ出力 C (Carry out) とすると、A+B = C S となるように真理値表を考えれば、表 2.12 のように表せる。S は排他的論理和の真理値表 2.6 と見比べると、$S = A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$ と表されることが分かる。C は $C = A \cdot B$ となる。

A	B	C	S	対応する演算
0	0	0	0	(0+0) = 0
0	1	0	1	(0+1) = 1
1	0	0	1	(1+0) = 1
1	1	1	0	(1+1) = 10

表 2.12: 半加算器の真理値表

2.4.3 全加算器

半加算器では、下の桁から繰り上がってきた数を加算していない。上の桁へ繰り上がる数 C (Carry out) だけでなく、下の桁から繰り上がってきた数 C_i (Carry in) も考えることで、全加算器 (Full adder) は最下位より上の桁どうしの加算でも用いられる。入力 A、入力 B、同じ桁の出力 S (Sum)、桁上げ出力 C (Carry out) とすると、A+B+C_i = C S となるように考えれば、表 2.13 のように表せる。S は $S = A \oplus B \oplus C_i$ で表され、C は $C = A \cdot B + B \cdot C_i + C_i \cdot A$ となる。

A	B	C_i	C	S	対応する演算
0	0	0	0	0	(0+0)+0 = 0
0	0	1	0	1	(0+0)+1 = 1
0	1	0	0	1	(0+1)+0 = 1
0	1	1	1	0	(0+1)+1 = 10
1	0	0	0	1	(1+0)+0 = 1
1	0	1	1	0	(1+0)+1 = 10
1	1	0	1	0	(1+1)+0 = 10
1	1	1	1	1	(1+1)+1 = 11

表 2.13: 全加算器の真理値表

n 桁と m 桁の 2 進数加算演算 ($n \geq m$) には、1 つの半加算器と n−1 つの全加算器を用いればよい。最下位から順に、桁上げ出力を 1 つ上の桁の C_i に順次つなげることで、複数ビットの演算も可能になる。

2.5 FPGA

2.5.1 FPGA ボード

FPGA (Field Programmable Gate Array) は、書き換え可能な論理素子の集合体であり、大規模な論理演算を任意に設計できる集積回路である。産業や研究用途で広く使われており、一般には複雑な設計をハードウェア記述言語 HDL (Hardware Description Language) から論理合成し、複雑な論理回路の実装設計が自動化されている。今回は HDL を用いず、回路図により FPGA へ論理回路をプログラミングする。

この実験で用いる Terasic 社の練習機「DE0-CV」は、約 5 万個の論理素子を持つ Intel 社の Cyclone V FPGA に、スイッチや LED 表示、USB 接続やメモリが搭載されているボードである。図 2.14 中に示すスライドスイッチ・プッシュボタン・7 セグメント LED・チップ LED を使って、FPGA にプログラムされた論理回路の操作・確認を行う。論理回路設計時に、パラメータ名を用いて各入出力を区別する。

