第1章
ブール代数のおさらいとディジタル回路の基礎

論理回路と回路記述

パソコン，マイコン，インターネット，携帯電話など，ディジタル回路を応用した電子機器が活躍する場面が数え切れないほどあります．最近の機器の複雑な機能を実現するには，大規模な回路が必要となる一方，機器の小型化や最適化には専用のLSIやカスタム・チップが使われます．もちろん，汎用ICやLSIを組み合わせて利用する場合もあります．いずれにしても，複雑で大規模な回路の設計は大変です．

そこで本書では，こうした複雑で大規模な回路の設計やシミュレーションを効率的に行うために近年よく使われる，VHDL(Very High-speed IC Hardware Description Language)を紹介します．

電子回路は長年，回路図で表現されてきました．その回路図に代わってVHDLでディジタル回路を記述することで，新たな設計パラダイムを導入することができます．

章を追って，その考えかたや具体的な手法について紹介していきますが，この章では，まずはディジタル回路の基礎でありVHDLによる設計にも欠かせない，論理回路の基本事項を整理しておきましょう．

1-1 二つの状態で回路の動作を決める ディジタル回路と2値論理

ディジタル回路ではすべての信号を，HighとLow，電圧の高低，電流の流れる向きや流れる/流れないなど，二つの状態で表して取り扱います．信号を‘1’や‘0’，True(真)，False(偽)といった値に対応づけて考えるわけです．

このように，信号の状態や回路のふるまいを二つの状態に対応づけることを2値論理と言います．2値論理の導入によって，ブール代数*1と呼ばれる数学体系の法則が適用できます．

もちろん，これは理論上の理想的な状態でのことです．ディジタル回路も電子回路である以上，その動作を仔細に解析すれば，特殊な動作モードのアナログ回路の一種と考えられますが，より複雑で大規模な回路の実現のために，一般に論理回路(logic circuit)と呼ばれるモデルで考えるようになりました．

理論上は2値論理だけでなく，3値論理や10進法に対応する10の状態をもつ論理回路も考えられます．実際，初期のコンピュータでは10進数で内部処理を行っていたようです．

しかし，回路構成の複雑さや信号の伝送など，2値論理に比べると明らかに大変になりそうです．コンピュータの回路で広く2値論理が採用されるようになったのは，かなり後になってから*2のようですが，ともあれ，論理回路は情報理論やコンピュータの設計手法として研究が進みました．

*1：ブール代数(Boolean algebra)は，この代数体系を考案した数学者George Bool(1815～1864)にちなんで名付けられた．
*2：ENIACやUNIVACなど，初期の有名なコンピュータの大半は，上記の2進法を採用したマシンと同時期か後に製作されたにもかかわらず，内部は10進法で動作していたようである．実は2進法によるコンピュータの製作やアーキテクチャが定着するのは意外と遅く，1950年代中盤から．

第2章
デザイン・フローからVHDLのフレームワークまで

回路設計手法の変化とHDL

近年のディジタル回路は複雑かつ大規模で，しかも高速動作が求められるため，その設計はますます難しくなっていくと言われています．そこで，新たな設計手法を導入し，こうした問題に対処していかなくてはなりません．

この章では，従来の回路図を中心としたディジタル回路設計から，ハードウェア記述言語へ移行する意義と，その基礎になる考えかたなどを紹介していきます．後半はVHDLの基本的な事項について紹介します．

2-1 複雑かつ大規模なディジタル回路設計が必要とされている デザイン・フローとHDL

● デザイン・フロー

ディジタル回路の設計は，所属する組織や扱う回路の規模や種類，用途などによってさまざまですが，おおまかに分類すると，

① 要求仕様の作成
② 設計仕様の作成
③ 回路の検討
④ 回路設計
⑤ 基板設計
⑥ 試作と動作検証
⑦ 設計修正
⑧ 完成

といった一連の機器開発の流れの一部に組み込まれた作業です．

LSIの設計なら，基板設計の代わりにレイアウトや配線，シミュレーションといった作業になります．試作を何度か繰り返すこともあるでしょうし，その際に基板を再設計したり，回路の検討時に回路の一部分を取り出して原理試作するといったこともあります．そうした事情も含めて，かなり大ざっぱですが，設計から完成までの一連の流れのことをデザイン・フローと呼びます(図1)．

