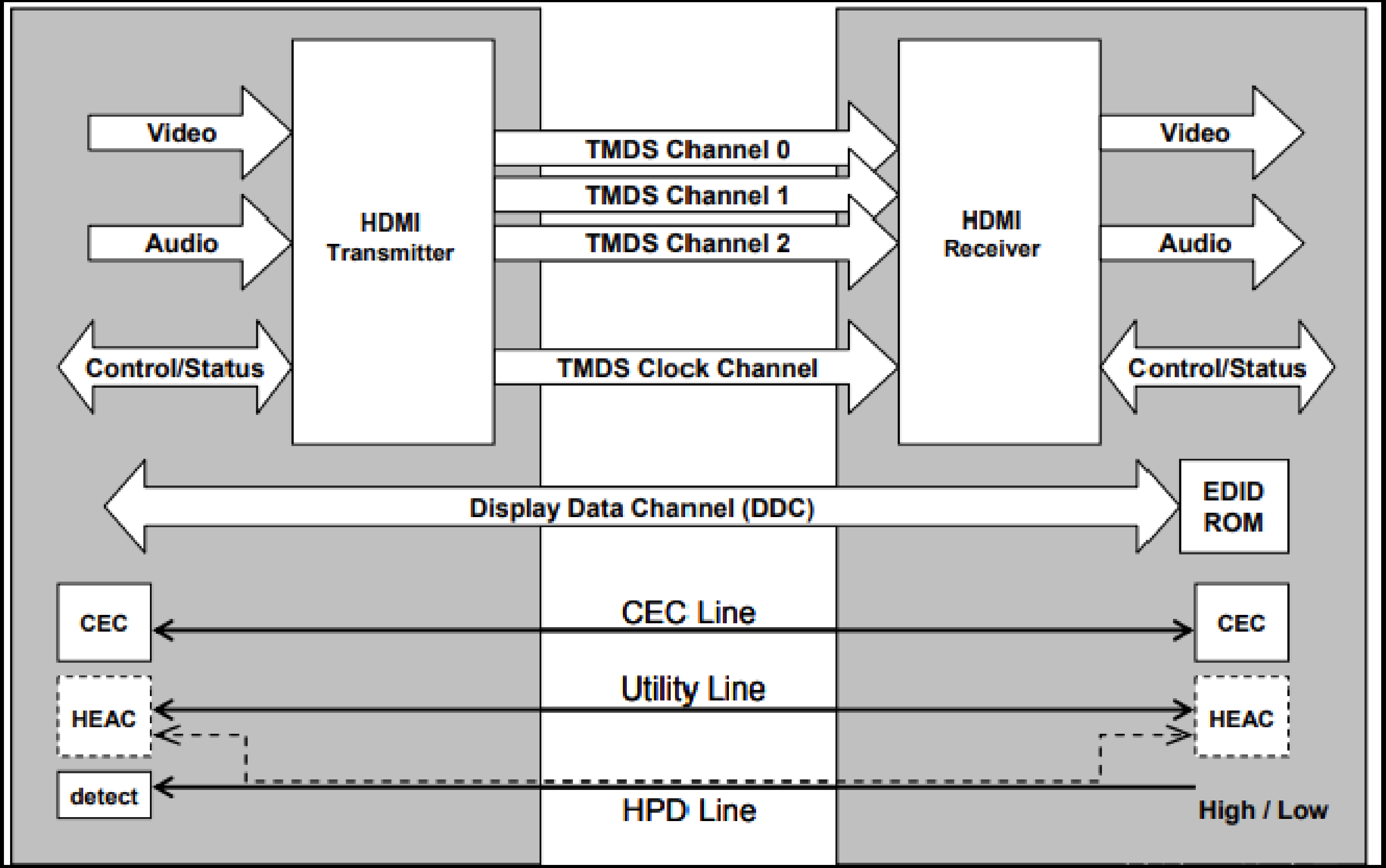
High Definition Multimedia Interface (HDMI)

HDMI Overview

HDMI透過4條TMDS差動信號線傳輸影音及時序，其餘信號為高級控制(非必須)，此次lab不會用到。



HDMI Overview

HDMI傳輸時需要透過Encode將一般的影音資料轉成TMDS編碼(8 bit to 10 bit)，
再將Parallel資料轉成Serial資料傳輸。

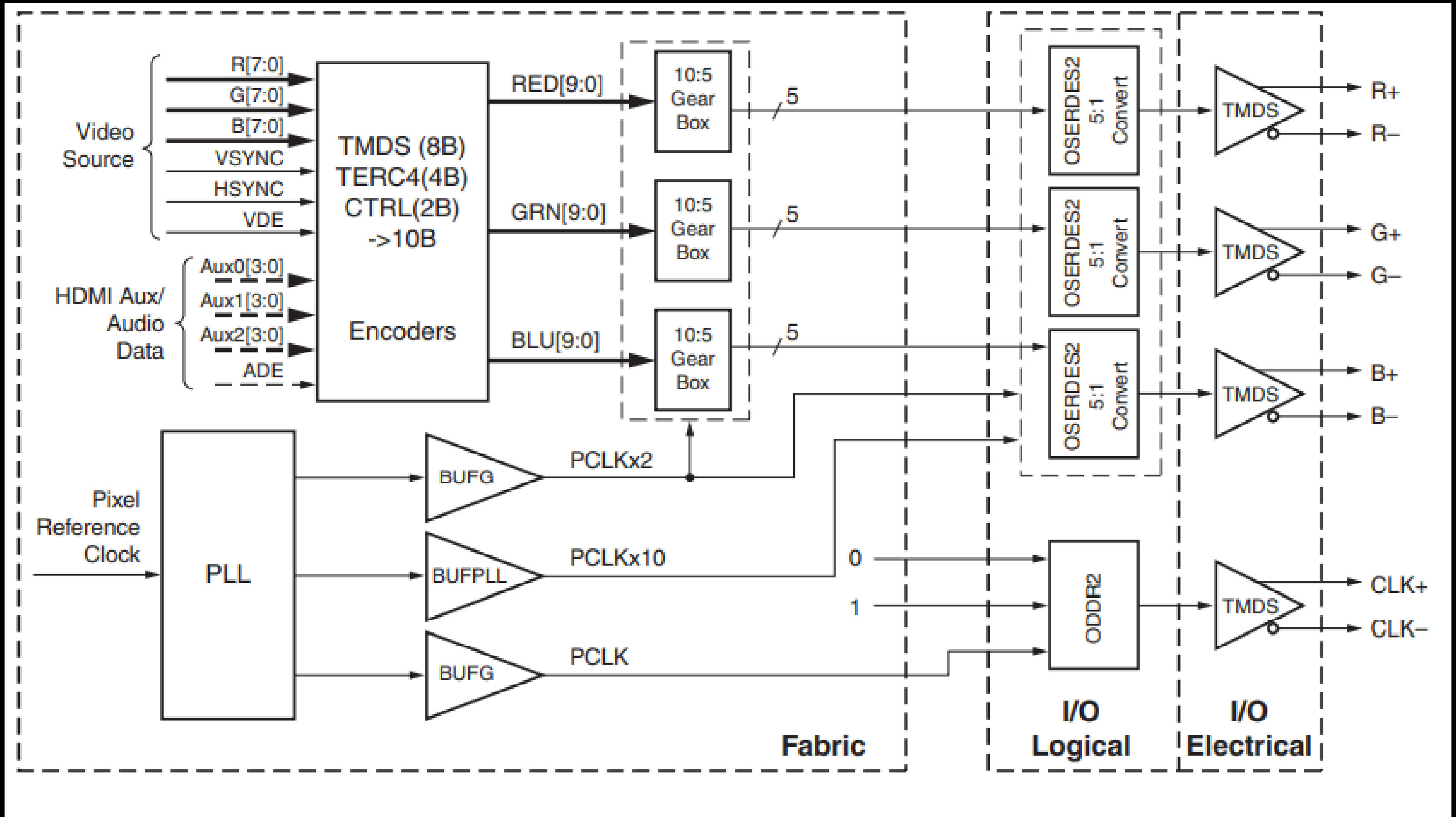
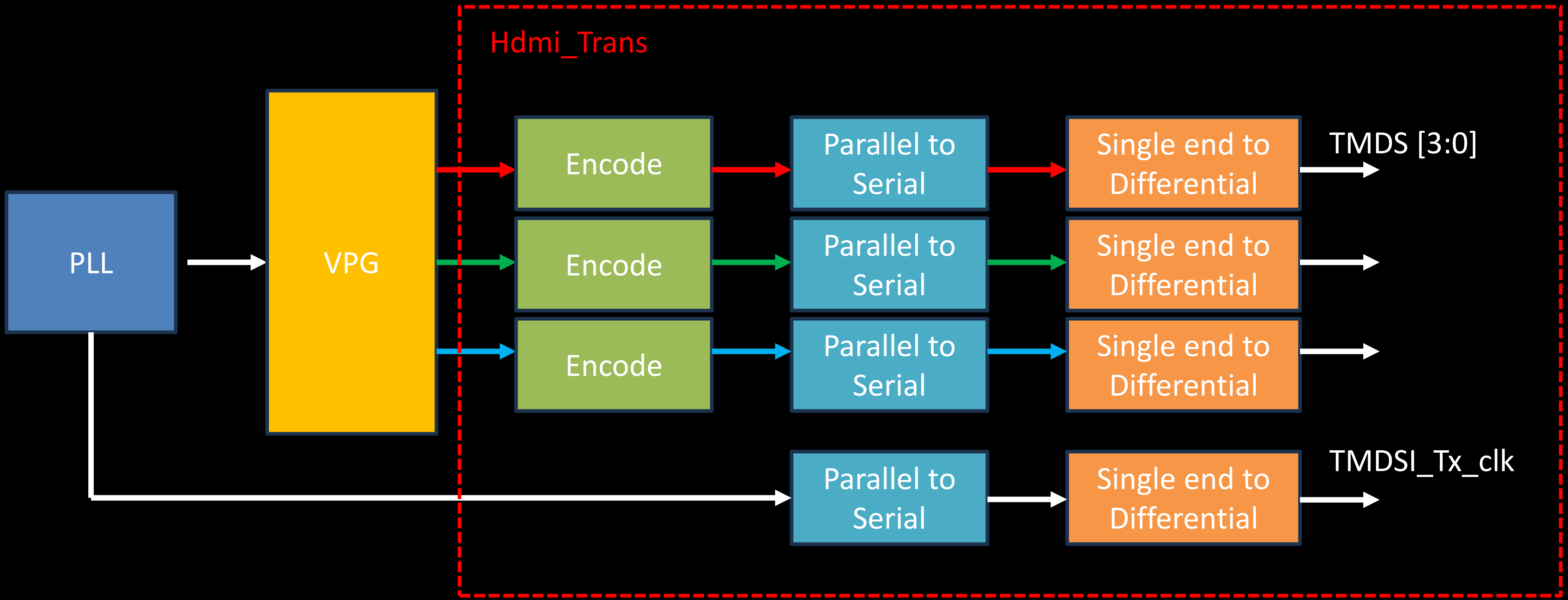


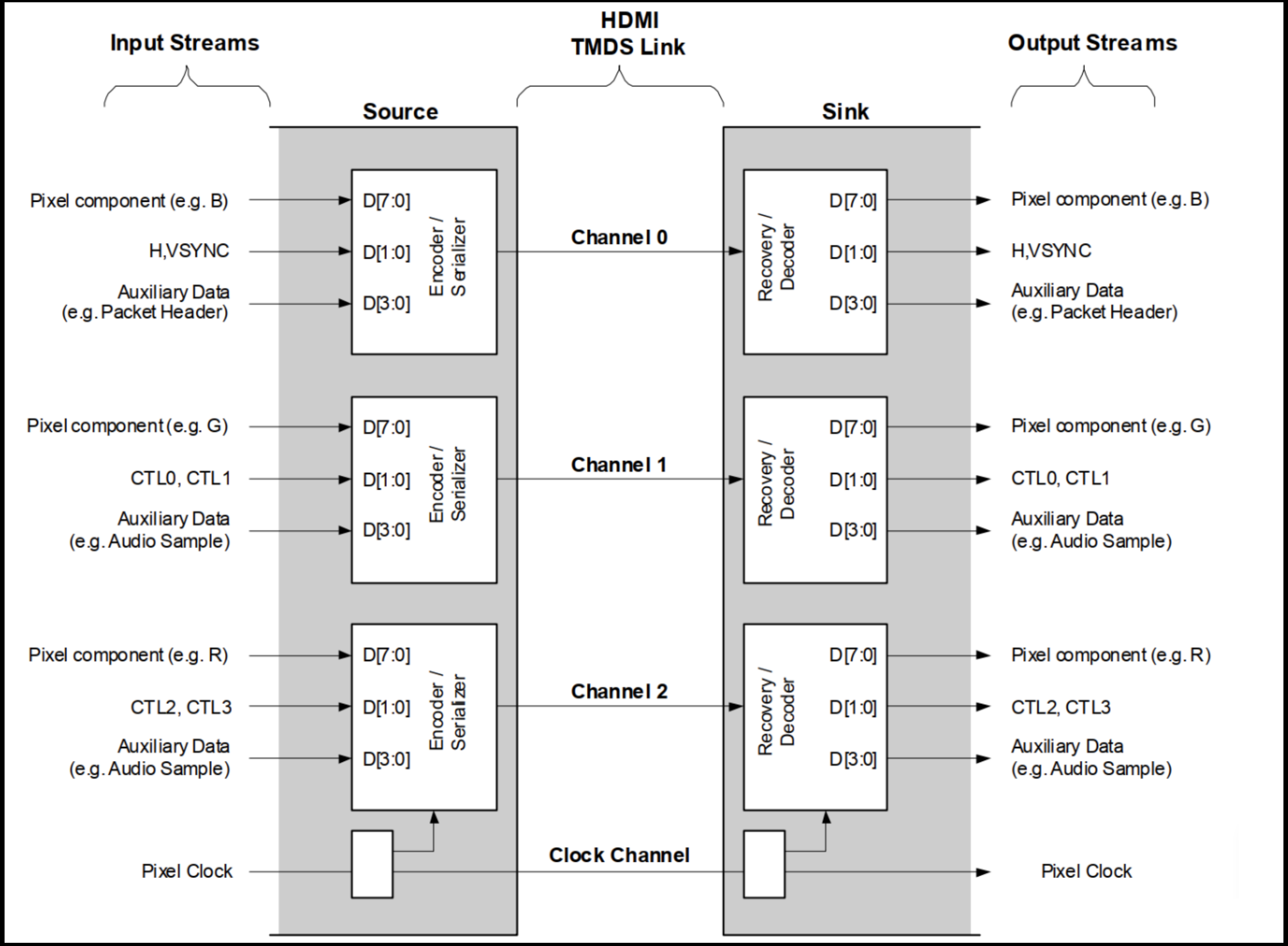
Figure 1: **TMDS Transmitter Design**

FPGA_HDMI


Hdmi_Trans
PLL
VPG
Encode
Encode
Encode
Parallel to Serial
Parallel to Serial
Parallel to Serial
Parallel to Serial
Single end to Differential
Single end to Differential
Single end to Differential
Single end to Differential
TMDS [3:0]
TMDSI_Tx_clk

FPGA_HDMI

Encode模塊將Pixel,H.V_SYNC及音訊等額外數據編碼成10bit的TMDS編碼



FPGA_HDMI

OSERDESE2為XILINX專門的串並轉換器，透過級聯可以一次輸出10bit的Serial資料(詳情可參考U.G 471)

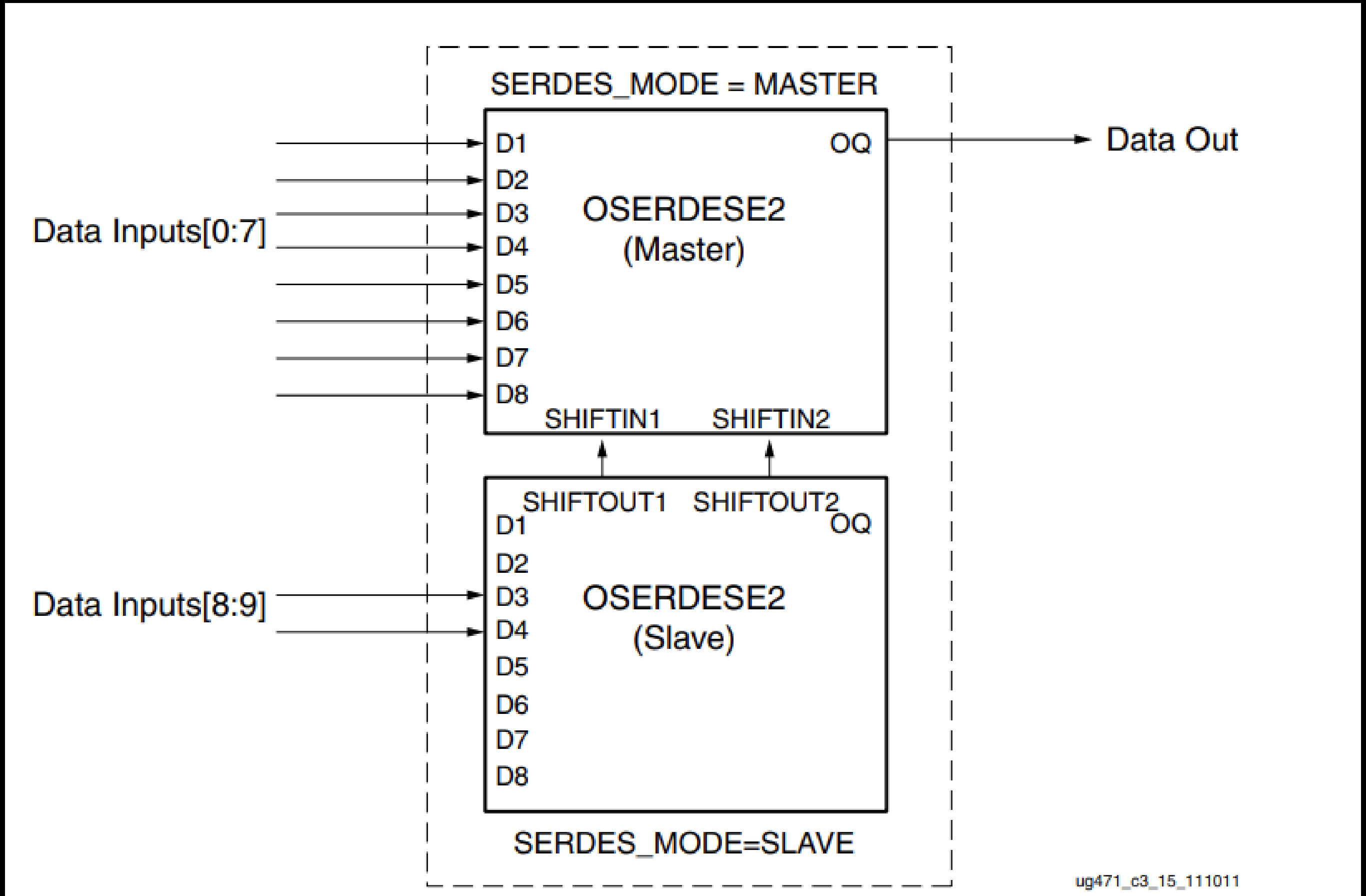
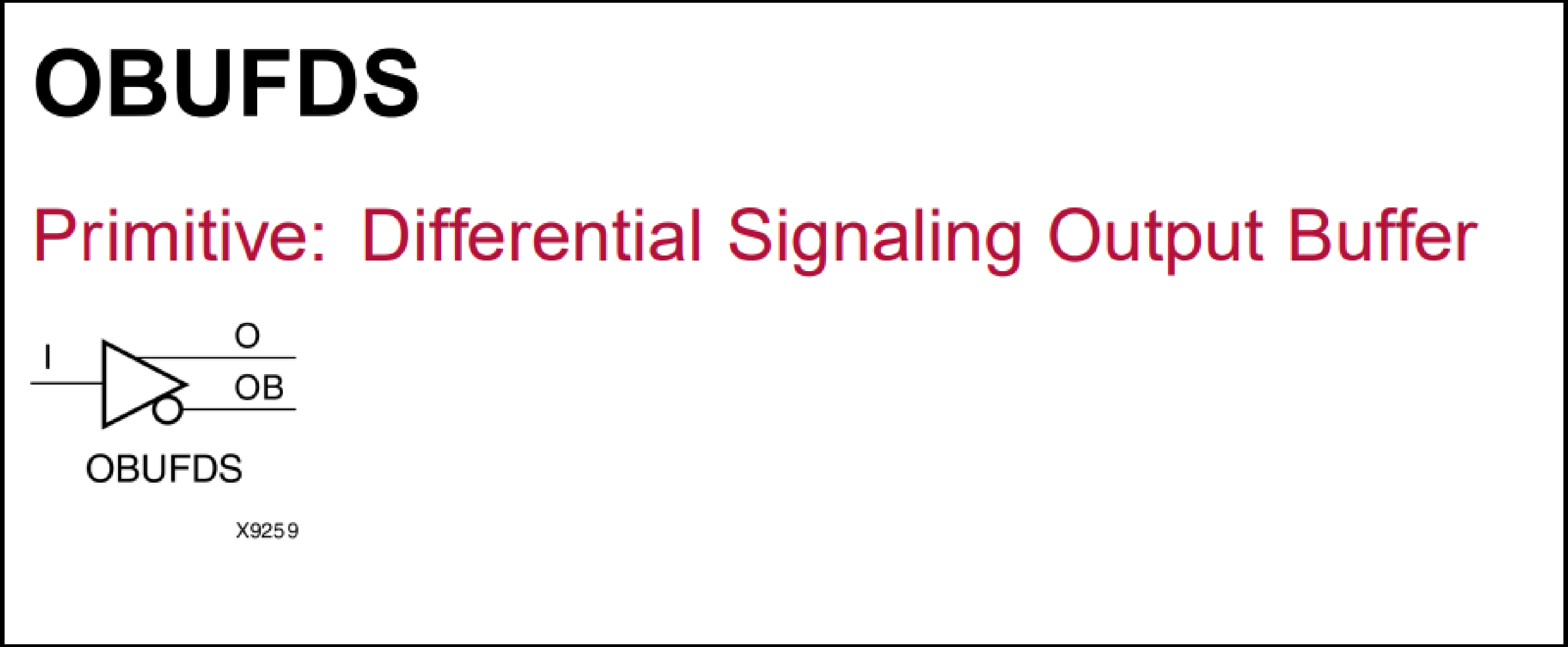