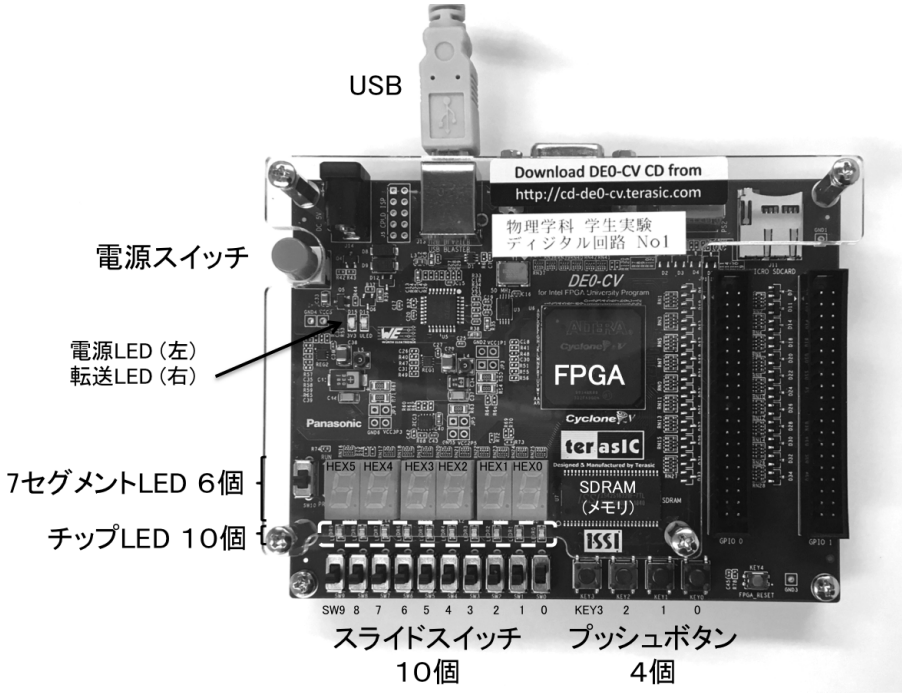

図 2.14: 練習機「DE0-CV」の外観

スライドスイッチ

左下に並ぶ１０個のスライドスイッチは、右からパラメータ名 SW0~SW9 で指定できる。

プッシュボタン

右下に並ぶ４個のスライドスイッチは、右からパラメータ名 KEY0~KEY3 で指定できる。

7 セグメント LED

10 進数の数字を表示したい場合、7 つのセグメントを持つ LED（7 セグメント LED）を用いると便利である（図 2.15）。各セグメント (a∼g) は独立しており、それぞれ独立に発光できるようになっている。図 2.16 をみてわかるように、電圧 V_{CC} 入力が共通してつながっており、他方を 0 V (GND) に繋げると順方向に電流が流れ、発光するようになっている。右端の 7 セグメント LED で a∼f に対応するパラメータ名は HEX0a∼HEX0g であり、6 個の 7 セグメント LED は右から HEX0 から HEX5 で区別される。

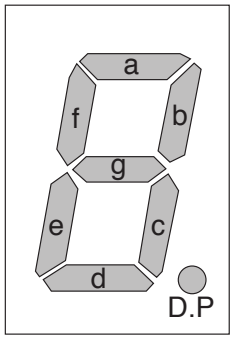

図 2.15: 7 セグメント LED

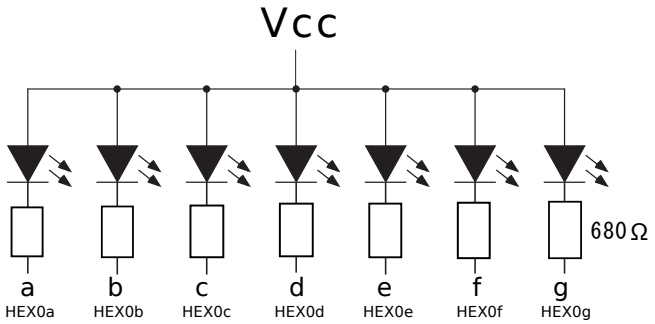

図 2.16: コモンアノード型 7 セグメント LED の内部構造

チップ LED

小さなチップ LED が 7 セグメント LED の下に 1 0 個配置されている。パラメータ名は右から LEDR0 から LEDR9 であり、High (V_{CC}) にすると点灯し、Low (GND) で消灯する。接続されず、Low でも High でもない場合は薄く点灯するが、無視してよい。

起動

動作させるために、まずコンピュータを電源をつないで起動させ、USB ケーブルで練習機と接続する。左上の赤い電源ボタンを押すと起動する。逆順で電源を落とせるが、赤い電源ボタンの近くの LED が 2 つ点灯している時に電源を切らないようにする。

2.5.2 プログラミング

コンピュータで、Intel 社製 FPGA をプログラムできるソフトウェア Quartus Prime (クォータス・プライム) の Lite Edition を用いて FPGA に論理回路を構築する。視覚的にマウスを使って、論理回路の配置が可能である。デスクトップ上から、該当実験課題のアイコンをダブル・クリックすると起動する。大まかに、以下の手順でプログラミングできる。

I. 論理回路作成

- マウスで、Input (KEY0–3、SW0–9) 、Symbol (GND、V_{CC}、論理ゲート等)、Output (LEDR0–9、HEX0a–HEX0g)、配線を追加し、繋げて回路図を作る。
- 追加中に Esc キーを押すと、追加モードが解除される。マウスアイコンが十字になる場所で配線を追加できる。
- 矢印十字マウスアイコンをクリックして選択されたものを Delete キーで消去でき、ドラッグすると移動できる。

