AN-1270
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

基于ADV7511/ADV7511W/ADV7513的视频发生器

作者：Witold Kaczurba

简介

本应用笔记描述一种基本配置，它将一个现场可编程门阵列(FPGA)用作信号源，产生同步时序和视频图案，并将ADV7511/ADV7511W/ADV7513配置为输出有效的高清多媒体接口(HDMI®)或数字视频接口(DVI)流。本应用笔记旨在利用最基本的实例来阐明产生有效视频流的方式。

基本要求

为了以HDMI或DVI格式输出，必须用适当的I^2C寄存器写操作(参见清单4)对ADV751x进行编程，并提供附带时钟的有效视频同步信号。利用端接的最小化传输差分信号(TMDS)线将ADV751x连接到有效的HDMI接收端。

视频流是HDMI传输的关键要素。它定义流的带宽，并支持插入携带HDMI数据包和音频信息的数据岛。

同步信息

视频标准可以分为两大类：隔行和非隔行。非隔行标准也称为逐行视频标准。逐行视频标准携带连续视频行的视频信息。其时序可利用垂直和水平同步参数来描述，例如：

- 总水平行长度
- 水平前肩和后肩
- 水平同步脉冲
- 垂直行总数
- 垂直前肩和后肩
- 垂直同步脉冲
- HV偏移
- 像素时钟频率

隔行视频标准携带奇偶视频行的信息，这些视频行分为奇数和偶数视频场。由于存在两类场，因此需要关于奇偶场垂直参数的额外信息。以下参数提供描述隔行视频标准同步信息所需的所有信息：

- 总水平行长度
- 水平前肩和后肩
- 水平同步脉冲
- 偶数场中的垂直行总数
- 偶数场的垂直前肩和后肩
- 偶数场中的垂直同步脉冲
- 奇数场中的垂直行总数
- 奇数场的垂直前肩和后肩
- 奇数场中的垂直同步脉冲长度
- HV偏移
- 像素时钟频率

