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12回 ボールを表示して動かす

今回は，ディスプレイにボールを表示して，動かしてみます．

ボールを表示するには壁やラケットのことも考える必要があります．壁，ラケット，ボールの三つの要素を画面表示するモジュールを作ります(**図12-1**)．

ボールの動作制御部は，製作するテレビ・ゲーム回路において，最も複雑な制御が必要です．とはいえ，状況や条件をよく考慮して，一つずつ解決していけば，問題ありません．

ボールの表示に要求されること

● ボールのサイズ

まず，ボールのサイズを決めます．制御を簡単にするために1ゲーム・ドットとします．形状も凝らずに正方形（描画点のドットで8×8）とします．

● ボールはどのように動くのか？

スカッシュ・ゲームでは，ボールは以下の三つの条件を満たすように動作する必要があります．

▶基本的に一定速度で直進させる

ボールを表示する位置座標を，一定時間ごとに一定ドットずつ変えていくことで実現できます．

更新の時間間隔と，表示座標の変更分を記憶させておく必要があります．

▶壁やラケットに衝突すると跳ね返る

壁やラケットの座標と重なることを検出して，座標の変化分の正負を変えれば実現できます．

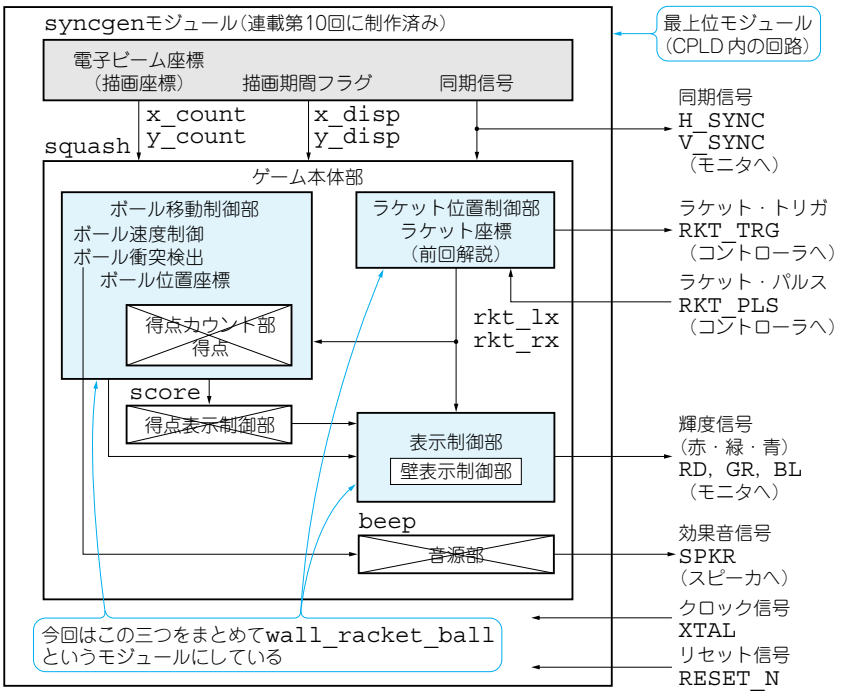

図12-1　ボール，ラケット，壁の三つを制御/表示する部分を作る
ラケットと壁の制御/表示については前回解説した

▶移動スピードは難易度にかかわるので切り替えられるようにする

位置座標を更新する時間間隔を変えることで可能になります．

● Verilog HDL記述が長いので分割して解説

処理が複雑なので，HDL記述も長くなっています．解説はリストを四つに分割して行っています．

ボールのスピードを決める更新タイミング信号を作る

ボールの移動速度はゲームの難易を決める重要な要素なので，2段階に切り替えられるようにします．切り替えはスイッチ(スライド・スイッチ0)で行います．

スライド・スイッチ0に繋がっているCPLDの端子の信号名はSLDSW[0]です．最上位モジュールでSLDSW[0]とボール速度信号ball_speedを接続します．

● 1画面の更新ごとに1ゲーム・ドットの移動

ボールの速度を検討してみましょう．1画面が16.8 msごとに更新されますので，これを単位にボールを1ゲーム・ドットぶん斜め方向に進行させるとすると，壁の上端から下端に移動するのにほぼ1秒，壁の右端から左端に移動するのに1.344秒と計算できます．ゲームとして考えると適度な速度といえます．

1画面ごとにボールを1ゲーム・ドットぶん移動させ，これを基準速度とします．ゲーム速度スイッチを切り替える(1にする)と，この半分の速度でボールを移動させます．

● V_sync信号を使って1画面に1回のパルスを得る

ボールの位置座標の更新は，クロック信号xtalのポジティブ・エッジに同期し，ボール更新指示信号new_ball_positionが1の場合に行うことにします．

1画面に1回パルスが出る垂直同期信号V_syncの立ち上がりエッジを検出して，1クロック間だけ1にすることで，new_ball_position信号を作ることにします．

● スピードを半分に落とすためには

ボール速度を1/2にする場合は，new_ball_position信号を2画面に1度出すように制御しなければなりません．そのための回路記述は，**リスト12−1**の行番号72～85です．

カウンタとしてv_count信号を使います．v_countは1ビットのレジスタで，ゲーム速度信号ball_speedが1の場合は，V_syncごとに反転します．ball_speed信号が0の場合，v_count信号は0のままにします．

v_count信号が0の場合だけV_syncの立ち上がりエッジを検出して1クロック間new_ball_position信号を1にすれば，1倍と1/2倍を切り替えられます．

リスト12−1　ボールのスピードを決める信号v_clockを作るHDL記述

外部スイッチに繋げたball_speed信号でスピードを変えられる

```
70:   // game speed control (tglsw==1 -> low  speed,
71:   //                          ==0 -> high speed)
72:     always @(posedge xtal or negedge reset_N) begin
73:         if(!reset_N) begin
74:             v_count <= 0;
75:         end else if(V_sync & ~vsyncd) begin
76:             v_count <= v_count ^ ball_speed;
77:         end
78:     end
79:
80:   // game basic clock generator
81:     always @(posedge xtal) begin
82:         vsyncd  <= V_sync;
83:     end
84:
85:     assign new_ball_position = V_sync & ~vsyncd &
                                                ~v_count;
86:
87:   // racket control
      (ラケット制御部分は前回と同じなので省略．
       ラケットの座標信号rkt_rxとrkt_lxが得られる)
112:
```

Keyword 1　wireとreg

Verilog HDLにはwire宣言するネット型信号とreg宣言するレジスタ型信号の二つの信号があります．

ネット型信号は物理的な配線とほぼ対応しています．レジスタ型信号は，「次に値が設定されるまで変化しない信号」であり，C言語などでいう変数に近いものです．実際にはレジスタではないケースもあります．

Verilog HDLの文法上，これらの信号に値を設定するための規定があります．

- ネット型信号はassign文で値を設定する
- レジスタ型信号はalways文内やfunction内において値を設定する

別の言い方をすると，レジスタ型信号はbegin～end内で設定(正確には手続き代入と呼ぶ)しなければならない信号となります．ただし，手続き代入文が一つしかない場合，begin～endは省略できます．

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レジスタではありませんが，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．この場合のreg信号は，レジスタというより中間変数的な意味合いをもった信号になります．