- Input と Output のパラメータ名は、ダブルクリックすると変更できる。

II. コンピュータでのコンパイル (論理合成し、FPGA のコンフィグレーションデータを生成する)

- ツールバーから青の右向き三角アイコンをクリックする (保存していないときにはダイアログが出てくるが、Yes をクリックして進める)。
- 右下の表示が 100%になるまで、数分間待機する。
- 緑文字で “successful. 0 errors” と下部に表示されると、正常にコンパイルが終了する。
- もし赤文字で “unsuccessful.” と下部に表示されると、論理回路に間違いがあるため、I. に戻って配線などを確認する。配線にバツ印がある場合は、配線が壊れている。
- Compilation Report タブはバツマークをクリックして閉じてよい。

III. FPGA へのプログラミング (コンフィグレーションデータを FPGA へ転送する)

- ツールバーから Programmer アイコンをクリックする。
- Start をクリックする。
- 右上の Progress が Success となって緑色に変わるまで、数十秒待機する (何も触れずに待つこと)。
- Programmer ウィンドウは右上のバツマークをクリックして閉じてよい。

IV. DE0-CV での動作確認

細かな手順と具体的な操作方法は、机上の操作マニュアルを参照する。

2.5.3 実験 II

Terasic 社の練習機「DE0-CV」を用いて、以下の実験を行い計算機を作成する。

(a) スライドスイッチ (SW0–9、上側が ON)、プッシュボタン (KEY0–3、押している間に ON)、チップ LED(LEDR0–9)、7 セグメント LED(HEX0–5) が、正論理と負論理のどちらで動作するか確認せよ。動作とは、スイッチの ON または点灯を言う。例えば、V_{CC} または GND 入力をつなげてそれぞれ 1 つで確かめられるとよい。確認の方法と結果を報告する。

(b) SW0 から 2 を用いて、3 つのスライドスイッチで 2 進数 000 ～ 111 を入力し、7 セグメント LED で対応する 1 0 進数の形 (0 ～ 7) を表現する回路を作成する。真理値表に入力 (3 つのスイッチ) と出力 (LED の 7 セグメント）の値をまとめ、論理式を構築し、練習機での動作を報告する。

(c) 半加算器を作成し、動作を真理値表にまとめて報告する。 2 つのスライドスイッチと、出力・桁上げ出力を表示する 2 つの LED を用いて確認する。

(d) 二進表示された 0 ～ 3 の 2 つの数 A_2A_1 と B_2B_1 で $A_2A_1 + B_2B_1$ を演算する回路を作れ。但し、0＝00, 1＝01, 2＝10, 3＝11 である。演算結果 CS_2S_1 の表示には、(b) で作成した 7 セグメント LED を使用する。半加算器 1 つと全加算器 1 つで構築できる。設計と結果をまとめ、実際に製作した回路の動作を報告する。

※ヒント
(a) で作成した回路は確認後に消去してよいが、(b) と (c) で作成する回路は (d) で用いるため、残しておくとよい。(b) では、真理値表を作成する際に正論理・負論理の違いを揃えるよう留意する。

2.6 順序回路

ここまでに解説してきた回路は、入力変数の組 X_i に対して一義的に出力状態が定まるものであった。このような回路を**組合せ回路**と言う。一方、論理ゲートの出力と入力を交差させる事により、出力が入力変数の組 X_i に対して一義的に定まらず、過去の出力状態にも影響されるような回路を構成する事が可能である。このような回路を**順序回路**と言う。順序回路は、情報を記憶する素子や周期的パルスで駆動される様々な論理演算素子の基礎となるものである。順序回路を構成する素子には、ラッチ（latch）はフリップ・フロップ（flip-flop）がある。以下に代表的なものを紹介する。

