特集

ソフト&ハード，両方知っててあたりまえ！

第2章

ソフトウェア技術者のための ハードウェア記述言語(HDL)入門

――回路記述はプログラミングではない

小林　優

設計データ

ここでは，ソフトウェア設計の知識のある技術者を対象に，ハードウェア記述言語(HDL)を使った回路設計手法について解説する．基本的な論理回路やHDL文法(ここではVerilog-HDLを紹介)，回路記述の定石などを説明する．HDLはプログラミング言語(C言語など)と似ている．そのため，プログラミング言語の知識があれば，HDLの文法は容易におぼえられる．しかし，回路記述とプログラム記述の間には明確な違いがある．たとえば，HDLの文法にしたがって記述しても，実際の回路として実現できない場合や，適切な回路に変換されない場合がある．こうした記述は論理合成の際にエラーとなることがある．　(編集部)

ASICやPLDを用いた論理設計では，HDLによる設計が主流になっています．言語による設計なので，ソフトウェア技術者にも簡単に理解できそうです．しかし，実際はそう甘くありません．言語体系や文法を理解できても，論理回路そのものを知らなくては設計できません．また回路記述に向く，もしくは向かない記述スタイルが存在します．

そこで，ここではソフトウェア技術者向けに，HDLによる回路設計をやさしく解説します．

1．HDL設計のための基本論理回路

まずは，論理回路について知っておかねばなりません．ここでは，HDL設計を行うために最低限必要な項目を解説します．

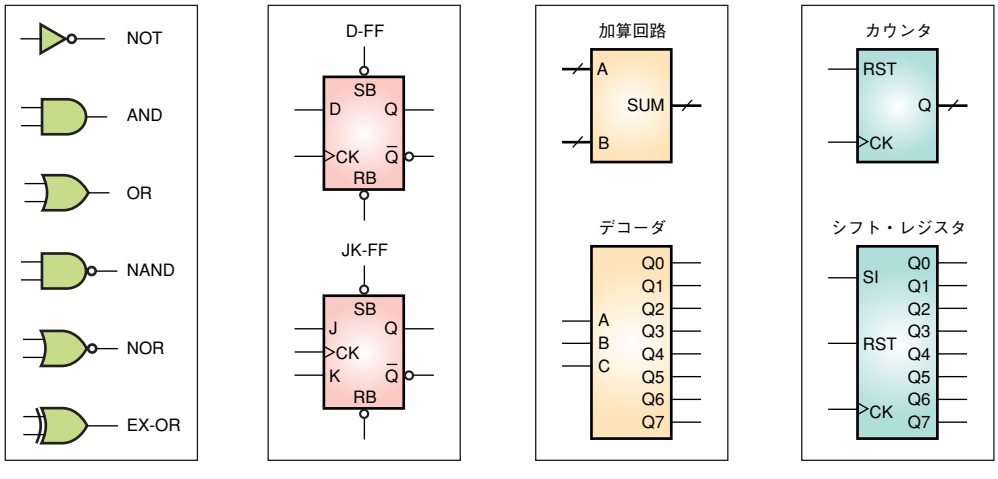
●論理回路記号

HDL設計では，回路をすべてHDLで表現するため，回路記号はまず使いません．しかし，論理合成によって得られた回路図から回路構成を把握したり，場合によっては論理を追いかけることもあります．

図1にいくつかの論理回路記号を示します．

(1) **ゲート回路(図1(a))**

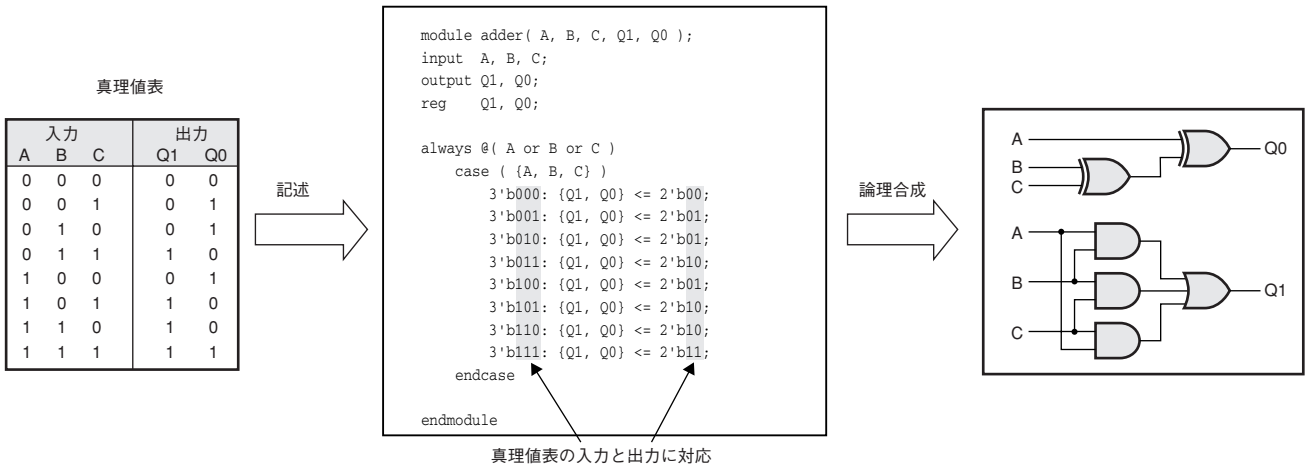
論理演算子に対応した回路記号があります．どのような組み合わせ回路でも，これらの回路記号を接続することで実現できます．



