An efficient architecture design for VGA monitor controller

Article · April 2011

DOI: 10.1109/CECNET.2011.5768261

2 authors, including:

Xuan-Tu Tran
Vietnam National University, Hanoi
92 PUBLICATIONS **559** CITATIONS

SEE PROFILE

An Efficient Architecture Design for VGA Monitor Controller

Van-Huan Tran, Xuan-Tu Tran
SIS Laboratory, University of Engineering and Technology, VNU Hanoi.
144 Xuan Thuy road, Hanoi 10000, Vietnam. Email: {huantv, tutx}@vnu.edu.vn

***Abstract*—In this paper, we present the design and implementation of an efficient hardware architecture for VGA monitor controllers based on FPGA technology. The design is compatible with PLB bus and has a high potential to be used in Xilinx FPGA-based systems. The ability to provide multiple display resolutions (up to WXGA 1280×800) and a customizable internal FIFO make the proposed architecture suitable for several FPGA devices. Furthermore, we have also offered a useful software library to enable the text mode feature. These highlight features have been validated through the demonstration of an application.**

I. Introduction

VGA (Video Graphics Array) has become a well-known standard interface in many embedded systems such as video surveillance systems, ATM machines, or video players. It provides a simple method to connect a system with a monitor for showing information/images, or for users to interact with the system. Depending on the needs of these applications, some systems may not require a high display quality. Therefore, VGA monitor controller, which is a logic circuit to control the VGA interface, can be easily realized by FPGA technology with a low cost and high flexibility. However, when moving to a higher display resolution, these FPGAs have to face with timing issues as well as logic resources overhead.

Several FPGA-based designs of VGA monitor controller have been released as commercial Intellectual Property (IP) cores [1], [2], [3] or open source cores [4]. They provide plenty of features and functionalities to be feasible in different running modes. Most of them can support multiple display resolutions and 24 bits color per pixel as the basic functions of a VGA monitor controller. Some additional functions, for example, the abilities to handle other color modes (e.g. RGBX5551, RGB232, etc.) and to program video timing of the design [2] might not commonly be used while they usually take more hardware resources. On the other hand, the design in [3] is more compact than the others, hence, it only works in a fixed display standard; or the lack in design [1] to support 64 bits data width makes it less efficient in usage.

This paper introduces an efficient architecture for VGA monitor controller with all necessary basic functions to work in both graphical mode and text mode. The efficiency of this design provides many choices for different FPGA devices, where system designers can select a proper display mode or configure the internal pixel buffer to be suitable with the application requirements. The design is intently integrated with Processor Local Bus (PLB) interfaces to be used in Xilinx FPGA-based systems.

The rest of this paper is organized as followed. Section II presents some basic ideas to design an efficient VGA monitor controller and the proposed architecture. A developed model to verify the design and its implementation are given in Section III. Section IV introduces a simple method to enable text display mode in the software driver. Section V presents an experiment as a case study of using VGA monitor controller. Finally, Section VI concludes some achievements of this work.

II. Architecture Design

A. Basic ideas and the needs of an efficient architecture

In order to display an image on screen, VGA monitor controller has to read every pixel data of the image while driving the color signals and synchronization signals of the VGA interface. All pixels are scanned in raster order at a frequency called pixel frequency. To ensure the visual quality, whole image will be re-drawn at a rate determined by refresh rate. The pixel frequency certainly depends on the display resolution and the refresh rate, better resolution or higher fresh rate will require higher pixel frequency. Some examples can be seen in Table I.

TABLE I
Display standard specification [5], [6]

Display standard	**Refresh rate** *(Hz)*	**Pixel frequency** *(MHz)*
VGA 640×480	60	25.175
SVGA 800×600	60	40.000
XGA 1024×768	60	65.000
WXGA 1280×800	60	83.460
WXGA+ 1440×900	60	106.47

The size of an image is usually larger than memory resources available on FPGA devices. For example, a 640×480 pixels at 24bpp image has the size of 912.6 Kbytes while the Spartan-3E family has only 81 Kbytes block RAM [7]. The image therefore can not be stored totally inside the design of VGA monitor controller. It should be held on an off-chip memory (e.g. SDRAM) and transferred into this VGA unit by small data blocks during the display time. For that reason, most of VGA monitor controllers have an internal FIFO memory to temporarily store these data blocks, e.g. [1], [3] or [4].