2.6.1 RS ラッチ

RS ラッチの “RS” は、Reset と Set の意味であり、Reset が 1 の場合に Q が 0 となり、Set が 1 の場合に Q が 1 となる回路である。 2 つの NAND ゲートを図 2.17 のように接続すると、負論理で当該動作が行われるため、入力端子記号 R, S に $\overline{\ }$ を付けている。つまり、この回路では、$\overline{S}$ が最後に “0” になったら Q 出力が “1” に（Set 状態）、$\overline{R}$ が最後に “0” になったら Q 出力が “0” に（Reset 状態）なる。また、$\overline{Q}$ には Q の反対の値が出力される（反転出力）。但し、$\overline{S}$, $\overline{R}$ ともに “0” を入力すると反転しないため、禁止入力としている。

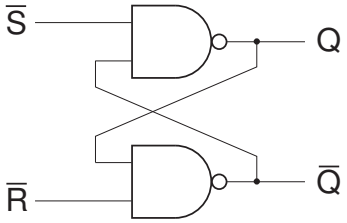

図 2.17: RS ラッチ

$\overline{S}$	$\overline{R}$	Q_{n+1}
0	0	禁止
0	1	1
1	0	0
1	1	Q_n

表 2.14: RS ラッチの真理値表

2.6.2 RST ラッチ

RS ラッチでは入力信号と同時に出力が変化するが、第 3 の入力 T (Trigger または Toggle) の変化に同期して出力が変わるようにしたものが、RST ラッチ（図 2.18）である。この RST ラッチでは、T＝0 の間の入力は出力に影響せず出力は固定され、T＝1 になると RS 入力に従って出力が変化する。S＝R＝1 の入力は禁止であり、Set Reset 動作は正論理で起動される（表 2.15）。

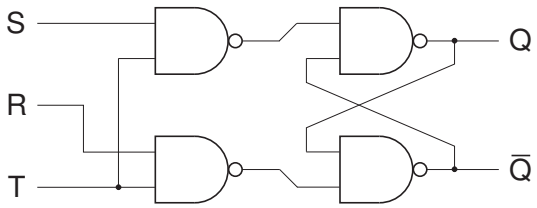

図 2.18: RST ラッチ

S	R	T	Q_{n+1}
d	d	0	Q_n
0	0	1	Q_n
0	1	1	0
1	0	1	1
1	1	1	禁止

表 2.15: RST ラッチの真理値表

2.6.3 D ラッチ

図 2.19 に示された回路は、ある時点の入力信号の値を、その後の入力信号の変化に関わらず保持する働きを持つ。図 2.15 から、D ＝ S ＝ $\overline{R}$ とすることで、T＝1 の間は D の値がそのまま Q に出力され、T＝0 に

なった時点で出力は固定される。この回路を D ラッチ（D は “Data” の意味）と呼ぶ。D ラッチは電子計算機の基本となるものである。$\overline{Q}$ は常に Q の反転出力となっている。

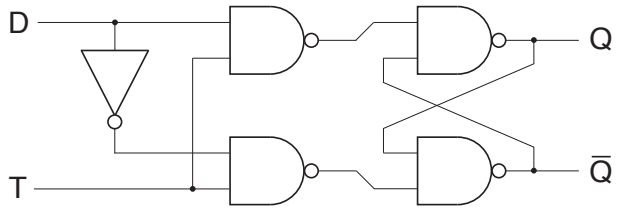

図 2.19: D ラッチ

D	T	Q_{n+1}
d	0	Q_n
0	1	0
1	1	1

表 2.16: D ラッチの真理値表

2.6.4 JK フリップフロップ

2 つの入力信号 J、K の状態に応じて、CLK（Clock の意味）信号に同期して、表 2.17 の真理値表に従った出力 Q を与える回路（図 2.20）を、JK フリップフロップ（JK F/F）と呼ぶ。

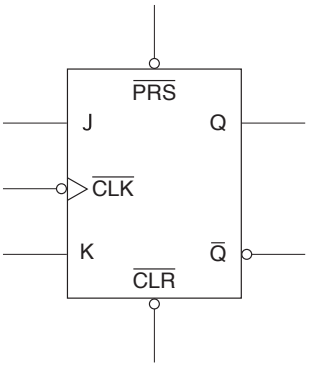

図 2.20: JK フリップフロップの回路記号

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

表 2.17: JK フリップフロップの真理値表

さらに、CLR（“Clear”）、PRS（“Preset”）という 2 つの入力が付け加わったものもあり、これらの入力は CLK、J、K に関わらず、それぞれ “0”、“1” を Q に出力する。図 2.20 の例では、端子名に $\overline{CLR}$、$\overline{PRS}$ と上線が付いており、それぞれの端子に○記号が付いている。これらは当該動作が負論理、つまり入力を “0” にすることによって駆動される事を意味している。