Figure 3-15: **Block Diagram of OSERDESE2 Width Expansion**

AMD

FPGA_HDMI

OBUFDS為XILINX的差分輸出緩衝器，可將FPGA內部信號轉成TMDS等差分信號(詳情參考U.G 471)

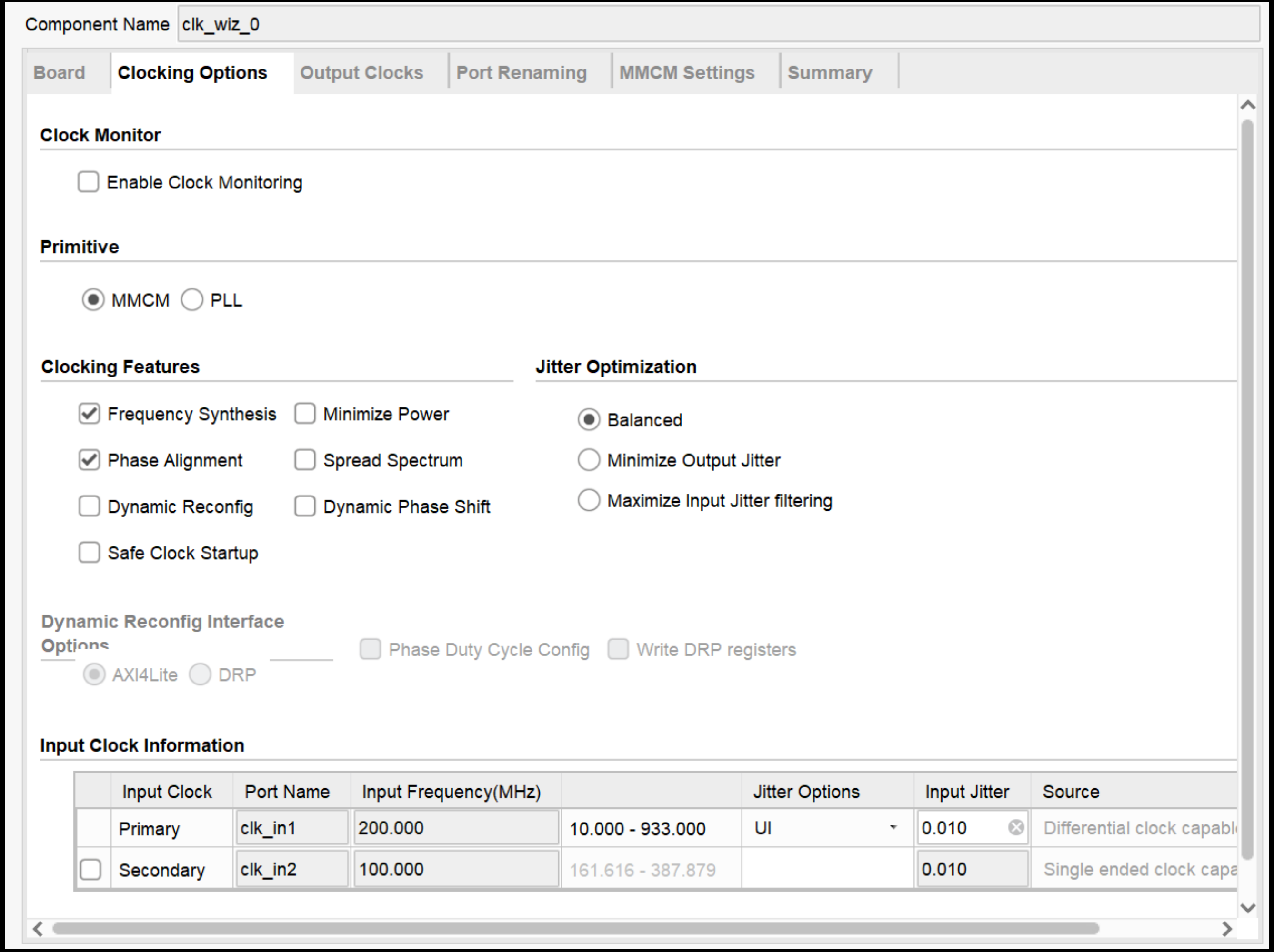


OBUFDS

Primitive: Differential Signaling Output Buffer

PLL IP Setting

PLL IP Setting



Component Name clk_wiz_0

Board | **Clocking Options** | Output Clocks | Port Renaming | MMCM Settings | Summary

Clock Monitor

☐ Enable Clock Monitoring

Primitive

[✓ MMCM] PLL

Clocking Features

- ☑ Frequency Synthesis
- ☐ Minimize Power
- ☑ Phase Alignment
- ☐ Spread Spectrum
- ☐ Dynamic Reconfig
- ☐ Dynamic Phase Shift
- ☐ Safe Clock Startup

Jitter Optimization

[✓ Balanced] Minimize Output Jitter / Maximize Input Jitter filtering

Dynamic Reconfig Interface Options

[✓ AXI4Lite] DRP

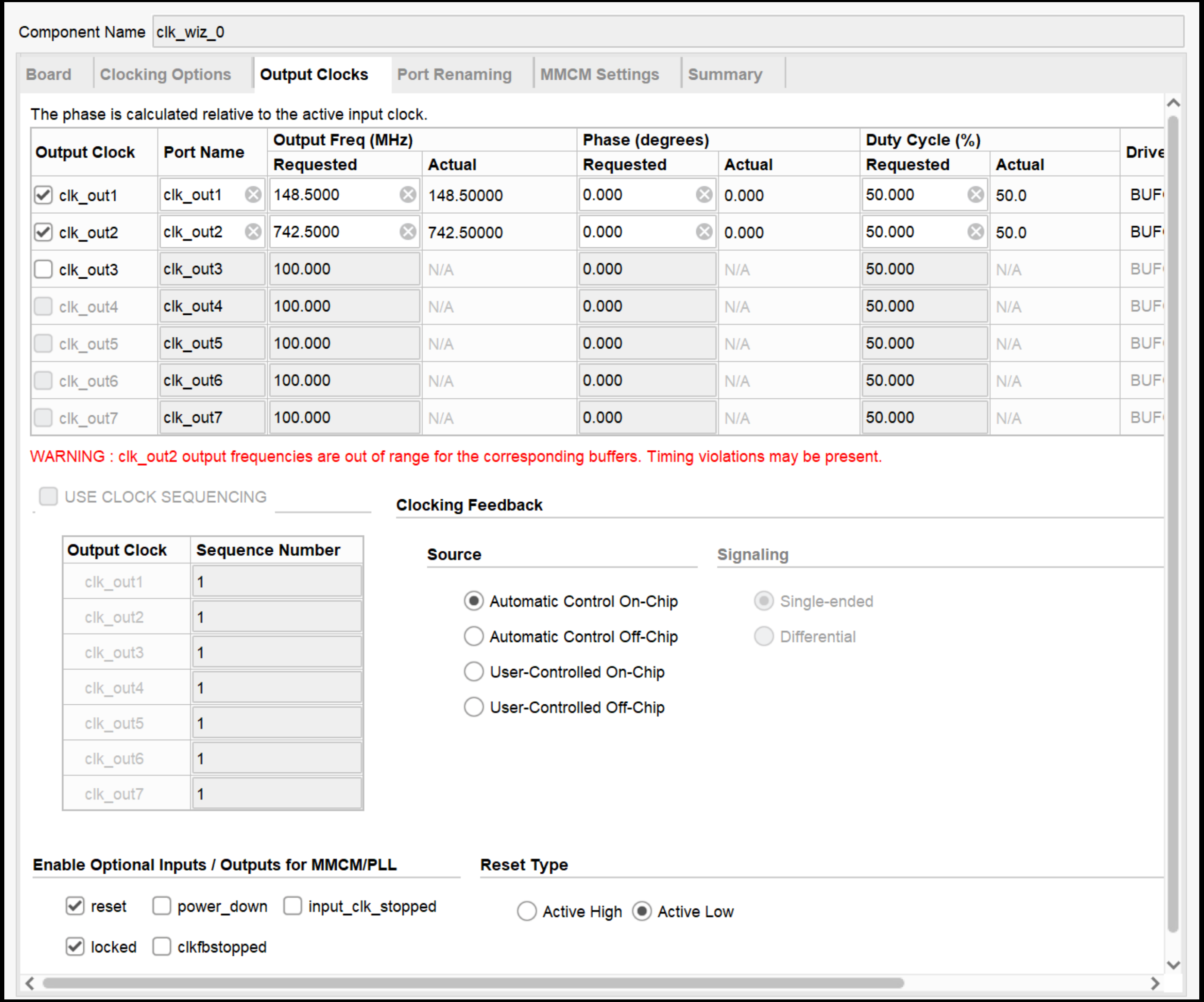
☐ Phase Duty Cycle Config ☐ Write DRP registers

Input Clock Information

	Input Clock	Port Name	Input Frequency(MHz)		Jitter Options	Input Jitter	Source
	Primary	clk_in1	200.000	10.000 - 933.000	UI	0.010	Differential clock capabl
☐	Secondary	clk_in2	100.000	161.616 - 387.879		0.010	Single ended clock capa

PLL IP Setting

PLL IP Setting



Component Name clk_wiz_0

Board | Clocking Options | **Output Clocks** | Port Renaming | MMCM Settings | Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drive
		Requested	Actual	Requested	Actual	Requested	Actual	
✓ clk_out1	clk_out1	148.5000	148.50000	0.000	0.000	50.000	50.0	BUF
✓ clk_out2	clk_out2	742.5000	742.50000	0.000	0.000	50.000	50.0	BUF
☐ clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUF
☐ clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUF
☐ clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUF
☐ clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUF
☐ clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUF

WARNING : clk_out2 output frequencies are out of range for the corresponding buffers. Timing violations may be present.

☐ USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Clocking Feedback

Source

- [✓ Automatic Control On-Chip]
- Automatic Control Off-Chip
- User-Controlled On-Chip
- User-Controlled Off-Chip

Signaling

- [✓ Single-ended]
- Differential

Enable Optional Inputs / Outputs for MMCM/PLL

- ✓ reset
- ☐ power_down
- ☐ input_clk_stopped
- ✓ locked
- ☐ clkfbstopped

Reset Type

Active High [✓ Active Low]

HDMI_trans_top

Port define

```
`timescale 1ns / 1ps
module hdmi_trans_top(
    input   wire      clk_in1_p          ,
    input   wire      clk_in1_n          ,
    input   wire      rst_n              ,
    output  wire      hdmi_oen           ,
    output  wire      hdmi_tx_clk_n      ,
    output  wire      hdmi_tx_clk_p      ,
    output  wire      hdmi_tx_chn_r_n    ,
    output  wire      hdmi_tx_chn_r_p    ,
    output  wire      hdmi_tx_chn_g_n    ,
    output  wire      hdmi_tx_chn_g_p    ,
    output  wire      hdmi_tx_chn_b_n    ,
    output  wire      hdmi_tx_chn_b_p
    );
```

HDMI_trans_top

wire define

```
parameter   CNT_MAX = 1000   ;
```

```
wire          locked       ;
wire          rst          ;
wire          clk1x        ;
wire          clk5x        ;
wire   [7:0]  rgb_r        ;
wire   [7:0]  rgb_g        ;
wire   [7:0]  rgb_b        ;
wire          vpg_de       ;
wire          vpg_hs       ;
wire          vpg_vs       ;
reg           hdmi_oen_r   ;
reg    [10:0] cnt          ;
```

```
assign rst = ~locked               ;
assign hemi_oen = hdmi_oen_r       ;
```

HDMI_trans_top

Counter & hdmi_enable

```
always@(posedge clk1x)begin
    if(rst)
        cnt <= 'd0;
    else if(cnt < CNT_MAX)
        cnt <= cnt + 1'b1;
    else
        cnt <= cnt;
end

always@(posedge clk1x)begin
    if(rst)
        hdmi_oen_r <= 1'b0;
    else if(cnt <= CNT_MAX)
        hdmi_oen_r <= 1'b1;
end
```

HDMI_trans_top

Pll inst & Video_Pattern inst

```
clk_wiz_0 inst_clock
   (
    // Clock out ports
    .clk_out1(clk1x),     // output clk_out1
    .clk_out2(clk5x),     // output clk_out2
    // Status and control signals
    .resetn(rst_n), // input reset
    .locked(locked),       // output locked
   // Clock in ports
    .clk_in1_p(clk_in1_p),    // input clk_in1_p
    .clk_in1_n(clk_in1_n)    // input clk_in1_n
);

vga_shift inst_vga_shift
    (
     .rst(rst),
     .vpg_pclk(clk1x),
     .vpg_de(vpg_de),
     .vpg_hs(vpg_hs),
     .vpg_vs(vpg_vs),
     .rgb_r(rgb_r),
     .rgb_g(rgb_g),
     .rgb_b(rgb_b)
    );
```

HDMI_trans_top

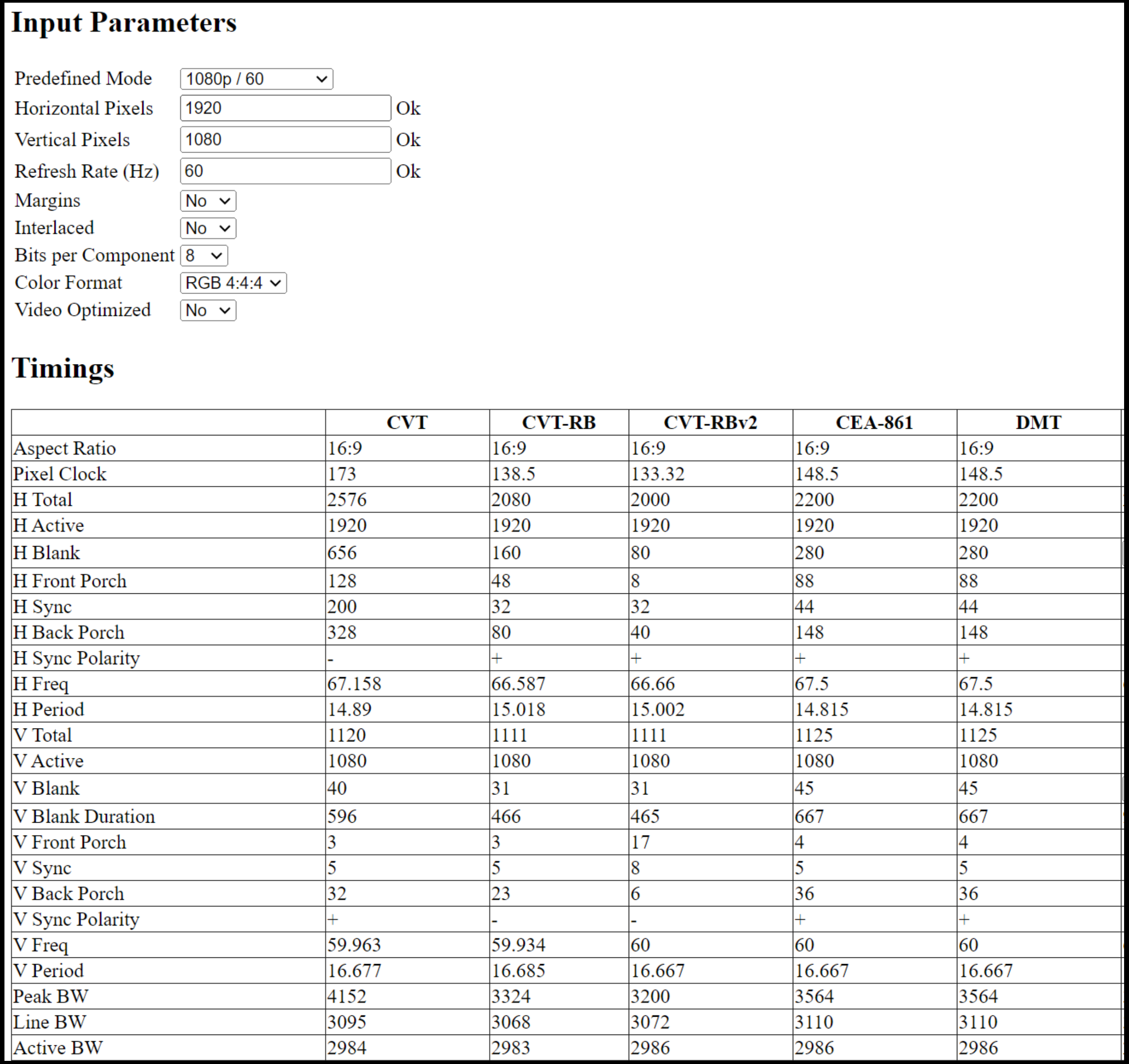
HDMI_trans inst

```
hdmi_trans inst_hdmi_trans
    (
     .clk1x(clk1x),
     .clk5x(clk5x),
     .rst(rst),
     .image_r(rgb_r),
     .image_g(rgb_g),
     .image_b(rgb_b),
     .vsync(vpg_vs),
     .hsync(vpg_hs),
     .de(vpg_de),
     .hdmi_tx_clk_n(hdmi_tx_clk_n),
     .hdmi_tx_clk_p(hdmi_tx_clk_p),
     .hdmi_tx_chn_r_n(hdmi_tx_chn_r_n),
     .hdmi_tx_chn_r_p(hdmi_tx_chn_r_p),
     .hdmi_tx_chn_g_n(hdmi_tx_chn_g_n),
     .hdmi_tx_chn_g_p(hdmi_tx_chn_g_p),
     .hdmi_tx_chn_b_n(hdmi_tx_chn_b_n),
     .hdmi_tx_chn_b_p(hdmi_tx_chn_b_p)
    );
endmodule
```

Vga_shift

Port degine & parameter

```
`timescale 1ns / 1ps
module vga_shift(
input   wire          rst       ,
input   wire          vpg_pclk  ,
output  reg           vpg_de    ,
output  reg           vpg_hs    ,
output  reg           vpg_vs    ,
output  wire  [7:0]   rgb_r     ,
output  wire  [7:0]   rgb_g     ,
output  wire  [7:0]   rgb_b
    );

parameter    H_TOTAL = 2200 - 1    ;
parameter    H_SYNC  = 44 - 1     ;
parameter    H_START = 190 - 1     ;
parameter    H_END = 2110 - 1      ;
parameter    V_TOTAL = 1125 - 1    ;
parameter    V_SYNC = 5 - 1       ;
parameter    V_START = 41 - 1     ;
parameter    V_END = 1121 - 1      ;
parameter    SQUARE_X = 500       ;
parameter    SQUARE_Y = 500       ;
parameter    SCREEN_X = 1920       ;
parameter    SCREEN_Y = 1080       ;
```



