HDL記述による設計法をマスタする

実験で学ぶロジック回路設計

木村 真也
Shinya Kimura

第8回 定石3 値を保持し数を数える回路の記述法

今回は，ロジック回路の定番記述として，レジスタとカウンタを取り上げます．

▶ビット列の値を記憶する回路…レジスタ

レジスタは，ビット列の値を記憶する回路です．

例えば，光センサで人の通過を検出して人数を数えるだけの簡単な回路を考えてみましょう．

光がさえぎられるとハイ・レベル(H)になるセンサがあったとします．人が通りかかったときは，信号を出しHになります．しかし，通り過ぎてしまえば，ロー・レベル(L)に戻ってしまいます．

このようなセンサで通過人数を数えるには，現在までの通過人数を覚えておく記憶回路(レジスタ)が必要不可欠です．

▶高度な機能を実現するためには必要不可欠

レジスタはデータの保持にも使いますが，それだけではありません．マイコンやCPUなどロジック回路の多くは，これまでの状況に依存する「現在の状態」をレジスタに保持させ，それに対応した動作をすることで，複雑かつ高度な機能を可能にしています．レジスタなしではロジック回路の醍醐味は味わえません．

▶信号パルスの回数を数えられる回路…カウンタ

記憶回路と加算回路を組み合わせると，カウンタと呼ばれる数を数える回路になります．

カウンタも非常に応用例が多い定番回路です．前述の例では，入力信号がHになった回数をカウンタで数えれば，人数を数えられます．本連載で作るゲーム回路も，多数のカウンタを使います．

値を保持するのに必要なレジスタ

レジスタは情報を記憶保持する部品で，各所で使用される重要なものです．レジスタはD型フリップフロップを必要な個数ぶん並べて構成します(**図8－1**)．通常，レジスタを1個の部品として扱います．

■ 基本的な記述方法

● Verilog HDLでは記憶機能のある信号で表現される

Verilog HDLでは，レジスタを記述する場合に基本ゲートを組み合わせるような方法は使いません．記憶機能のある信号を定義し，どのような条件で記憶するかを規定すると，実際にはレジスタが構成されます．

● reg宣言した信号は記憶機能をもつレジスタになる

記憶機能のある信号はregで宣言します．wireで定義しても文法エラーになります．記述例は以下です．

```
reg D_ff;                    //1本の信号
reg [7:0] A_reg, B_reg;      //二つの8ビット信号
```

Keyword 1 レジスタ

情報(＝ロジック信号)を記憶しておく機能をレジスタと呼んでいます．実態はD型のフリップフロップを必要なビット数ぶん，並べて構成しています．

Verilog HDLでレジスタを記述する場合，レジスタの出力信号をreg型信号として定義し，レジスタ本体をalways @()文で記述します．

レジスタ以外にも情報(ロジック信号)を記憶する機能としてメモリがあります．メモリの一種，スタティックRAM(SRAM)はD型フリップフロップで構成されています．1ビットあるいは複数ビットを1ワード(語)とし，その単位で読み書きができるようになっています．実際のSRAMには複数個のワードがあり，どのワードかを指定する信号としてアドレス線があります．複数のワードを同時に読み出したり，書き込んだりすることはできません．

ただし，SRAMのフリップフロップは，一般的なエッジ・トリガではなくレベル・センシティブ型です．書き込み信号が1の間は入力と出力が等しく，書き込み信号が0になればそのときの値を保持するタイプです．

文法上の規約はwireと同じで，ビット幅を有する信号の定義も可能です．異なるビット幅の信号は，別途regとして宣言します．

● レジスタへの信号入力はalways文を使う

regとして宣言した信号に値を設定するにはalways文を使用します．これまで使ってきたassign文ではありません．例えば以下のような記述になります．

```
wire clock;
wire d;
reg q;

always @(posedge clock) begin
    q <= d;
end
```

● always文の使い方

▶ reg宣言された信号にしか代入できない

always文の中で代入する信号（上の例ではq）は，reg宣言しておく必要があります．

▶ 実行条件を@( )の中に記述する

@以下のカッコ内において，always文の中を実行する条件（イベント式）を記述します．

記述例のposedge clockは，clock信号のポジティブ・エッジ（立ち上がりのとき）の意味です．立ち下がりのときはnegedge clockとなります．

イベント式が満たされた場合，begin~end内の文が実行されます．記述例では，入力信号dの値がqとして記憶，保持されます（レジスタに保持される）．

● 8ビットのレジスタの記述例

リスト8－1に8ビット長の単純なレジスタをVerilog HDLで記述したものを示します．

単純なレジスタは非常に簡単に記述することができます．Dフリップフロップ，つまり，1ビットのレジスタも同様に記述できます．

ただし，**リスト8－1**のように記述した場合，例えば16ビットのレジスタが必要な場合には，ビット幅の部分だけ異なるほとんど同じ記述のモジュールを用意することになります．

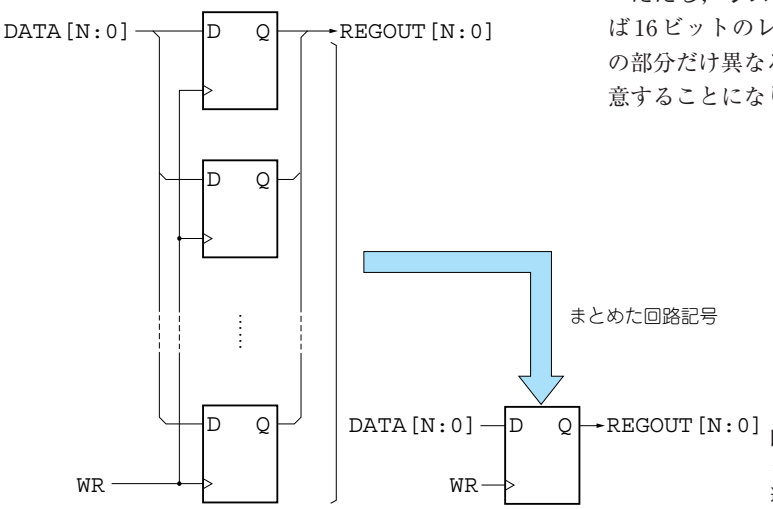

図8－1　レジスタのロジック回路
クロックが立ち上がったときだけ入力の値を読み込み，それ以外のときはずっと値を保持し続ける

Keyword 2　wireとreg

Verilog HDLには2種類の信号の型があります．一つはwire宣言するネット型信号で，もう一つはregで宣言するレジスタ型信号です．

ネット型信号は，単に「つなぐ」ための配線に相当し，物理的な配線と対応していると考えてよいでしょう．これに対してレジスタ型信号は実際のロジック回路上の配線とは異なり，Verilog HDL上の抽象化された概念になります．

レジスタ型信号と聞くとレジスタそのものを連想してしまいますが，レジスタ型信号＝レジスタではないケースがあります．具体例は後述します．

Verilog HDLでは，regで宣言する信号は「次に値が設定されるまで変化しない信号」であり，プログラム（C言語など）でいう変数に近いものと考えるとイメージしやすいかもしれません．

function化した部分は，論理合成すると必ず組み合わせ回路になります．しかし，その中で使用するローカルな信号はregで宣言する必要があります．これは中間変数的な意味合いをもった信号と理解してください．

レジスタ型信号を宣言していても組み合わせ回路になるケースは，その信号に関係する入力信号のすべての組み合