Rostock University
Faculty of Computer Science and Electronic Engineering
Electronic Circuits and Appliances Division (Institute GS)

VHDL-Lab 3: Controlling a Display over HDMI (DVI) in SXGA-Mode

(Version for Spartan 6 on Scarab miniSpartan6+ board)

Task:
In a previous lab we generated synchronization signals (HSYNC, VSYNC, BLANK) and colour information for a direct VGA-output over Video-DAC.

In this lab we will use the synchronization and colour data for a full digital output over HDMI (High Definition Multimedia Interface).
We will not include Audio (embedded in video data stream), EDID (Extended Display Identification Data), DDC with HDCP-Handshaking (Encryption).
The given hardware limits the resolution to 1280x1024 pixels (as we used for VGA). To meet the common aspect ratio of digital display of 16:9 you may change the resolution to 1366x768. Table 1 shows the timing data for both 1280x1024 and 1366x768. If you choose another resolution be aware that a pixel rate (VIDEOCLOCK) of about 105 MHz (1050 MHz at the digital outputs) is the maximum for our hardware.

Table 1: Timing Information for different Resolutions

Resolution Refresh	Apect	Clock Mult/Div	H-Active	H-Front-Porch	H-SYNC	H-Back-Porch	V-Active	V-Front-Porch	V-SYNC	V-Back-Porch
800x600 @72Hz	4:3	50 MHz 2/2	800	56	120 high	64	720	37	6 high	23
1280x1024 @60Hz	4:3	108 MHz 13/6.	1280	48	112 high	248	600	1	3 high	38
1600x1200 @60Hz	4:3	162.5 MHz 13/4	1600	64	193 high	304	1200	1	3 high	46
1280x720 @60Hz	16:9	74.2 MHz 37/25	1280	110	40 high	220	720	5	5 high	20
1360x768 @60Hz	16:9	85,5 MHz 171/100	1360	64	112 high	256	768	3	6 high	18

There is a frequency synthesis mode (DFS) available for Digital Clock Modules (DCM). Using this mode we can derive a clock of $f = \frac{f_{in} \cdot MULT}{DIV}$. Be aware that $f_{in} \cdot MULT$ determines the VCO frequency that should be in a range 600 … 1600 MHz and MULT is selectable from 2.000 to 64.000 in steps of 0.125, while DIV is selectable from 2.000 to 128.000 in steps of 0.125.
In VHDL based design the MULT and DIV values are set as generics into the DCM model. (Project -> New Source -> Coregen -> FPGA Features and Design -> Clocking -> Clocking Wizard).
For 100 MHz input and 108 MHz output MULT/DIV = 10.75/10.0 (exactly 107.5 MHz) are practical values.

For analogous output he card uses a 240 MHz true colour digital to analogous converter (ADV7125KST240). Digital colour inputs are binary encoded with 8 bits per colour.

HDMI (like DVI and Display Port) uses differential wire pairs for the transmission of colour-information and clock. Information is 8 to 10 bit encoded in a manner that there is a minimal occurrence of transitions and no dc-component on the differential wire pair. Encoding is TMDS (Transition Minimized Differential Signalling).

In HDMI encoded data is shifted out by 10 times data rate of the video pixel clock. The pixel clock itself is transmitted on a separate channel.

FPGAs have a special output cell to shift out serial data with the appropriate data rate.

Tasks:

Clock generation
8b/10b-encoding of video data
Serializing data and implementation of output cells

Other than in the previous lab we have not only to create digital hardware in VHDL but we must implement the special FPGA-cells in the right manner.

From the VGA-Sync- and image generation module we get colour information as three 8-Bit vectors and the signals HSYNC, VSYNC and BLANK.
Shortly said, TMDI the underlying encoding scheme of HDMI does the following:

- For the time of visible pixels the 8-Bit colour vectors are encoded into 10 bits using a code with minimum number of transitions and no dc component over longer times.
- During the blanking time a special dc-free code is transmitted. Every colour channel may use 4 different codes hence encoding 4 states. The BLUE-channel is used to encode HSYNC and VSYNC. (It is possible to include audio date during blanking time.)

The encoded data (10 bits wide) are sent to differential output drivers by a clock of 10 times the picture clock. For serialization with high speed clocks the XILINX devices have special serializers (8 bits in our device) in their output cells. They may become cascaded to have 10 bits. TMDS-encoding is the candidates for our VHDL-code. All the other parts (yellow in figure 1) will be instantiations of XILINX library parts.

