A FPGA Paint Brush Application

Vince Edwards
Vince.Edwards@selu.edu

Michael Courtney
Michael.Courtney@selu.edu

Kuo-pao Yang
kyang@selu.edu

Southeastern Louisiana University
Hammond, Louisiana 70402, USA

Abstract

This paper describes a Field Programmable Gate Array (FPGA) paint program similar to the one found on the Windows operating system. This paint brush application uses a modern educational kit that has recently been available for Computer Science hardware courses. The educational package employs state-of-the-art technology in both hardware and software. This new technology is currently being used in many universities and many electronic commercial products. With an integrated design environment and its reconfigurable capabilities on the board where hardware can be changed on the fly, we implement the system-on-chip application and immediately see execution results. Further development of this program can lead to a more complex and sophisticated application.

Keywords: Hardware/software co-design, Embedded computing education, FPGA-based design, System-on-chip

1. INTRODUCTION

We have developed a FPGA-based program similar to the Microsoft Paint application on the Windows operating system. This FPGA paint brush application is displayed on a VGA monitor and uses a mouse to draw different figures, to fill colors, and to clear drawing area. The VGA monitor and the USB mouse are connected to a reconfigurable education board. This project is hardware and software co-design accomplished using the custom hardware and its software toolkit.

Program execution is conventionally accomplished by one of two primary methods. The first is to use hardwired technology, an Application Specific Integrated Circuit (ASIC) to perform operations in hardware. ASICs are designed to perform specific sets of instructions for accelerating a variety of applications, and thus they are very fast and efficient. However, the circuits cannot be changed after fabrication. The circuits need to be redesigned and re-fabricated if any parts of the application require modification.

The second method is to use software-programmed microprocessors, which execute a set of instructions to perform computations for general purposes. By interpreting the software instructions, the functionality of the system is altered without changing any hardware. However, the downside of this flexibility is that the performance degrades. The processor needs to read each instruction from memory, decode its meaning, and then execute it, so the performance is significantly poorer than that of an ASIC.

Reconfigurable computing is intended to fill the gap between hardware and software methods, achieving potentially much higher performance than microprocessors, while maintaining a higher level of flexibility than ASICs. Reconfigurable devices, field programmable gate arrays (FPGAs), contain arrays of computational logic blocks whose functionality is determined through multiple programmable configuration bits. Custom digital circuits are mapped into the reconfigurable hardware to form the necessary circuit. A variety of applications that have been shown to exhibit significant speedups using reconfigurable FPGA hardware include data encryption (Elbirt 2000), automatic target recognition (Rencher 1997), error detection (Atieno 2006), string pattern matching (Weinhardt 1999), boolean satisfiability (Zhong 1998), data compression (Huang 2000), and genetic algorithms (Graham 1996).

This project is based on an existing one at Cornell University (ECE 2008) and then we made a number of fine adjustments to fit it into our environment. In this project, an interface is created between the hardware and software by taking full advantage of the system-on-a-programmable-chip (SOPC) builder, which provides features to ease writing software and to accelerate system simulation. We then developed and altered the application using the Integrated Development Environment (IDE), which comes with the software package. The mouse is integrated and gives the user the ability to draw or paint, or basically just manipulate all functions of the program. The monitor is used to display the functions that are being transmitted from the DE2 board onto the screen.

2. EQUIPMENT AND SOFTWARE PACKAGE

The DE2 package that we used includes the hardware board and software (Yang 2008). The Cyclone II chip on the DE2 board is reconfigurable and is currently used in many electronic commercial products. A block diagram of the DE2 board is shown in Figure 1. The Cyclone II is a SRAM based FPGA device with 475 I/O pins. All connections are made through the Cyclone II device, and thus developers can configure the FPGA through the USB blaster to implement any system design. The FPGA will retain this configuration as long as power is applied to the board. The EPCS16 chip provides non-volatile storage of the bit stream, so that the information is retained even when the power supply to the DE2 board is turned off. When the board's power is turned on, the configuration data in the EPCS16 device is automatically loaded into the Cyclone II.

