THE DESIGN OF A FPGA BASED RECONFIGUARABLE IP VIDEO SURVEILLANCE AND CONTROL SYSTEM

Nam Pham Ngoc[1] **and Minh Tien Nguyen**[1]

[1] Department of Electronics and Computer Engineering, Hanoi University of Science and Technology, Ha Noi, Vietnam, e-mail: pnnam-fet@mail.hut.edu.vn

Received Date: July 11, 2011

Abstract

This paper presents the HW/SW co-design of a reconfigurable IP video surveillance and control system based on low cost Xilinx Spartan 3E FPGA family. The system allows users to observe video images captured by a camera via any standard browser and control remote devices. Our design focuses on the design of the HW base line JPEG encoder on FPGA and the design of a SW web server running on uCLinux operating system ported on MicroBlaze, a soft-core microprocessor embedded on the FPGA. With the reconfigurability of FPGA, the system can be upgraded easily to support multiple cameras and more control functionality without incurring high additional design cost.

Keywords: Embedded system, FPGA, HW/SW co-design

Introduction

Internet Protocol (IP) video surveillance and control systems have been used widely in many applications such as in intelligent building management systems (IBMS) or in remote controlling of base transceiver station (BTS) in wireless communication etc. In a typical scenario in these applications, a user uses a PC or a Laptop or even a mobile phone to login into the remote system at her home via a web browser. She can then view the images captured by cameras and may activate controls such as turning on or off lights or fans with verification of the results via cameras. The user may also view sensors such as temperature, water flow, or electrical usage via a simple user interface on the browser.

Due to the dynamic nature of user requirements in surveillance and control systems such as the number of cameras and the number of devices that need to be controlled, flexibility becomes a very important design metric in the system design. With its reconfigurable property, Field Programmable Gate Array (FPGA) provides a flexible, short time to market and low cost solution for designing and implementing such a system in comparison with ASIC and software based systems [10, 11]. There has been a number of work exploiting FPGA advantages to design remote control systems. In our previous work [2], we presented a cost-effective and flexible design of a remote control system using FPGA and microcontrollers technologies. This system did not have video surveillance feature. In [3] a conceptual design of a remote controller for digital cameras was proposed. However, the design relied on digital cameras outputting compressed images which are more expensive than analog cameras or CMOS image sensors outputting uncompressed raw images. The authors of [4] and [5] presented the designs of reconfigurable high resolution network cameras using CMOS image sensor. These designs used a processor made by Axis Communications running Linux operating system to realize web server functionality and a Xilinx FPGA chip to perform the compression of the images taken from the CMOS sensor. In this paper, we propose a one chip solution to perform both web server functionality and image compression using embedded design flow provided by

Xilinx [1]. Integrating both functionalities onto a single FPGA leads to the reduction on the chip count, PCB cost, inter-chip communication overhead and power consumption.

The platform we use to demonstrate our design consists of a Spartan 3E1600 main development board and a custom designed board to control the various devices. The main FPGA board is connected to a low cost analog camera via an extended board.

The rest of the paper is organized as follows: In Section 2, the system architecture is described. The hardware (HW) design of the JPEG encoder is given in Section 3. Section 4 presents software (SW) design and in Section 5, we discuss the results. Finally, the paper is concluded in Section 6.

System Architecture

The architecture of the FPGA based reconfigurable IP video surveillance and control system is shown in Figure 1. The system architecture consists of a FPGA chip connected to other components including memories, external boards and I/O devices. The control board is a custom designed board in charge of receiving control signal form the FPGA to control the I/O devices such as lights, fan etc. The extended board consists of a TVP5150 chip which converts the analog video signal from the camera to YCbCr digital format. This board is not needed when using a CMOS image sensor instead of an analog camera. The hardware platform on the FPGA is designed using Xilinx Platform Studio (XPS) tool. The heart of the hardware platform is a 32 bit soft-core RISC microprocessor MicroBlaze provided by Xilinx. All software components of the system including the uCLinux operating system [6], TCP/IP protocol stacks and web server application will be executed on the MicroBlaze microprocessor.