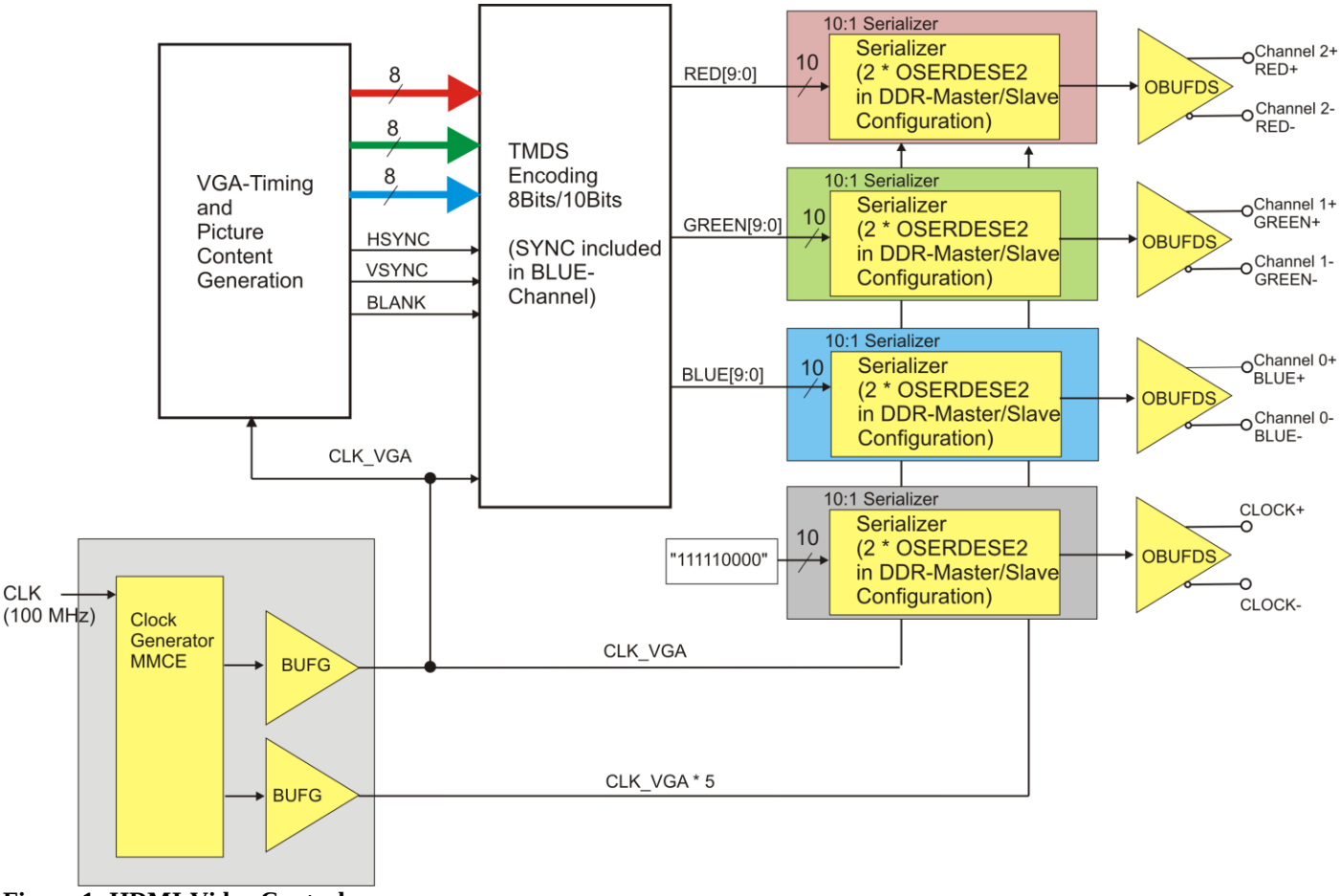


Figure 1: HDMI-Video Control

Figure 2 shows the data encoding algorithm for the BLUE-colour with the SYNC-signals included. On the other colour channels use the code for "00" during the blanking periods.

There is a counter "cnt" implemented for the deviance from DC-balance of the code sent in the past. This counter is used during encoding to correct the DC-balance.

A possible solution is the implementation of the hardware for the calculation of the number of ones in input data and intermediate code and the generation of the intermediate code as combinatorial parallel processes and the generation of the final code as clocked processes.

The use of the OSERDES2-cells for the 5 to 1 serializer will be given as a VHDL-description. It would be beyond this instruction paper to explain all the possible features of OSERDES2.
You may read www.xilinx.com/support/documentation/user_guides/**ug380**.pdf *pp.91.* For our design we use cascaded IOSERDES-cells to form a 5 to 1 serializer.
Least significant bit is transferred first.
CLK is transferred with 0-value for the first 5 data bits and 1-value for the higher 5 data bits.