The DE2 board offers a rich set of features as shown in Figure 1 and Figure 2. This board contains standard connectors for microphone, line-in, line-out (16-bit audio CODEC), video-in (TV Decoder), and VGA (10-bit DAC); these features can be used to create CD-quality audio applications and professional-looking video. It is also possible to connect other user defined boards to the DE2 board by means of two expansion headers.

The Altera Quartus II software is comprised of an integrated design environment that includes everything from design entry to device programming. Developers can combine different types of design files in a hierarchical project. The software recognizes schematic capture diagrams and hardware description languages such as VHDL and Verilog. The Quartus II Compiler analyzes and synthesizes the designed files and then generates the configuration bit stream for the assigned device. It then downloads the configuration bit stream into the target device via the USB connection. System developers can simulate the designed component, examine the timing issues related to the target device, and modify the I/O pin assignments before the configuration is downloaded onto the chip on the DE2 board. The Quartus II computer-aided design tools work with both the chips on the DE2 and other Altera devices. NIOS II processors implement a 32-bit instruction set based on a general-purpose RISC architecture (Yiannacouras 2005, 2006). Because it is a soft-core configurable processor (Sheldon 2006), FPGA developers can choose from a myriad of system configurations, picking the best-fit CPU core (Sharma 2004), selecting processor peripherals, and meeting performance goals.

The software development environment of the NIOS II IDE uses a standard GNU GCC compiler tool chain and Eclipse IDE to compile projects. The NIOS II IDE provides a helpful interface for building complex embedded programs. System designers can create their own custom peripherals that can be integrated with NIOS II processor systems. For performance-critical systems that spend most CPU cycles executing a specific section of code, it is a common technique to create a custom peripheral that implements the same function in hardware. Using this approach, performance is improved significantly. The processor is free to perform other functions in parallel while the custom peripheral operates on data.

3. RESULTS AND DISCUSSIONS

This project focuses on the development of the FPGA paint brush application shown in the Figure 3. It downloads the configuration bit streams from the host computer onto the Cyclone II FPGA through the USB blaster port. An existing VGA controller on the board handles a VGA monitor with the resolution of 640x480. The NIOS II software in this project is used to interface the ISP1362 USB drive for the mouse and also interacts with the VGA adapter implemented in the hardware. The movement of the mouse cursor is sent to the VGA adapter. The software development module sends coordinates and colors to be displayed at particular pixels on the VGA screen. The mouse is used to control all functions of the application such as drawing lines or circles, etc.

This software module runs the various algorithms for drawing the various shapes and sends these values to the SRAM. The VGA controller reads from the SRAM and displays on the VGA screen. The complete functionality of the paint brush application and the realization of different figures and shapes on the VGA screen was replicated using various graphics algorithms. Bresenham's Algorithm (Harris 2004) is used for line and circle drawing. The Edge Fill Algorithm (Smith 1979) is used for detecting the edges of a closed figure and filling color inside its boundary. The Fencing Technique (Chalmers 2003) is applied to reduce the scan area. This decreases the number of pixels being inverted for every polygon color fill and thus achieves better speed and performance. The functionalities provided by this paint brush application are drawing a point, line, square, circle, polygon, clearing the pixels using the eraser, spray paint, fill color, pick color, clear drawing area and various color options in a color pallet at the bottom of the screen.

Now that we have completed our implementation, our FPGA drawing program works as the follows. First, the user chooses a specific function such as a line or circle, and then chooses the color at the bottom of the screen. The default color is black and the color remains the current color at the bottom left corner of the screen until another color is chosen. Next, the user selects a starting point on the canvas to start the shape and then an endpoint to complete the shape.

