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 XGA-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 54/25	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 $f = \frac{f_{in} \cdot MULT}{DIV}$ with MULT an integer in the range of 2…256 and DIV an integer in the range of 1…256.
In VHDL based design simulation the MULT and DIV values are set as generics into the DCM model. For synthesis we use entries in the XILINX constraint file (ucf-file). One can use the XILINX core generator wizard to generate the clock source
(Project -> New Source -> Coregen -> FPGA Features and Design -> Clocking -> Clocking Wizard).
For 50 MHz input and 108 MHz output MULT/DIV = 28/13 (107.69 MHz] or 13/6 (108.33 MHZ) are practical values. The wizard will calculate them from inputs for input and output frequency.

The 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 direct 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 (4 bit in our device) in their output cells. They may become cascaded to have up to 8 bits. We would need 10 bits. So we cascade to 5 bits and use a multiplexer with twice the VIDEOCLOCK for 10 to 5 bit conversion. Multiplexers and the TMDS-encoding are the candidates for our VHDL-code. All the other parts (yellow in figure 1) will be instantiations of XILINX library parts.

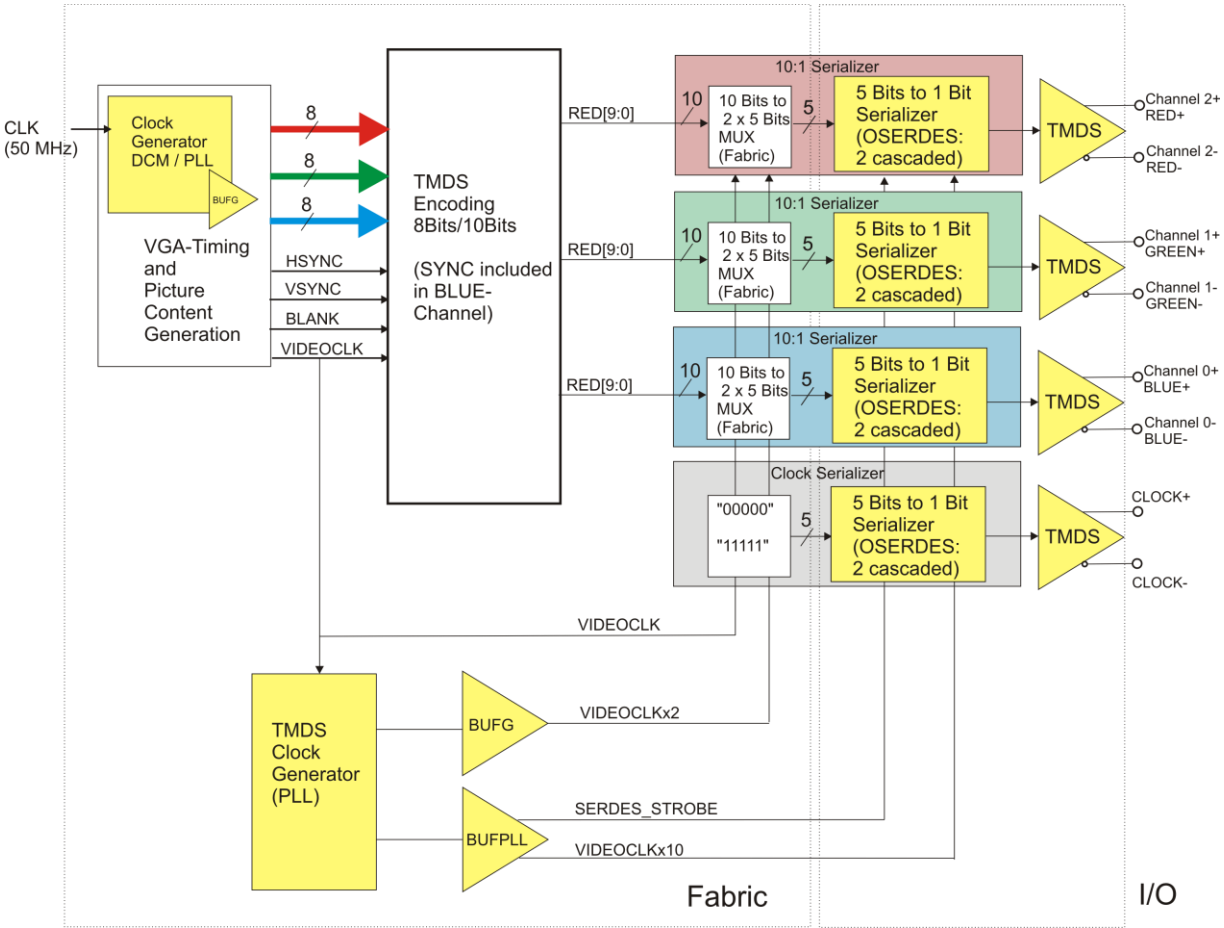


Figure 1: HDMI-Video Control

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 xc6xlx9-3ftg256

HDMI-Signal	Spartan6+ Scarab Board	
CLOCK (50 MHz Input)	K3	IOSTANDARD = LVCMOS33
TMDS_BLUE_P (Channel 0)	C13	IOSTANDARD = TMDS_33
TMDS_BLUE_N (Channel 0)	A13	IOSTANDARD = TMDS_33
TMDS_GREEN_P (Channel 1)	B12	IOSTANDARD = TMDS_33
TMDS_GREEN_N (Channel 1)	A12	IOSTANDARD = TMDS_33
TMDS_RED_P (Channel 2)	C11	IOSTANDARD = TMDS_33
TMDS_RED_N (Channel 2)	A11	IOSTANDARD = TMDS_33
TMDS_TCLK_P	B14	IOSTANDARD = TMDS_33
TMDS_TCLK_N	A14	IOSTANDARD = TMDS_33