Input Parameters

Predefined Mode	1080p / 60	
Horizontal Pixels	1920	Ok
Vertical Pixels	1080	Ok
Refresh Rate (Hz)	60	Ok
Margins	No	
Interlaced	No	
Bits per Component	8	
Color Format	RGB 4:4:4	
Video Optimized	No	

Timings

	CVT	CVT-RB	CVT-RBv2	CEA-861	DMT
Aspect Ratio	16:9	16:9	16:9	16:9	16:9
Pixel Clock	173	138.5	133.32	148.5	148.5
H Total	2576	2080	2000	2200	2200
H Active	1920	1920	1920	1920	1920
H Blank	656	160	80	280	280
H Front Porch	128	48	8	88	88
H Sync	200	32	32	44	44
H Back Porch	328	80	40	148	148
H Sync Polarity	-	+	+	+	+
H Freq	67.158	66.587	66.66	67.5	67.5
H Period	14.89	15.018	15.002	14.815	14.815
V Total	1120	1111	1111	1125	1125
V Active	1080	1080	1080	1080	1080
V Blank	40	31	31	45	45
V Blank Duration	596	466	465	667	667
V Front Porch	3	3	17	4	4
V Sync	5	5	8	5	5
V Back Porch	32	23	6	36	36
V Sync Polarity	+	-	-	+	+
V Freq	59.963	59.934	60	60	60
V Period	16.677	16.685	16.667	16.667	16.667
Peak BW	4152	3324	3200	3564	3564
Line BW	3095	3068	3072	3110	3110
Active BW	2984	2983	2986	2986	2986

Vga_shift

Reg define

```
reg  [12:0] cnt_h   ;
reg  [12:0] cnt_v   ;
reg  [11:0] x       ;
reg         flag_x  ;
reg  [11:0] y       ;
reg         flag_y  ;
reg  [23:0] rgb     ;
```

```
assign {rgb_r , rgb_g , rgb_b } = rgb ;
```

Vga_shift

Counter

```
always@(posedge vpg_pclk)begin
    if(rst)
        cnt_h <= 'd0;
    else if(cnt_h == H_TOTAL)
        cnt_h <= 'd0;
    else
        cnt_h <= cnt_h + 1'b1;
end

always@(posedge vpg_pclk)begin
    if(rst)
        cnt_v <= 'd0;
    else if(cnt_v == V_TOTAL && cnt_h == H_TOTAL)
        cnt_v <= 'd0;
    else if(cnt_h == H_TOTAL)
        cnt_v <= cnt_v + 1'b1;
end
```

Vga_shift

H_sync & V_sync

```
always@(posedge vpg_pclk)begin
    if(rst)
        vpg_hs <= 1'b1;
    else if(cnt_h == H_TOTAL)
        vpg_hs <= 1'b1;
    else if(cnt_h == H_TOTAL)
        vpg_hs <= 1'b0;
end

always@(posedge vpg_pclk)begin
    if(rst)
        vpg_vs <= 1'b1;
    else if(cnt_v == V_TOTAL && cnt_h == H_TOTAL)
        vpg_vs <= 1'b1;
    else if(cnt_v == V_SYNC && cnt_h == H_TOTAL)
        vpg_vs <= 1'b0;
end
```

Vga_shift

Vpg_enable

```
always@(posedge vpg_pclk)begin
    if(rst)
        vpg_de <= 1'b1;
    else if((cnt_h >= H_START) && (cnt_h < H_END) && (cnt_v >= V_START) &&
(cnt_v < V_END))
        vpg_de <= 1'b1;
    else
        vpg_de <= 1'b0;
end
```

Vga_shift

Video output x_axis

```
always@(posedge vpg_pclk)begin
    if(rst)
        x <= 'd0;
    else if(flag_x == 1'b0 && cnt_v == V_TOTAL && cnt_h == H_TOTAL)
        x <= x + 1'b1;
    else if(flag_x == 1'b1 && cnt_v == V_TOTAL && cnt_h == H_TOTAL)
        x <= x - 1'b1;
end

always@(posedge vpg_pclk)begin
    if(rst)
        flag_x <= 1'b0;
    else if(flag_x == 1'b0 && cnt_v == V_TOTAL && cnt_h == H_TOTAL && x
==(H_END - H_START - SQUARE_X - 1'b1))
        flag_x <= 1'b1;
    else if(flag_x == 1'b1 && cnt_v == V_TOTAL && cnt_h == H_TOTAL)
        flag_x <= 1'b0;
end
```

Vga_shift

Video output y_axis

```
always@(posedge vpg_pclk)begin
    if(rst)
        y <= 'd0;
    else if(flag_y == 1'b0 && cnt_v == V_TOTAL && cnt_h == H_TOTAL)
        y <= y + 1'b1;
    else if(flag_y == 1'b1 && cnt_v == V_TOTAL && cnt_h == H_TOTAL)
        y <= y - 1'b1;
end

always@(posedge vpg_pclk)begin
    if(rst)
        flag_y <= 1'b0;
    else if(flag_y == 1'b0 && cnt_v == V_TOTAL && cnt_h == H_TOTAL && y
==(V_END - V_START - SQUARE_Y - 1'b1))
        flag_y <= 1'b1;
    else if(flag_y == 1'b1 && cnt_v == V_TOTAL && cnt_h == H_TOTAL && y
=='d1)
        flag_y <= 1'b0;
end
```

Vga_shift

Video output pattern

```
always@(posedge vpg_pclk)begin
    if(rst)
        rgb <= 'd0;
    else if(cnt_h >= H_START+x && cnt_h <= H_START+SQUARE_X+x && cnt_v >= V_START+y && cnt_v <= V_START+SQUARE_Y+y)
        rgb <= 24'hFFB6C1;
    else if(cnt_h >= H_START && cnt_h < H_END && cnt_v >= V_START && cnt_v <= V_END && cnt_h[4:0] >= 'd20)
        rgb <= 24'h00FF00;
    else if(cnt_h>=H_START && cnt_h<H_END && cnt_v>=V_START && cnt_v<=V_END && (cnt_h[4:0]>='d10 && cnt_h[2:0]<'d20))
        rgb <= 24'h0000FF;
    else if(cnt_h >= H_START && cnt_h < H_END && cnt_v >= V_START && cnt_v <= V_END && cnt_h[4:0] < 'd10)
        rgb <= 24'hFF0000;
    else
        rgb <= 'd0;
end
endmodule
```

HDMI_trans

Port define

```
`timescale 1ns / 1ps
module hdmi_trans(
    input       clk1x,
    input       clk5x,
    input       rst,
    input [7:0] image_r,
    input [7:0] image_g,
    input [7:0] image_b,
    input       vsync,
    input       hsync,
    input       de,
    output      hdmi_tx_clk_n,
    output      hdmi_tx_clk_p,
    output      hdmi_tx_chn_r_n,
    output      hdmi_tx_chn_r_p,
    output      hdmi_tx_chn_g_n,
    output      hdmi_tx_chn_g_p,
    output      hdmi_tx_chn_b_n,
    output      hdmi_tx_chn_b_p
    );

wire    [9:0]   encode_chn_r;
wire    [9:0]   encode_chn_g;
wire    [9:0]   encode_chn_b;
```

HDMI_trans

Encode inst

```
    encode inst_encode_chn_r(
        .clkin(clk1x),
        .rstin(rst),
        .din(image_r),
        .c0(1'b0),
        .c1(1'b0),
        .de(de),
        .dout(encode_chn_r));
    encode inst_encode_chn_g(
        .clkin(clk1x),
        .rstin(rst),
        .din(image_g),
        .c0(1'b0),
        .c1(1'b0),
        .de(de),
        .dout(encode_chn_g));
    encode inst_encode_chn_b(
        .clkin(clk1x),
        .rstin(rst),
        .din(image_b),
        .c0(1'b0),
        .c1(1'b0),
        .de(de),
        .dout(encode_chn_b));
```

HDMI_trans

P_to_S inst

```
parallel_to_serial inst_parallel_to_serial_clk(
        .clk1x(clk1x),
        .clk5x(clk5x),
        .rst(rst),
        .din(10'b11111_00000),
        .dout_p(hdmi_tx_clk_p),
        .dout_n(hdmi_tx_clk_n)
    );
```

HDMI_trans

P_to_S inst

```
parallel_to_serial inst_parallel_to_serial_chn_r(
        .clk1x(clk1x),
        .clk5x(clk5x),
        .rst(rst),
        .din(encode_chn_r),
        .dout_p(hdmi_tx_chn_r_p),
        .dout_n(hdmi_tx_chn_r_n));
    parallel_to_serial inst_parallel_to_serial_chn_g(
        .clk1x(clk1x),
        .clk5x(clk5x),
        .rst(rst),
        .din(encode_chn_g),
        .dout_p(hdmi_tx_chn_g_p),
        .dout_n(hdmi_tx_chn_g_n));
    parallel_to_serial inst_parallel_to_serial_chn_b(
        .clk1x(clk1x),
        .clk5x(clk5x),
        .rst(rst),
        .din(encode_chn_b),
        .dout_p(hdmi_tx_chn_b_p),
        .dout_n(hdmi_tx_chn_b_n));

endmodule
```

HDMI_trans

P_to_S inst

```
parallel_to_serial inst_parallel_to_serial_chn_r(
       .clk1x(clk1x),
       .clk5x(clk5x),
       .rst(rst),
       .din(encode_chn_r),
       .dout_p(hdmi_tx_chn_r_p),
       .dout_n(hdmi_tx_chn_r_n));
    parallel_to_serial inst_parallel_to_serial_chn_g(
       .clk1x(clk1x),
       .clk5x(clk5x),
       .rst(rst),
       .din(encode_chn_g),
       .dout_p(hdmi_tx_chn_g_p),
       .dout_n(hdmi_tx_chn_g_n));
    parallel_to_serial inst_parallel_to_serial_chn_b(
       .clk1x(clk1x),
       .clk5x(clk5x),
       .rst(rst),
       .din(encode_chn_b),
       .dout_p(hdmi_tx_chn_b_p),
       .dout_n(hdmi_tx_chn_b_n));

endmodule
```

encode

Refer to Xilinx.com

```
`timescale 1ns / 1ps
module encode(
    input                   clkin,
    input                   rstin,
    input           [7:0]   din,
    input                   c0,
    input                   c1,
    input                   de,
    output reg [9:0]        dout
    );

reg [3:0]   n1d;
reg [7:0]   din_q;

always @ (posedge clkin) begin
    n1d <= din[0] + din[1] + din[2] + din[3] + din[4] + din[5] + din[6] + din[7];

    din_q <= din;
  end
```

encode

Refer to Xilinx.com

```
///////////////////////////////////////////////////////////
// Stage 2: 9 bit -> 10 bit
// Refer to DVI 1.0 Specification, page 29, Figure 3-5
///////////////////////////////////////////////////////////
reg [3:0] n1q_m, n0q_m; // number of 1s and 0s for q_m
always @ (posedge clkin) begin
  n1q_m  <= q_m[0] + q_m[1] + q_m[2] + q_m[3] + q_m[4] + q_m[5] + q_m[6] + q_m[7];
  n0q_m  <= 4'h8 - (q_m[0] + q_m[1] + q_m[2] + q_m[3] + q_m[4] + q_m[5] + q_m[6] + q_m[7]);
end

parameter CTRLTOKEN0 = 10'b1101010100;
parameter CTRLTOKEN1 = 10'b0010101011;
parameter CTRLTOKEN2 = 10'b0101010100;
parameter CTRLTOKEN3 = 10'b1010101011;

reg [4:0] cnt; //disparity counter, MSB is the sign bit
wire decision2, decision3;

assign decision2 = (cnt == 5'h0) | (n1q_m == n0q_m);
/////////////////////////////////////////////////////////////////////////
// [(cnt > 0) and (N1q_m > N0q_m)] or [(cnt < 0) and (N0q_m > N1q_m)]
/////////////////////////////////////////////////////////////////////////
assign decision3 = (~cnt[4] & (n1q_m > n0q_m)) | (cnt[4] & (n0q_m > n1q_m));
```

encode

Refer to Xilinx.com

```
////////////////////////////////////
// pipe line alignment
////////////////////////////////////
reg       de_q, de_reg;
reg       c0_q, c1_q;
reg       c0_reg, c1_reg;
reg [8:0] q_m_reg;

always @ (posedge clkin) begin
  de_q    <= de;
  de_reg  <= de_q;

  c0_q    <= c0;
  c0_reg  <= c0_q;
  c1_q    <= c1;
  c1_reg  <= c1_q;

  q_m_reg <= q_m;
end
```

encode

Refer to Xilinx.com

```
///////////////////////////////
  // 10-bit out
  // disparity counter
  ///////////////////////////////
  always @ (posedge clkin or posedge rstin) begin
    if(rstin) begin
      dout <= 10'h0;
      cnt <= 5'h0;
    end else begin
      if (de_reg) begin
        if(decision2) begin
          dout[9]   <= ~q_m_reg[8];
          dout[8]   <= q_m_reg[8];
          dout[7:0] <= (q_m_reg[8]) ? q_m_reg[7:0] : ~q_m_reg[7:0];

          cnt <= (~q_m_reg[8]) ? (cnt + n0q_m - n1q_m) : (cnt + n1q_m - n0q_m);
        end else begin
          if(decision3) begin
            dout[9]   <= 1'b1;
            dout[8]   <= q_m_reg[8];
            dout[7:0] <= ~q_m_reg[7:0];

            cnt <= cnt + {q_m_reg[8], 1'b0} + (n0q_m - n1q_m);
          end else begin
```

encode

Refer to Xilinx.com

```
dout[9]    <= 1'b0;
            dout[8]    <= q_m_reg[8];
            dout[7:0] <= q_m_reg[7:0];

            cnt <= cnt - {~q_m_reg[8], 1'b0} + (n1q_m - n0q_m);
          end
        end
      end else begin
        case ({c1_reg, c0_reg})
          2'b00:    dout <= CTRLTOKEN0;
          2'b01:    dout <= CTRLTOKEN1;
          2'b10:    dout <= CTRLTOKEN2;
          default: dout <= CTRLTOKEN3;
        endcase

        cnt <=#1 5'h0;
      end
    end
  end
endmodule
```

Parallel_to_serial

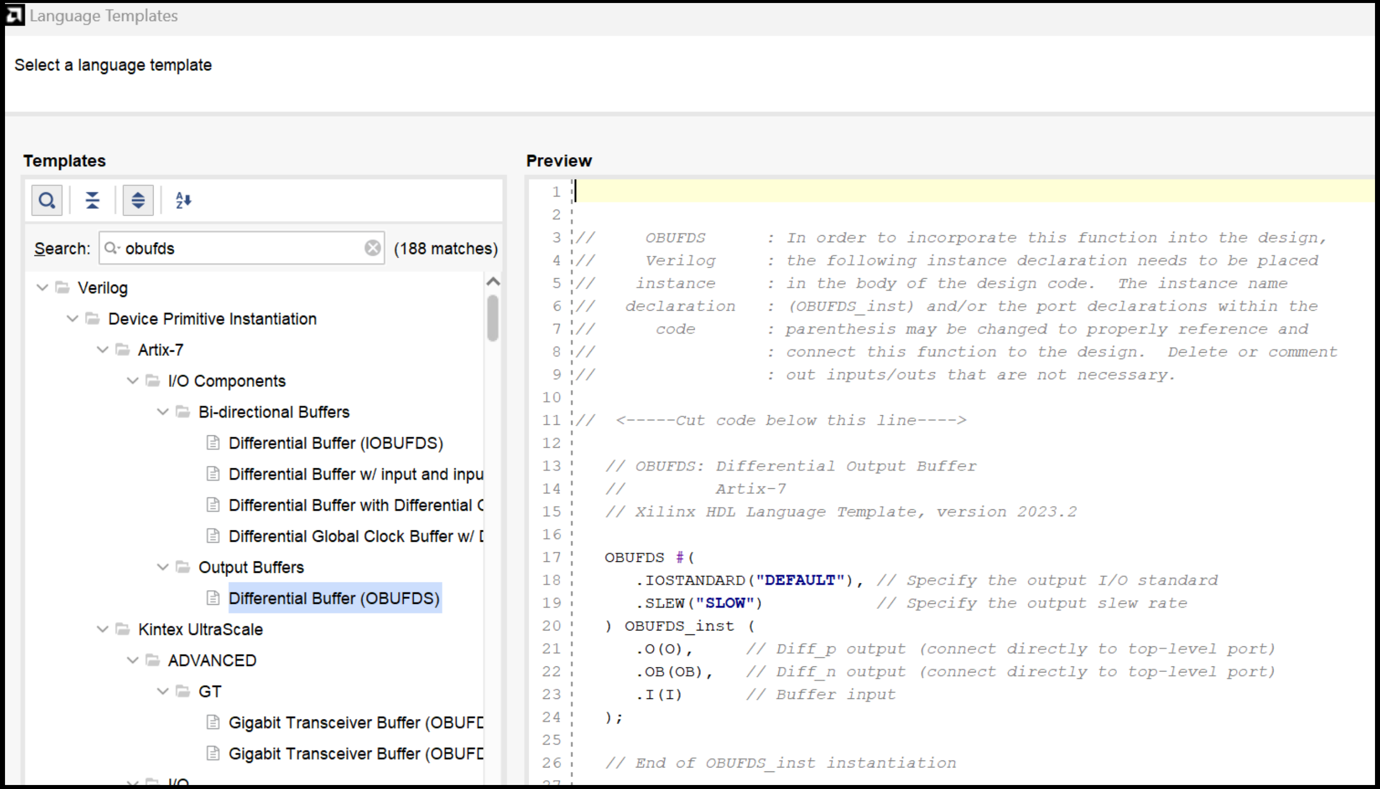
Port define

```
`timescale 1ns / 1ps

module parallel_to_serial(
  input      wire            clk1x,
  input      wire            clk5x,
  input      wire            rst,
  input      wire     [9:0]  din,
  output     wire            dout_p,
  output     wire            dout_n
);

  //wire define
  wire         dout;
  wire         shift_in1;
  wire         shift_in2;
```

Parallel_to_serial

OBUFDS inst

```
//******************************************************
  //**                    main code
  //******************************************************

  OBUFDS #(
    .IOSTANDARD("DEFAULT"),
    .SLEW("SLOW"))
  OBUFDS_inst(
    .O(dout_p),
    .OB(dout_n),
    .I(dout)
  );
```


Language Templates
Select a language template
Templates
Preview
Verilog
Device Primitive Instantiation
Artix-7
I/O Components
Bi-directional Buffers
Differential Buffer (IOBUFDS)
Differential Buffer w/ input and inpu
Differential Buffer with Differential C
Differential Global Clock Buffer w/ D
Output Buffers
Differential Buffer (OBUFDS)
Kintex UltraScale
ADVANCED
GT
Gigabit Transceiver Buffer (OBUFD
Gigabit Transceiver Buffer (OBUFD
I/O