We discovered some drawbacks to the Altera system. Analysis of the memory and CPU usage of the program revealed that Altera's IDE takes a lot of time and consums a lot of memory in order to compile the code and transfer downloadable bit streams, when even a small change is made to the program. With this being the first year FPGA was introduced to our IS department we encountered issues adapting to the learning curve of the software. First issue was finding the right drivers and getting them to work with the board. There where specific instructions on obtaining the updated drivers located on the Altera website and downloading it to the system. We could not get the computers to recognize the USB blaster from the DE2 board. We finally got that to work. Then, we had second issue with the VGA monitor not showing anything except for a white screen with very faint lines going up and down. It started to work after we slightly modified the VGA controller from the DE2 default design settings.

The most important thing we have learned from this project is the Principle of Equivalence of Hardware and Software (Null 2006). We choose this FPGA Paint Brush project because we are interested in learning how to work with hardware and understanding the difference between this hardware approach and Microsoft Paint. This project encourages student research and development. We also realize that using FPGA gives a concrete structure to abstract lessons. IS students may also discover after programming the board, that the FPGA can be reprogrammed at any time which can reduce development cost since the instructions can be programmed at any time. We now believe this principle: anything that can be done with software can also be done with hardware, and anything that can be done with hardware can also be done with software.

4. CONCLUSION

This project introduces a FPGA Paint Brush application similar to Microsoft Paint. It is based on the existing project at Cornell University and then we made a number of fine adjustments to fit it into our environment. Altera's Quartus II and NIOS II IDE are employed to compile the implementation, simulate the results, program and control the Cyclone II FPGA chip through USB ports on the DE2 board. This project broadens the traditional knowledge of our IS majors, since we usually do not have opportunities to work with real hardware. Though the paint program that we are using is just a mock version of the original Microsoft Paint features, the implementation of this program helps our IS students to understand and appreciate the work that goes into larger, more complex hardware solutions.

5. REFERENCES

Atieno, L., Allen, J., Goeckel, D., Tessier, R. (2006) "An Adaptive Reed-Solomon Errors-and-Erasures Decoder." Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays, Monterey, California, 150-158.

Chalmers, A., Cater, K., Maflioli, D. (2003) "Visual attention models for producing high fidelity graphics efficiently." Proceedings of the 19th Spring Conference on Computer Graphics (SCCG'03), 39-46.

Elbirt, A., Paar, C. (2000) "A FPGA Implementation and Performance Evaluation of the Serpent Block Cipher." ACM/SIGDA International Symposium on FPGAs, 33-40.

Electrical and Computer Engineering (ECE) at Cornell University (2008) "Advanced Microcontroller Design and system-on-chip." http://www.ece.cornell.edu

Graham, P., Nelson, B. (1996) "Genetic Algorithms in Software and in Hardware—A Performance Analysis of Workstations and Custom Computing Machine Implementations." IEEE Symposium on FPGAs for Custom Computing Machines, Napa, CA, 216-225.

Harris, M., Reingold E. (2004) "Line Drawing, Leap Years, and Euclid." ACM Computing Surveys (CSUR), vol. 36, no. 1, 68-80.

Huang, W., Saxena, N., Mccluskey, E. (2000) "A Reliable LZ Data Compressor on Reconfigurable Coprocessors." IEEE Symposium on Field-Programmable Custom Computing Machines, 249-258.

Null, L., Lobur, J. (2006) "The Essentials of Computer Organization and Architecture, Second Edition." Jones and Barlett Publishers.

Rencher, M., Hutchings, B. (1997) "Automated Target Recognition on SPLASH2." IEEE Symposium on Field-Programmable Custom Computing Machines, 192-200.

Sharma, A., Compton, K., Ebeling, C., Hauck, S. (2004) "Exploration of Pipelined FPGA Interconnect Structures." Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, Monterey, California, 13-22.

Sheldon, D., Kumar, R., Vahid, F., Tullsen, D., Lysecky, R. (2006) "Conjoining Soft-core FPGA Processors." Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided Design, San Jose, California, 694-701.