Tasks:
1. Create a design for the 1280x1024@60Hz resolution with CLK_50MHz as input clock and the differential digital tmds-encoded video data stream as outputs. In addition it shall produce the colour information for a picture on 3*8 bit. The picture should be a predefined pattern. 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 (PLL, IOSERDES, IO-Driver)
   - Creation of RGB signals
2. Carry out 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 xc6xlx9-3ftg256 FPGA from XILINX. Can your design work with the given pixel clock?
5. Carry out 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 OSERDES2_cells!
2. Use the prepared template hdmi_top.vhd for your design and the prepared .ucf-file for pin assignment!

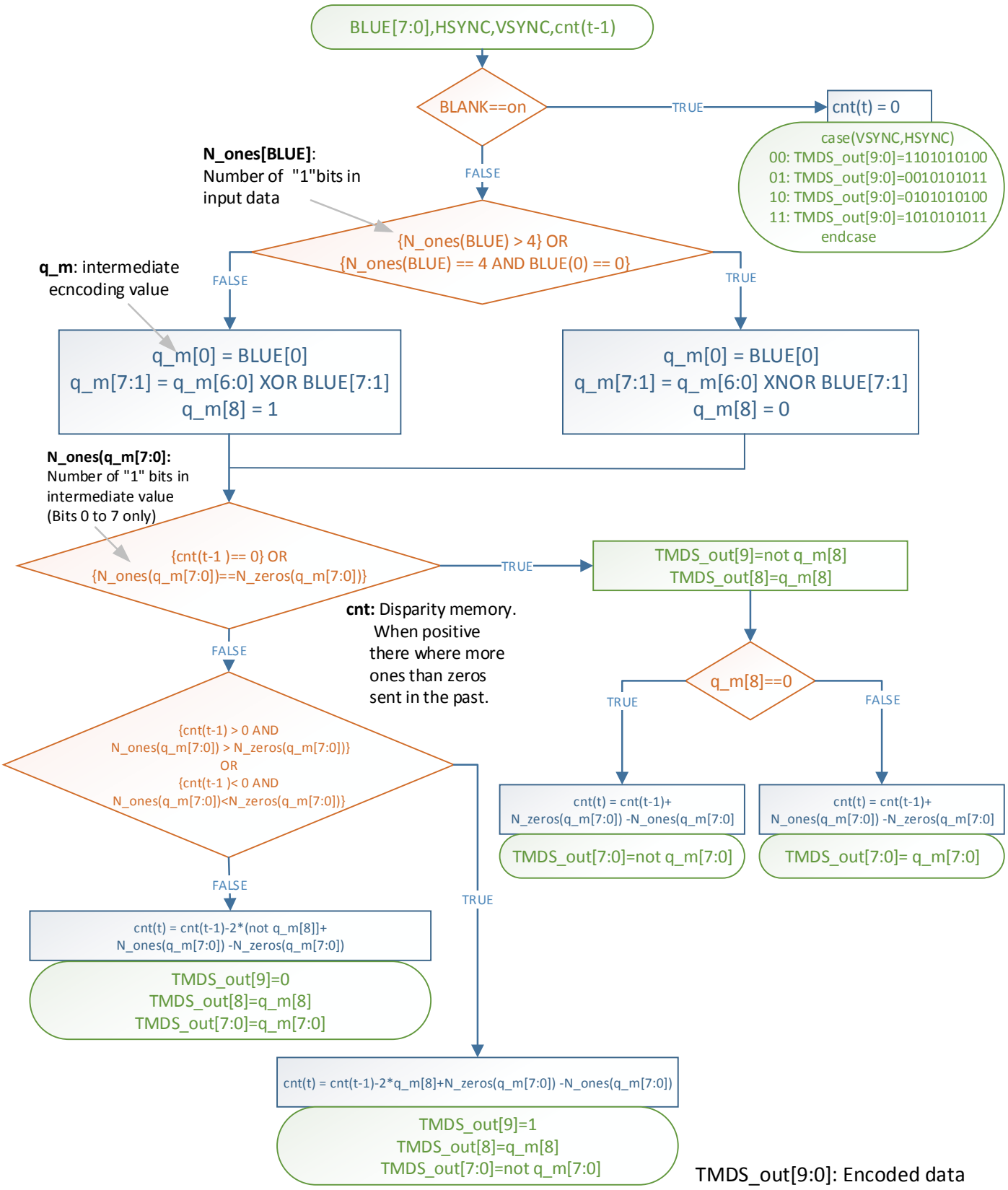


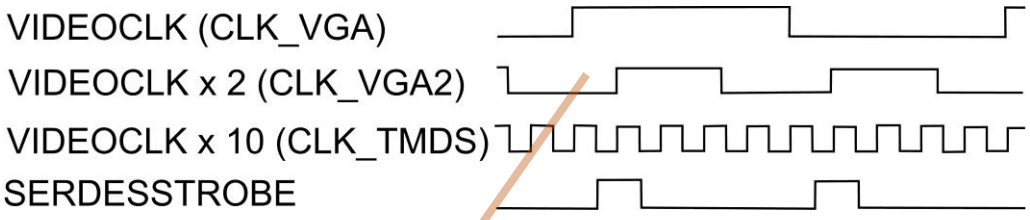
Figure 2: TMDS Encoding

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 OSERDES2 I/O-cell primitive:




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library UNISIM;
use UNISIM.VComponents.all;

--Serialize 10 bist as 2*5 bits using OSERDES2-Blocks
--5 bits serializer needs a 1 bit master and a 4 bit slave OSERDES2 in cascade

entity serializer is
    Port ( CLK_TMDS : in  STD_LOGIC; --Pixelclock x 10
           CLK_VGA : in STD_LOGIC;      --Pixelclock
           CLK_VGA2 : in STD_LOGIC;    --Pixelclock x 2
           TMDS_IN : in  STD_LOGIC_VECTOR (9 downto 0);
           TMDS_OUT: out std_logic;
           SERDESSTROBE: in  STD_LOGIC);
end serializer;

architecture mixed of serializer is
           signal parin: std_logic_vector (4 downto 0);
           signal clock_enable, io_reset   : std_logic;
            signal ocascade_ms_d, ocascade_ms_t,,  signal ocascade_sm_d, ocascade_sm_t          : std_logic;
            signal data_out_to_pin          : std_logic;
begin

  mux_data: process (CLK_VGA2)
                                begin
                                if rising_edge(CLK_VGA2) then
                                 if CLK_VGA = '1' then
                                  parin  <= TMDS_IN(4 downto 0);
                                 else
                                  parin  <= TMDS_IN(9 downto 5);
                                 end if;
                                end if;