Table 2: Pin assignment (HDMI-related signals) for demonstration board with xc7a50tfgg484

HDMI-Signal	Artix7 on MIMAS A7 Board	
CLOCK (50 MHz Input)	K3	IOSTANDARD = LVCMOS33
TMDS_BLUE_P (Channel 0)	B1	IOSTANDARD = TMDS_33
TMDS_BLUE_N (Channel 0)	A1	IOSTANDARD = TMDS_33
TMDS_GREEN_P (Channel 1)	E1	IOSTANDARD = TMDS_33
TMDS_GREEN_N (Channel 1)	D1	IOSTANDARD = TMDS_33
TMDS_RED_P (Channel 2)	G1	IOSTANDARD = TMDS_33
TMDS_RED_N (Channel 2)	F1	IOSTANDARD = TMDS_33
TMDS_TCLK_P	L3	IOSTANDARD = TMDS_33
TMDS_TCLK_N	K3	IOSTANDARD = TMDS_33

Tasks:

1. Create a design for the 1280x1024@60Hz resolution with CLK_100MHz as input clock and the differential digital TMDS-encoded video data stream as outputs. In addition the design shall produce the colour information for a picture on 3*8 bit. The picture should be a predefined pattern. You may use the results from the previous VGA-lab for the generation of the RGB-picture and the timing. Divide the design into the parts:
   - Generation and control of clocks
   - VHDL-code for TMDS-Encoding of RGB-data and synchronization signals
   - Instantiation of the appropriate special FPGA-Blocks (MMCE, IOSERDESE2, IO-Driver)
   - Creation of RGB signals
2. Run a functional (VHDL-) simulation in Modelsim or XILINX ISim!
3. Synthesize the design using XSE from XILINX or Design Compiler from SYNOPSYS!
4. Implement the design in a xc7a50tfgg484 FPGA from XILINX. Can your design work with the given pixel clock?
5. Run the post-layout simulation in Modelsim or Xilinx ISim!
6. Open the design inside fpga_editor and try to recognize your description in the placed and routed hardware.
7. Download the design to a testboard and show that it works.
8. Measure the signals using of an oscilloscope and/or logic analyzer.

Some Useful Hints:

1. Use the prepared VHDL-skeleton-file for the serializers with OSERDES2E2 cells!
2. Use the prepared template hdmi_top.vhd for your design and the prepared .xdc-file for pin assignment!