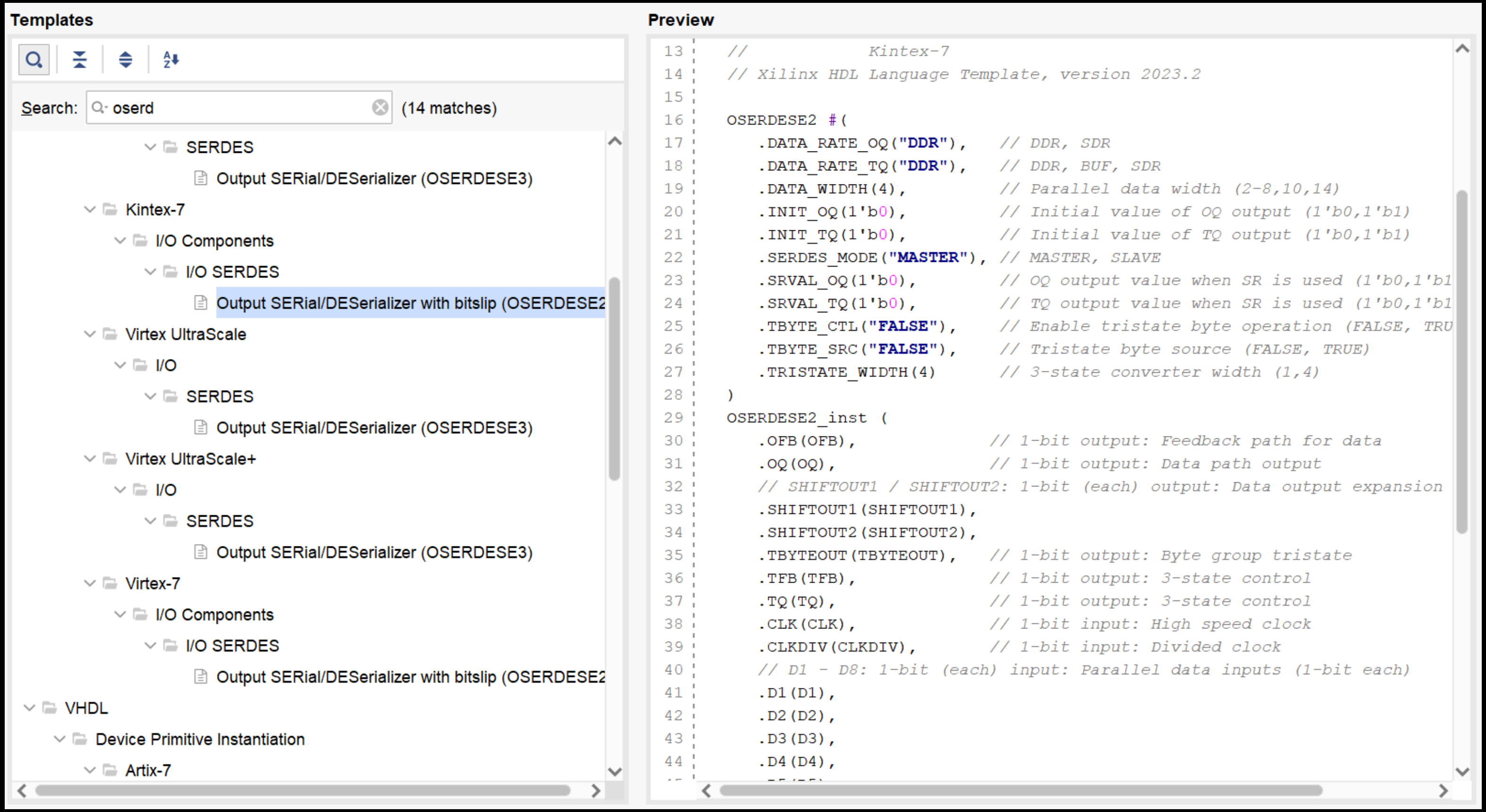
Search: obufds (188 matches)

```


//     OBUFDS      : In order to incorporate this function into the design,
//     Verilog     : the following instance declaration needs to be placed
//    instance     : in the body of the design code.  The instance name
//   declaration   : (OBUFDS_inst) and/or the port declarations within the
//      code       : parenthesis may be changed to properly reference and
//                 : connect this function to the design.  Delete or comment
//                 : out inputs/outs that are not necessary.

//  <-----Cut code below this line---->

   // OBUFDS: Differential Output Buffer
   //         Artix-7
   // Xilinx HDL Language Template, version 2023.2

   OBUFDS #(
      .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
      .SLEW("SLOW")           // Specify the output slew rate
   ) OBUFDS_inst (
      .O(O),     // Diff_p output (connect directly to top-level port)
      .OB(OB),   // Diff_n output (connect directly to top-level port)
      .I(I)      // Buffer input
   );

   // End of OBUFDS_inst instantiation
```

Parallel_to_serial

OSERDESE2_Master inst

```
OSERDESE2 #(
    .DATA_RATE_OQ   ("DDR"),
    .DATA_RATE_TQ   ("SDR"),
    .DATA_WIDTH     (10),
    .INIT_OQ        (1'b0),
    .INIT_TQ        (1'b0),
    .SERDES_MODE    ("MASTER"),
    .SRVAL_OQ       (1'b0),
    .SRVAL_TQ       (1'b0),
    .TBYTE_CTL      ("FALSE"),
    .TBYTE_SRC      ("FALSE"),
    .TRISTATE_WIDTH (1)
  )
OSERDESE2_inst_Master (
    .OFB          (),
    .OQ           (dout),
    .SHIFTOUT1    (),
    .SHIFTOUT2    (),
    .TBYTEOUT     (),
    .TFB          (),
    .TQ           (),
    .CLK          (clk5x),
    .CLKDIV       (clk1x),
```



Templates

Search: oserd (14 matches)

- SERDES
  - Output SERial/DESerializer (OSERDESE3)
- Kintex-7
  - I/O Components
    - I/O SERDES
      - Output SERial/DESerializer with bitslip (OSERDESE2
- Virtex UltraScale
  - I/O
    - SERDES
      - Output SERial/DESerializer (OSERDESE3)
- Virtex UltraScale+
  - I/O
    - SERDES
      - Output SERial/DESerializer (OSERDESE3)
- Virtex-7
  - I/O Components
    - I/O SERDES
      - Output SERial/DESerializer with bitslip (OSERDESE2
- VHDL
  - Device Primitive Instantiation
    - Artix-7

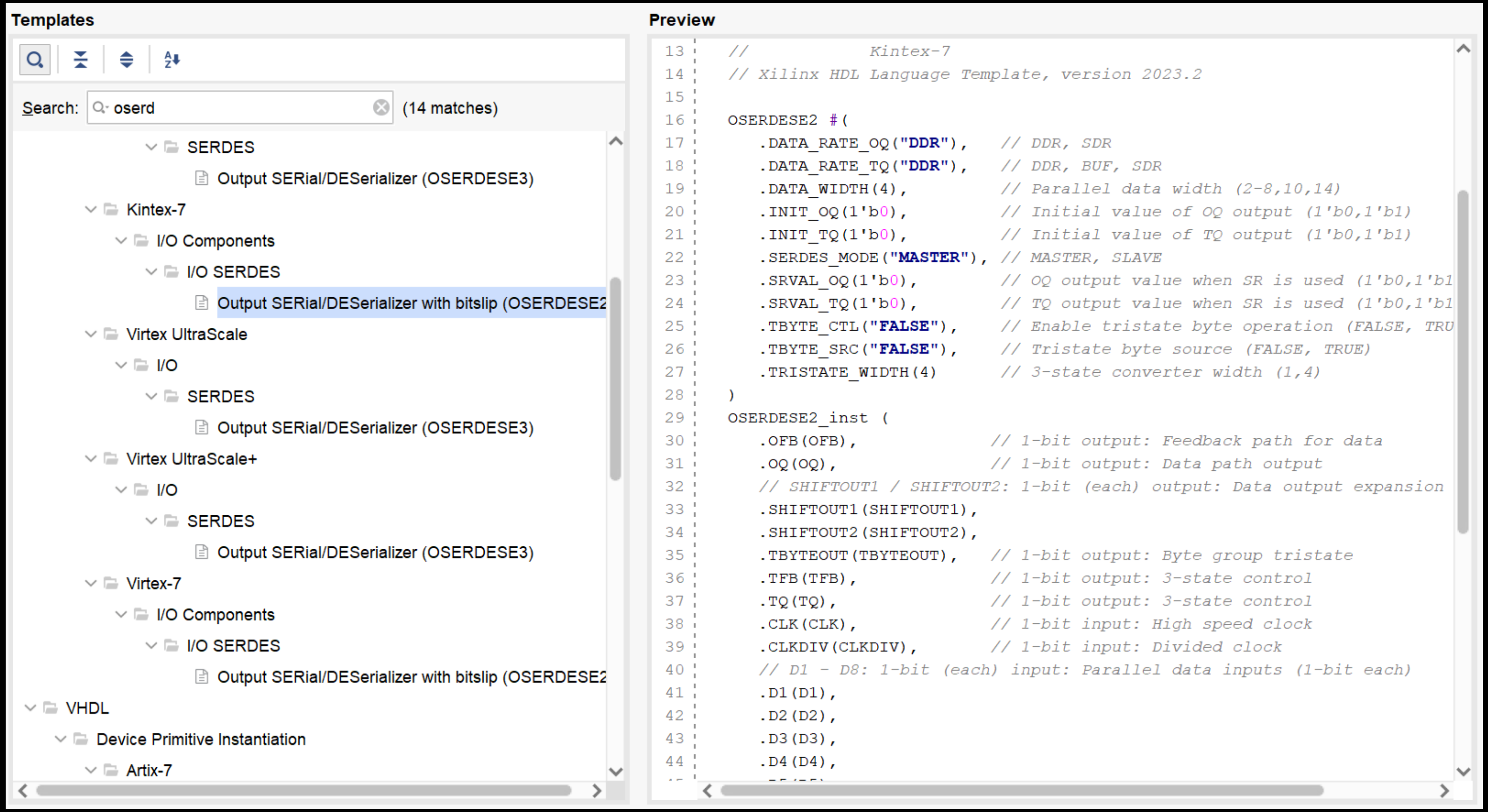
Preview

```
//              Kintex-7
// Xilinx HDL Language Template, version 2023.2

OSERDESE2 #(
   .DATA_RATE_OQ("DDR"),   // DDR, SDR
   .DATA_RATE_TQ("DDR"),   // DDR, BUF, SDR
   .DATA_WIDTH(4),         // Parallel data width (2-8,10,14)
   .INIT_OQ(1'b0),         // Initial value of OQ output (1'b0,1'b1)
   .INIT_TQ(1'b0),         // Initial value of TQ output (1'b0,1'b1)
   .SERDES_MODE("MASTER"), // MASTER, SLAVE
   .SRVAL_OQ(1'b0),        // OQ output value when SR is used (1'b0,1'b1
   .SRVAL_TQ(1'b0),        // TQ output value when SR is used (1'b0,1'b1
   .TBYTE_CTL("FALSE"),    // Enable tristate byte operation (FALSE, TRU
   .TBYTE_SRC("FALSE"),    // Tristate byte source (FALSE, TRUE)
   .TRISTATE_WIDTH(4)      // 3-state converter width (1,4)
)
OSERDESE2_inst (
   .OFB(OFB),             // 1-bit output: Feedback path for data
   .OQ(OQ),               // 1-bit output: Data path output
   // SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion
   .SHIFTOUT1(SHIFTOUT1),
   .SHIFTOUT2(SHIFTOUT2),
   .TBYTEOUT(TBYTEOUT),   // 1-bit output: Byte group tristate
   .TFB(TFB),             // 1-bit output: 3-state control
   .TQ(TQ),               // 1-bit output: 3-state control
   .CLK(CLK),             // 1-bit input: High speed clock
   .CLKDIV(CLKDIV),       // 1-bit input: Divided clock
   // D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
   .D1(D1),
   .D2(D2),
   .D3(D3),
   .D4(D4),
```

Parallel_to_serial

OSERDESE2_Master inst

```
    .D1           (din[0]),
    .D2           (din[1]),
    .D3           (din[2]),
    .D4           (din[3]),
    .D5           (din[4]),
    .D6           (din[5]),
    .D7           (din[6]),
    .D8           (din[7]),
    .OCE          (1'b1),
    .RST          (rst),
    .SHIFTIN1     (shift_in1),
    .SHIFTIN2     (shift_in2),

    .T1           (1'b0),
    .T2           (1'b0),
    .T3           (1'b0),
    .T4           (1'b0),
    .TBYTEIN      (1'b0),
    .TCE          (1'b0)
  );
```



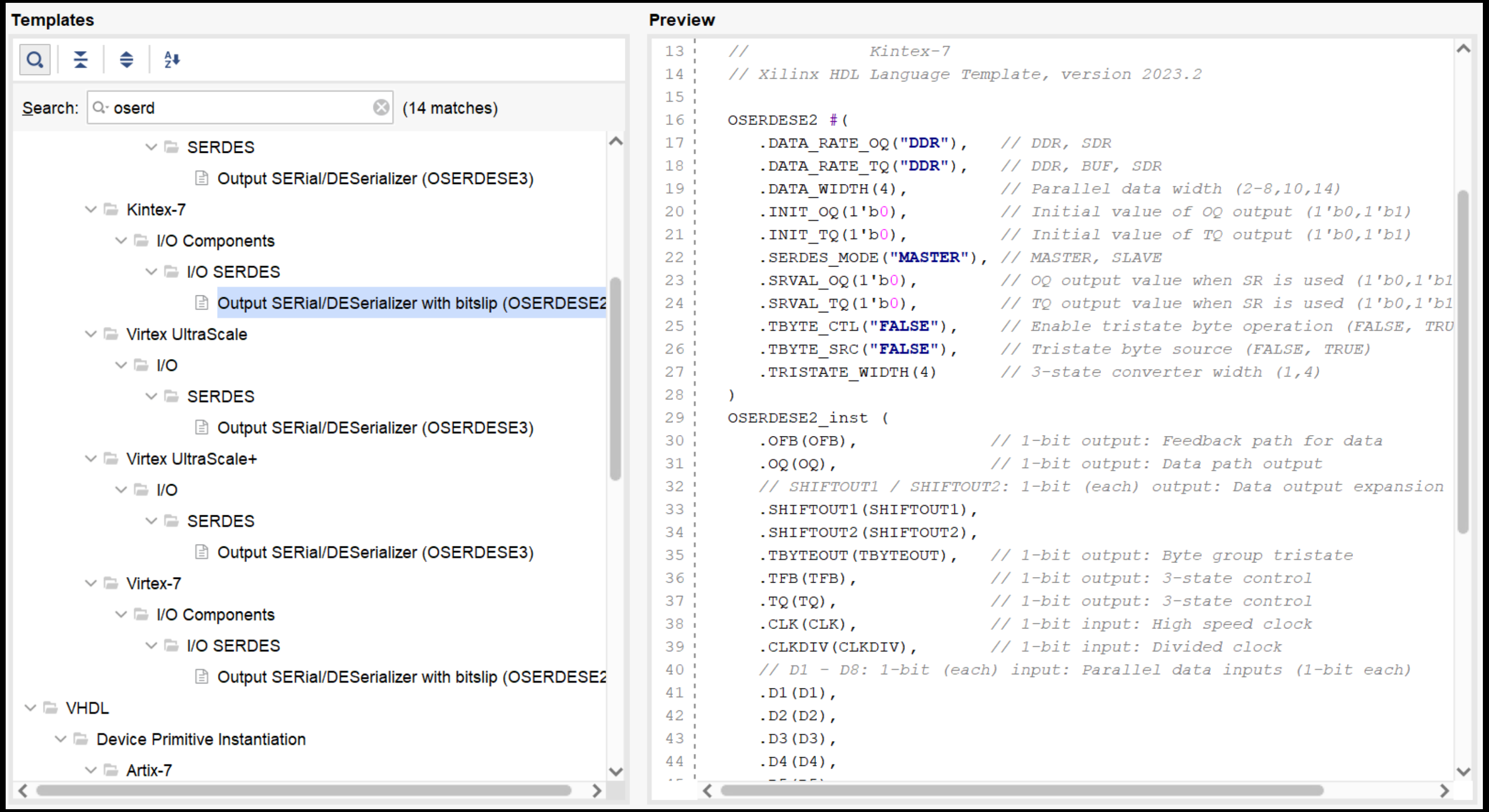
Templates
Search: oserd (14 matches)
SERDES
Output SERial/DESerializer (OSERDESE3)
Kintex-7
I/O Components
I/O SERDES
Output SERial/DESerializer with bitslip (OSERDESE2
Virtex UltraScale
I/O
SERDES
Output SERial/DESerializer (OSERDESE3)
Virtex UltraScale+
I/O
SERDES
Output SERial/DESerializer (OSERDESE3)
Virtex-7
I/O Components
I/O SERDES
Output SERial/DESerializer with bitslip (OSERDESE2
VHDL
Device Primitive Instantiation
Artix-7
Preview

```
//              Kintex-7
// Xilinx HDL Language Template, version 2023.2

OSERDESE2 #(
   .DATA_RATE_OQ("DDR"),   // DDR, SDR
   .DATA_RATE_TQ("DDR"),   // DDR, BUF, SDR
   .DATA_WIDTH(4),         // Parallel data width (2-8,10,14)
   .INIT_OQ(1'b0),         // Initial value of OQ output (1'b0,1'b1)
   .INIT_TQ(1'b0),         // Initial value of TQ output (1'b0,1'b1)
   .SERDES_MODE("MASTER"), // MASTER, SLAVE
   .SRVAL_OQ(1'b0),        // OQ output value when SR is used (1'b0,1'b1
   .SRVAL_TQ(1'b0),        // TQ output value when SR is used (1'b0,1'b1
   .TBYTE_CTL("FALSE"),    // Enable tristate byte operation (FALSE, TRU
   .TBYTE_SRC("FALSE"),    // Tristate byte source (FALSE, TRUE)
   .TRISTATE_WIDTH(4)      // 3-state converter width (1,4)
)
OSERDESE2_inst (
   .OFB(OFB),             // 1-bit output: Feedback path for data
   .OQ(OQ),               // 1-bit output: Data path output
   // SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion
   .SHIFTOUT1(SHIFTOUT1),
   .SHIFTOUT2(SHIFTOUT2),
   .TBYTEOUT(TBYTEOUT),   // 1-bit output: Byte group tristate
   .TFB(TFB),             // 1-bit output: 3-state control
   .TQ(TQ),               // 1-bit output: 3-state control
   .CLK(CLK),             // 1-bit input: High speed clock
   .CLKDIV(CLKDIV),       // 1-bit input: Divided clock
   // D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
   .D1(D1),
   .D2(D2),
   .D3(D3),
   .D4(D4),
```

Parallel_to_serial

OSERDESE2_Slave inst

```
OSERDESE2 #(
    .DATA_RATE_OQ    ("DDR"),
    .DATA_RATE_TQ    ("SDR"),
    .DATA_WIDTH      (10),
    .INIT_OQ         (1'b0),
    .INIT_TQ         (1'b0),
    .SERDES_MODE     ("SLAVE"),
    .SRVAL_OQ        (1'b0),
    .SRVAL_TQ        (1'b0),
    .TBYTE_CTL       ("FALSE"),
    .TBYTE_SRC       ("FALSE"),
    .TRISTATE_WIDTH (1)
  )
  OSERDESE2_inst_slave (
    .OFB         (),
    .OQ          (),
    .SHIFTOUT1   (shift_in1),
    .SHIFTOUT2   (shift_in2),
    .TBYTEOUT    (),
    .TFB         (),
    .TQ          (),
    .CLK         (clk5x),
    .CLKDIV      (clk1x),
```



Templates

Search: oserd (14 matches)

- SERDES
  - Output SERial/DESerializer (OSERDESE3)
- Kintex-7
  - I/O Components
    - I/O SERDES
      - Output SERial/DESerializer with bitslip (OSERDESE2
- Virtex UltraScale
  - I/O
    - SERDES
      - Output SERial/DESerializer (OSERDESE3)
- Virtex UltraScale+
  - I/O
    - SERDES
      - Output SERial/DESerializer (OSERDESE3)
- Virtex-7
  - I/O Components
    - I/O SERDES
      - Output SERial/DESerializer with bitslip (OSERDESE2
- VHDL
  - Device Primitive Instantiation
    - Artix-7