清单1是针对所需视频标准而生成的Verilog代码。

VERILOG清单

清单 1. 生成的视频同步Verilog模块

```
module sync_vg
#(
  parameter X_BITS=12,
            Y_BITS=12
)
(
  input wire clk,
  input wire reset,
  input wire interlaced,
  input wire [Y_BITS-1:0] v_total_0,
  input wire [Y_BITS-1:0] v_fp_0,
  input wire [Y_BITS-1:0] v_bp_0,
  input wire [Y_BITS-1:0] v_sync_0,
  input wire [Y_BITS-1:0] v_total_1,
  input wire [Y_BITS-1:0] v_fp_1,
  input wire [Y_BITS-1:0] v_bp_1,
  input wire [Y_BITS-1:0] v_sync_1,
  input wire [X_BITS-1:0] h_total,
  input wire [X_BITS-1:0] h_fp,
  input wire [X_BITS-1:0] h_bp,
  input wire [X_BITS-1:0] h_sync,
  input wire [X_BITS-1:0] hv_offset_0,
  input wire [X_BITS-1:0] hv_offset_1,
  output reg vs_out,
  output reg hs_out,
  output reg de_out,
  output reg [Y_BITS:0]   v_count_out,
  output reg [X_BITS-1:0] h_count_out,
  output reg [X_BITS-1:0] x_out,
  output reg [Y_BITS:0]   y_out,
  output reg field_out,
  output wire clk_out
);

reg [X_BITS-1:0] h_count;
reg [Y_BITS-1:0] v_count;
reg              field;
reg [Y_BITS-1:0] v_total;
reg [Y_BITS-1:0] v_fp;
reg [Y_BITS-1:0] v_bp;
reg [Y_BITS-1:0] v_sync;
reg [X_BITS-1:0] hv_offset;

assign clk_out = !clk;

/* horizontal counter */
always @(posedge clk)
  if (reset)
    h_count <= 0;
  else
    if (h_count < h_total - 1)
      h_count <= h_count + 1;
    else
      h_count <= 0;

/* vertical counter */
always @(posedge clk)
  if (reset)
    v_count <= 0;
  else
    if (h_count == h_total - 1)
    begin
      if (v_count == v_total - 1)
        v_count <= 0;
      else
        v_count <= v_count + 1;
    end

/* field */
always @(posedge clk)
  if (reset)
  begin
    field <= 0;
    v_total   <= v_total_0;
    v_fp      <= interlaced ? v_fp_1 : v_fp_0; // In the interlaced mode this value must be inverted as v_fp_1 is still in
field0
    v_bp      <= v_bp_0;
    v_sync    <= v_sync_0;
    hv_offset <= hv_offset_0;
  end
```

```
  else
    if ((interlaced) && ((v_count == v_total - 1) && (h_count == h_total - 1)))
    begin
      field     <= field + interlaced;
      v_total   <= field ? v_total_0 : v_total_1;
      v_fp      <= field ? v_fp_1    : v_fp_0; // This order is inverted as v_fp_1 is still in field0
      v_bp      <= field ? v_bp_0    : v_bp_1;
      v_sync    <= field ? v_sync_0  : v_sync_1;
      hv_offset <= field ? hv_offset_0 : hv_offset_1;
    end

always @(posedge clk)
if (reset)
  { vs_out, hs_out, de_out, field_out } <= 4'b0;
else begin
  hs_out <= ((h_count < h_sync));
  de_out <= (((v_count >= v_sync + v_bp) && (v_count <= v_total - v_fp - 1)) && \
             ((h_count >= h_sync + h_bp) && (h_count <= h_total - h_fp - 1)));

  if ((v_count == 0) && (h_count == hv_offset))
     vs_out <= 1'b1;
  else if ((v_count == v_sync) && (h_count == hv_offset))