JK F/F は、CLK 信号が “0” から “1” に変化する時に出力が変化する**立ち上がりエッジトリガ型**と、CLK 信号が “1” から “0” に変化する時に出力が変化する**立ち下がりエッジトリガ型**とがある。図 2.20 の例では、端子名が $\overline{CLK}$ かつ端子に○記号が付いているので、**立ち下がりエッジトリガ型**である事が分かる。

2.6.5 カウンタ

カウンタ（計数器）とは、入力信号（例えばクロックパルス）の数をカウントする回路である。この機能を持つ回路を作るために、数個の JK F/F を使用する。全ての JK F/F の出力が同じタイミングで動作するものを「同期式カウンタ」、時間ずれを伴って動作するものを「非同期式カウンタ」という。また、計数器のカウントするサイクルが n であるとき、n 進カウンタとよぶ。カウンタはパルス間隔を遅らせる分周器としても使用できる。

非同期式カウンタ

立ち下がりエッジトリガ型の JK F/F の入力を J = 1、K = 1 に固定しておくと、CLK が立ち下がるたびに出力 Q が反転する。このようなフリップフロップは **T フリップフロップ**（T は “Toggle” の意味）と呼ばれる。この場合、$\overline{\mathrm{CLK}}$ は $\overline{\mathrm{T}}$ と言う名に書き換えられる。n 個の T フリップフロップを図 2.21 のように接続すると、各フリップフロップの出力 Q_n はカウント数の二進表示を与え、2^n 進計数器として働く（図 2.22）。この型のカウンタでは、後段の JK F/F が前段の出力を $\overline{\mathrm{T}}$ 入力として使用するため、各桁の状態変化が順次遅延する。つまり後段の桁へ状態変化が細波（ripple）のように伝わるため、非同期式カウンタは**リップルカウンタ**とも呼ばれる。

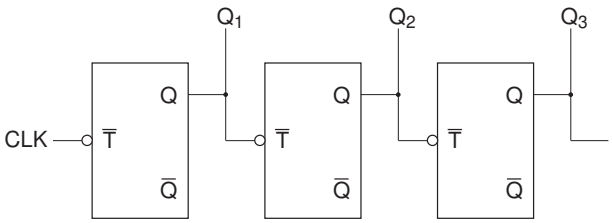

図 2.21: 非同期式カウンタ

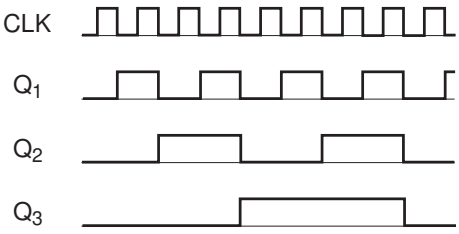

図 2.22: 非同期式カウンタのタイミングチャート

同期式カウンタ

非同期式カウンタに見られるような、上位桁の変化の遅れをなくすには、全ての JK F/F を同一 CLK に同期して動作させる必要がある。このように作ったものが同期式カウンタである。例として、同期式 4 進カウンタを設計してみよう。4 進カウンタとは、入力されたパルス毎に出力が $0(00) \to 1(01) \to 2(10) \to 3(11) \to 0(00)$ と変化するものである。つまり 2 桁の出力が必要である。2^1 および 2^0 の桁を、それぞれ一個の JK F/F A、B に対応させ、双方に同一の CLK を入力する（図 2.24 実線部分のみ）。