To handle the data transfer operations, there are two solutions could be addressed:

- The first solution is to attach a bus master interface to VGA unit to access data on the external memory. Hence the data transfer between memory and VGA unit is continuous, the bus master interface should support burst transfer mode to speed up this process. In general, this

method may give VGA unit more complex due to the appearance of the bus master interface.

- The second solution is to use a Direct Memory Access (DMA) unit to cooperate with VGA unit. The VGA unit therefore does not need to include any bus master interfaces. Every data transfer from the external memory to VGA unit is handled by the DMA. This DMA core can be reused from an existing IP delivered by Xilinx [8], so we can reduce the development time for VGA unit. A processor, however, has to initiate the operation of the DMA core and handle its interrupt during active time of VGA unit. Consequently, these issues will affect software applications.

In our design, we prefer the first solution to gain the portability and elegance for VGA unit, by using the PLB Master Burst [9] as the bus master interface.

Furthermore, one of the most challenges in the design of VGA unit is to estimate the size of FIFO memory within the core and the buffering strategy. Our goal is to minimize the size of FIFO while it has to ensure a good display quality (without flickering or fragmenting effect). To select a suitable FIFO, there are some factors should be considered, including pixel clock, operating system clock, data bus width, and the bus occupation of the system which contains VGA unit.

Obviously, the communication throughput depends on the operating system clock, data bus width, bus latency, and bus occupation. The throughput required by VGA unit is equivalent to the pixel clock. To successfully display, the first throughput must be higher than the second one. If the first throughput is much higher than the second throughput, we can use a small FIFO. Otherwise, we need a bigger.

In addition, to adapt with several SoCs on different FPGA devices, a good design of VGA monitor controller should provide multiple display standards, resizable data bus width and a customizable FIFO memory (which is able to change its depth or data width).

B. Overall architecture

To deal with those things above, we propose an efficient architecture for VGA monitor controllers as presented in Figure 1. It is composed of the following modules:

- PLB Master Burst, which supports burst transfer mode, is used for fast transfer data from an external memory to the FIFO module.
- PLB Slave Single serves the read/write operations from/to VGA monitor controller.
- VGA Registers is a set of registers to hold the control data and other information such as width, height or base address of an image. The accesses to these registers are performed via PLB Slave Single interface.
- VGA BRAMS (FIFO) is a dual-port RAM to temporarily store pixel data during display time. The depth of this FIFO is parameterized to be 64/ 128/ 256/ 512 bytes. The read and write pointer controllers are implemented within VGA Controller.

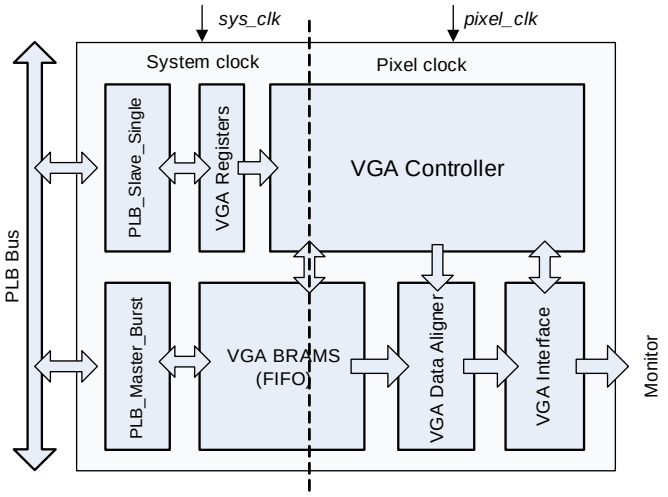


Fig. 1. VGA monitor controller architecture.

- VGA Data Aligner (VDA) aligns data between the output of the FIFO module and the input of the VGA Interface module.
- VGA Interface directly drives the color signals and synchronization signals of the monitor.
- VGA Controller generates video timing according to an expected display standard and controls the operation of other modules, exception of the two bus interface modules.