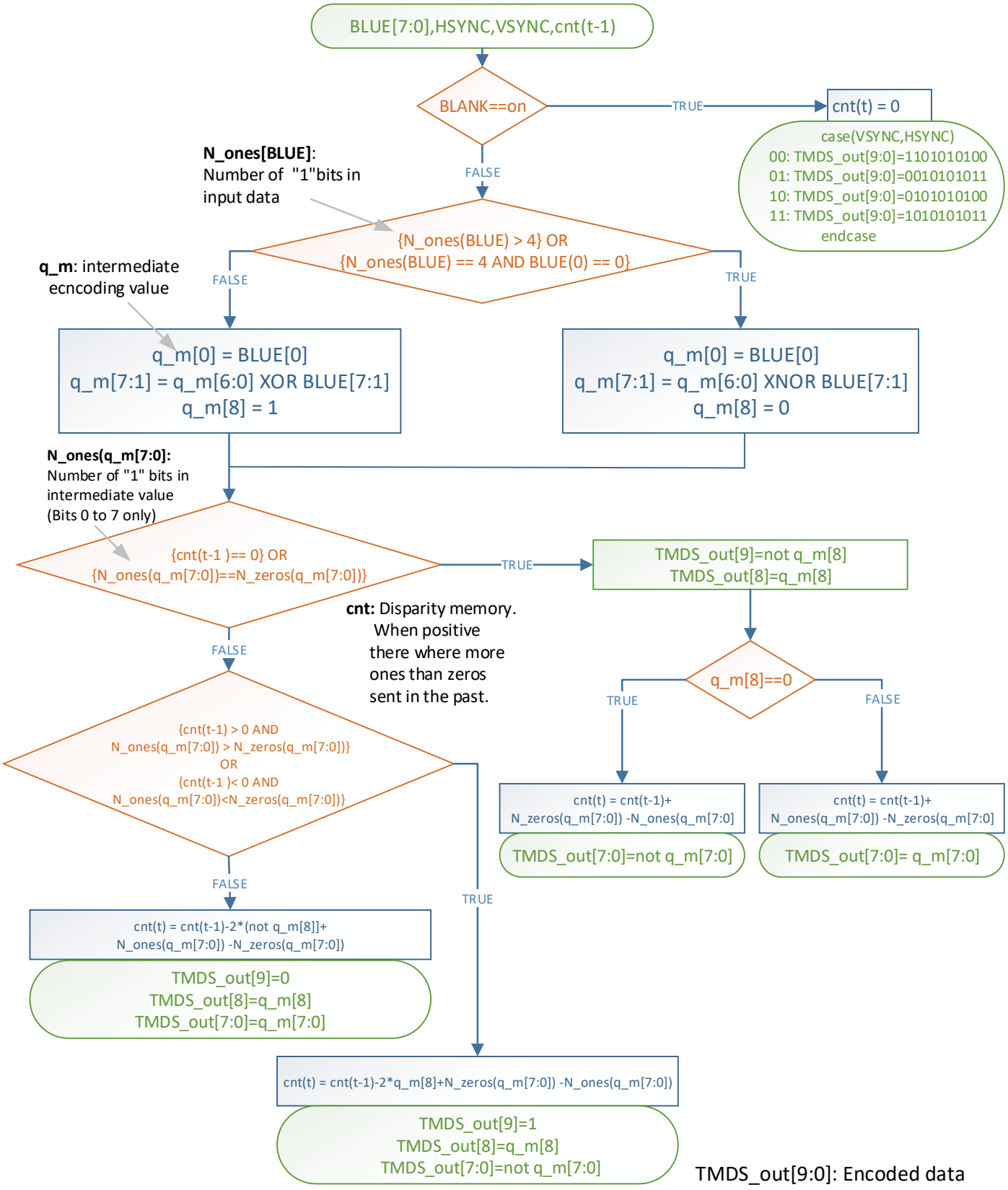


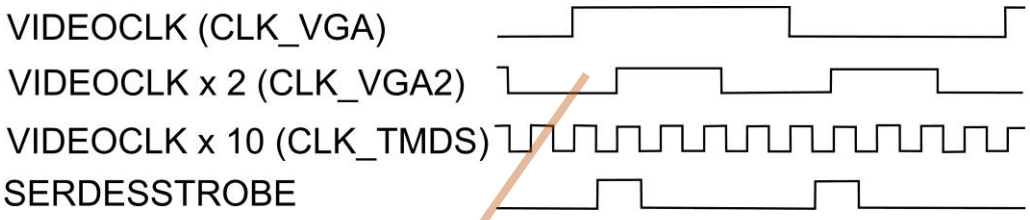
Figure 2: TMDS Encoding

Additional Tasks:

The DCM-Module can be configured dynamically using an internal SPI-Interface. So it will be possible to change the resolution of the image at runtime. Therefore you will have to implement a table (memory) with multiplier/divider pairs for the DCM and values for the picture dimensions

(width, sync-time etc.) for every resolution. An additional module must be designed to transfer multiplier/divider into the DCM via SPI. Changes may be started from input switches on the board.