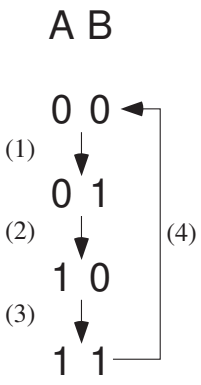

図 2.23: 4 進カウンタの動作

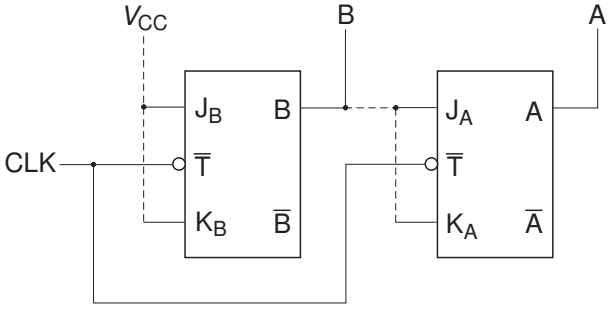

図 2.24: 同期式 4 進カウンタの回路図

この上で表示 AB が、上記の順で変化するために、各ステップにおける各 JK F/F の J、K 入力を考える。

(1) 最初の CLK パルス入力で A 出力が $0 \to 0$ と変化する為には、$(\mathrm{J_A}, \mathrm{K_A}) = (0, 0)$ もしくは $(\mathrm{J_A}, \mathrm{K_A}) = (0, 1)$、つまりは $\mathrm{J_A} = 0$ であればよい。一方、B 出力が $0 \to 1$ と変化する為には、$(\mathrm{J_B}, \mathrm{K_B}) = (1, 0)$ もしくは $(\mathrm{J_B}, \mathrm{K_B}) = (1, 1)$、つまりは $\mathrm{J_B} = 1$ であればよい。

(2) 第 2 のパルス入力で、A 出力が $0 \to 1$、B 出力が $1 \to 0$ となる為には、$\mathrm{J_A} = 1$、$\mathrm{K_B} = 1$ であればよい。

(3) 第 3 のパルス入力で、A 出力が $1 \to 1$、B 出力が $0 \to 1$ となる為には、$K_A = 0$、$J_B = 1$ であればよい。

(4) 第 4 のパルス入力で、A 出力が $1 \to 0$、B 出力が $1 \to 0$ となる為には、$K_A = 1$、$K_B = 1$ であればよい。

これらの J, K 入力を各ステップ AB 出力の始状態で構成する為に、カルノー図を描いて整理すると、表 2.18 の実線部分のようになる。このカルノー図は、半数の欄が空欄であり、“d”(don't care) で埋められる。J, K 入力がなるべく簡単な A, B の論理関数になるように、一例として表 2.18 では “d” をグレーの文字とみなして論理式で表すと $J_A = B, K_A = B, J_B = 1, K_B = 1$ となる。それを回路にすると、図 2.24 点線部分を含むもののようになる。空欄の埋め方によっては、異なる回路を作ることもできる。

J_A

A \ B	0	1
0	0	1
1	0	1

K_A

A \ B	0	1
0	0	1
1	0	1

J_B

A \ B	0	1
0	1	1
1	1	1

K_B

A \ B	0	1
0	1	1
1	1	1

表 2.18: 同期式 4 進カウンタのカルノー図

2.6.6 実験 III

練習機「DE0-CV」を用いて、以下の実験を行いカウンタを作成する。

(a) NAND ゲートを用いて RS ラッチを作成し、その動作を確かめよ。禁止入力以外全ての $\overline{S}$, $\overline{R}$ 入力について、始状態の Q 出力（Q_n）と終状態の Q 出力（Q_{n+1}）の対応関係を真理値表にまとめて結果を報告する。遷移が分かるように、$\overline{S}$, $\overline{R}$, Q_n の各値で Q_{n+1} 出力をまとめる。

(b) JK F/F の動作を確かめよ。立ち上がり・立ち下がりエッジトリガ型のどちらになるかを注意して確認する。全ての J, K 入力について、始状態の Q 出力（Q_n）と終状態の Q 出力（Q_{n+1}）の対応関係を真理値表にまとめて結果を報告する。遷移が分かるように、Q_n と Q_{n+1} は共に表に含める。

(c) 3 つの JK F/F を使用して、非同期式 8 進カウンタを作成せよ。表示には 3 個の LED を使用して 2 進数で表す。各フリップフロップの CLK 入力と Q 出力についてタイミングチャートを描き、動作原理をまとめよ。

