第2章

特集 入門！ VHDLによる回路設計の基礎

VHDL

組み合わせ回路とVHDL記述

吉田たけお/尾知博

前章で扱ったディジタル回路は，じつは「組み合わせ回路」である．組み合わせ回路とは，その出力が過去の入力には依存せず，現在の入力のみによって一意に定まるようなディジタル回路のことである．本章では，実際のコンピュータなどで使用されている実用的で重要な組み合わせ回路を紹介していこう．また，それらの組み合わせ回路を実際にVHDLで記述し設計してみることにしよう．（筆者）

2.1 実用的な組み合わせ回路

2.1.1 加算器

コンピュータ内部のCPUには，必ずといってよいほど加算器が組み込まれており，ディジタル回路設計では，もっとも基本的でかつ重要な回路である．**図2.1**に示すように，4ビット加算器は，キャリ入力なしの1ビット加算器（半加算器（HA））とキャリ入力ありの1ビット加算器（全加算器（FA））の縦続接続で構成できる．

以下で全加算器を設計してみよう．

例題2-1 全加算器の設計

全加算器の真理値表を作成し，カルノー図を利用して回路設計しなさい．

［解］

全加算器の真理値表は，**表2.1**に示すとおりである．

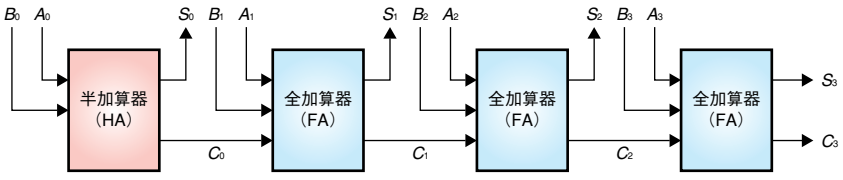
加算結果S_iに関してカルノー図を描くと**図2.2**(a)となる．図よりこれ以上の圧縮はできないことがわかる．したがって，S_iの論理関数は以下のようになる．

$$S_i = \overline{A_i} \cdot \overline{B_i} \cdot C_{i-1} + \overline{A_i} \cdot B_i \cdot \overline{C_{i-1}} + A_i \cdot B_i \cdot C_{i-1} + A_i \cdot \overline{B_i} \cdot \overline{C_{i-1}}$$

またキャリC_iに関しては，カルノー図（**図2.2**(b)）により圧縮が可能である．そこで得られたC_iは，以下のとおりであった．

$$C_i = A_i \cdot B_i + B_i \cdot C_{i-1} + C_{i-1} \cdot A_i$$

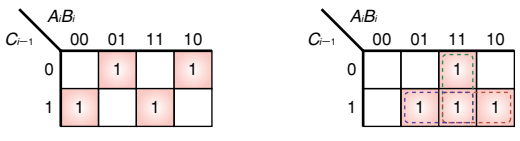
以上より，1ビット全加算器の回路は，**図2.3**となる．



〔図2.1〕 4ビット加算器

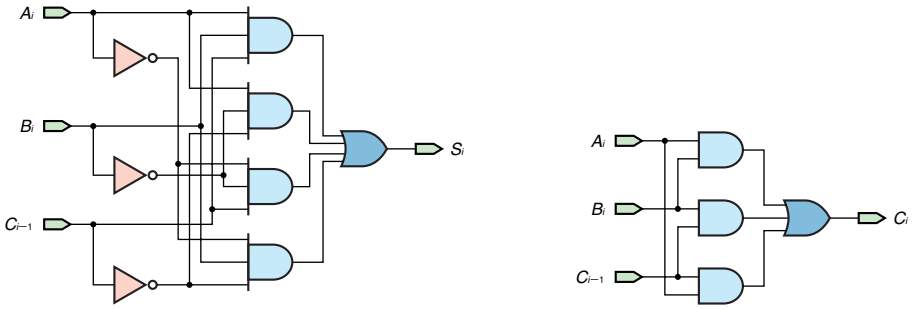
〔表2.1〕全加算器の真理値表

A_i	B_i	C_{i-1}	S_i	C_i
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



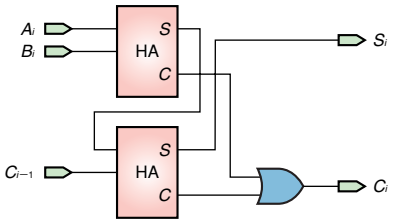
〔図2.2〕全加算器のカルノー図

(a) 加算結果S_iのカルノー図　(b) キャリC_iのカルノー図



〔図2.3〕全加算器の回路図

(a) 加算結果S_i生成部　(b) キャリC_i生成部



〔図2.4〕半加算器による全加算器の構成

◆階層設計

なお全加算器は，**図2.4**のように，半加算器を二つ用いて構成することもできる．このように，すでに設計された回路ブロック（コンポーネント：component という）を用いて，新しい回路を設計する方法を**階層設計**（hierarchical design）という．

以下では，全加算器を階層設計してみよう．そのために，コンポーネントとして用いる半加算器を設計する．

演習2-1　半加算器

半加算器をVHDLにより設計しなさい．

［解］

リスト2.1に示すとおりである．

なお，**リスト2.1**は，bit型の代わりにstd_logic型を用いている点を除けば，**リスト1.1**（p.24）と同じである．また，**リスト2.1**を論理合成した結果は，**リスト1.1**を論理合成した結果と同じであり，**図1.6**のようになる．

演習2-2　全加算器

全加算器をVHDLにより設計しなさい．その際，**演習2-1**で設計した半加算器をコンポーネントとして階層設計すること．

［解］

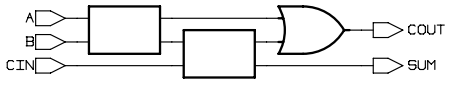
リスト2.2および**図2.5**に示すとおりである．

〔リスト2.1〕半加算器のVHDL記述（2）

```
-- std_logic型を使用するための
-- ライブラリ宣言とパッケージ呼び出し
library IEEE;
use IEEE.std_logic_1164.all;

entity HALF_ADDER is
    port ( A, B : in  std_logic;
           S, C : out std_logic );
end HALF_ADDER;

architecture STRUCTURE of HALF_ADDER is
begin
    S <= A xor B;
    C <= A and B;
end STRUCTURE;
```

〔リスト2.2〕全加算器のVHDL記述

```
library IEEE;
use IEEE.std_logic_1164.all;

entity FULL_ADDER is
    port ( A, B, CIN : in  std_logic;
           SUM, COUT : out std_logic );
end FULL_ADDER;

architecture STRUCTURE of FULL_ADDER is

-- HALF_ADDERのコンポーネント宣言
component HALF_ADDER
    port ( A, B : in  std_logic;
           S, C : out std_logic );
end component;

signal C1_C, C1_S, C2_C : std_logic;

begin
    -- コンポーネント・インスタンス文
    COMP1 : HALF_ADDER port map ( A, B, C1_S, C1_C );
    COMP2 : HALF_ADDER port map ( C1_S, CIN, SUM, C2_C );
    COUT <= C1_C or C2_C;
end STRUCTURE;
```



〔図2.5〕全加算器の論理合成結果（Design Compilerによる）