There are two clock domains in this design: system clock (*sys_clk*) and pixel clock (*pix_clk*). The system clock is the source clock for the side of bus interface while the pixel clock is used for the side of VGA interface. The pixel clock frequency is required according to the display standard (as provided by [5] or [6]). It can be driven by an on-chip clock generator (using Digital Clock Manager and PLL blocks of FPGA chips) or an off-chip clock generator.

Currently, the architecture can treat with 32 bits or 64 bits data width of the bus master interface without any functions of the Bus Width Adapter (a module inside the PLB Master Burst). There are different architectures for some modules to be feasible in both 32 bits mode and 64 bits mode, especially for the FIFO and the VGA Data Aligner. For the FIFO module, its data width is always equal to the data width of the bus master interface. Most of the changes take place in the VGA Data Aligner. The next section will give more details about its architecture.

C. VGA Data Aligner

From the overall architecture of VGA unit, the FIFO can output 4 or 8 bytes data per clock cycle (corresponding to the two modes, 32 bits and 64 bits data width of the bus master interface, respectively. However, the VGA Interface module can only push 3 bytes data (or three color components of a pixel) per clock cycle. The VGA Data Aligner therefore will aim to adjust the data flow between FIFO and VGA Interface. On the other hand, it also has to ensure the pixel order (first in, first out).

Basically, VGA Data Aligner is formed as the second FIFO of VGA unit. It has two different architectures for the 32 bits mode and the 64 bits mode, as depicted in Figure 2 and Figure 3. For the 32 bits mode, the FIFO has the size of 12 bytes and this size will be 24 bytes for the 64 bits mode. The read and write pointer controllers of this FIFO are controlled by the VGA Controller module. The read address and write address specified for each memory cell are shown in every square of these figures, where the read address is on the right and the write address is on the left. The output data width is always 3 bytes, which is equivalent to three color components RGB.

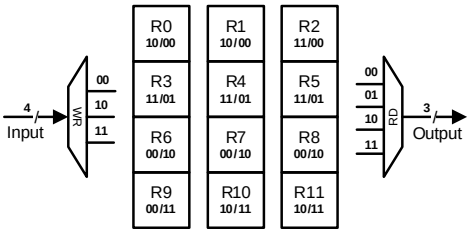


Fig. 2. VGA Data Aligner for 32 bits mode.

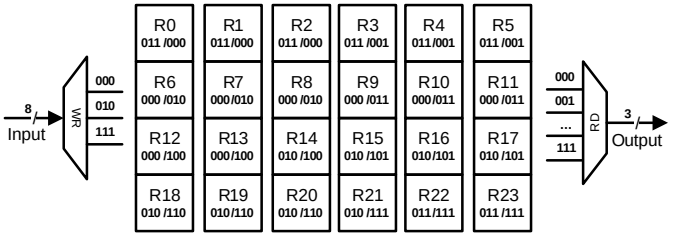


Fig. 3. VGA Data Aligner for 64 bits mode.

D. VGA Controller

Operations of those modules inside VGA unit are controlled by VGA Controller module. For a selected display standard, VGA Controller will determine horizontal and vertical timing parameters to generate corresponding video timing and to drive synchronization signals. During the display time, PLB Master Burst requests the bus to transfer data from external memory to the FIFO module. The addresses to access the memory and the burst length in every transfer session are calculated by VGA Controller. It also makes a handshake with the bus master interface to control read/write operations of the FIFO and VGA Data Aligner modules.

To handle these tasks, VGA Controller includes the following functional modules: Video Timing Generator, Video Address Calculator, FIFO Read/Write Pointer Controller, VDA Read/Write Pointer Controller, as shown in Figure 4. These modules are driven by a finite state machine (FSM).