Smith, A. (1979) "Tint fill." ACM SIGGRAPH Computer Graphics, vol. 13, no. 2, 276-283.

Weinhardt, M. and Luk, W. (1999) "Pipeline Vectorization for Reconfigurable Systems." IEEE Symposium on Field-Programmable Custom Computing Machines, 52-62.

Yang, K.P., Beaubouef, T., (2008) "A Field Programmable Gate Array Media Player for Realmedia Files." Journal of

Computing Sciences in Colleges, vol. 23, no. 6, June, 2008, 133-139.

Yiannacouras, P., Rose, J., Steffan, G. (2005) “The Microarchitecture of FPGA-based Soft Processors.” Proceedings of the 2005 international conference on Compilers, architectures and synthesis for embedded systems, San Francisco, California, 202-212.

Yiannacouras, P., Steffan, G., Rose, J. (2006) “Application-Specific Customization of Soft Processor Microarchitecture.” Proceedings of the 2006 ACM/SIGDA 14th Int. Symposium on Field Programmable Gate Array, Monterey, California, 201-210.

Zhong, P., Martinosi, M., Ashar, P., Malik, S. (1998) “Accelerating Boolean Satisfiability with Configurable Hardware.” IEEE Symposium on Field-Programmable Custom Computing Machines, 186-195.