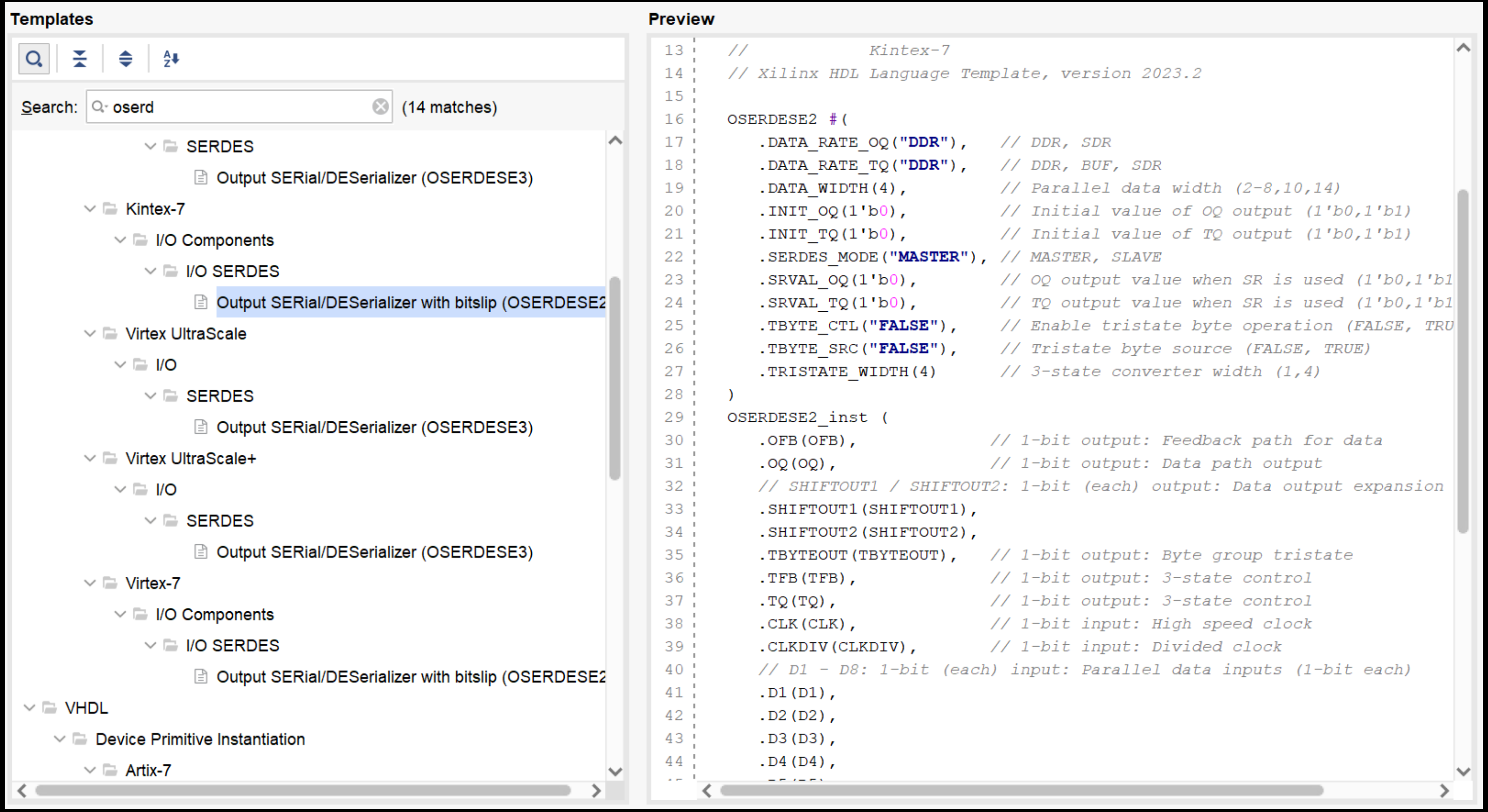
Preview

```
//              Kintex-7
// Xilinx HDL Language Template, version 2023.2

OSERDESE2 #(
   .DATA_RATE_OQ("DDR"),   // DDR, SDR
   .DATA_RATE_TQ("DDR"),   // DDR, BUF, SDR
   .DATA_WIDTH(4),         // Parallel data width (2-8,10,14)
   .INIT_OQ(1'b0),         // Initial value of OQ output (1'b0,1'b1)
   .INIT_TQ(1'b0),         // Initial value of TQ output (1'b0,1'b1)
   .SERDES_MODE("MASTER"), // MASTER, SLAVE
   .SRVAL_OQ(1'b0),        // OQ output value when SR is used (1'b0,1'b1
   .SRVAL_TQ(1'b0),        // TQ output value when SR is used (1'b0,1'b1
   .TBYTE_CTL("FALSE"),    // Enable tristate byte operation (FALSE, TRU
   .TBYTE_SRC("FALSE"),    // Tristate byte source (FALSE, TRUE)
   .TRISTATE_WIDTH(4)      // 3-state converter width (1,4)
)
OSERDESE2_inst (
   .OFB(OFB),             // 1-bit output: Feedback path for data
   .OQ(OQ),               // 1-bit output: Data path output
   // SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion
   .SHIFTOUT1(SHIFTOUT1),
   .SHIFTOUT2(SHIFTOUT2),
   .TBYTEOUT(TBYTEOUT),   // 1-bit output: Byte group tristate
   .TFB(TFB),             // 1-bit output: 3-state control
   .TQ(TQ),               // 1-bit output: 3-state control
   .CLK(CLK),             // 1-bit input: High speed clock
   .CLKDIV(CLKDIV),       // 1-bit input: Divided clock
   // D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
   .D1(D1),
   .D2(D2),
   .D3(D3),
   .D4(D4),
```

Parallel_to_serial

OSERDESE2_Slave inst

```
    .D1          (),
    .D2          (),
    .D3          (din[8]),
    .D4          (din[9]),
    .D5          (),
    .D6          (),
    .D7          (),
    .D8          (),
    .OCE         (1'b1),
    .RST         (rst),
    .SHIFTIN1    (),
    .SHIFTIN2    (),

    .T1          (1'b0),
    .T2          (1'b0),
    .T3          (1'b0),
    .T4          (1'b0),
    .TBYTEIN     (1'b0),
    .TCE         (1'b0)
  );

endmodule
```



Templates

Search: oserd (14 matches)

- SERDES
  - Output SERial/DESerializer (OSERDESE3)
- Kintex-7
  - I/O Components
    - I/O SERDES
      - Output SERial/DESerializer with bitslip (OSERDESE2
- Virtex UltraScale
  - I/O
    - SERDES
      - Output SERial/DESerializer (OSERDESE3)
- Virtex UltraScale+
  - I/O
    - SERDES
      - Output SERial/DESerializer (OSERDESE3)
- Virtex-7
  - I/O Components
    - I/O SERDES
      - Output SERial/DESerializer with bitslip (OSERDESE2
- VHDL
  - Device Primitive Instantiation
    - Artix-7

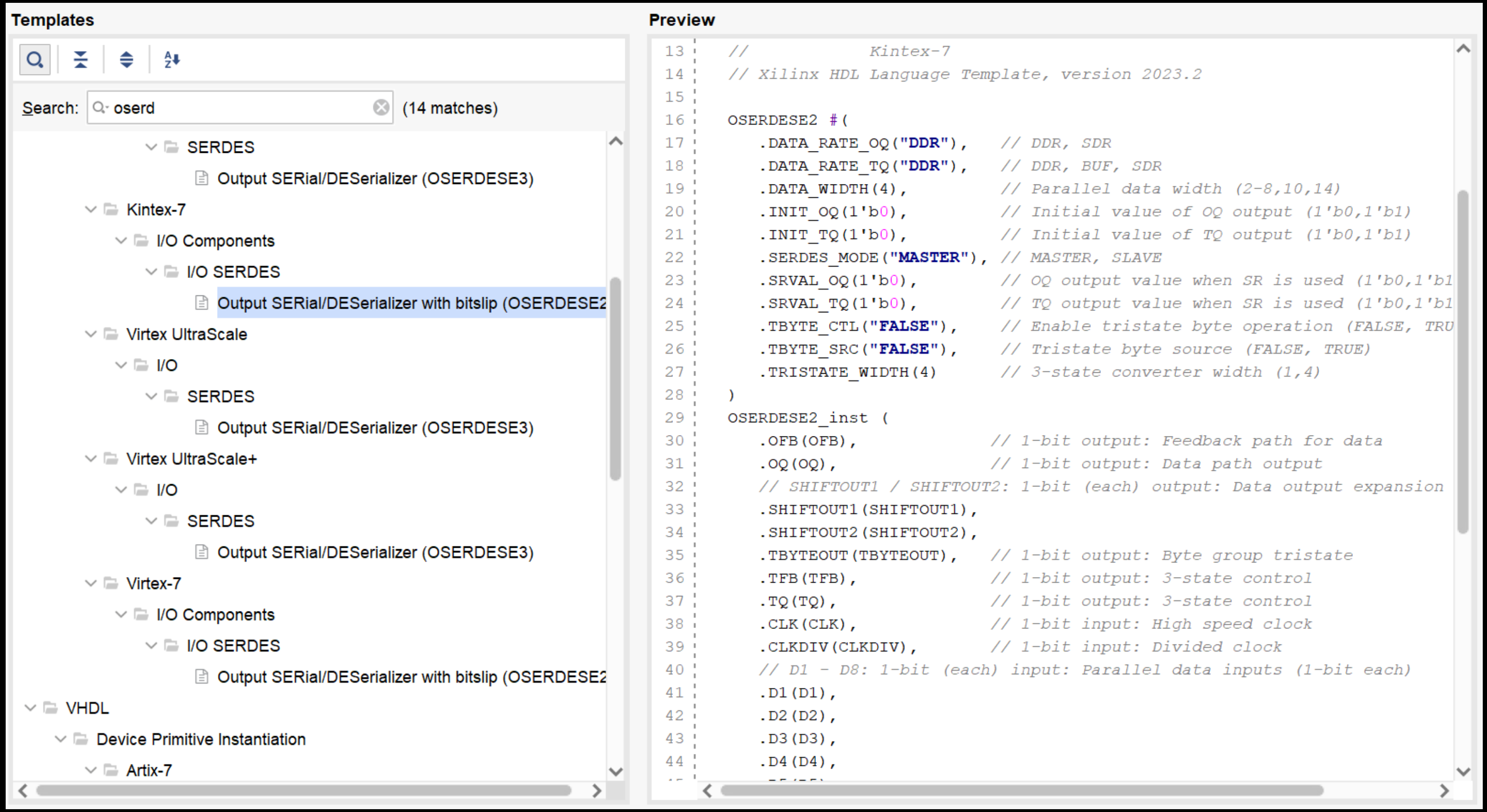
Preview

```
//              Kintex-7
// Xilinx HDL Language Template, version 2023.2

OSERDESE2 #(
   .DATA_RATE_OQ("DDR"),   // DDR, SDR
   .DATA_RATE_TQ("DDR"),   // DDR, BUF, SDR
   .DATA_WIDTH(4),         // Parallel data width (2-8,10,14)
   .INIT_OQ(1'b0),         // Initial value of OQ output (1'b0,1'b1)
   .INIT_TQ(1'b0),         // Initial value of TQ output (1'b0,1'b1)
   .SERDES_MODE("MASTER"), // MASTER, SLAVE
   .SRVAL_OQ(1'b0),        // OQ output value when SR is used (1'b0,1'b1
   .SRVAL_TQ(1'b0),        // TQ output value when SR is used (1'b0,1'b1
   .TBYTE_CTL("FALSE"),    // Enable tristate byte operation (FALSE, TRU
   .TBYTE_SRC("FALSE"),    // Tristate byte source (FALSE, TRUE)
   .TRISTATE_WIDTH(4)      // 3-state converter width (1,4)
)
OSERDESE2_inst (
   .OFB(OFB),             // 1-bit output: Feedback path for data
   .OQ(OQ),               // 1-bit output: Data path output
   // SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion
   .SHIFTOUT1(SHIFTOUT1),
   .SHIFTOUT2(SHIFTOUT2),
   .TBYTEOUT(TBYTEOUT),   // 1-bit output: Byte group tristate
   .TFB(TFB),             // 1-bit output: 3-state control
   .TQ(TQ),               // 1-bit output: 3-state control
   .CLK(CLK),             // 1-bit input: High speed clock
   .CLKDIV(CLKDIV),       // 1-bit input: Divided clock
   // D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
   .D1(D1),
   .D2(D2),
   .D3(D3),
   .D4(D4),
```

Parallel_to_serial

OSERDESE2_Slave inst

```
    .D1           (),
    .D2           (),
    .D3           (din[8]),
    .D4           (din[9]),
    .D5           (),
    .D6           (),
    .D7           (),
    .D8           (),
    .OCE          (1'b1),
    .RST          (rst),
    .SHIFTIN1     (),
    .SHIFTIN2     (),

    .T1           (1'b0),
    .T2           (1'b0),
    .T3           (1'b0),
    .T4           (1'b0),
    .TBYTEIN      (1'b0),
    .TCE          (1'b0)
  );

endmodule
```



Templates

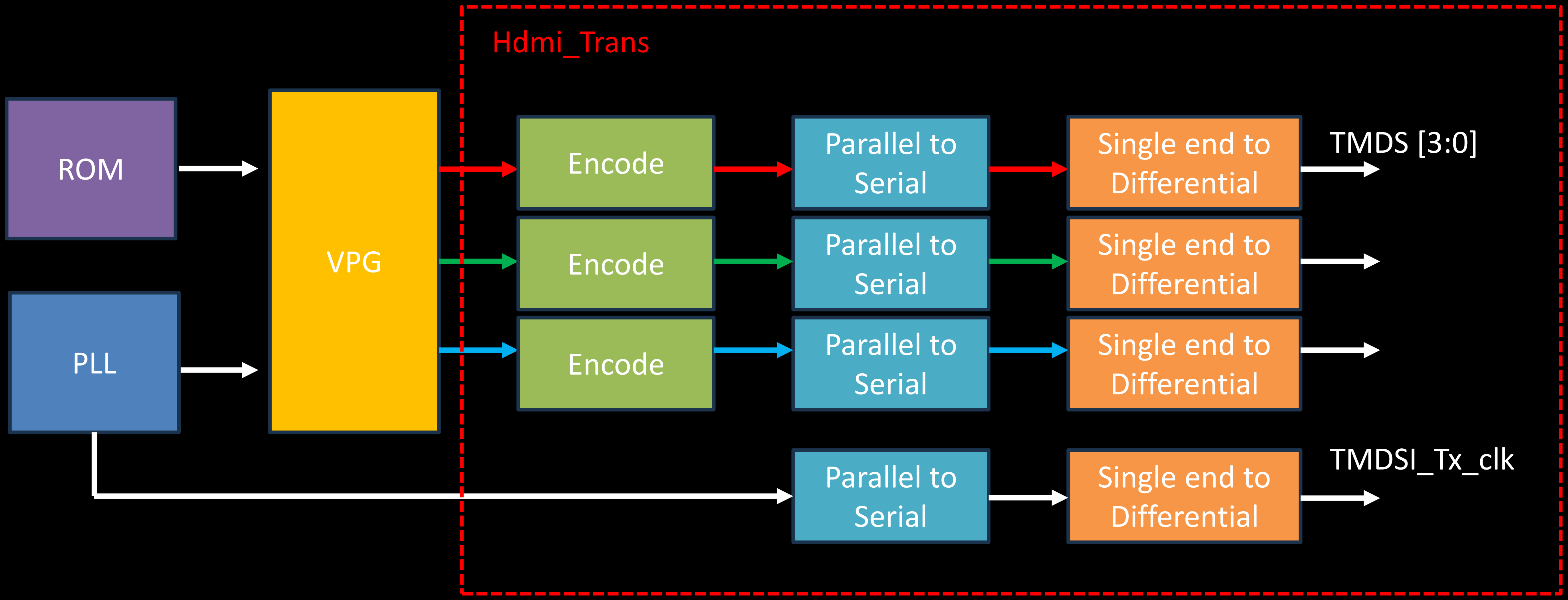
Search: oserd (14 matches)

- SERDES
  - Output SERial/DESerializer (OSERDESE3)
- Kintex-7
  - I/O Components
    - I/O SERDES
      - Output SERial/DESerializer with bitslip (OSERDESE2
- Virtex UltraScale
  - I/O
    - SERDES
      - Output SERial/DESerializer (OSERDESE3)
- Virtex UltraScale+
  - I/O
    - SERDES
      - Output SERial/DESerializer (OSERDESE3)
- Virtex-7
  - I/O Components
    - I/O SERDES
      - Output SERial/DESerializer with bitslip (OSERDESE2
- VHDL
  - Device Primitive Instantiation
    - Artix-7

Preview

```
//              Kintex-7
// Xilinx HDL Language Template, version 2023.2

OSERDESE2 #(
    .DATA_RATE_OQ("DDR"),   // DDR, SDR
    .DATA_RATE_TQ("DDR"),   // DDR, BUF, SDR
    .DATA_WIDTH(4),         // Parallel data width (2-8,10,14)
    .INIT_OQ(1'b0),         // Initial value of OQ output (1'b0,1'b1)
    .INIT_TQ(1'b0),         // Initial value of TQ output (1'b0,1'b1)
    .SERDES_MODE("MASTER"), // MASTER, SLAVE
    .SRVAL_OQ(1'b0),        // OQ output value when SR is used (1'b0,1'b1
    .SRVAL_TQ(1'b0),        // TQ output value when SR is used (1'b0,1'b1
    .TBYTE_CTL("FALSE"),    // Enable tristate byte operation (FALSE, TRU
    .TBYTE_SRC("FALSE"),    // Tristate byte source (FALSE, TRUE)
    .TRISTATE_WIDTH(4)      // 3-state converter width (1,4)
)
OSERDESE2_inst (
    .OFB(OFB),             // 1-bit output: Feedback path for data
    .OQ(OQ),               // 1-bit output: Data path output
    // SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion
    .SHIFTOUT1(SHIFTOUT1),
    .SHIFTOUT2(SHIFTOUT2),
    .TBYTEOUT(TBYTEOUT),   // 1-bit output: Byte group tristate
    .TFB(TFB),             // 1-bit output: 3-state control
    .TQ(TQ),               // 1-bit output: 3-state control
    .CLK(CLK),             // 1-bit input: High speed clock
    .CLKDIV(CLKDIV),       // 1-bit input: Divided clock
    // D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
```

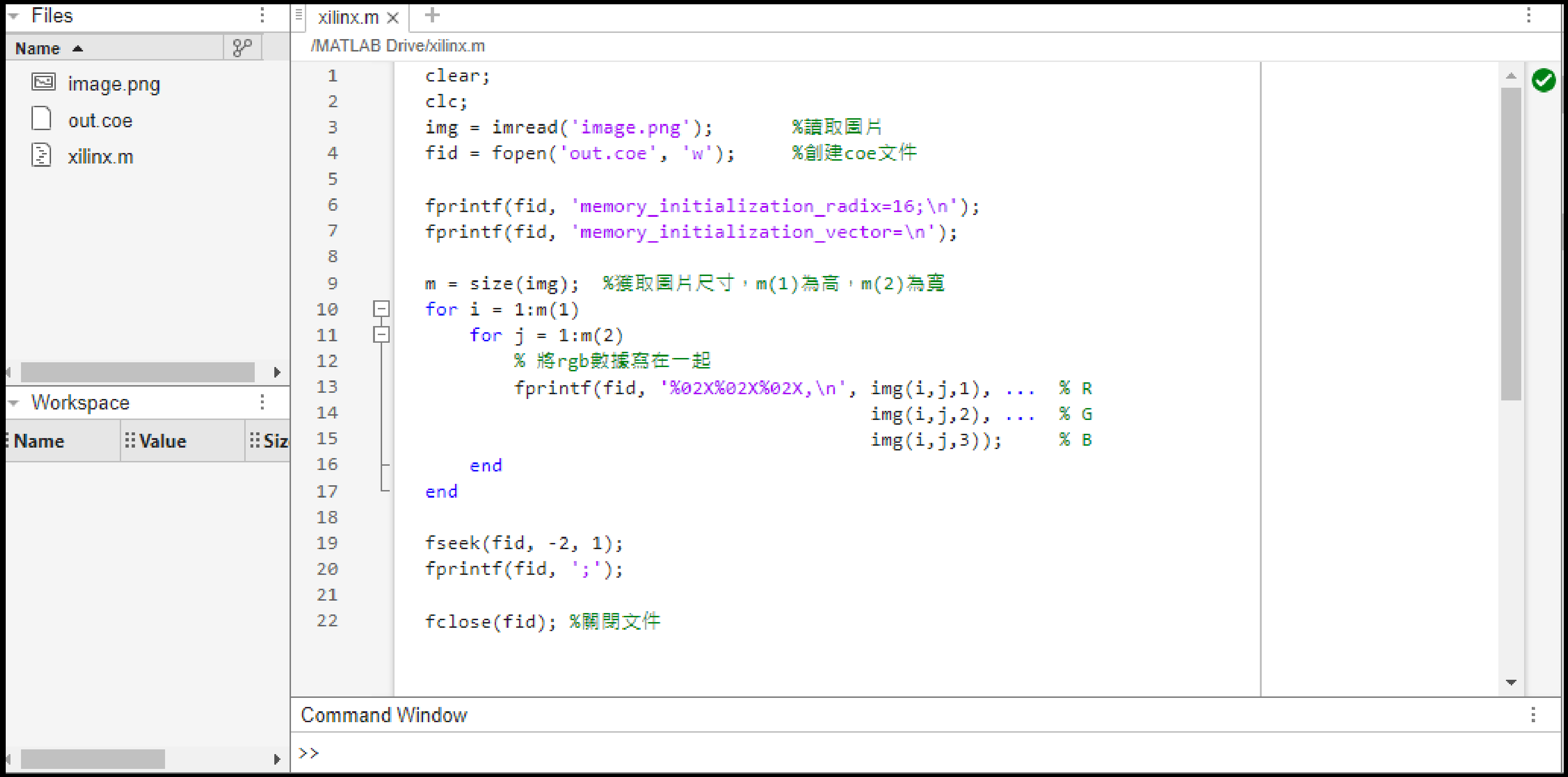
HDMI_Img


Hdmi_Trans
ROM
PLL
VPG
Encode
Encode
Encode
Parallel to Serial
Parallel to Serial
Parallel to Serial
Parallel to Serial
Single end to Differential
Single end to Differential
Single end to Differential
Single end to Differential
TMDS [3:0]
TMDSI_Tx_clk