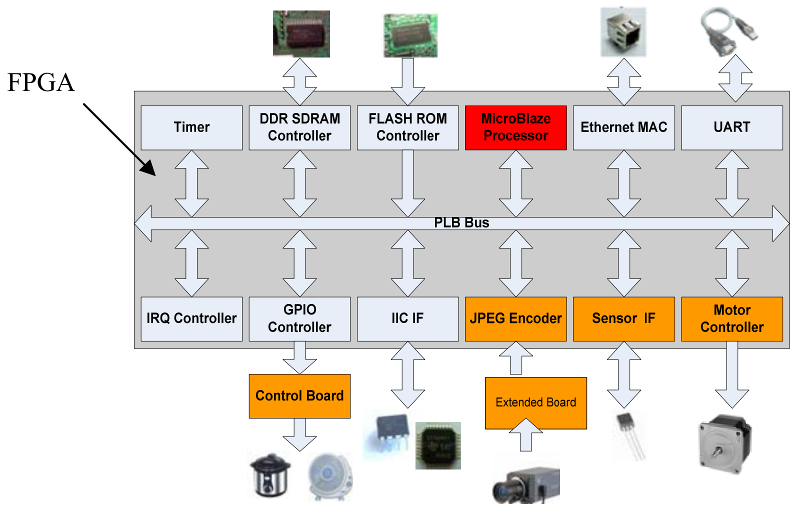


Figure 1. System architecture

The MicroBlaze is connected to other hardware components via Processor Local Bus (PLB). All the on-FPGA hardware components except for the JPEG Encoder, sensor Interface (IF) and Motor controller components are provided by XPS tool. The JPEG encoder is a customer Intellectual Property (IP) that receives the 4:2:2 ITU format data stream from the extended board and compresses it following the base line JPEG compression algorithm [7]. The sensor IF is also a customer IP that interfaces sensors with the FPGA. In our system, we design a controller for temperature sensor DS1820 in the sensor IF component. Finally, the motor controller is another customer IP that receives the signals from the MicroBlaze and controls the viewing angle of the camera. If in the future we want to add more sensors to the system, the sensor interface blocks can be easily added thanks to the reconfigurable feature of the FPGA.

The system functions as follows. The images captured by the analog camera are converted to YCbCr digital format by the extended board and input to the JPEG encoder. The JPEG encoder compresses the raw image data into the JPEG format with the compression ratio of a factor 30. The compressed images are sent to the MicroBlaze to put into Motion JPEG (MJPEG) stream which is then transmitted via the Ethernet connection to the user browser. The user can view the images and see the status of different remote devices and other remote information such as room temperature. The user may control the viewing angle of the camera via a user interface on the browser. The camera viewing angle control signals are sent back to the MicroBlaze and forwarded to the motor controller block to control the motor attached to the camera.

Hardware Design

This section presents the design of the most important customer and also the most computation intensive IP block, which is the JPEG Encoder. This block and other customer IP blocks are designed using VHDL. The block diagram of the JPEG encoder is shown in Figure 2.

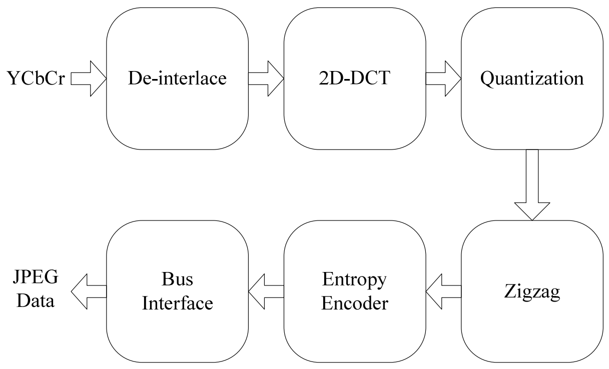


Figure 2. Block diagram of the JPEG encoder

The de-interlace block receives data stream in 4:2:2 ITU format from the video decoder TVP5150 and stores 8 scan lines of data stream to its buffers. It gets the data stored in the buffer and gathers the same component of luminance or chroma by 8x8 pixels block. The detailed structure of the de-interlace block is depicted in Figure 3. The Identifier is used to determine the data part in 4:2:2 ITU data stream. The FSM Controller controls the signals to read data from and write the data to two separated buffers which are implemented using dedicated block RAM resource of Xilinx FPGA. Because of the continuous data stream, the de-interlace block needs two Block RAM (BRAM) buffers, one buffer for reading and one for writing.

