ISSN 2277-3061

A COST EFFECTIVE DVI INTERFACE ON VIRTEX 5 FPGA THROUGH VERILOG HDL

Asif Ahmad A S
ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT
USN:1BM12EC022
BMS COLLEGE OF ENGINEERING,ACCREDITED BY NBA
BANGALORE,INDIA
asif949@gmail.com

ABSTRACT

There is a definite need for video and image processing technologies in today's world. However the computer vision technologies need to be tested and optimized. There is need for testing these interfaces for the platform which we work on. This modeling is a cost effective architecture for interfacing Digital Visual Interface(DVI) on Virtex5 FPGA's. The architecture is modeled in such a way that it does not use XPS micro blaze or Power PC processor but simple pixel feeder design, configuration of the Chrontel 7301C chip design and the interface between them.

Indexing terms/Keywords

PixelFeeder, DVI, pixel counter, SDR2DDR, I2CDRAM master, FIFO initial, PLL.

Academic Discipline And Sub-Disciplines

Computing sciences, Electrical Engineering, VLSI, FPGA Image Processing.

SUBJECT CLASSIFICATION

VLSI design and Embedded Systems, Electrical and Computing Sciences

TYPE (METHOD/APPROACH)

An experimental approach is used here. The result used here can be used straightaway for other VLSI image processing applications. The Verilog design with each of its coding lines with its relevant diagrams is used here. The lines of code are highlighted in bold. The language used is simple, clear and concise for the ease for the reader's understanding.

.

Council for Innovative Research

Peer Review Research Publishing System

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 13, No. 2

editor@cirworld.com

www.cirworld.com, www.ijctonline.com

ISSN 2277-3061

INTRODUCTION

The design of the DVI interface on virtex 5 requires the modeling of a pixel feeder module design and the correct DVI interfaces between them. The DVI chip present on the FPGA is Chrontel 7301C. The CH7301C is a display controller chip which accepts a digital input signal, encodes it and transmits data through a DVI(Digital Visual Interface or a DFP(Digital flat panel). The DVI module handles communication with the Chrontel DVI chip and initializes its control registers via an I2C bus to set the desired color format and update the pixel rate. Thus the DVI driver needs a constant stream supply of pixels to initialize the clock rate and to output the pixel. Thus a pixel feeder module is necessary. The DVI driver causes the Chrontel DVI chip to generate Synchronous VGA 800x600 at 75hz.The pixel feeder module is responsible for continuously reading from the frame buffer of the SDRAM and feeding the pixel values to the DVI driver module.