III. Hardware Verification and Implementation

A. Verification model

The VGA monitor controller is implemented at RTL (VHDL) and simulated by ModelSim simulator. The verification model is created by using the IBM PLB Bus Functional Model (BFM) toolkit [10]. This package offers a set

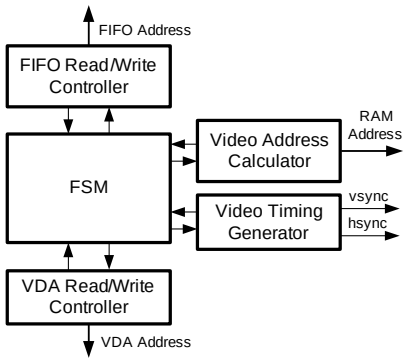


Fig. 4. Composition of the VGA Controller module.

of verification IPs, includes three main components: Master BFM (plbv46_master_bfm), Slave BFM (plbv46_slave_bfm) and Monitor BFM (plbv46_monitor_bfm) to simulate systems which contain the PLB bus (version 4.6). The model is depicted in Figure 5.

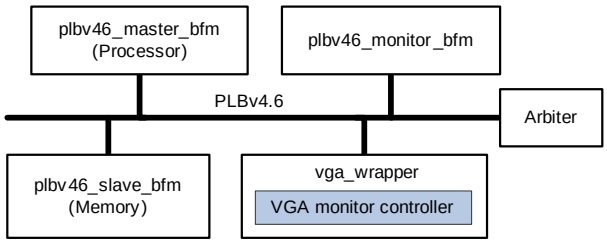


Fig. 5. Verification model for VGA monitor controller.

In this model, the Master BFM is a bus master core which functions as a processor to initiate and control the operation of VGA unit. The size of an image and its base address will be written to VGA unit by this processor. The Slave BFM is a bus slave core which functions as a memory. A sample image is stored in this memory and will be read by VGA unit during simulation. The monitor BFM checks for bus compliance or violations and notifies warnings or errors to the simulator.

For simulating, we pre-configure the VGA unit to run in a specific mode whose the FIFO depth is 256 bytes and display mode is SVGA 800×600. A small image is initialized in the memory. The processor then writes the image properties (width, height, base address) to the VGA unit and makes it active. Bus transactions are described by bus functional language. They will be translated to bus stimulus to run the simulation in ModelSim.

B. Implementation results

The VGA monitor controller has been implemented on different Xilinx FPGA devices with a FIFO depth of 256 bytes. To have a fair comparison of performance and resource utilization, we implemented our design and some related works (with minimized functionalities) on Xilinx Spartan3E-1600E. The result comparison is shown in Table II. Our design uses less logic and memory resources in comprared with the designs in [1] and [4] while it can achieve a higher performance than the others. Actually, the design [1] provides some additional

TABLE III
COMPARISON OF SUPPORTED DISPLAY STANDARD AND BUS TECHNOLOGY

	Display standard						Bus technology		
Design	**QVGA**	**VGA**	**SVGA**	**XGA**	**WXGA**	**Higher resolution**	**PLBv4.6**	**AHB**	**WISHBONE**
CAST [1]	✔	✔	✔	✔	✔	✔	-	✔	-
Oc_vga [4]	✔	✔	✔	-	-	-	-	-	✔
Xl_tft [3]	-	✔	-	-	-	-	✔	-	-
Proposed	✔	✔	✔	✔	✔	-	✔	-	

functions such as Integrated Test mode and Built-in Power Save mode; or [4] can support hardware cursor and color look-up-table. All these stuffs are the causes of logic consumption. The design [3] has a better result than ours but it only supports VGA standard, its logic circuit is therefore much simpler than the others. Table III shows the comparison of display standard and the bus technology supported by these works. The design [1] uses AMBA Advanced High-performance Bus (AHB), then it can work with higher resolutions than WXGA. The design [4] uses the open source WISHBONE bus while our design and [3] support CoreConnect PLB bus.