(a) ゲート回路　(b) フリップフロップ　(c) 組み合わせ回路　(d) 順序回路

〔図1〕回路記号

HDL設計でも，論理合成によって得られた回路図から回路構成を把握したり，場合によっては論理を追いかけることもある．したがってゲート回路やフリップフロップの動作を理解している必要がある．HDLで回路を記述する際に，デコーダやカウンタのレベルの回路図(詳細なブロック図)を書くことはある．この程度の抽象度なら，そのままHDLで記述できる．



〔図2〕真理値表から回路へ

真理値表の入出力の関係をそのままcase文に置き換えて論理合成すれば，回路が得られる．組み合わせ回路の場合，回路の動作を正確に真理値表で表現できれば，設計の大半が終了したことになる．

⑵ **フリップフロップ（図1(b)）**

値を保持する基本ユニットです．Dフリップフロップ（D-FF）が基本で，JKフリップフロップ（JK-FF）やTフリップフロップ（T-FF）などがあります．

⑶ **組み合わせ回路（図1(c)）**

組み合わせ回路（詳しくは後述）の機能を一つの箱で表現したものです．内部はゲート回路によって構成されています．

⑷ **順序回路（図1(d)）**

値を保持する順序回路（詳しくは後述）の機能を一つの箱で表現したものです．内部はフリップフロップとゲート回路によって構成されています．

HDL設計では，アーキテクチャ設計（概略設計）の際に回路構成を検討する資料として，回路図を用いる場合もあります．**図1(c)**や**図1(d)**のように，回路の動作を一つの箱で置き換えて，これらを組み合わせて全体構成を考えます．ゲート回路やフリップフロップのレベルまで具体化していなくても，この程度の抽象度でHDL記述できます．

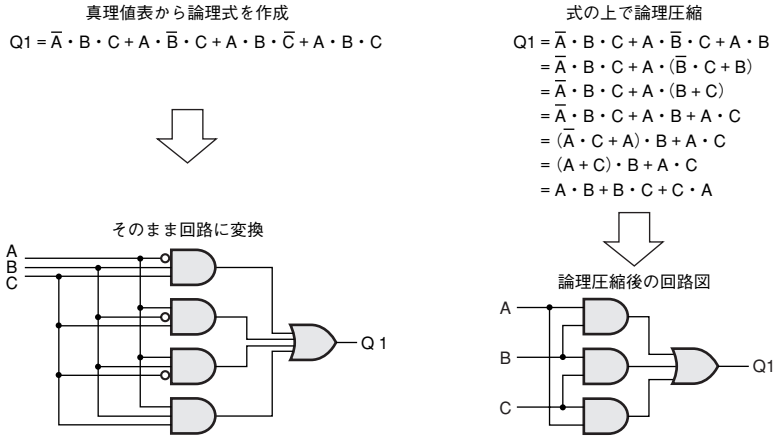
なお，**図1(c)**や**図1(d)**は，回路記号として正式なものではなく，慣例的に使われている形状を紹介したものです．他の形状を用いる場合もあります．

●真理値表

真理値表とは，回路の入力と出力の関係を表で示したものです．おもに組み合わせ回路を表現するために使います．HDL設計では，真理値表が完成すれば，簡単に回路に変換することができます．

図2の真理値表は，1ビットの加算回路（フル・アダー）です．真理値表の入出力の関係をそのままHDLのcase文に置き換え，論理合成して回路を得ます．

したがって，組み合わせ回路の設計では，回路の動作を正確に真理値表で表現できれば，設計の大半が終了したことになります．



〔図3〕論理式と論理圧縮

論理式上で式を簡単化し，回路の冗長部分を取り除くことを論理圧縮と呼ぶ．論理合成ツールは，高度の論理圧縮機能を備えているので，設計者自身が論理圧縮を行う必要はない．