HDMI_img

透過Matlab將圖片轉成rgb編碼，並儲存成rom可以存取的coe檔案

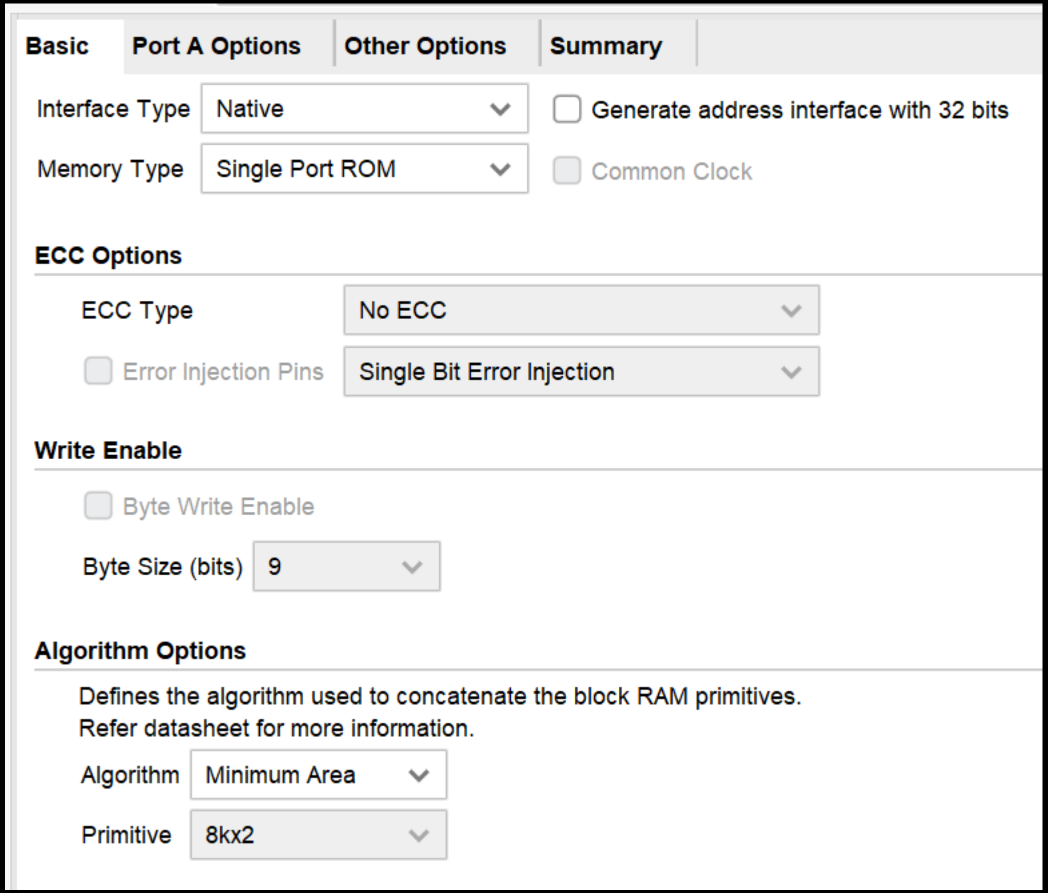
```
clear;
clc;
img = imread('image.png'); %讀取圖片
fid = fopen('out.coe'  'w'); %創建coe文件

fprintf(fid, 'memory_initialization_radix=16;\n'
fprintf(fid, 'memory_initialization_vector=\n'

m = size(img); %獲取圖片尺寸，m(1)為高，m(2)為寬
for i = 1:m(1)
for j = 1:m(2)
% 將rgb數據寫在一起
fprintf(fid, '%02X%02X%02X,\n', img(i,j,1), ... % R
img(i,j,2), ... % G
img(i,j,3)); % B
end
end

fseek(fid, -2, 1);
fprintf(fid  ';');

fclose(fid); %關閉文件
```



```
clear;
clc;
img = imread('image.png');      %讀取圖片
fid = fopen('out.coe', 'w');    %創建coe文件

fprintf(fid, 'memory_initialization_radix=16;\n');
fprintf(fid, 'memory_initialization_vector=\n');

m = size(img);  %獲取圖片尺寸，m(1)為高，m(2)為寬
for i = 1:m(1)
    for j = 1:m(2)
        % 將rgb數據寫在一起
        fprintf(fid, '%02X%02X%02X,\n', img(i,j,1), ...  % R
                                        img(i,j,2), ...  % G
                                        img(i,j,3));     % B

    end
end

fseek(fid, -2, 1);
fprintf(fid, ';');

fclose(fid); %關閉文件
```

HDMI_img

Rom IP Setting



Basic | Port A Options | Other Options | Summary

Interface Type: Native

[ ] Generate address interface with 32 bits

Memory Type: Single Port ROM

[ ] Common Clock

ECC Options

ECC Type: No ECC

[ ] Error Injection Pins: Single Bit Error Injection

Write Enable

[ ] Byte Write Enable

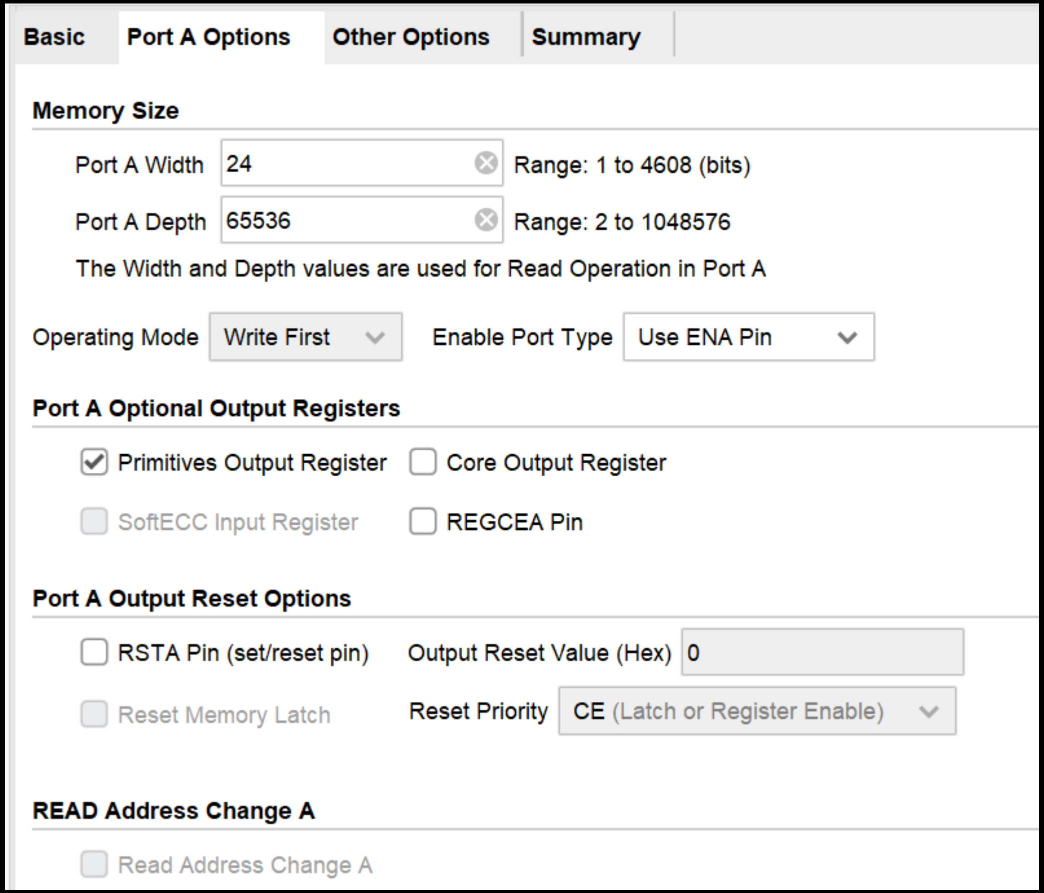
Byte Size (bits): 9

Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.

Algorithm: Minimum Area

Primitive: 8kx2



Basic | Port A Options | Other Options | Summary

Memory Size

Port A Width: 24 — Range: 1 to 4608 (bits)

Port A Depth: 65536 — Range: 2 to 1048576

The Width and Depth values are used for Read Operation in Port A

Operating Mode: Write First

Enable Port Type: Use ENA Pin

Port A Optional Output Registers

[✓] Primitives Output Register

[ ] Core Output Register

[ ] SoftECC Input Register

[ ] REGCEA Pin

Port A Output Reset Options

[ ] RSTA Pin (set/reset pin)

Output Reset Value (Hex): 0

[ ] Reset Memory Latch

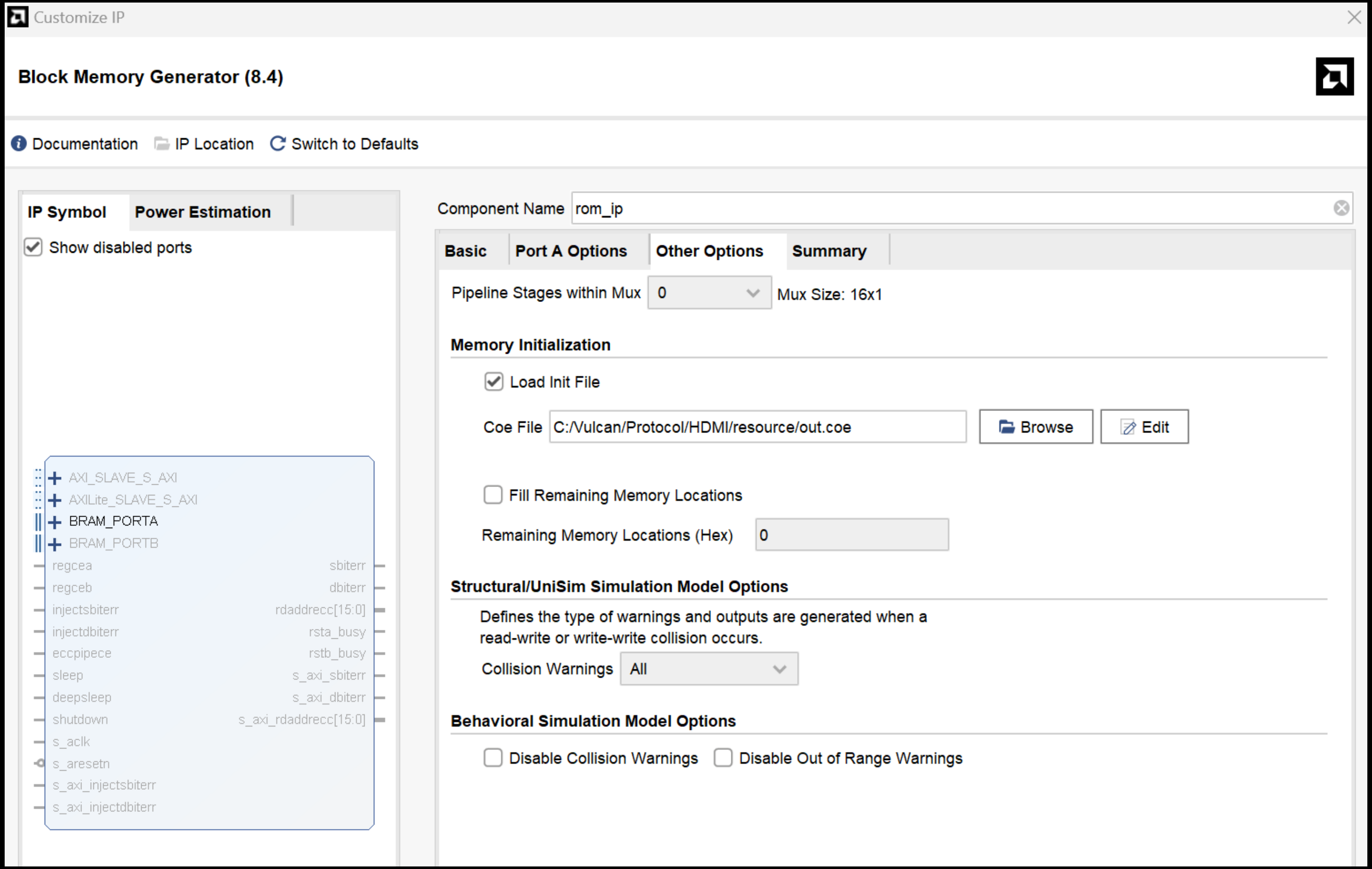
Reset Priority: CE (Latch or Register Enable)

READ Address Change A

[ ] Read Address Change A

HDMI_img

Rom IP Setting



Customize IP

Block Memory Generator (8.4)

[✓] Show disabled ports

Component Name: rom_ip

Pipeline Stages within Mux: 0 — Mux Size: 16x1

Memory Initialization

[✓] Load Init File

Coe File: C:/Vulcan/Protocol/HDMI/resource/out.coe

[ ] Fill Remaining Memory Locations

Remaining Memory Locations (Hex): 0

Structural/UniSim Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.

Collision Warnings: All

Behavioral Simulation Model Options

[ ] Disable Collision Warnings [ ] Disable Out of Range Warnings

HDMI_trans_top

Rd_data & rd_req port

```
vga_shift inst_vga_shift
    (
     .rst(rst),
     .vpg_pclk(clk1x),
     .rd_data(rd_data),
     .rd_req(rd_req),
     .vpg_de(vpg_de),
     .vpg_hs(vpg_hs),
     .vpg_vs(vpg_vs),
     .rgb_r(rgb_r),
     .rgb_g(rgb_g),
     .rgb_b(rgb_b)
    );

rd_image inst_rd_image
    (
     .clk(clk1x),
     .rst(rst),
     .rd_req(rd_req),
     .rd_data(rd_data)
    );
```

Vga_shift

Port degine

```
`timescale 1ns / 1ps
module vga_shift(
input  wire         rst       ,
input  wire         vpg_pclk  ,
input  wire [23:0]  rd_data   ,
output wire         rd_req    ,
output reg          vpg_de    ,
output reg          vpg_hs    ,
output reg          vpg_vs    ,
output wire [7:0]   rgb_r     ,
output wire [7:0]   rgb_g     ,
output wire [7:0]   rgb_b
    );
```

Vga_shift

Rd_data & rd_req

```
reg  [12:0] cnt_h    ;
reg  [12:0] cnt_v    ;
reg  [11:0] x        ;
reg         flag_x   ;
reg  [11:0] y        ;
reg         flag_y   ;
reg  [23:0] rgb      ;
reg         rd_req_r;
```

```
assign {rgb_r , rgb_g , rgb_b } = rgb ;
assign rd_req = rd_req_r;
```

Vga_shift

Rd_data & rd_req

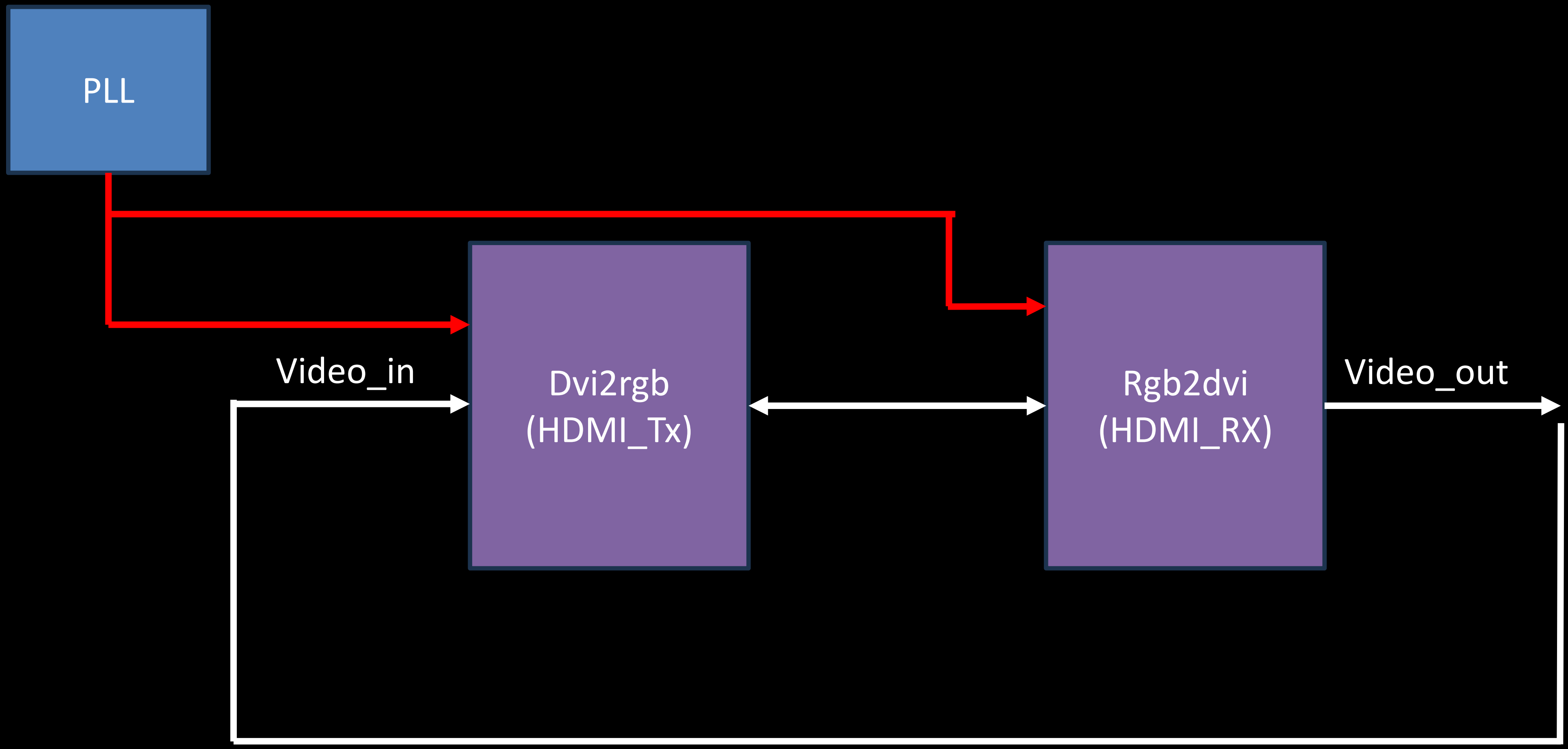
```
always@(posedge vpg_pclk)begin
    if(rst)
        rd_req_r <= 1'b0;
    else if(cnt_h >= H_START+x-2 && cnt_h < H_START+SQUARE_X+x-2 && cnt_v >= V_START+y && cnt_v <=V_START+SQUARE_Y+y)
        rd_req_r <= 1'b1;
    else
        rd_req_r <= 1'b0;
end
always@(posedge vpg_pclk)begin
    if(rst)
        rgb <= 'd0;
    else if(cnt_h >= H_START+x && cnt_h < H_START+SQUARE_X+x && cnt_v >= V_START+y && cnt_v <= V_START+SQUARE_Y+y)
        rgb <= {rd_data[7:0],rd_data[15:8],rd_data[23:16]};
    else if(cnt_h >= H_START && cnt_h < H_END && cnt_v >= V_START && cnt_v <= V_END && cnt_h[4:0] >= 'd20)
        rgb <= 24'h00FF00;
    else if(cnt_h>=H_START && cnt_h<H_END && cnt_v>=V_START && cnt_v<=V_END && (cnt_h[4:0]>='d10 && cnt_h[2:0]<'d20))
        rgb <= 24'h0000FF;
    else if(cnt_h >= H_START && cnt_h < H_END && cnt_v >= V_START && cnt_v <= V_END && cnt_h[4:0] < 'd10)
        rgb <= 24'hFF0000;
    else
        rgb <= 'd0;
end
```