TABLE II
PERFORMANCE AND RESOURCE UTILIZATION BENCHMARKS ON XILINX SPARTAN3E (XC3S1600E-5-FG484)

Design	**Slices**	**LUTs**	**Block RAMs**	**Fmax** (*MHz*)
CAST [1]	1009	N/A	2	95
Oc_vga [4]	844	1302	2	130
Xl_tft [3]	519	521	1	141
Proposed	690	701	1	141

IV. ENABLING TEXT MODE

Displaying information as text is very useful in many applications. On the basic of the hardware design of VGA unit, we develop a set of methods in its software driver to enable text mode. In some context, it can be realized by hardware model, e.g. [11] and [12]. However, it is obvious that the implementing a character generator by software is much easier and more flexible than by hardware. For instance, we can easily change the attributes of text such as font table, size and color of characters, etc. In addition, the supporting a character generator in VGA unit may introduce more logic resources as well as memory overhead on FPGA devices.

To generate a character, a standard font (on Windows or Linux operating system) is converted to bitmap format. Every character is described as a matrix of pixels. Bit '0' presents a pixel in the background and bit '1' presents a pixel in the foreground of character. Figure 6 shows an example of character "A" with the size of 8×12 pixels. The image of character "A" is converted to the binary format and represented as an array in C language.

Based on these matrices, display a character can be handled by modifying the memory space which corresponds to the position of the character on the screen.

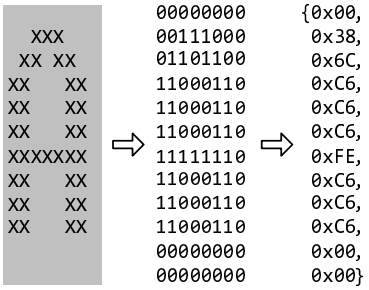


Fig. 6. The representation of character "A".

V. APPLICATION – A CASE STUDY

The VGA monitor controller can be used in several systems which have video output, such as portable video systems, video games, or digital cameras with video capabilities. In this section, we provide an FPGA-based system which uses VGA monitor controller as a functional module to display visual data in both graphics mode and text mode. This system plays a role as a remote camera system.

To build such system, we create a simple system-on-chip with the architecture presented in Figure 7. It is composed of: a PowerPC processor; Ethernet controller unit to communicate with other system via computer network; a memory controller unit, namely MPMC, to interface with an external memory; PS/2 unit to interface with a PS/2 keyboard; VGA unit; and UART unit. The system uses a channel of PLBv4.6 bus to interconnect these components. The VGA unit is configured to operate in XGA mode, the pixel clock is driven by an internal clock generator. In order to provide good display quality and also meet the timing constraints, we implement the system on Xilinx Virtex-4 ML410 development kit.

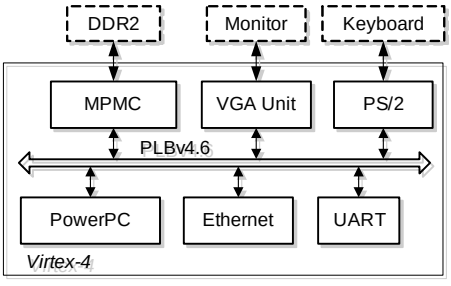


Fig. 7. FPGA-based system for the application.

A remote desktop PC, which connects to a camera, is used to capture images and send them to the FPGA board.

Communication between this PC and the FPGA board is performed on a LAN network. Figure 8 presents the sequence diagram of the whole remote camera system.

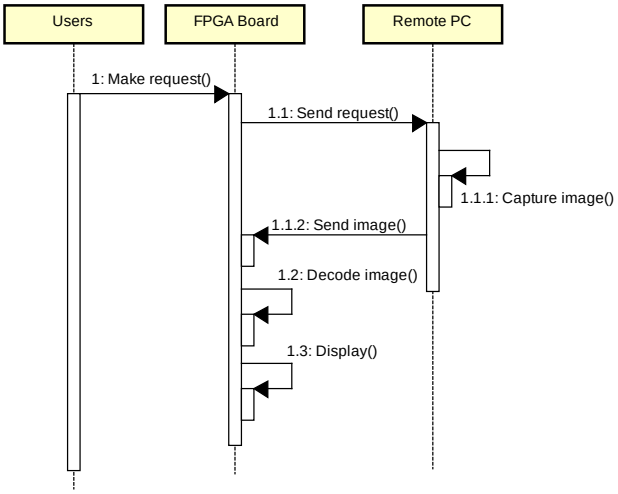


Fig. 8. Sequence diagram of the remote camera system.

