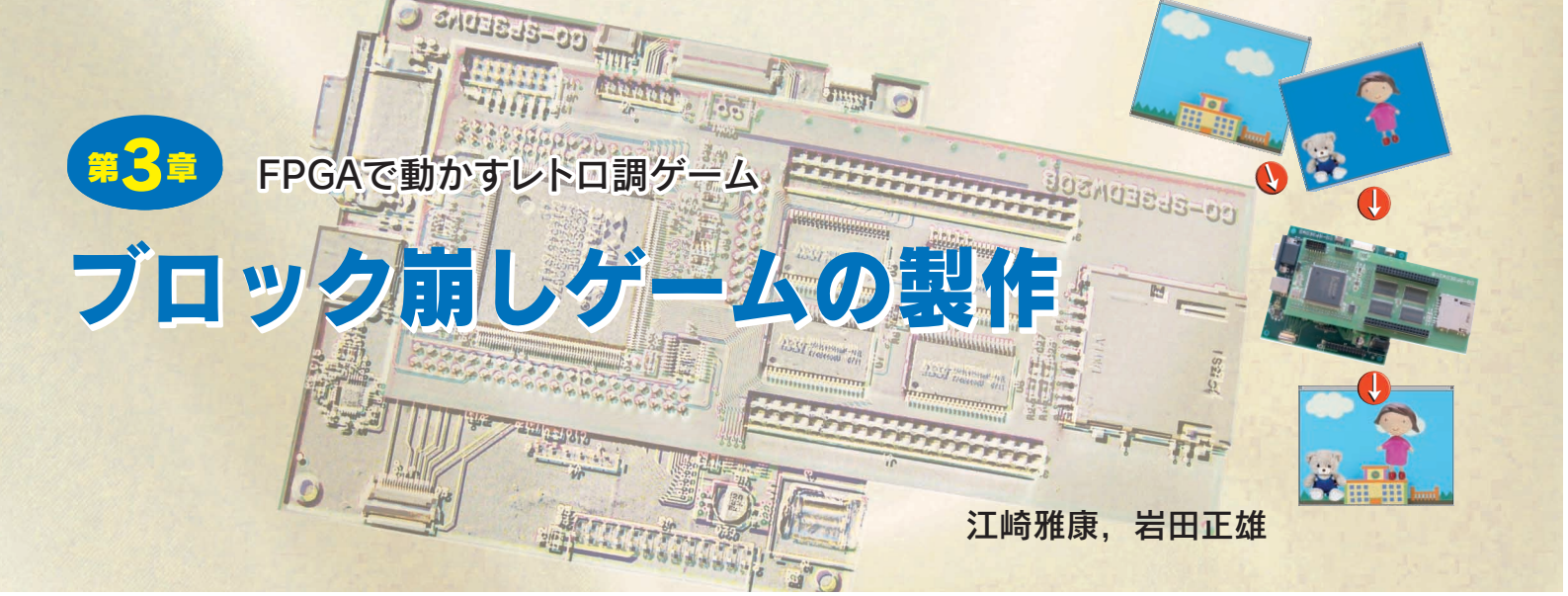

第3章 FPGAで動かすレトロ調ゲーム

ブロック崩しゲームの製作

江崎雅康，岩田正雄

第2章のマイクロプロセッサ回路と，本誌2007年8月号特集1 第2章のアナログRGB出力回路を使って「何か，意味のある作品を…」と思い立って，「ブロック崩しゲーム」の制作に着手した．（筆者）

1 FPGAでアーケード・ゲーム製作

黎明期のテレビ・ゲームの基板には100個近くのTTL標準ロジックICが並んでいました．標準ロジックICだけで，テレビの同期信号から画面上のアクションまで組み上げられていたアーケード・ゲームもありました．

当時のロジックIC 1個を100ゲートとしても，25万ゲートのSpartan-3E（XC3S250）には2,500個分のICを組み込めるパワーがあります．

FPGAであれば，多くのICをコツコツはんだ付けする手間も不要です．パソコン上でHDLコードを記述するだけで回路が完成します．30年前には考えられなかった25万ゲートのFPGAを使ってアーケード・ゲームを作ってみよう．これが開発の動機です．

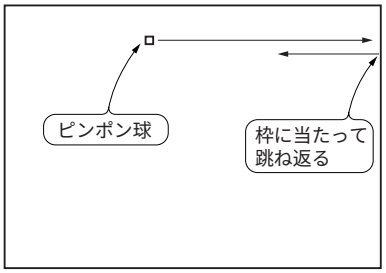

図1 ピンポン玉の動作（水平方向のみ）
10×10ドットのピンポン球（正方形で表現）を表示し，それがゆっくり画面上を移動し，画面の枠で跳ね返る様子を表現．

2 ピンポン球の表示

本誌2007年8月号特集1の第2章でVGA画面表示の同期信号を作りました．まず**図1**に示すように10ドット×10ドットのピンポン球（画面上では正方形）を表示してみましょう．次にこのピンポン球を画面上をゆっくり移動させて画面の枠で跳ね返る様子を表現します．

リスト1に示すように，

- ピンポン球の水平方向位置を示すh_pos
- 垂直方向位置を示すv_pos

を定義します．初期値としてピンポン球の左上角の座標を（144，244）としました．**リスト2**の記述を追加するとピンポン球が画面の中ほどに表示されます．

リスト1 ピンポン玉の表示（内部信号の定義）

```
signal h_pos        : std_logic_vector (9 downto 0)
                                :="0010010000"; --144
signal v_pos        : std_logic_vector (9 downto 0)
                                :="0011110100"; --244
```