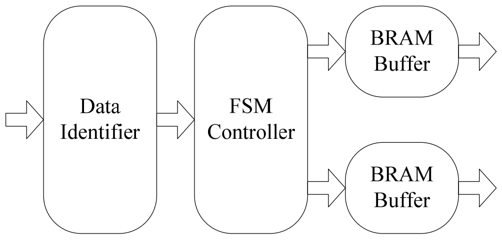


Figure 3. Components of the de-interlace block

The 2D-Discrete Cosine Transform (2D-DCT) block is formed from two 1D-DCT blocks, each of which contains a writing/reading controller to control two 64-word BRAM buffers which allows one reading and one writing simultaneously. Data from the buffer is sent to the multiplier unit and multiplied with the coefficient from an approximate coefficient array to calculate the DCT output for each input data [8].

The quantization module takes 12 bit data from the 2D-DCT block, uses multiplication by 12 bit together with shift right 12 bit instead of division by 8 bit to produce 8 bit data at the output. The zigzag module arranges the data stream from the quantization module and sorts the data values in low to high frequency order.

The Entropy Encoder block handles the data stream from zigzag block at every clock cycle. The first part of this block is the Delta DC block, which calculates the difference between two continuous DC values of two continuous data block for each data component. The second part is Run Length Encoder (RLE) which finds the run length step for the similar data values of data block for each data component. Data stream is then transmitted to Length coder and Value coder. At Length coder, each data value is coded by an appropriate Huffman code which is stored in a Huffman coding table [9]. The Combiner links two parts of the code word into a single word and stores it into a buffer. Because the lengths of code words are different, the buffer gathers all of the coded words and sends valid signal out to write the coded data stream to the next block whenever the coded data stream reaches 32-bit length.

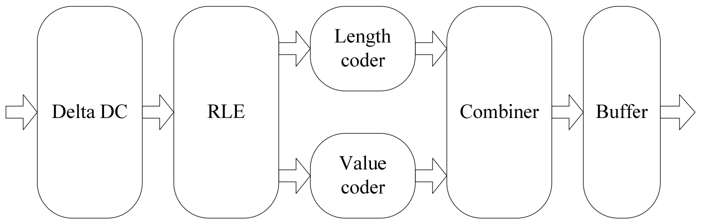


Figure 4. Block diagram of the entropy encoder

Software Design

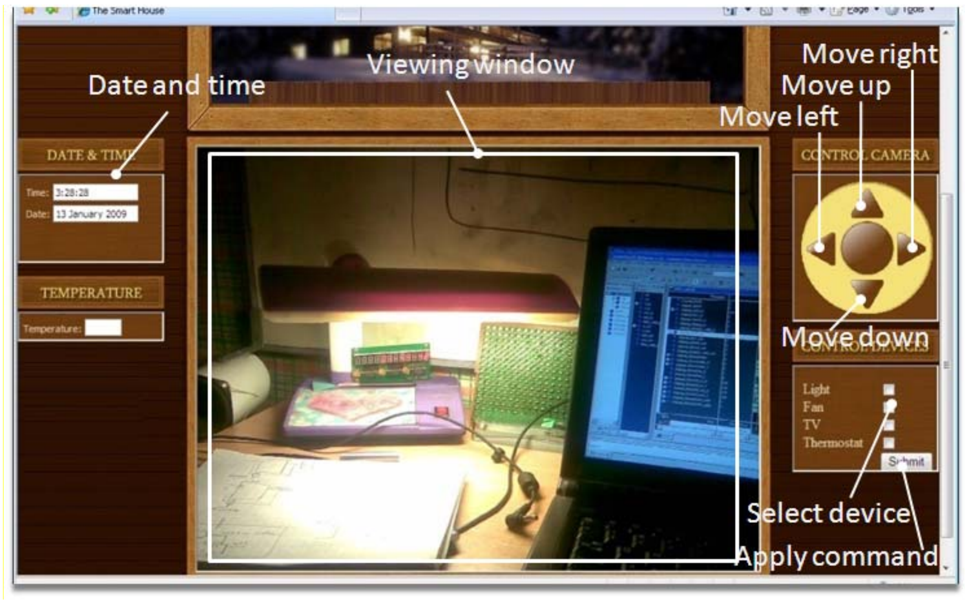
The embedded software in our design is running on uClinux operating system. The use of a Linux based OS eases reuse of existing open-source software modules, which allows us to design the full system in a short time.

The main tasks of uClinux OS are making MJPEG data stream, controlling devices and providing web-server service for the system. For making a MJPEG picture data stream, uClinux runs the MJPEG CGI (Common Gateway Interface) program. When the hardware JPEG encoder generates an interrupt to inform the OS that a JPEG data block is available, the MJPEG CGI program receives the JPEG data block and sends it to the user’s browser.