To start, the users first make a capturing request on the FPGA board, and then the FPGA board dispatches this request to the PC. Whenever PC receives a proper request, it captures an image from camera, and then sends it back to the FPGA board. After successfully receiving the image, the testing board will decode and display it on the screen.

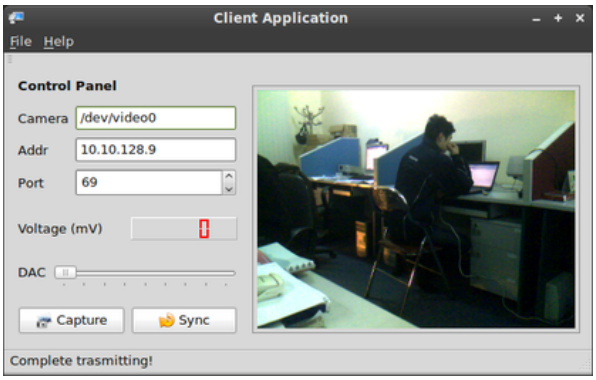
The software application on the FPGA board provides a command-line interface (CLI) to interact with users. Users thereby can invoke a capturing request by executing a specific command on this interface.

Figure 9 presents the result of the experiment, the captured image is shown on both PC application and the FPGA board. The contents (image and text) on the screen of the FPGA board are displayed in 24 bits color – XGA resolution (1024×768).

This application is just a case study for demonstrating how this VGA monitor controller can be used in a real system. Its result has already evaluated the functionalities of the VGA unit in both graphics mode and text mode. The design is also used in developing a system-on-chip platform as presented in [13].

VI. Conclusions

We have presented an efficient hardware architecture for VGA monitor controller which has a high potential to be used in Xilinx FPGA-based systems. The highlighted features such as multiple display resolutions supporting capability, customizable internal FIFO memory, 32/64-bit data bus width, independence to system clock and pixel clock…, make the design suitable for several FPGA devices and able to meet different requirements of targeted applications. In addition, a software library to enable text mode is also introduced. These useful features of the design have been validated through real-application demonstrations.



(a)

(b)

Fig. 9. Captured images on PC application (a) and FPGA application (b).

Acknowlegment

This work is partly supported by Vietnam National University, Hanoi (VNU) under the research projects, No. PUF.08.06 and No. QGDA.10.02 (VENGME).

References

[1] CAST. High Resolution Display Controller – Datasheet. Technical report, October 2010.

[2] Think Silicon Ltd. Think LCD Display Controller – Product Specification. Technical report, 2010.

[3] Xilinx. XPS Thin Film Transistor (TFT) Controller v1.00a. Technical report, Product specification, July 2008.

[4] Richard Herveille. VGA/LCD Controller v2.0. Technical report, 2009.

[5] VESA. Monitor Timing Specification Version 1.0 Rev. 0.8. Technical report, September 1998.

[6] SECONS Ltd. VGA Signal Timing. Technical report, 2008.

[7] Xilinx. Spartan-3E FPGA Family: Data Sheet. Technical report, August 2009.

[8] Xilinx. XPS Central DMA Controller (v2.00b) – Product Specification. Technical report, April 2008.

[9] Xilinx. PLBV46 Master Burst (v1.01a) – Product Specification. Technical report, May 2008.

[10] Xilinx. BFM Simulation in Platform Studio – User Guide. Technical report, 2009.

[11] Guohui Wang, Yong Guan, and Yan Zhang. Designing of VGA Character String Display Module Based on FPGA. In *International Symposium on Intelligent Ubiquitous Computing and Education(IUCE)*, pages 499–502, 2009.

[12] Javier Valcarce Garcia. Monochrome Text-Mode VGA Video Display Adapter. Technical report, 2009.

[13] Van-Huan Tran and Xuan-Tu Tran. CoMoSy: a Flexible System-on-Chip for Embedded Applications. *Journal for Research, Development and Application on Information and Communication Technology*, 2, 2011.