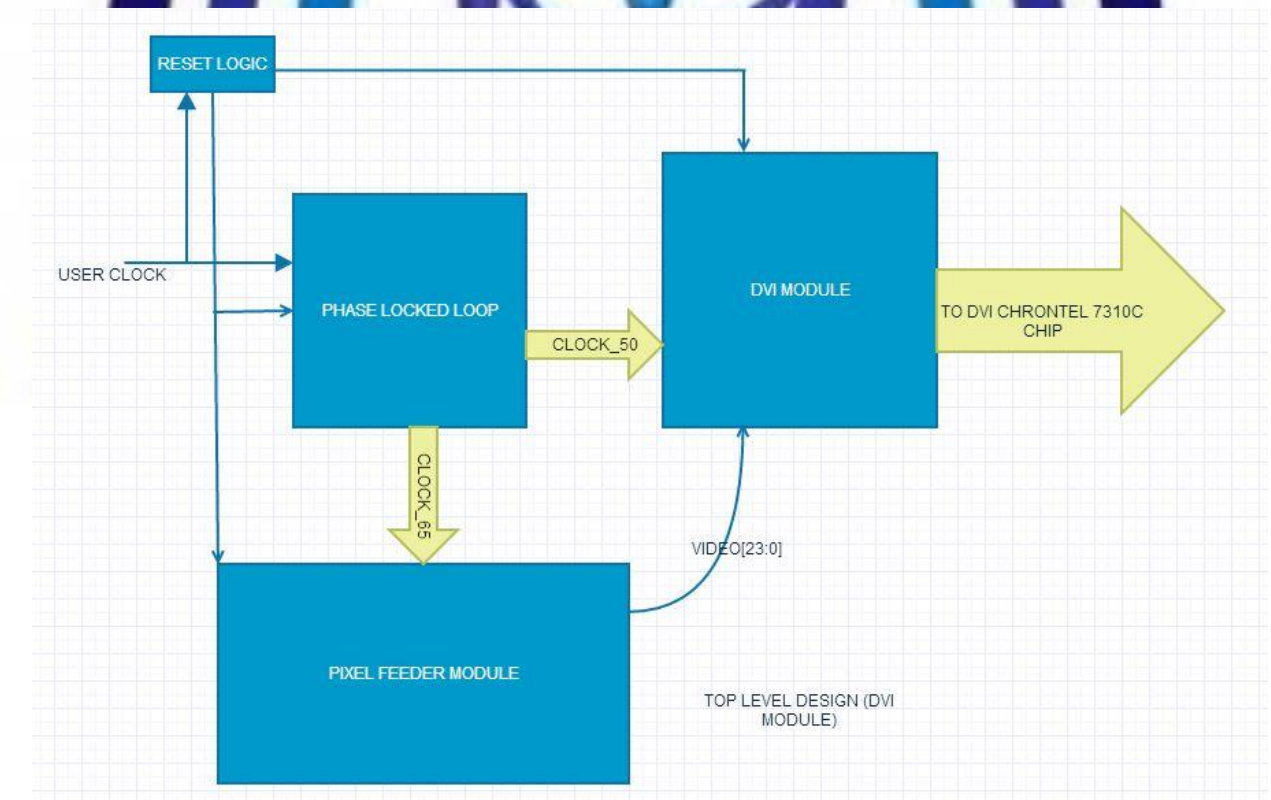
The aim of this paper is to design a fully customized design or a DVI interface by Verilog HDL. This is a cost effective algorithm as it does not require the inclusion of external libraries of XPS and Xilinx EDK. The use of these software forces the inclusion of a MICROBLAZE or a POWER PC soft core processor which gives an unnecessary load for including xps_tft, input output modules such as GPIO,UART, bus peripherals. Another disadvantage of these software is that these IP's cost a lot. Even after addition of these IP's the test software(.elf files) IP's to run on these soft core processors also has to be bought. Also running a software on a processor reduces its speed of execution unless and until there is software parallelization**.** Amdahl's law states that the maximum speed up of the parallelized version is 1.136 times as fast as the non-parallelized implementation[courtesy-Wikipedia].

Thus this type of approach is money-spinning and not efficient. My algorithm is to completely model a hardware architecture on the FPGA that increases its performance and reduce its cost when used for high speed image processing, video processing and computer vision applications.

II. ALGORITHM AND ARCHITECTURE

A. TOP LEVEL ARCHITECTURE AND BLOCK DIAGRAM

Starting from the clock input to the whole design the user clock is inputted to the PLL base module. The PLL base module generates a clock input for the DVI Chrontel chip and the pixel feeder module. The pixel feeder module accepts the clock and reset signals and generates the pixels to produce a 24 bit video signals to the input of the DVI. The DVI module accepts the inputs reset, clock from the PLL and the 24-bit video signal from the pixel feeder module and outputs that to the DVI Chrontel chip which is interfaced to an external DVI monitor.