     vs_out <= 1'b0;

  /* H_COUNT_OUT and V_COUNT_OUT */
  h_count_out <= h_count;
  if (field)
    v_count_out <= v_count + v_total_0;
  else
    v_count_out <= v_count;

  /* X and Y coords – for a backend pattern generator */
  x_out <= h_count - (h_sync + h_bp);
  if (interlaced)
    y_out <= { (v_count - (v_sync + v_bp)) , field };
  else
    y_out <= { 1'b0, (v_count - (v_sync + v_bp)) };
  field_out <= field;
end

endmodule
```

清单 2. 生成的视频图案Verilog模块

```
module pattern_vg
  #(
    parameter B=8, // number of bits per channel
              X_BITS=13,
              Y_BITS=13,
              FRACTIONAL_BITS = 12
  )
  (input reset, clk_in,
   input wire [X_BITS-1:0] x,
   input wire [Y_BITS-1:0] y,
   input wire vn_in, hn_in, dn_in,
   input wire [B-1:0] r_in, g_in, b_in,
   output reg vn_out, hn_out, den_out,
   output reg [B-1:0] r_out, g_out, b_out,
   input wire [X_BITS-1:0] total_active_pix,
   input wire [Y_BITS-1:0] total_active_lines,
   input wire [7:0] pattern,
   input wire [B+FRACTIONAL_BITS-1:0] ramp_step);

reg [B+FRACTIONAL_BITS-1:0] ramp_values; // 12-bit fractional end for ramp values

always @(posedge clk_in)
begin
  vn_out <= vn_in;
  hn_out <= hn_in;
  den_out <= dn_in;

  if (reset)
     ramp_values <= 0;
  else if (pattern == 8'b0) // no pattern
    begin
      r_out <= r_in;
      g_out <= g_in;
      b_out <= b_in;
    end
  else if (pattern == 8'b1) // border
    begin
      if (dn_in && ((y == 12'b0) || (x == 12'b0) || (x == total_active_pix - 1) || (y == total_active_lines - 1)))
      begin
        r_out <= 8'hFF;
        g_out <= 8'hFF;
        b_out <= 8'hFF;
      end
      else
      begin
        r_out <= r_in;
        g_out <= g_in;
        b_out <= b_in;
      end
    end
  else if (pattern == 8'd2) // moireX
  begin
    if ((dn_in) && x[0] == 1'b1)
    begin
      r_out <= 8'hFF;
      g_out <= 8'hFF;
      b_out <= 8'hFF;
    end
    else
    begin
      r_out <= 8'b0;
      g_out <= 8'b0;
      b_out <= 8'b0;
    end
  end
  else if (pattern == 8'd3) // moireY
  begin
    if ((dn_in) && y[0] == 1'b1)
    begin
      r_out <= 8'hFF;
      g_out <= 8'hFF;
      b_out <= 8'hFF;
    end
    else
    begin
      r_out <= 8'b0;
      g_out <= 8'b0;
      b_out <= 8'b0;
    end
  end
  else if (pattern == 8'd4) // Simple RAMP
  begin
```

```
    r_out <= ramp_values[B+FRACTIONAL_BITS-1:FRACTIONAL_BITS];
    g_out <= ramp_values[B+FRACTIONAL_BITS-1:FRACTIONAL_BITS];
    b_out <= ramp_values[B+FRACTIONAL_BITS-1:FRACTIONAL_BITS];
    if ((x == total_active_pix - 1) && (dn_in))
      ramp_values <= 0;
    else if ((x == 0) && (dn_in))
      ramp_values <= ramp_step;
    else if (dn_in)
      ramp_values <= ramp_values + ramp_step;
  end
end

endmodule
```