リスト2 ピンポン玉の表示（画面の中ほどにピンポン玉を白色で表示．ただし四角の球？）

```
rgb<="111" when h_counter>=h_pos and
                                h_counter<h_pos+10 and
                v_counter>=v_pos and v_counter<v_pos+10
else
    "000";
LR<=(others=>rgb(2));
LG<=(others=>rgb(1));
LB<=(others=>rgb(0));
```

KeyWord ADuC7026，XC3S500E-VQ208，Spartan-3E，VGA，RGB，CQ-SP3EDW，CQ-SP3E208，画像フレーム・メモリ，LDO，ブロック崩しゲーム

3 ピンポン球の移動

次にリスト3に示すように，三つの内部信号を定義します．

- クロック　puls
- ピンポン球の水平方向の動き　h_dir
- ピンポン球の垂直方向の動き　v_dir

pulsは，垂直カウンタの一部から取り出した適度な速さのクロックです．h_dir，v_dirはピンポン球の移動方向を示します．pulsの立ち上がり時に，ピンポン球の移動方向に位置レジスタの値を1ずつ加減します．これによってピンポン球が移動します．移動の結果，ピンポン球の位置が画面の端に到達した場合はピンポン球の移動方向を逆にします．

リスト4は水平方向の動きに対する制御の記述です．同様に垂直方向の制御も追加すると，ピンポン球は図2に示すように画面の枠内を移動します．

4 パドルの追加

さて次は図3のように画面の中央下部分にパドル（サイズ80ドット×8ドット）を配置します．パドルに当たるとピンポン球が跳ね返るように記述を変更します．

リスト5はパドルの属性定義で，パドルの左上角の座標を（280，384）としています．パドルのサイズは固定値（constant），位置座標は変化可能なレジスタ値（signal）として定義します．

リスト6はパドルの配置とピンポン球の跳ね返りの記述です．パドルの上側だけでなく，下側でもピンポン球が跳ね返るよう，2カ所に分けて記述してあります．

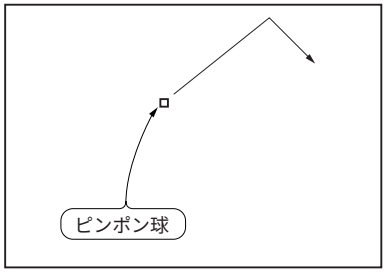

図2
ピンポン球の動作（上下左右に移動）
画面の上下左右枠で跳ね返りながら，斜め上下の移動を繰り返す．

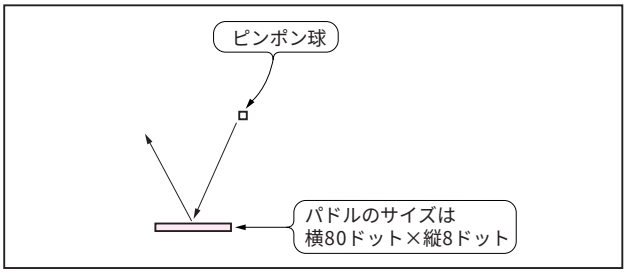

図3　パドルの配置
画面の中央下部分にパドル（サイズ80ドット×8ドット）を配置し，その部分でもピンポン球が跳ね返るように変更．

5 ブロック崩しの記述

今度はピンポン球の跳ね返りを利用して，図4のように配置したブロックを消していく「ブロック崩し」にしてみましょう．

まずブロックを配置します．ブロックの有無はリスト7に示すようにレジスタblk1のビットで示します（1：あり，0：なし）．

リスト3　動きのあるピンポン玉の表示（内部信号定義）
クロックとしてpuls，ピンポン玉の動きの方向は，水平方向を示すh_dir，垂直方向を示すv_dirを定義

```
signal h_dir   : std_logic :='0';   ··· 0:左へ  1:右へ
signal v_dir   : std_logic :='0';   ··· 0:下へ  1:上へ
signal puls    : std_logic;
```

リスト4　動きのあるピンポン玉の表示（水平方向のみ制御）
ピンポン玉移動用のクロックの生成と，水平方向の動きに対する制御部分．

```
puls<=v_counter(6);

process (puls)
begin
      if puls'event and puls='1' then
                if h_dir='1' then
                          h_pos<=h_pos+1;
                          if h_pos>630 then
                                    h_dir<='0';
                          end if;
                else
                          h_pos<=h_pos-1;
                          if h_pos<=1 then
                                    h_dir<='1';
                          end if;
                end if;

      end if;
end process;
```

リスト5　パドルの配置（内部信号定義の追加）
パドルの左上角の座標を（280，384）とする．パドルのサイズは固定（constant）で位置座標は変化できるよう定義．

```
signal pad_h_pos    : std_logic_vector (9 downto 0)
                                  :="0100011000"; --280
signal pad_v_pos    : std_logic_vector (9 downto 0)
                                  :="0110000000"; --384
constant pad_h_size : std_logic_vector (9 downto 0)
                                  :="0001010000"; --80
constant pad_v_size : std_logic_vector (9 downto 0)
                                  :="0000001000"; --8
```