Every time a JPEG frame is compressed completely, the hardware JPEG encoder generates another interrupt to inform the CGI program to insert the boundary section within the data stream of MJPEG. The boundary section has the following format:

```
HTTP/1.0 200 OK
Connection: Close
Server: FPGA
Content-Type: multipart/x-mixed-replace;        boundary=--myboundary
--myboundary
Content-Type: image/jpeg
JPEG DATA STARTS HERE
```

In this format, the first two lines are the connection establishment procedure, the third line is the server name, the fourth line is the content type of data and boundary name declared. Another CGI program is the program that controls the devices. When the user clicks on a button on the webpage, this CGI program accesses the driver of corresponding device and writes or reads the data to or from the device through an operation file. All of CGI programs run on the web server service of uClinux OS. The web server service provides a webpage that includes windows to display the video in MJPEG format, buttons to control the devices and textboxes to display the status of devices as shown in Figure 5.



Time: 3:28:28
Date: 13 January 2009

Figure 5. A snapshot of the user webpage

Implementation Results

The whole system with one camera has been implemented successfully on a single Spartan3E1600 FPGA with the resource usage listed in Table 1.

Table 1. Synthesis Result of the System with One Camera on Xilinx Spartan3E 1600

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	7,013	14,752	48%
Number of BRAMs	35	36	97%
Number of Multipliers	18	36	50%

We also measured the size of the complete software including the operating system and the application software. The measured size is 2.8 MB.

The experimental results show that the JPEG encoder processes each 8x720 pixel block of a frame in 13781 clock cycles, which consists of buffering time and processing time. The total encoding time for the whole frame of 720x625 pixels is thus 1076640 clock cycles, which is about 39.5 ms at 27 MHz. The reason for using a clock frequency of 27 MHz is to synchronize the JPEG encoder with the video decoder TVP5150 running at 27 MHz. With the processing time of 39.5 ms, the JPEG encoder can ensure a frame rate of 25 fps.

For other picture resolutions of HxV pixels, the hardware resources in terms of number of slices and number of multipliers used by our JPEG encoder will not change. The number of BRAMs used by the two buffers of the De-Interlace block, however, will change and will affect the total number N of BRAMs used by the system. N can be calculated using the following formula:

$$N=\frac{H(pixels\ per\ line)*8(lines)*16(bits\ per\ pixel)*2(buffers)}{18(Kbits\ per\ BRAM)}+25$$

where 25 is the number of BRAMs used by the system excluding the two buffers of the De-Interlace block.

Multiple Camera Support

In order to support multiple cameras simultaneously, we have extended our design as shown in Figure 6.