CADを使った設計では，それぞれの段階でデータの受け渡しを行うEDA(Electronic Design Automation)と呼ばれるシステムを構築するケースも増えています．ただ，すべての工程をカバーする単一のソフトウェアはありませんし，通常は社内の他部門や社外とのデータのやりとりが必要になります．

● 回路図からHDLへ

従来，ディジタル回路の設計は回路図に図記号を使って記述してきました．しかし，回路が複雑化し，大規模な回路になると，1枚の回路図には収まりきらなくなり，何枚もの紙に分割した回路図を作成することになります．

実際，ちょっとしたLSI規模になると100ページを越えてしまうことも多く，回路図のままでは回路の設計もデバッグも大変です．何よりも，回路図は，作成時にCADを使って入力しないかぎり，部品情報はもちろん，配線情報など，あとでコンピュータ情報として扱うことは，まったく不可能ではないにしても非常に困難です．

たとえば，30年前の手書きの回路図を現在のEDAで扱うには，回路図を再入力するしかありません．また，図形情報として描かれてはいますが，配線情報は複雑なデータ構造としてコンピュータ内部で表現され，その取り扱いには相当なコンピュータ資源が必要です．

もちろん，回路図ならではのメリットもありますが，これまで述べてきたような回路図の限界や

第3章
使用する文字の規則から基本的な文法説明まで

VHDLの基礎

この章では，第2章の後半で紹介したVHDLの基本的な知識を補強し，さまざまな回路の記述に必要な基本構文を，簡単な例を挙げながら紹介していきます．

なお，本書はVHDLの文法書ではありません．膨大で複雑なVHDLのすべての構文を網羅するのは，言語に対する見通しを良くする反面，初学者にとってはしばしば混乱の原因にもなります．そのため，最初からすべてを網羅するのではなく，まずは慣れることを重視して最小限の紹介にとどめます．

ある程度慣れてから，さらに高度な記述を習得するには，IEEEの言語マニュアルなどを参照するようにしてください．

3-1 記述に使用できる文字と命名規則 VHDLの基本的な約束ごと

まずは，VHDLの基本的な約束ごとについて，まとめておきます．

● 大文字と小文字

前章でも触れましたが，基本的にVHDLは大文字と小文字を識別しません．すべてを大文字で記述しても，すべてを小文字で記述しても，コンパイラは同じものとして認識します．

VHDLでは大文字と小文字を区別しない

すなわち，Moduleとmodule，MODULEは，すべて同じです．

ただし例外があります．いずれも少し高度な話題ですが，あとで参照するときのために，ここで紹介しておきましょう．

シングル・クォート「'」で囲んだ文字，またはダブル・クォート「"」で囲んだ文字列と，後述するcharacter型またはstring型の変数で文字を扱う場合，そして列挙型で宣言した場合には，大文字と小文字を区別します．

文字型を扱う場合は当然として，シングル・クォートのほうは，IEEEライブラリのstd_logicの‘X’と‘Z’は大文字で定義されているため，小文字で記述するとエラーとなります．列挙型も，宣言した際の文字が大文字なら大文字で，小文字なら小文字で扱われます．

● コメント

VHDLのソース・コード中に，任意にコメント（注釈）を記述することができます．コメントは，「--」（マイナス記号が二つ）で示され，その行の終わりまでがコメントになります．

コメントは「--」で始まり，その行の終わりまで

複数行にわたるコメント指定はできません．複数行のコメントを記述する場合は，それぞれの行頭に「--」を記入する必要があります．

なお，「--」は必ずしも行頭である必要はなく，VHDLの有効な記述の後ろに「--」を記入することで，それ以降がコメントとみなされます．複数の行にまたがるコメントを記述するには，たとえばコメント行のすべての行頭に「--」を記述するしかありません．

● 文の終わり

VHDLは基本的に文単位で構文解析が行われていきます．そのため，一つの文の終わりには，文の終わりを示す記号としてセミコロン「;」が必要になります．