(d) 3 つの JK F/F を使用して、同期式 5 進カウンタを作成せよ。要求される機能に基づき、各フリップフロップへの J, K 入力についてカルノー図を描く事で回路設計を行う。表示には 3 個の LED を使用して 2 進数で表す。

2.6.7 実験 IV

練習機「DE0-CV」を用いて、以下の機能を持った「早撃ちゲーム」を作成せよ。

(1) ゲーム開始のスイッチを入れてしばらく経つと、数秒後に合図 LED が点灯する。点灯は保持される。

(2) 合図 LED の点灯を見て、競技者 A, B が各々のスイッチを入れる。先に入れた方が勝ち。但し、合図 LED 点灯前は、先に入れた方が負け（お手つき）。結果は 2 つの LED で表示する。

(3) 勝敗が着いた後は、スイッチの状態を変化させても表示結果が変わらないように固定する。

(4) 次のゲームに移る為のリセットスイッチを装備する。

設計方法（各コンポーネントの動作原理）を示し、実際に作成した回路の動作、ゲームの対戦結果 (5 回以上で各勝敗をまとめる) を報告する。

※ヒント
各機能をもつ回路をブロックとして動作確認しながら作成すれば、能率的に進めることが出来る。機能 (1) にはカウンタを使用する。機能 (2) の設計においては、合図出力 S とスイッチ入力 A, B についてのカルノー図がポイントである。機能 (3) には D ラッチを使用するとよい。$\overline{\mathrm{T}}$ 入力の論理設計がポイントとなる。機能 (4) は、リセットするべきものを冷静に考え、そこにリセット信号を正しく入れてやればよい。

2.7 実験レポート

レポートは、実験 I〜IV に沿って、ディジタル回路の原理・設計・動作・考察をまとめて報告するものとする。本テキストの図・表を参照してよい。また、Quartus Prime のスクリーンショット図を用いてよい。Win キー+PrtSc キーを同時に押すと、スクリーンショットフォルダに全画面が表示される。コンピュータで画像を開いて、編集と作成→編集から、トリミングで一部画面を切り取ることができる。コンピュータからのデータ持ち出しは、このスクリーンショットのみ許可される。自分で作成した図と分かるように、ファイルは移動せずコピーで持ち出し、実験に使用したコンピュータにあるファイルは消さずに残しておく。

以下にレポート内容の一例を挙げる。

論理ゲート回路： 実験 I(a) に沿って論理ゲート回路の動作を確認し、その結果を真理値表にまとめて、正しい動作であったかどうかを報告する。

論理回路設計： 実験 I(b)(c) に沿って、それぞれの回路設計法（真理値表、論理式、カルノー図）を示し、実際に作成した回路の動作を報告する。

FPGA 動作： FPGA についてまとめ、実験 II(a) に沿って、どのように確認したかを説明し、動作を報告する。

加算器動作： 実験 II(b)(c)(d) に沿って、(b) 7 セグメント LED の真理値表・論理式と動作報告、(c) 半加算器の回路図と真理値表にまとめた動作報告、(d) 計算機の回路図と結果、動作報告を記す。

順序回路の基礎： 実験 III(a)(b) に沿って、RS ラッチおよび JK F/F の動作を確認し、その結果を真理値表にまとめる。

カウンタ設計： 実験 II(c)(d) に沿って、カウンタ設計方法（回路図と JK F/F 使用法、タイミングチャート、(d) ではカルノー図）を示し、実際に作成した回路の動作を報告する。

早撃ちゲーム： 実験 IV「早撃ちゲーム」の設計方法（合図 LED・勝敗の決定・勝敗の固定・リセットの動作原理）を示し、実際に作成した回路の動作、ゲームの対戦結果を報告する。

別途配布する評価項目を漏れなく含むように留意する。

参 考 文 献

[1] 桜井捷海、霜田光一：「応用エレクトロニクス」、裳華房

[2] 猪飼國夫、本田中二：「定本 ディジタル・システムの設計」、CQ 出版社

[3] 相磯秀夫、松下 温：「電子計算機 I - 基礎編 -」、コロナ社

[4] 小林優：「改訂 2 版 FPGA ボードで学ぶ組込みシステム開発入門 Intel FPGA 編」、技術評論社