Rd_img

Read rom data

```
`timescale 1ns / 1ps
module rd_image(
    input  wire          clk,
    input  wire          rst,
    input  wire          rd_req,
    output wire [23:0]   rd_data
    );
parameter   STOP_ADDR = 256*256-1;
reg  [15:0]  rd_addr;
wire [23:0]  dout;

assign rd_data = dout;
always@(posedge clk)begin
    if(rst)
        rd_addr <= 'd0;
    else if(rd_req == 1'b1)
        rd_addr <= (rd_addr == STOP_ADDR) ? 'd0 : rd_addr + 1'b1;
end

rom_ip inst_rom(
    .clka(clk),
    .addra(rd_addr),
    .douta(dout));
endmodule
```

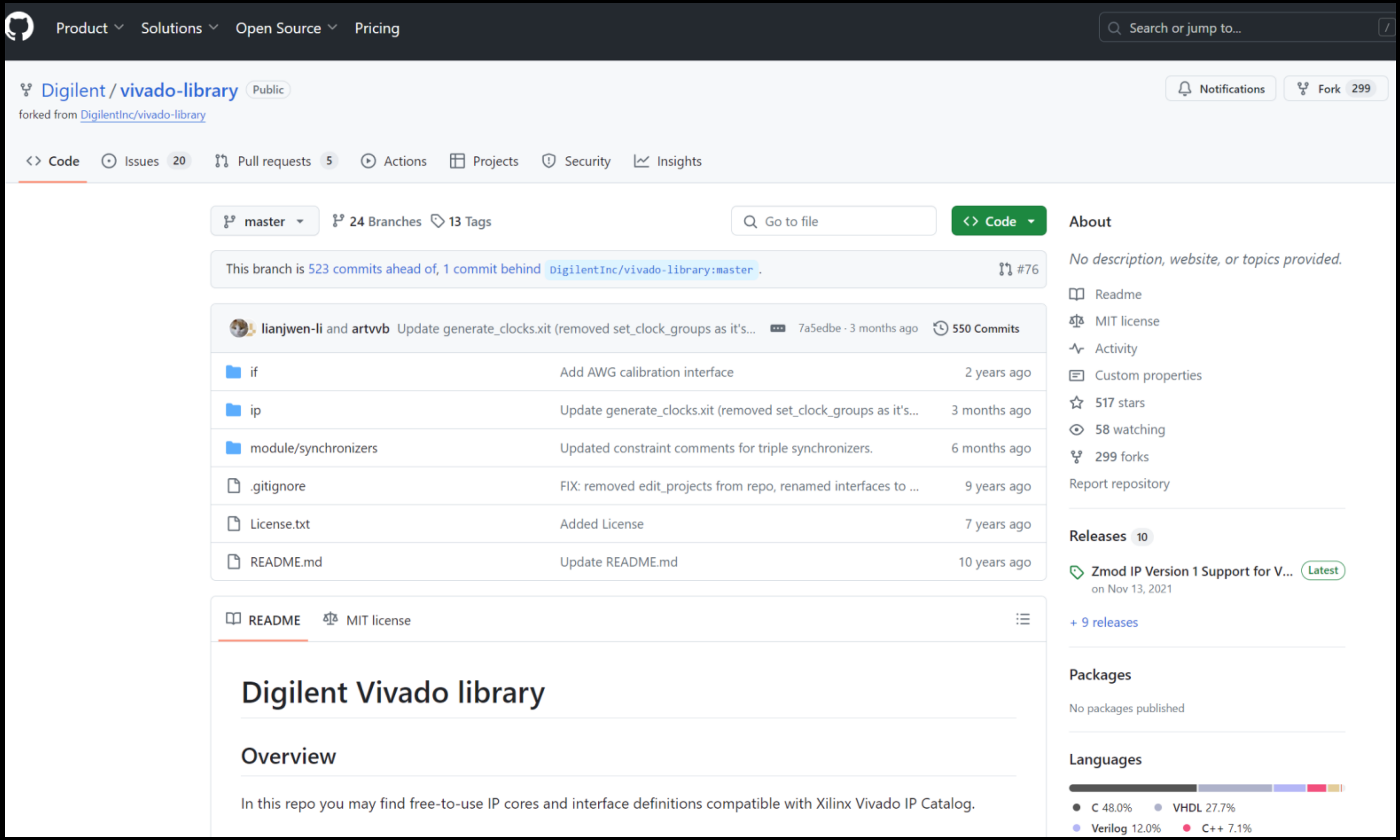
HDMI_loop



HDMI_loop

這次改使用現成IP的方式來完成HDMI Loop

https://github.com/Digilent/vivado-library



Digilent / vivado-library Public

forked from DigilentInc/vivado-library

This branch is 523 commits ahead of, 1 commit behind DigilentInc/vivado-library:master. #76

lianjwen-li and artvvb Update generate_clocks.xit (removed set_clock_groups as it's... 7a5edbe · 3 months ago 550 Commits

if	Add AWG calibration interface	2 years ago
ip	Update generate_clocks.xit (removed set_clock_groups as it's...	3 months ago
module/synchronizers	Updated constraint comments for triple synchronizers.	6 months ago
.gitignore	FIX: removed edit_projects from repo, renamed interfaces to ...	9 years ago
License.txt	Added License	7 years ago
README.md	Update README.md	10 years ago

README MIT license

Digilent Vivado library

Overview

In this repo you may find free-to-use IP cores and interface definitions compatible with Xilinx Vivado IP Catalog.

About

No description, website, or topics provided.

Readme
MIT license
Activity
Custom properties
517 stars
58 watching
299 forks
Report repository

Releases 10

Zmod IP Version 1 Support for V... Latest
on Nov 13, 2021

+ 9 releases

Packages

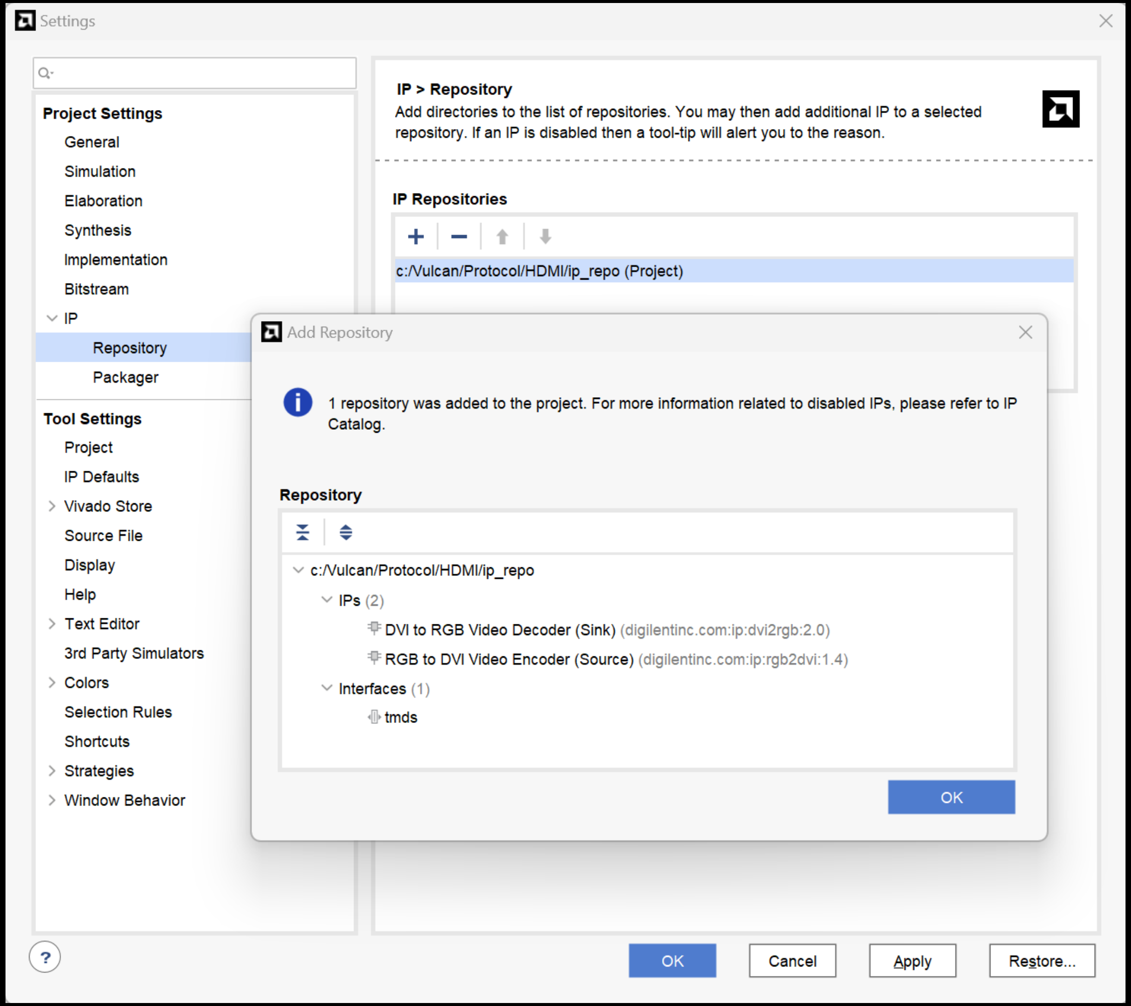
No packages published

Languages

● C 48.0% ● VHDL 27.7%
● Verilog 12.0% ● C++ 7.1%

HDMI_loop

使用tmds(interface)及dvi2rgb & rgb2dvi(ip)



Settings

Project Settings
- General
- Simulation
- Elaboration
- Synthesis
- Implementation
- Bitstream
- IP
  - Repository
  - Packager

Tool Settings
- Project
- IP Defaults
- Vivado Store
- Source File
- Display
- Help
- Text Editor
- 3rd Party Simulators
- Colors
- Selection Rules
- Shortcuts
- Strategies
- Window Behavior

IP > Repository
Add directories to the list of repositories. You may then add additional IP to a selected repository. If an IP is disabled then a tool-tip will alert you to the reason.

IP Repositories

c:/Vulcan/Protocol/HDMI/ip_repo (Project)

Add Repository

1 repository was added to the project. For more information related to disabled IPs, please refer to IP Catalog.

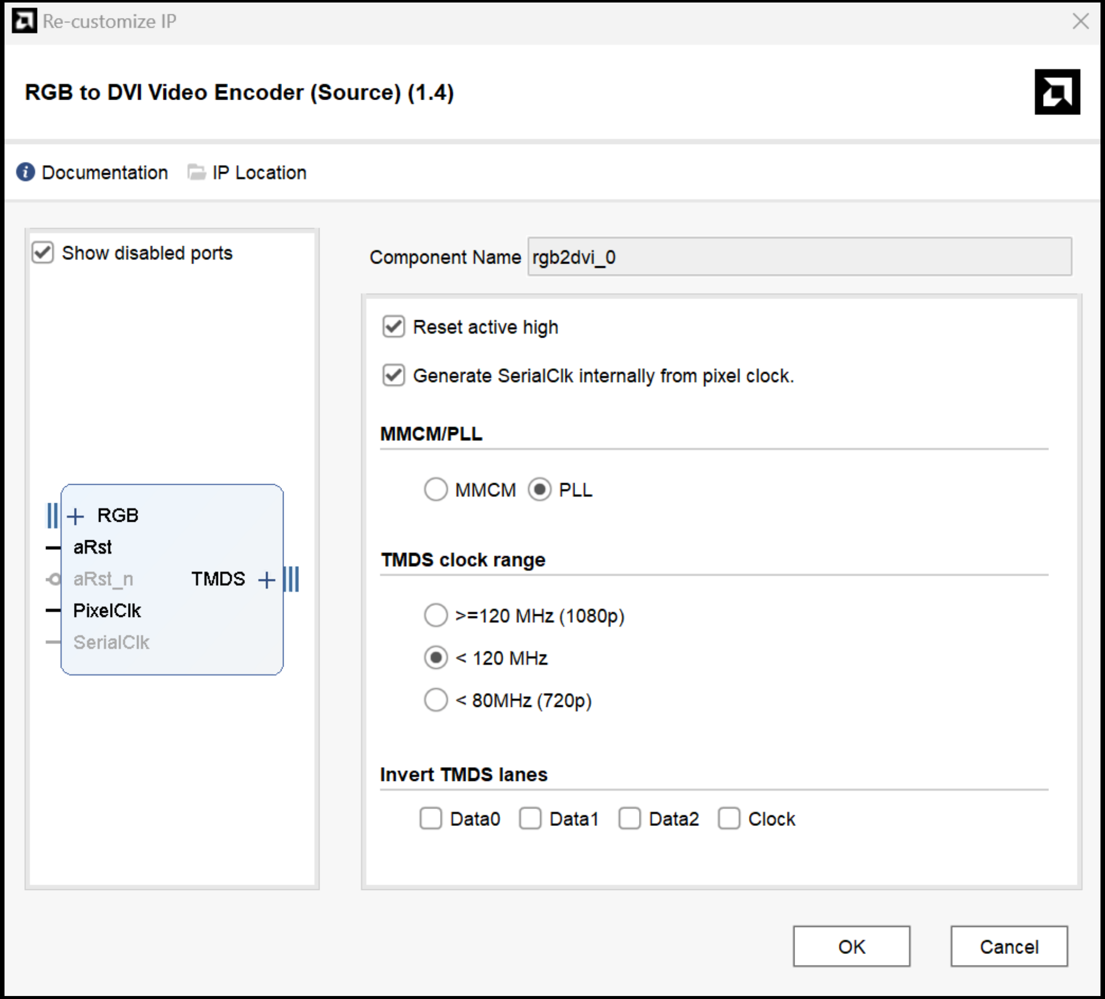
Repository

```
c:/Vulcan/Protocol/HDMI/ip_repo
  IPs (2)
    DVI to RGB Video Decoder (Sink) (digilentinc.com:ip:dvi2rgb:2.0)
    RGB to DVI Video Encoder (Source) (digilentinc.com:ip:rgb2dvi:1.4)
  Interfaces (1)
    tmds
```

AMD

HDMI_loop

IP Setting


Re-customize IP

RGB to DVI Video Encoder (Source) (1.4)

Documentation IP Location

☑ Show disabled ports

Component Name: rgb2dvi_0

☑ Reset active high

☑ Generate SerialClk internally from pixel clock.

MMCM/PLL

MMCM [✓ PLL]

TMDS clock range

>=120 MHz (1080p)

[✓ < 120 MHz]

< 80MHz (720p)

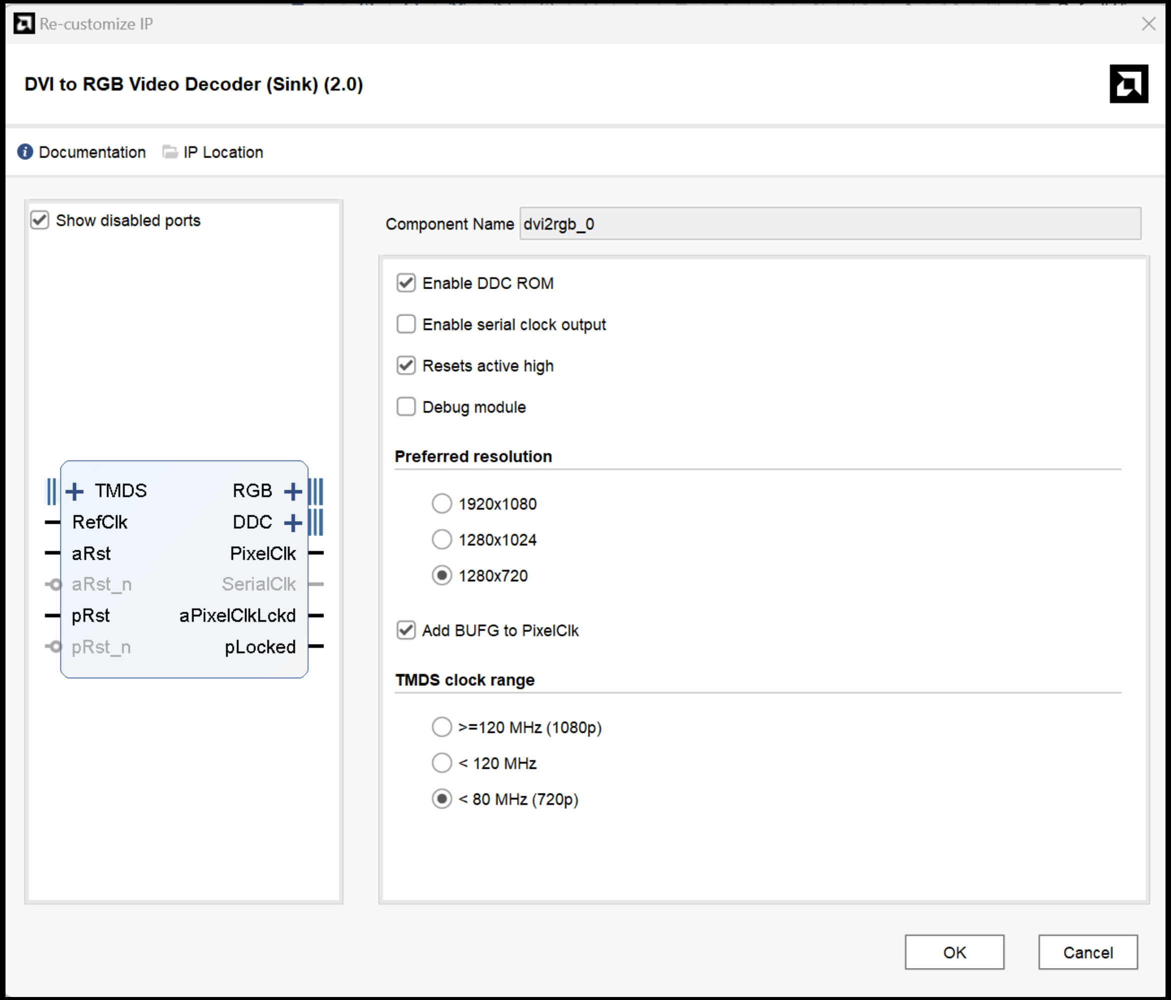
Invert TMDS lanes

☐ Data0 ☐ Data1 ☐ Data2 ☐ Clock

OK Cancel

HDMI_loop

IP Setting


Re-customize IP

DVI to RGB Video Decoder (Sink) (2.0)

Documentation IP Location

[✓] Show disabled ports

Component Name dvi2rgb_0

[✓] Enable DDC ROM

[ ] Enable serial clock output

[✓] Resets active high

[ ] Debug module

Preferred resolution

- ○ 1920x1080
- ○ 1280x1024
- ● 1280x720

[✓] Add BUFG to PixelClk

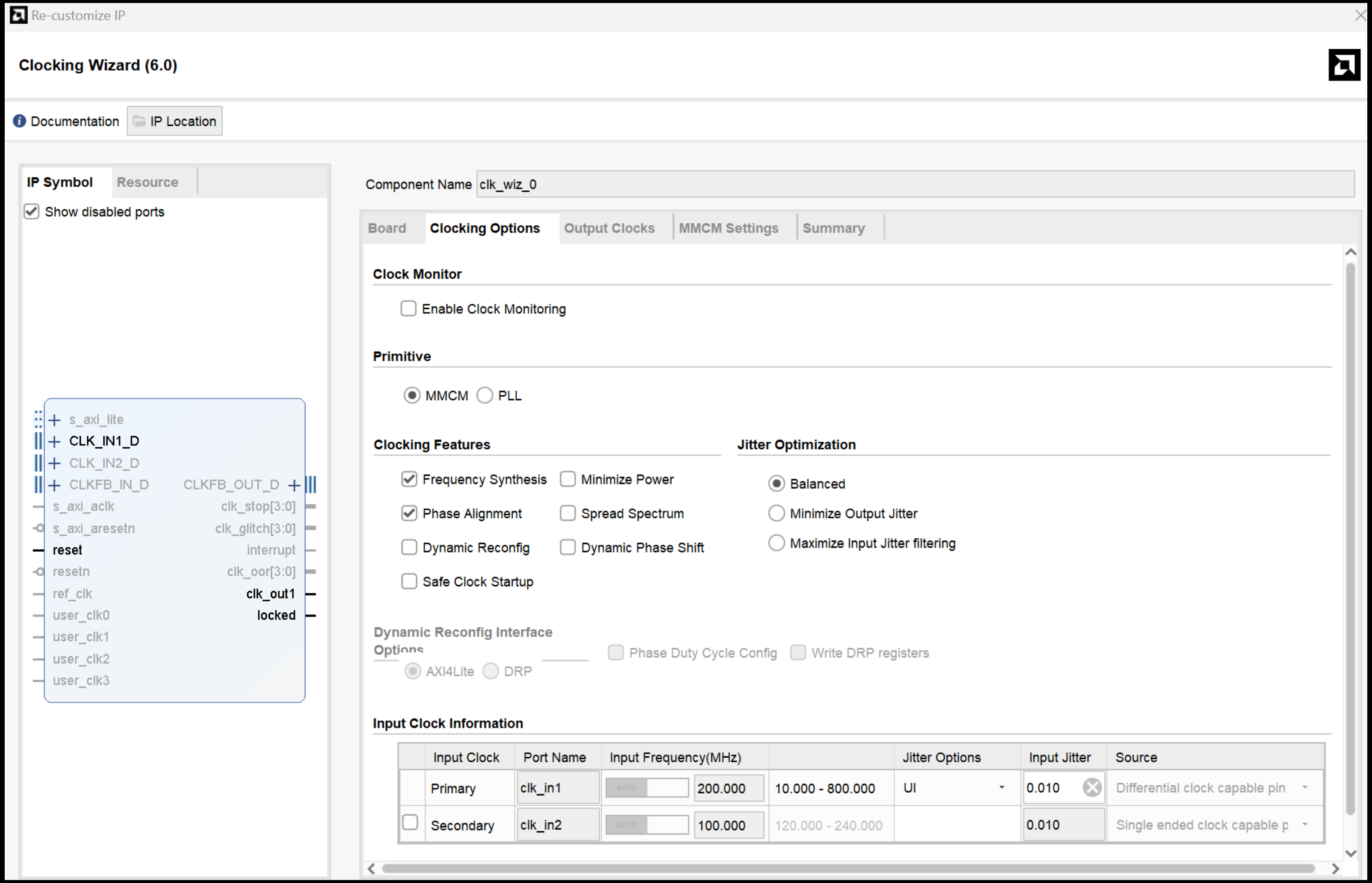
TMDS clock range

- ○ >=120 MHz (1080p)
- ○ < 120 MHz
- ● < 80 MHz (720p)