清单3. 生成的与图案发生器相关的视频同步发生器Verilog顶层模块

```
module top_sync_vg_pattern
  (
   input wire clk_in,
         input wire resetb,
         output reg adv7511_hs,       // HS output to ADV7511
         output reg adv7511_vs,       // VS output to ADV7511
         output wire adv7511_clk,       // ADV7511: CLK
         output reg [35:0] adv7511_d,   // data
         output reg adv7511_de,             // ADV7511: DE
         input wire [5:0]  pb
  );
/* ********************************** */

/* SELECT ONE OF MODES: */
`define MODE_1080p
//`define MODE_1080i
//`define MODE_720p

`ifdef MODE_1080p /* FORMAT 16 */
parameter INTERLACED  = 1'b0;
parameter V_TOTAL_0   = 12'd1125;
parameter V_FP_0      = 12'd4;
parameter V_BP_0      = 12'd36;
parameter V_SYNC_0    = 12'd5;
parameter V_TOTAL_1   = 12'd0;
parameter V_FP_1      = 12'd0;
parameter V_BP_1      = 12'd0;
parameter V_SYNC_1    = 12'd0;
parameter H_TOTAL     = 12'd2200;
parameter H_FP        = 12'd88;
parameter H_BP        = 12'd148;
parameter H_SYNC      = 12'd44;
parameter HV_OFFSET_0 = 12'd0;
parameter HV_OFFSET_1 = 12'd0;
parameter PATTERN_RAMP_STEP = 20'h0222;
parameter PATTERN_TYPE = 8'd4; // RAMP
//parameter PATTERN_TYPE = 8'd1; // OUTLINE
`endif
`ifdef MODE_1080i /* FORMAT 5 */
parameter INTERLACED  = 1'b1;
parameter V_TOTAL_0   = 12'd562;
parameter V_FP_0      = 12'd2;
parameter V_BP_0      = 12'd15;
parameter V_SYNC_0    = 12'd5;
parameter V_TOTAL_1   = 12'd563;
parameter V_FP_1      = 12'd2;
parameter V_BP_1      = 12'd16;
parameter V_SYNC_1    = 12'd5;
parameter H_TOTAL     = 12'd2200;
parameter H_FP        = 12'd88;
parameter H_BP        = 12'd148;
parameter H_SYNC      = 12'd44;
parameter HV_OFFSET_0 = 12'd0;
parameter HV_OFFSET_1 = 12'd1100;
parameter PATTERN_RAMP_STEP = 20'h0222; // 20'hFFFFF / 1920 act_pixels per line = 20'h0222
parameter PATTERN_TYPE = 8'd4; // RAMP
//parameter PATTERN_TYPE = 8'd1; // OUTLINE
`endif
`ifdef MODE_720p /* FORMAT 4 */
parameter INTERLACED  = 1'b0;
parameter V_TOTAL_0   = 12'd750;
parameter V_FP_0      = 12'd5;
parameter V_BP_0      = 12'd20;
parameter V_SYNC_0    = 12'd5;
parameter V_TOTAL_1   = 12'd0;
parameter V_FP_1      = 12'd0;
parameter V_BP_1      = 12'd0;
parameter V_SYNC_1    = 12'd0;
parameter H_TOTAL     = 12'd1650;
parameter H_FP        = 12'd110;
parameter H_BP        = 12'd220;
parameter H_SYNC      = 12'd40;
parameter HV_OFFSET_0 = 12'd0;
parameter HV_OFFSET_1 = 12'd0;
parameter PATTERN_RAMP_STEP = 20'h0333; // 20'hFFFFF / 1280 act_pixels per line = 20'h0333
//parameter PATTERN_TYPE = 8'd1; // BORDER.
parameter PATTERN_TYPE = 8'd4; // RAMP
`endif

wire reset;
assign reset = !resetb;
```

```
wire [11:0] x_out;
wire [12:0] y_out;
wire [7:0] r_out;
wire [7:0] g_out;
wire [7:0] b_out;

/* ******************** */
sync_vg #(.X_BITS(12), .Y_BITS(12)) sync_vg
(
  .clk(clk_in),
  .reset(reset),
  .interlaced(INTERLACED),
  .clk_out(), // inverted output clock - unconnected

  .v_total_0(V_TOTAL_0),
  .v_fp_0(V_FP_0),
  .v_bp_0(V_BP_0),
  .v_sync_0(V_SYNC_0),
  .v_total_1(V_TOTAL_1),
  .v_fp_1(V_FP_1),
  .v_bp_1(V_BP_1),
  .v_sync_1(V_SYNC_1),
  .h_total(H_TOTAL),
  .h_fp(H_FP),
  .h_bp(H_BP),
  .h_sync(H_SYNC),
  .hv_offset_0(HV_OFFSET_0),
  .hv_offset_1(HV_OFFSET_1),
  .de_out(de),
  .vs_out(vs),
  .v_count_out(),
  .h_count_out(),
  .x_out(x_out),
  .y_out(y_out),
  .hs_out(hs),
  .field_out(field)
);

   pattern_vg #(
     .B(8), // Bits per channel
     .X_BITS(12),
     .Y_BITS(12),
     .FRACTIONAL_BITS(12)) // Number of fractional bits for ramp pattern
   pattern_vg (
     .reset(reset),
     .clk_in(clk_in),
     .x(x_out),
     .y(y_out[11:0]),
     .vn_in(vs),
     .hn_in(hs),
     .dn_in(de),
     .r_in(8'h0), // default red channel value
     .g_in(8'h0), // default green channel value
     .b_in(8'h0), // default blue channel value
     .vn_out(vs_out),
     .hn_out(hs_out),
     .den_out(de_out),
     .r_out(r_out),
     .g_out(g_out),
     .b_out(b_out),
     .total_active_pix(H_TOTAL - (H_FP + H_BP + H_SYNC)), // (1920) // h_total - (h_fp+h_bp+h_sync)
     .total_active_lines(INTERLACED ? (V_TOTAL_0 - (V_FP_0 + V_BP_0 + V_SYNC_0)) + (V_TOTAL_1 - (V_FP_1 + V_BP_1 +
V_SYNC_1)) : (V_TOTAL_0 - (V_FP_0 + V_BP_0 + V_SYNC_0))),         // originally: 13'd480
     .pattern(PATTERN_TYPE),
     .ramp_step(PATTERN_RAMP_STEP));

assign adv7511_clk = ~clk_in;

always @(posedge clk_in)
begin
  adv7511_d[35:24] <= { r_out, 4'b0 };
  adv7511_d[23:12] <= { g_out, 4'b0 };
  adv7511_d[11:0]  <= { b_out, 4'b0 };
  adv7511_hs <= hs_out;
  adv7511_vs <= vs_out;
  adv7511_de <= de_out;
end

endmodule
```