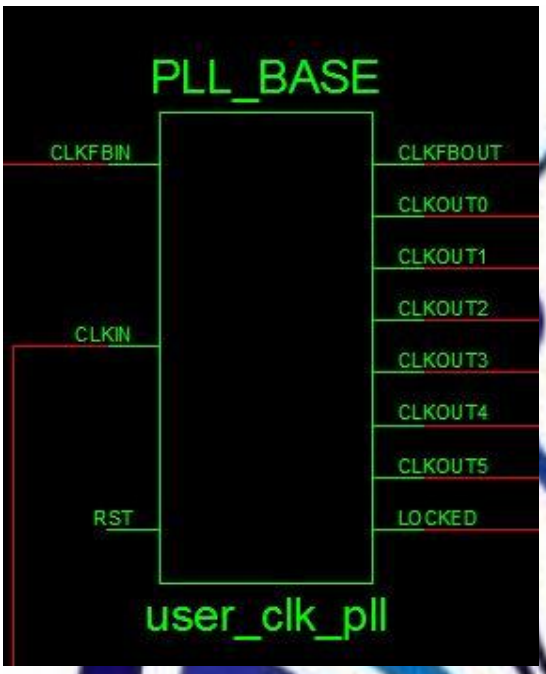
ISSN 2277-3061

B. PLL_BASE MODULE

The Chrontel chip needs an external clock supply for active functioning. The PLL(Phase Locked Loop) clock jitter module can eliminate this need. The user clock is input from one of the crystal oscillators on the board and is multiplied by this module. The output from this module consists of a series of clock frequencies which can be input to the pixel feeder and the DVI modules. Below are some lines of code.

This code is used to output certain clock frequencies out from the PLL module.

```
IBUFG user_clk_buf ( .I(USER_CLK), .O(user_clk_g) ); BUFG cpu_clk_buf ( .I(cpu_clk), .O(cpu_clk_g) );
BUFG clk200_buf ( .I(clk200), .O(clk200_g) );
BUFG clk0_buf ( .I(clk0), .O(clk0_g) );
BUFG clkdiv50_buf ( .I(clk50), .O(clk50_g) );
BUFG clk90_buf ( .I(clk90), .O(clk90_g) );
BUFG clkdiv0_buf ( .I(clkdiv0), .O(clkdiv0_g) ));
```



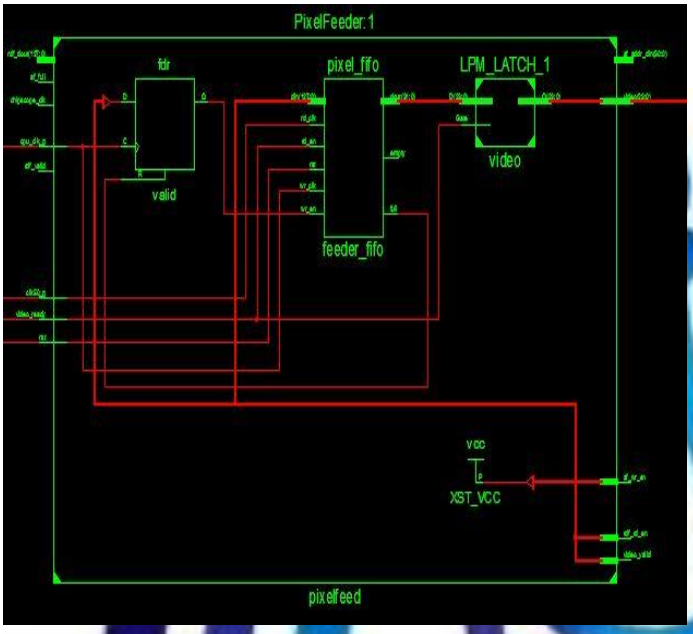
C. PIXEL FEEDER MODULE

The first step to display pixels on the screen is to model the PixelFeeder module. This module is very much responsible to continuously read from the frame buffer address space of the SDRAM and to feed the pixel values to the DVI module. The challenge will be to deal with the SDRAM latency levels. When we submit a request we will have to make sure that there is enough room on the FIFO when the request returns from the SDRAM. To accomplish this task we count the number of pixels on the request area and the FIFO. The code lines below explain this:

```
always @(posedge cpu_clk_g)
begin
        if(!feeder_full)
 begin
      valid <= 1;
      dout <=128'hffffffffffffffffffffffffffffffff; //rdf_dout; //read in processed output pixels
end
else begin
valid <= 0;
end
  end
```

ISSN 2277-3061

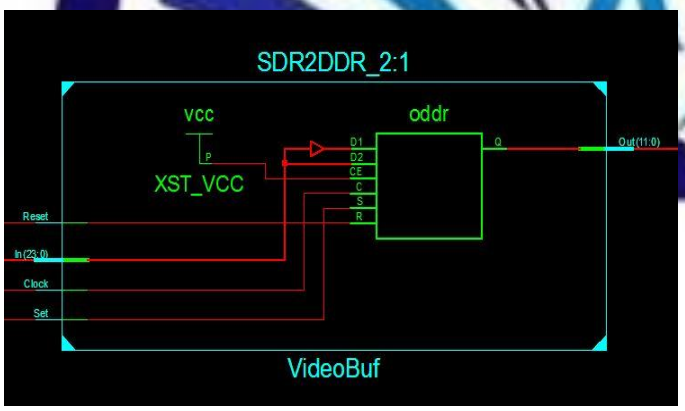
```
//assign dout = 128'hffffffffffffffffffffffffffffffff;
//assign valid = 1'b1;

always @(*) begin
   if (video_ready) begin
      video = feeder_dout[23:0];
      video_valid = 1'b1;
   end
```



The pixel feeder module is as shown above.