Appendices

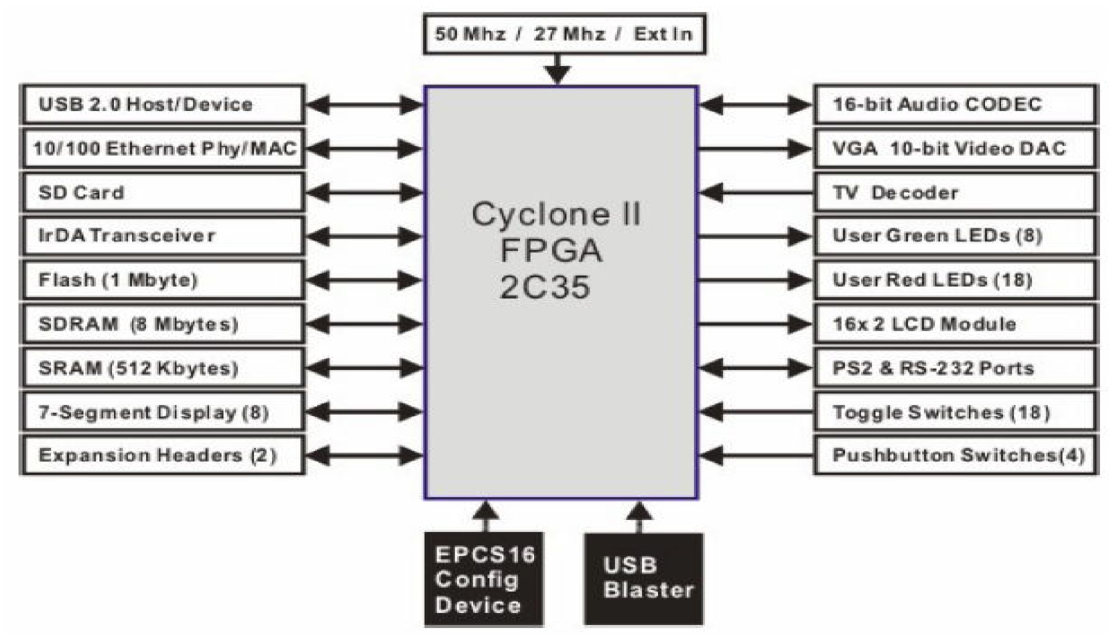

Figure 1: Block Diagram of the DE2 Board

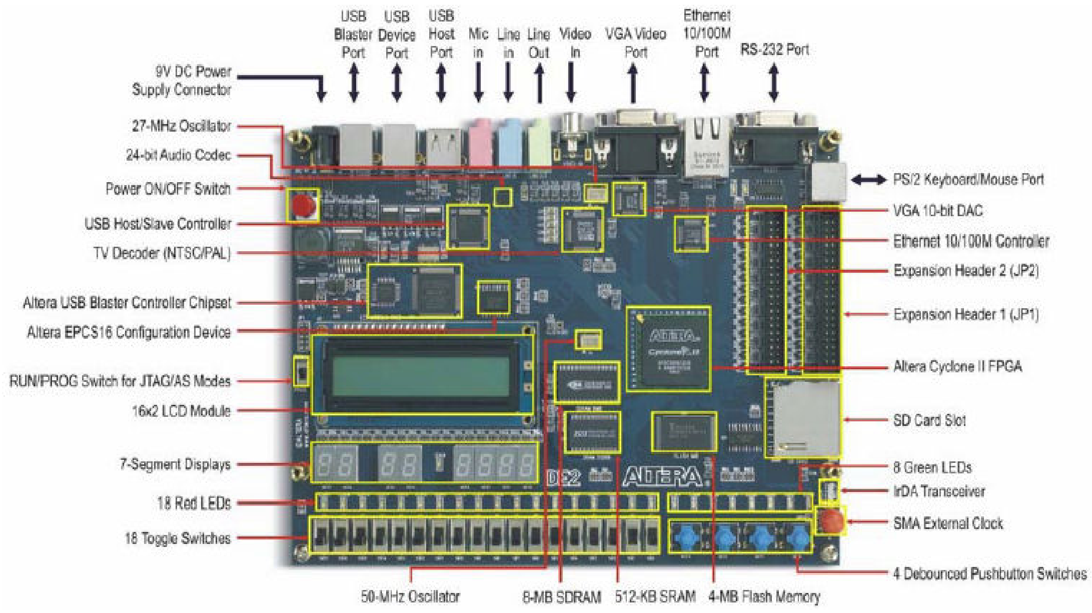

Figure 2: Layout and Components of the Altera DE2 Board

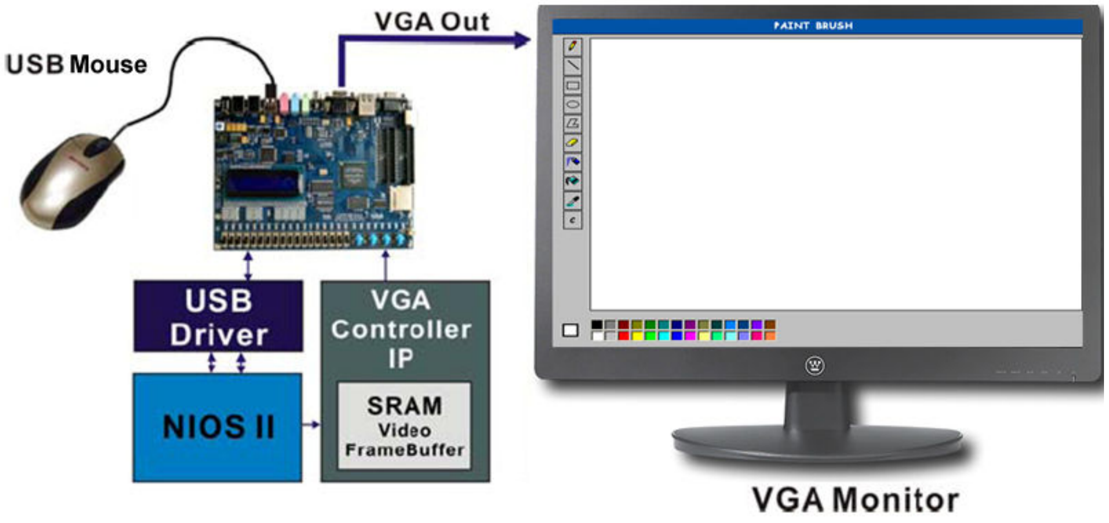

Figure 3: FPGA Paint Brush Application