OK Cancel

HDMI_loop

IP Setting



Clocking Wizard (6.0)

[✓] Show disabled ports

Component Name: clk_wiz_0

Clock Monitor

[ ] Enable Clock Monitoring

Primitive

[✓ MMCM] PLL

Clocking Features

[✓] Frequency Synthesis [ ] Minimize Power
[✓] Phase Alignment [ ] Spread Spectrum
[ ] Dynamic Reconfig [ ] Dynamic Phase Shift
[ ] Safe Clock Startup

Jitter Optimization

[✓ Balanced]
Minimize Output Jitter
Maximize Input Jitter filtering

Dynamic Reconfig Interface Options

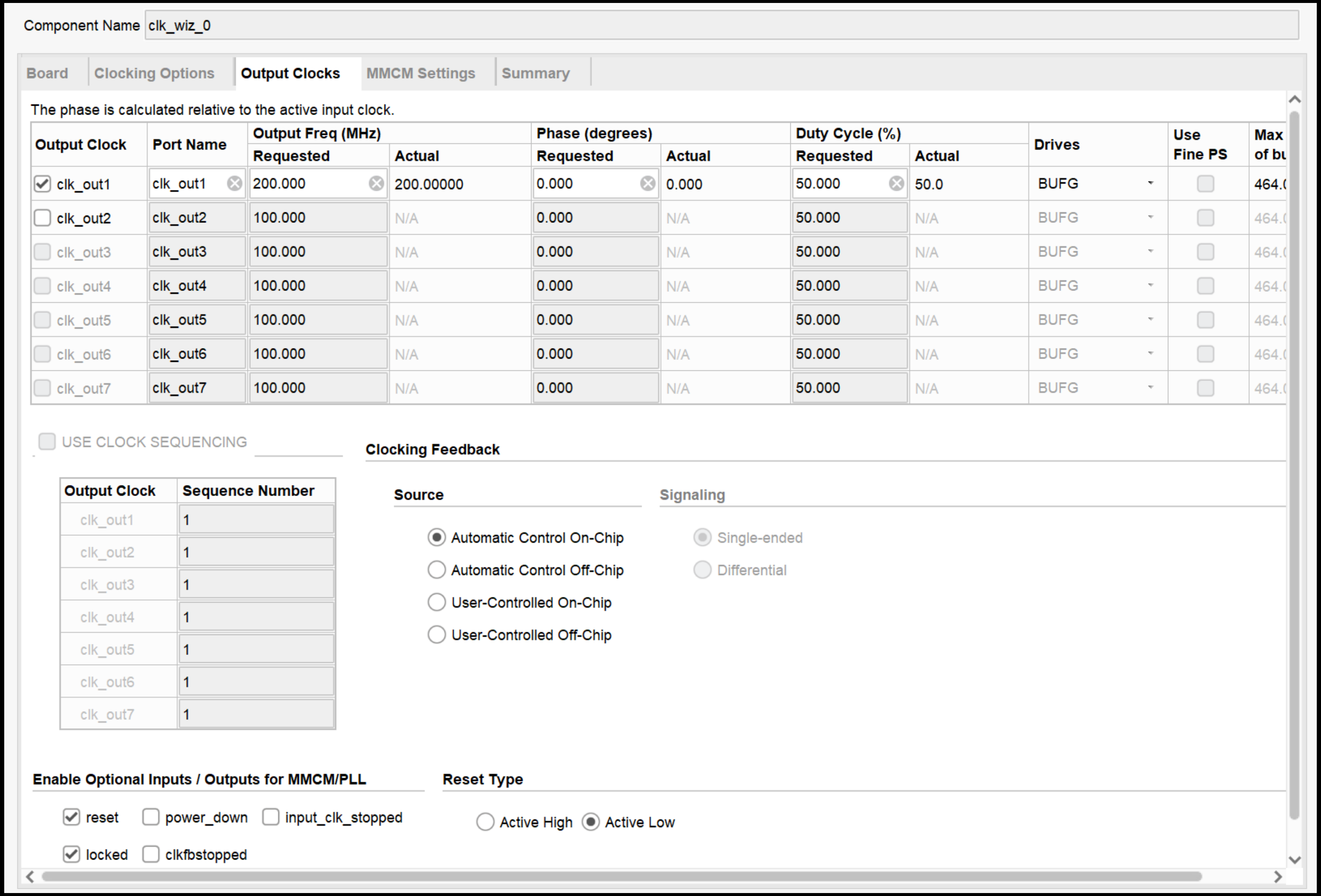
[✓ AXI4Lite] DRP
[ ] Phase Duty Cycle Config [ ] Write DRP registers

Input Clock Information

Input Clock	Port Name	Input Frequency(MHz)		Jitter Options	Input Jitter	Source
Primary	clk_in1	200.000	10.000 - 800.000	UI	0.010	Differential clock capable pin
[ ] Secondary	clk_in2	100.000	120.000 - 240.000		0.010	Single ended clock capable p

HDMI_loop

IP Setting



Component Name clk_wiz_0

Board | Clocking Options | **Output Clocks** | MMCM Settings | Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	Use Fine PS	Max of bu
		Requested	Actual	Requested	Actual	Requested	Actual			
✓ clk_out1	clk_out1	200.000	200.00000	0.000	0.000	50.000	50.0	BUFG	☐	464.
☐ clk_out2	clk_out2	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐	464.
☐ clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐	464.
☐ clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐	464.
☐ clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐	464.
☐ clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐	464.
☐ clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐	464.

☐ USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Clocking Feedback

Source

- [✓ Automatic Control On-Chip]
- Automatic Control Off-Chip
- User-Controlled On-Chip
- User-Controlled Off-Chip

Signaling

- [✓ Single-ended]
- Differential

Enable Optional Inputs / Outputs for MMCM/PLL

[✓ reset] ☐ power_down ☐ input_clk_stopped

[✓ locked] ☐ clkfbstopped

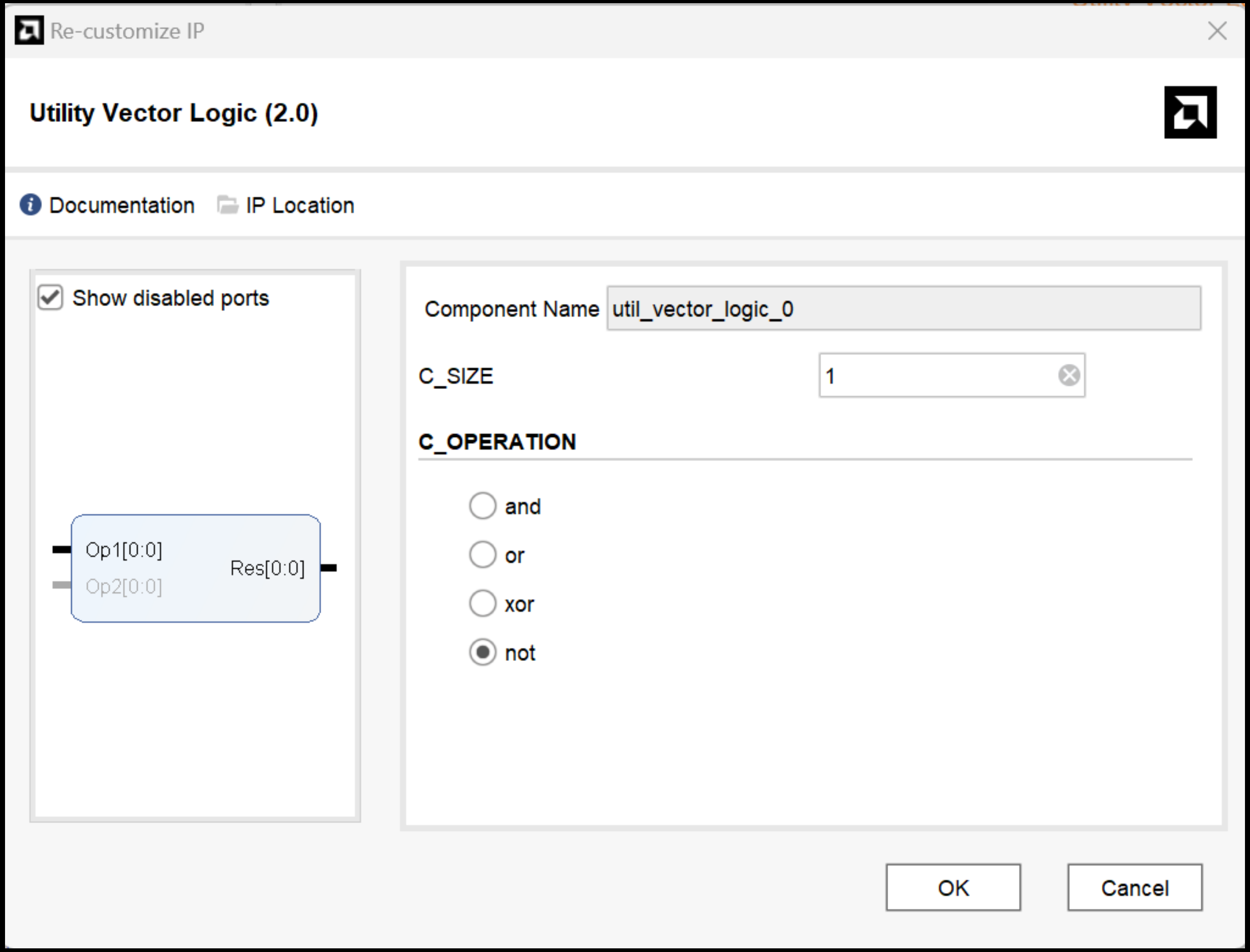
Reset Type

Active High [✓ Active Low]

AMD

HDMI_loop

IP Setting


Re-customize IP

Utility Vector Logic (2.0)

Documentation IP Location

[✓] Show disabled ports

Component Name: util_vector_logic_0

C_SIZE: 1

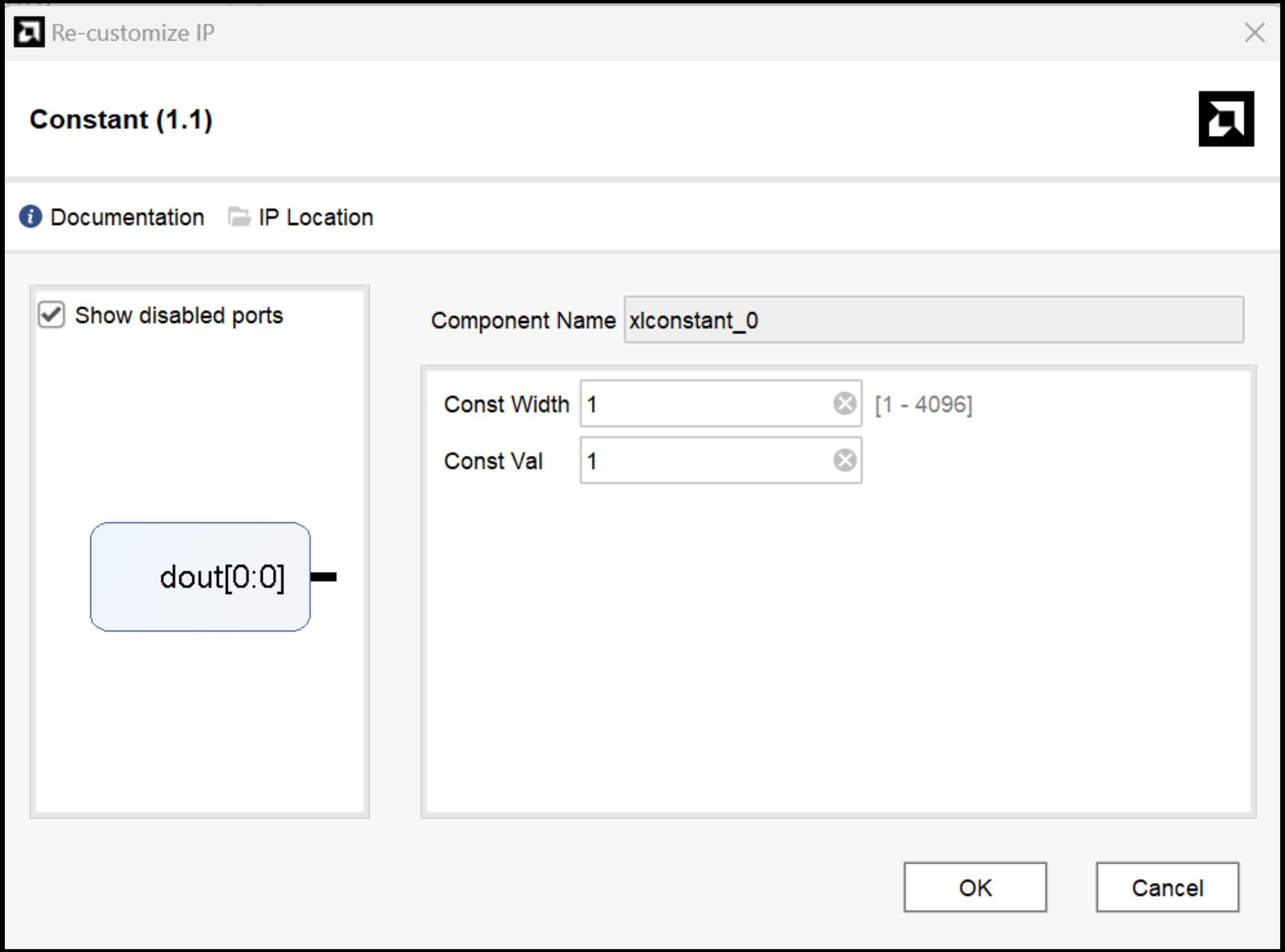
C_OPERATION

( ) and
( ) or
( ) xor
[✓ not]

OK Cancel

HDMI_loop

IP Setting



Re-customize IP

Constant (1.1)

Documentation | IP Location

☑ Show disabled ports

Component Name: xlconstant_0

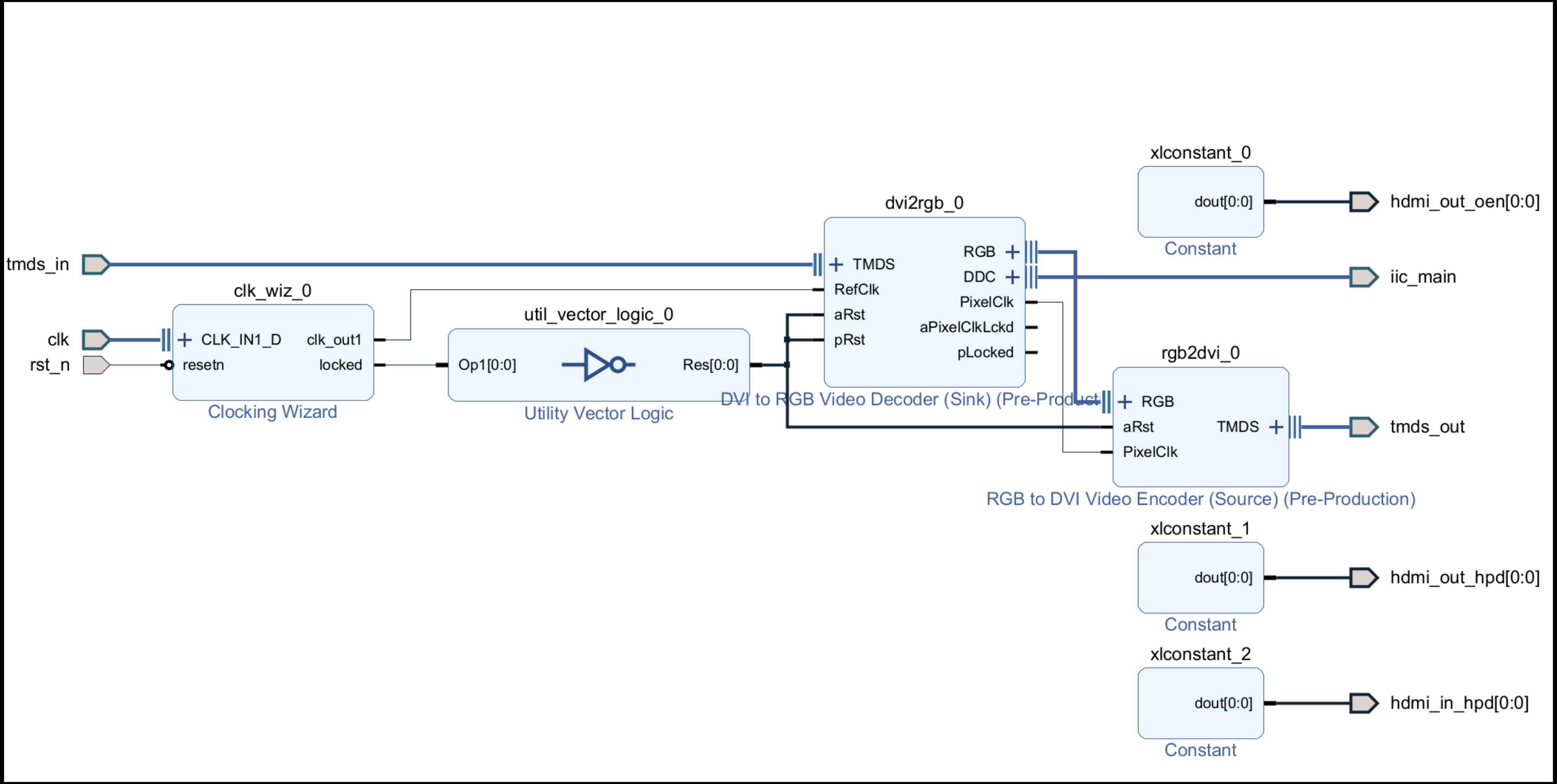
Const Width: 1 [1 - 4096]

Const Val: 1

OK | Cancel

HDMI_loop

Final Block Design


xlconstant_0
dout[0:0]
Constant
hdmi_out_oen[0:0]
dvi2rgb_0
tmds_in
TMDS
RGB
DDC
iic_main
RefClk
aRst
pRst
PixelClk
aPixelClkLckd
pLocked
DVI to RGB Video Decoder (Sink) (Pre-Production)
clk_wiz_0
clk
CLK_IN1_D
clk_out1
rst_n
resetn
locked
Clocking Wizard
util_vector_logic_0
Op1[0:0]
Res[0:0]
Utility Vector Logic
rgb2dvi_0
RGB
aRst
PixelClk
TMDS
tmds_out
RGB to DVI Video Encoder (Source) (Pre-Production)
xlconstant_1
dout[0:0]
Constant
hdmi_out_hpd[0:0]
xlconstant_2
dout[0:0]
Constant
hdmi_in_hpd[0:0]