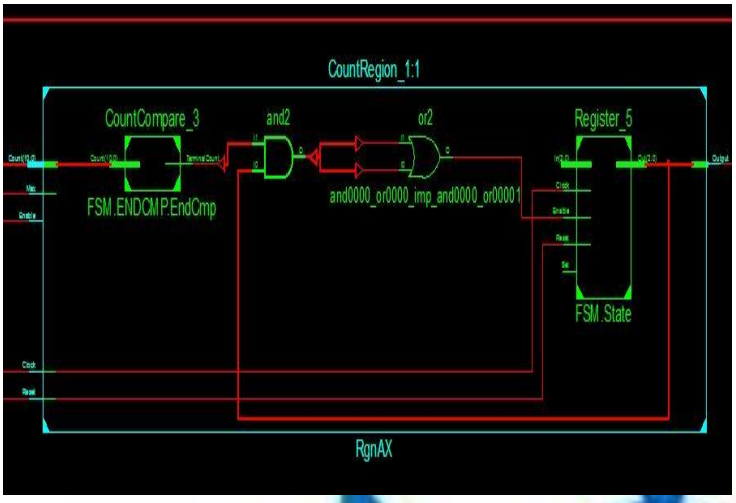
D. DVI MODULE

The DVI module accepts the input from the pixel feeder,(Video[23:0]).This signal is routed into the SDR2DDR memory. The output DDR(ODDR) is the output data interface for any SDR2DDR memory. This data is latched out to the DVI_D(11:0) Port. This forms the video buffer of the DVI module.



For Hsync and Vsync of the DVI interface display, an IORegister module has a simple D flip flop to and produces a DVI_H,DVI_V,DVI_D signals. The pixel active and the video ready signals must be fed back into the pixel feeder FIFO so that the pixel feeder must output a next pixel. As said above the challenge in this kind of design is to maintain and deal with the SDRAM latency when the above operation is accomplished. The best way to accomplish this task is to count the number of pixels on the request area and the FIFO. The videoready signal is fed back to the pixel feeder module through the pixel counter. Thus the pixel counter counts the number of pixels in the request area and is fed back to the LPM_LATCH in the pixel feeder. The pixel must be count in both the X and Y directions, stored in a register and must be compared with the request area. Similarly the Hsync and Vsync signals are compared and stored. Thus at the end of a pixel write onto the DVI chip, the SDRAM pulls out an interrupt and feeds it back into the pixel FIFO to know it is full. The

module below shows the count request area for the X-direction. The register is a behavioral register whose code is as shown below.




```
generate if (AsyncReset) begin:AR
                if (AsyncSet) begin:AS
                        always @ (posedge Clock or posedge Reset or posedge Set) begin
                        if (Reset) Out <=                       ResetValue;
                        else if (Set) Out <=                    SetValue;
                        else if (Enable) Out <=                 In;
        end
        end else begin:SS
        always @ (posedge Clock or posedge Reset) begin
if (Reset) Out <= ResetValue;
        else if (Set) Out <= SetValue;
        else if (Enable) Out <=In;
        end
end
end else begin:SR
        if (AsyncSet) begin:AS
                always @ (posedge Clock or posedge Set) begin
if (Reset) Out <=ResetValue;
        else if (Set) Out <=SetValue;
else if (Enable) Out <=In;
        end
                end
 else begin:SS
                        always @ (posedge Clock) begin
if (Reset) Out <=ResetValue;
        else if (Set) Out <=        SetValue;
else if (Enable) Out <=In;
end
```

```
end

        end endgenerate
```

Similarly the Y-direction is checked and thus the Hsync and Vsync signals.

Thus the interface is established to the DVI display controller.