end process mux_data;

             clock_enable <= '1';
             io_reset <= '0';

    -- Instantiate the serdes primitive
 oserdes2_master : OSERDES2
      generic map (
        DATA_RATE_OQ   => "SDR",
        DATA_RATE_OT   => "SDR",
        TRAIN_PATTERN  => 0,
        DATA_WIDTH     => 5,
        SERDES_MODE    => "MASTER",
        OUTPUT_MODE    => "SINGLE_ENDED")
```

```
      port map (
       D1        => parin(4),
       D2        => '0',
       D3        => '0',
       D4        => '0',
       T1        => '0',
       T2        => '0',
       T3        => '0',
       T4        => '0',
       SHIFTIN1   => '1',
       SHIFTIN2   => '1',
       SHIFTIN3   => ocascade_sm_d,
       SHIFTIN4   => ocascade_sm_t,
       SHIFTOUT1  => ocascade_ms_d,
       SHIFTOUT2  => ocascade_ms_t,
       SHIFTOUT3  => open,
       SHIFTOUT4  => open,
       TRAIN      => '0',
       OCE        => clock_enable,
       CLK0       => CLK_TMDS, --10*pixelclock
       CLK1       => '0',
       CLKDIV     => CLK_VGA2, --2*pixelclock
       OQ         => TMDS_OUT,
       TQ         => open,
       IOCE       => SERDESSTROBE,
       TCE        => clock_enable,
       RST        => io_reset);

 oserdes2_slave : OSERDES2
      generic map (
        DATA_RATE_OQ   => "SDR",
        DATA_RATE_OT   => "SDR",
        DATA_WIDTH     => 5,
        SERDES_MODE    => "SLAVE",
        TRAIN_PATTERN  => 0,
        OUTPUT_MODE    => "DIFFERENTIAL")
      port map (
       D1        => parin(0),
       D2        => parin(1),
       D3        => parin(2),
       D4        => parin(3),
       T1        => '0',
       T2        => '0',
       T3        => '0',
       T4        => '0',
       SHIFTIN1   => ocascade_ms_d,
       SHIFTIN2   => ocascade_ms_t,
       SHIFTIN3   => '1',
       SHIFTIN4   => '1',
       SHIFTOUT1  => open,
       SHIFTOUT2  => open,
       SHIFTOUT3  => ocascade_sm_d,
       SHIFTOUT4  => ocascade_sm_t,
       TRAIN      => '0',
       OCE        => clock_enable,
       CLK0       => CLK_TMDS,
       CLK1       => '0',
       CLKDIV     => CLK_VGA2,
       OQ         => open,
       TQ         => open,
       IOCE       => SERDESSTROBE,
       TCE        => clock_enable,
       RST        => io_reset);
end mixed;
```