It is possible to transmit audio data embedded in the video stream. For audio a special encoding in the "nonvisible" time is used replacing the standard transfer of HSYNC and VSYNC.
You may add a simple audio generator (SIN-wave or something like that) and add an audio transmission.
Example for the serializer 10 to 1 Bit using XILINX OSERDES2E2 I/O-cell primitive:




```
entity serializer is
    Port ( CLK_TMDS : in  STD_LOGIC;
                          CLK_VGA : in STD_LOGIC;
           TMDS_IN : in  STD_LOGIC_VECTOR (9 downto 0);
                          TMDS_OUT: out std_logic;
                          LOCKED: in  STD_LOGIC);
end serializer;


architecture Behavioral of serializer is

    signal shift1 : std_logic := '0';
    signal shift2 : std_logic := '0';
    signal reset : std_logic;
--     signal clk_vga_int, clk_tmds_int : std_logic;

begin
reset <= not LOCKED;


master_serdes : OSERDESE2
   generic map (
      DATA_RATE_OQ => "DDR",   -- DDR, SDR
      DATA_RATE_TQ => "DDR",   -- DDR, BUF, SDR
      DATA_WIDTH => 10,         -- Parallel data width (2-8,10,14)
      INIT_OQ => '1',          -- Initial value of OQ output (1'b0,1'b1)
      INIT_TQ => '1',          -- Initial value of TQ output (1'b0,1'b1)
      SERDES_MODE => "MASTER", -- MASTER, SLAVE
      SRVAL_OQ => '0',         -- OQ output value when SR is used (1'b0,1'b1)
      SRVAL_TQ => '0',         -- TQ output value when SR is used (1'b0,1'b1)
      TBYTE_CTL => "FALSE",    -- Enable tristate byte operation (FALSE, TRUE)
      TBYTE_SRC => "FALSE",    -- Tristate byte source (FALSE, TRUE)
      TRISTATE_WIDTH => 1      -- 3-state converter width (1,4)  !! Design rule check warns that it should be 4, but
bitgen gives an error using 4 and depends 1
   )
   port map (
      OFB       => open,            -- 1-bit output: Feedback path for data
      OQ        => TMDS_OUT,              -- 1-bit output: Data path output
      -- SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion (1-bit each)
      SHIFTOUT1 => open,
      SHIFTOUT2 => open,
      TBYTEOUT  => open,   -- 1-bit output: Byte group tristate
      TFB       => open,            -- 1-bit output: 3-state control
      TQ        => open,             -- 1-bit output: 3-state control
      CLK       => clk_tmds,            -- 1-bit input: High speed clock
      CLKDIV    => clk_vga,       -- 1-bit input: Divided clock
      -- D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
```

```
    D1 => TMDS_IN(0),
    D2 => TMDS_IN(1),
    D3 => TMDS_IN(2),
    D4 => TMDS_IN(3),
    D5 => TMDS_IN(4),
    D6 => TMDS_IN(5),
    D7 => TMDS_IN(6),
    D8 => TMDS_IN(7),
    OCE => LOCKED, --ce_delay(0),            -- 1-bit input: Output data clock enable
    RST => reset,            -- 1-bit input: Reset
    -- SHIFTIN1 / SHIFTIN2: 1-bit (each) input: Data input expansion (1-bit each)
    SHIFTIN1 => SHIFT1,
    SHIFTIN2 => SHIFT2,
    -- T1 - T4: 1-bit (each) input: Parallel 3-state inputs
    T1 => '0',
    T2 => '0',
    T3 => '0',
    T4 => '0',
    TBYTEIN => '0', -- 1-bit input: Byte group tristate
    TCE => '0'                -- 1-bit input: 3-state clock enable
  );

slave_serdes : OSERDESE2
  generic map (
    DATA_RATE_OQ   => "DDR",   -- DDR, SDR
    DATA_RATE_TQ   => "DDR",   -- DDR, BUF, SDR
    DATA_WIDTH     => 10,      -- Parallel data width (2-8,10,14)
    INIT_OQ        => '1',     -- Initial value of OQ output (1'b0,1'b1)
    INIT_TQ        => '1',     -- Initial value of TQ output (1'b0,1'b1)
    SERDES_MODE    => "SLAVE", -- MASTER, SLAVE
    SRVAL_OQ       => '0',     -- OQ output value when SR is used (1'b0,1'b1)
    SRVAL_TQ       => '0',     -- TQ output value when SR is used (1'b0,1'b1)
    TBYTE_CTL      => "FALSE", -- Enable tristate byte operation (FALSE, TRUE)
    TBYTE_SRC      => "FALSE", -- Tristate byte source (FALSE, TRUE)
    TRISTATE_WIDTH => 1        -- 3-state converter width (1,4) !! Design rule check warns that it should be 4, but
bitgen gives an error using 4 and depends 1
  )
  port map (
    OFB       => open,         -- 1-bit output: Feedback path for data
    OQ        => open,         -- 1-bit output: Data path output
    -- SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion (1-bit each)
    SHIFTOUT1 => shift1,
    SHIFTOUT2 => shift2,

    TBYTEOUT  => open,    -- 1-bit output: Byte group tristate
    TFB       => open,    -- 1-bit output: 3-state control
    TQ        => open,    -- 1-bit output: 3-state control
    CLK       => clk_tmds,  -- 1-bit input: High speed clock
    CLKDIV    => clk_vga,     -- 1-bit input: Divided clock
    -- D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
    D1        => '0',
    D2        => '0',
    D3        => TMDS_IN(8),
    D4        => TMDS_IN(9),
    D5        => '0',
    D6        => '0',
    D7        => '0',
    D8        => '0',
    OCE       => LOCKED,  -- ce_delay(0),
    RST       => reset,
    SHIFTIN1 => '0',
    SHIFTIN2 => '0',
    -- T1 - T4: 1-bit (each) input: Parallel 3-state inputs
    T1        => '0',
    T2        => '0',
```

```
      T3       => '0',
      T4       => '0',
      TBYTEIN  => '0',
      TCE      => '0'
   );
--delay_ce: process(clk_x5)
--    begin
 --       if rising_edge(clk_x5) then
--            ce_delay <= not reset & ce_delay(ce_delay'high downto 1);
--        end if;
--    end process;
end Behavioral;
```