清单4. ADV7511/ADV7511W/ADV7513脚本

```
72 01 00 ; Set N Value(6144)
72 02 18 ; Set N Value(6144)
72 03 00 ; Set N Value(6144)
72 15 00 ; Input 444 (RGB or YCrCb) with Separate Syncs
72 16 61 ; 44.1kHz fs, YPrPb 444
72 18 46 ; CSC disabled
72 40 80 ; General Control Packet Enable
72 41 10 ; Power Down control
72 48 48 ; Reverse bus, Data right justified
72 48 A8 ; Set Dither_mode - 12-to-10 bit
72 4C 06 ; 12 bit Output
72 55 00 ; Set RGB444 in AVinfo Frame
72 55 08 ; Set active format Aspect
72 96 20 ; HPD Interrupt clear
72 98 03 ; ADI required Write
72 98 02 ; ADI required Write
72 9C 30 ; ADI required Write
72 9D 61 ; Set clock divide
72 A2 A4 ; ADI required Write
72 43 A4 ; ADI required Write
72 AF 16 ; Set HDMI Mode
72 BA 60 ; No clock delay
72 DE 9C ; ADI required write
72 E4 60 ; ADI required Write
72 FA 7D ; Nbr of times to search for good phase
```

视频发生器

将清单1描述的视频同步发生器连接到一个视频图案发生器。图案发生器将一个像素颜色值分配给由计数器(H_CNT, V_CNT)表示的屏幕上特定位置(X, Y)。

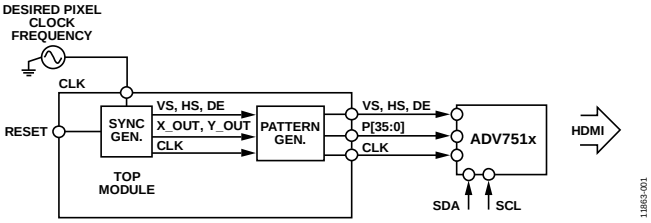

图1. 简化功能框图

ADV751x所有器件都包含颜色空间转换模块、上变频器和下变频器。因此，使用能够产生RGB 444信号的视频图案发生器即足以从ADV7511输出其它标准，如YCbCr 422和YCbCr 444等。

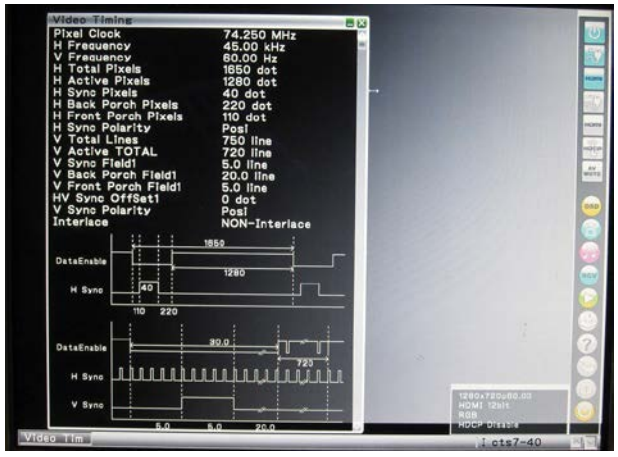

图2. FPGA生成的720p、60 Hz斜坡图案视频流

参考文献

ADV7511 Data Sheet. Analog Devices, Inc.
ADV7511W Data Sheet. Analog Devices, Inc.
ADV7513 Data Sheet. Analog Devices, Inc.
Keith Jack, *Video Demystified: A Handbook for the Digital Engineer*, 5th Edition. Newnes, 2007.
CEA-861-D Specification. Consumer Electronics Association.

修订历史

2013年11月—修订版0：初始版

AN11863sc-0-11/13(0)