III. RESULTS

The best results were obtained and the output was checked on a DVI monitor. The interface gave an output of a white screen on the DVI display. The display was also used for checking the changing inverting patterns and was also successful. Thus DVI interface chip was successfully modeled. The applications was also modeled for video processing which still gave me the best results.

IV. APPLICATIONS

The interface can be directly used for interfacing video processing, image processing algorithms. The algorithm module must be connected to the inputs of the pixel feeder circuit. The video interface can be input into a VGA input video codec, the pixels must be processed, and fed into the pixel feeder again and then output to the DVI module where the pixel is output onto the screen. Thus this low cost algorithm is very efficient in performance and speed for video processing applications and can be used even without the XPS or EDK software.

V. CONCLUSION

The DVI interface was a proposed base work to establish an easy way to interface core image processing and video processing projects and was completed successfully. I have also checked the system with some interfacing of video and image algorithms such as clear video acquisition, edge detection on an image, edge detection on a continuous video stream - video processing, all of which gave me high quality results and a very cost effective technique. I guarantee that this method of DVI interface on an FPGA can boost the engineer's knowledge and performance in the field of image processing and segmentation and also in the field of video processing and also on a cost effective scale factor. The future plans of this project are to take this algorithm onto computer vision applications. The high cost of a vision software can be replaced on a hardware accelerator or an interface chip and all the processing could be done above this interface. This also doubles the speed of the CPU, increases power efficiency, and also is a cost effective technology. Thus efficient use of resources is established. I guarantee that this work can be readily used in all image processing and Video processing applications.

ACKNOWLEDGMENTS

I sincerely thank my department of Electronics and Communication Engineering, BMS College of Engineering, accredited by NBA, Bangalore-560019, for providing me with the Xilinx XUPV5LX110T Virtex 5 board. My special thanks to Prof. Harish V Mekali for helping me to procure this FPGA from my college authority.

REFERENCES

[1] MICHAEL D. CLIETTI, ADVANCED DIGITAL DESIGN OF VERILOG HDL, PHI LEARNING.

[2] LECTURES ON VLSI AND DIGITAL CIRCUITS OF IIT MADRAS , nptel.iitm.ac.in.

[3] XILINX ML505 DOCUMENTATION,USER GUIDE,APPLICATION NOTES- HIGH PERFORMANCE DDR2SDRAM INTERFACE,VIRTEX 5 LIBRARY GUIDE FOR HDL DESIGNS,VIRTEX 5 FPGA CONFIGURATION USER GUIDE.

[4] W.F LEE, VLIW MICROPROCESSOR AND HARDWARE DESIGN FOR ASIC AND FPGA.

[5] Moy C., Raulet M., "High-Level Design for Ultra-Fast Software Defined Radio Prototyping on Multi-Processors Heterogeneous Platforms", Journal on Advances in Electronics and Telecommunications – Radio Communication Series: special issue on Recent Advances and Future Trends in Wireless (Communications, Vol. 1, n° 1, pp. 67-85, April 2010[in press].)

[6] PONG.P.CHU, FPGA PROTOTYPING BY VERILOG EXAMPLES.

[7] CLASS LECTURES OF Prof. AJAY D KUMAR ON VHDL AND VERILOG DESCRIPTION LANGUAGES, BMS COLLEGE OF ENGINEERING, ACCREDITED BY NBA, BANGALORE-560078.

Author' biography

Asif Ahmad A S currently pursuing his 2nd year engineering degree in Electronics and Communication Engineering,BMSCE,accredited by NBA,NAAC(A grade),BANGALORE,INDIA. He has been the member of NXP laboratories,BMSCE(Council Of Innovation and Reasearch) has worked on "smart low power irrigation in remorte areas" which bagged 2nd place in ARM technical symposium.He also earned a "Certificate of Merit" for the national level robotics championship held in INDIA.

His interests are VLSI design and architectures for signal processing algorithms, efficient algorithms for signal processing filter designs, etc. Now he has been working on Efficient VLSI Architectures for image impainting, filter designs. He has worked on many projects such as "Sleep low power modes of ARM Cortex M0,M3",GSM communication and data transfer", "SPI protocol between GSM and ARM Cortex M4", etc. He has a wider vision of application and confines his works to a research degree.