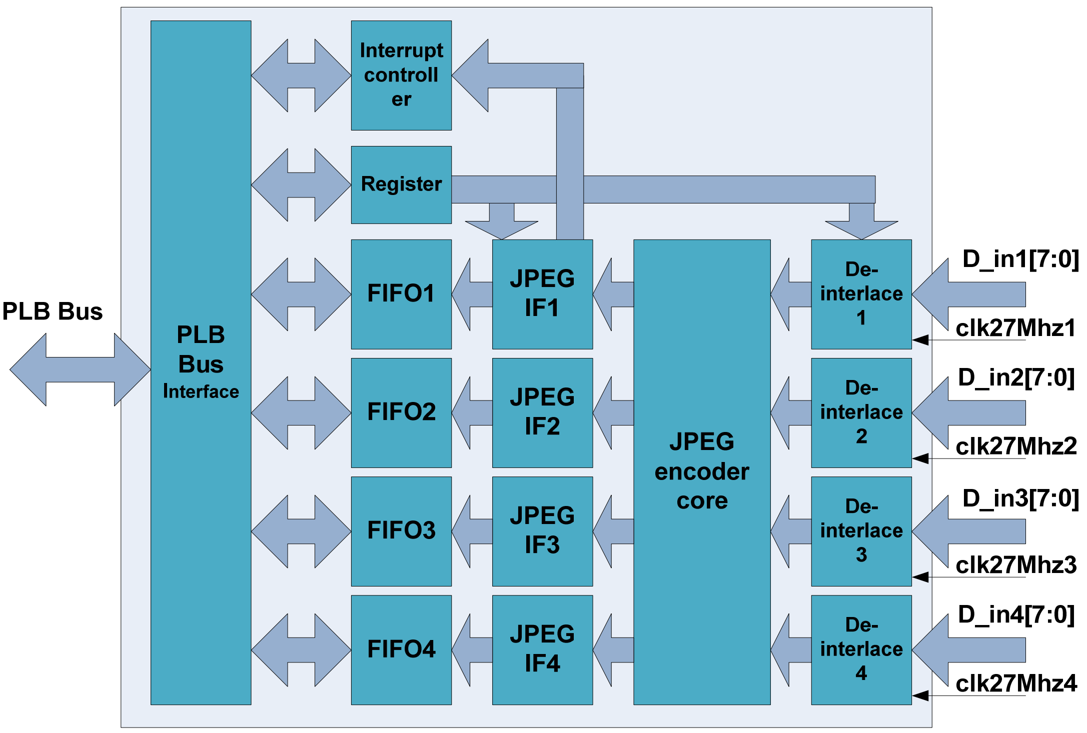


Figure 6. Jpeg encoder for multiple cameras

In Figure 6, video signals from four cameras are de-interlaced by four separate de-interlace blocks and then time-multiplexed and encoded by a shared JPEG encoder core. The encoded video streams are then put to separate FIFO buffers. Thanks to sharing of the JPEG encoder core, the system which connects to four cameras requires small additional resources: 1 Block RAM for FIFOs, 600 slices for de-interlaces and 30 of slice for JPEG IFs. Thus the hardware resource usage for the 4 cameras system is: 7643 slices, 36 BRAMs and 18 Multipliers, which can be fit on a single Spartan3E1600 FPGA chip. We can see that with the proposed encoding scheme, we can better utilize the hardware resources. For a bigger number of cameras, our design can be migrated to a bigger FPGA chip without much additional effort.

Conclusions

In this paper we described the design of a FPGA based IP video surveillance and control system in both hardware and software aspects. The system was successfully designed and implemented on a single FPGA chip using Xilinx embedded design flow. The system implements video streaming functionality using Motion JPEG standard with the JPEG encoder implemented in hardware. In our future work, we will use a better video compression standard such as H.264 and use data encryption to obtain a secure connection between the server and the client.

References

[1] XILINX, (n.d.), 2011, [Online]. Available: http://www.xilinx.com/ [Accessed: April 2011]

[2] H.H. Dung, P.N. Nam, and H.A. Dung, “A cost-effective and flexible design of a security system using CPLD/FPGA and microcontrollers technologies,” Paper presented at *The First International Conference on Communication and Electronics (ICCE’06)*, Hanoi, October 10-11, 2006.

[3] D. Wilburn, D. Hafeman, A. Rogers, and H. Yu, "*Using Spartan-3 FPGAs as Low-Cost Controllers for Remote Digital Cameras*," 2003, [Online]. Available: http:// www.xilinx.com [Accessed: April 2011]
[4] D. Verkest, D. Desmet, P. Avasare, P. Coene, S. Decneut, F.Hendrickx, T. Marescaux, J.-Y. Mignolet, R. Pasko, and P. Schaumont, "Design of a secure, intelligent, and reconfigurable web cam using a C based system design flow," In: *The 35th Asilomar Conference on Signals Systems & Computers Pacific Grove*, CA, pp. 463-467, 2001.
[5] A. Filippov, "Reconfigurable high resolution network camera," In: *The 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM03)*, pp. 276-277, 2003.
[6] Petalogix, *Petalinux User Guide*, [Online]. Available: http://developer.petalogix.com [Accessed: April 2010]
[7] ITU, *Information Technology-Digital Compression and Coding of Continuous-Tone Still Images-Requirements and Guidelines*, 1993, [Online]. Available: http:// www.w3.org [Accessed: April 2011]
[8] A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto, "A pipelined fast 2D-DCT accelerator for FPGA-based SoCs," In: *IEEE Computer Society Annual Symposium on VLSI*, Porto Alegre, pp. 331-336, 2007.
[9] Xilinx, *Huffman Coding*, 2003, [Online]. Available: http://www.xilinx.com [Accessed: April 2011]
[10] Xilinx, (n.d.) [Online]. Available: http://www.xilinx.com/company/gettingstarted/ fpgavsasic.htm [Accessed: April 2011]
[11] K. Parnell, and R. Bryner, *Comparing and Contrasting FPGA and Microprocessor System Design and Development*, (n.d.) [Online]. Available: http://www.xilinx.com. [Accessed: April 2011].