International Journal of Future Generation Communication and Networking
Vol. 9, No. 11 (2016), pp. 299-308
http://dx.doi.org/10.14257/ijfgcn.2016.9.11.27

Research on Embedded Network Real-Time Video Monitoring System Based on Zynq

Yang Nie, Xiaofei Yin, Yu Hu, Hanbin Xu and Ruofei Yan

Department of Physics, Jining Normal University, Ulanqab, China
E-mail: nieyangwork@163.com

Abstract

With the improvement of the performance of the embedded processor, the real-time network video monitoring system based on embedded technology becomes a developing direction of the network video with its low price and portability. In this paper, software and hardware co-design method is adopted to design an embedded system. ARM of the Zynq chip is responsible for embedded system structures and high definition video processing, and FPGA is used to design other logic and hardware expansion. Compared with the traditional network video monitoring technology, the embedded network video monitoring technology based on Zynq not only improves the quality of the image, but also has better real-time performance and scalability.

Keywords*: Embedded system; Network Video Monitoring; Zynq*

1. Introduction

With the rapid development of video monitoring market, digital monitoring has become the mainstream, and network, personalized and intelligent will be an important development trend in the future video monitoring market. However, the network condition of video monitoring network based on streaming media transmission technology is higher, and the fixed monitoring cost is too high. So it can not be popularized in large area [1-3]. It is an urgent problem of modern monitoring technology that how to combine the video monitoring with the Internet. On the other hand, ARM9 and ARM11 are mostly used for the current video monitoring system. These hardware peripherals of micro controller have been fixed, which is not conducive to the user to expand and upgrade the hardware. The multi chip combination solution of ARM and FPGA will not only lead to high cost of the system, but also cause the waste of system resources. Therefore, the single chip solution is needed, and this chip is integrated with ARM and hardware extensibility.

That is the case with the Xilinx Zynq-7000 family, all programmable SoC that includes a dual-core ARM Cortex-A9 processor and a 28nm FPGA. The Zynq comprises two sections: the Processing System (PS), and the Programmable Logic (PL). These can be used independently or together, and in fact the power circuitry is configured with separate domains for each. However, the most compelling use model for Zynq is when both of its constituent parts are used in conjunction, and therefore it is important to appreciate the structure of both sections, as well as the interfaces between them. This means that the processor and logic can each be used for what they do best, without the overhead of interfacing between two physically separate devices.

The paper is structured as follows: In Section 2, we introduce Zynq-7000 All Programmable SoC. Section 3 discusses embedded network video monitoring system based on Zynq, which are the main contribution of this work. In Section IV, testing and verification of the monitoring system is completed. Finally, Section 5 summarizes the main conclusions of this work.

ISSN: 2233-7857 IJFGCN

2. Zynq-7000 All Programmable SoC

The Zynq-7000 is based on the Xilinx All Programmable SoC architecture, which is shown in Figure 1. These products integrate a feature-rich dual-core ARM Cortex-A9 based PS and PL in a single device[4-5]. The ARM Cortex-A9 CPUs are the heart of the PS and also include on-chip memory, external memory interfaces, and a rich set of peripheral connectivity interfaces. The Zynq-7000 architecture enables implementation of custom logic in the PL and custom software in the PS. It allows for the realization of unique and differentiated system functions. The architecture is completed by industry standard AXI interfaces, which provide high bandwidth, low latency connections between the two parts of the device. This means that the processor and logic can each be used for what they do best, without the overhead of interfacing between two physically separate devices.

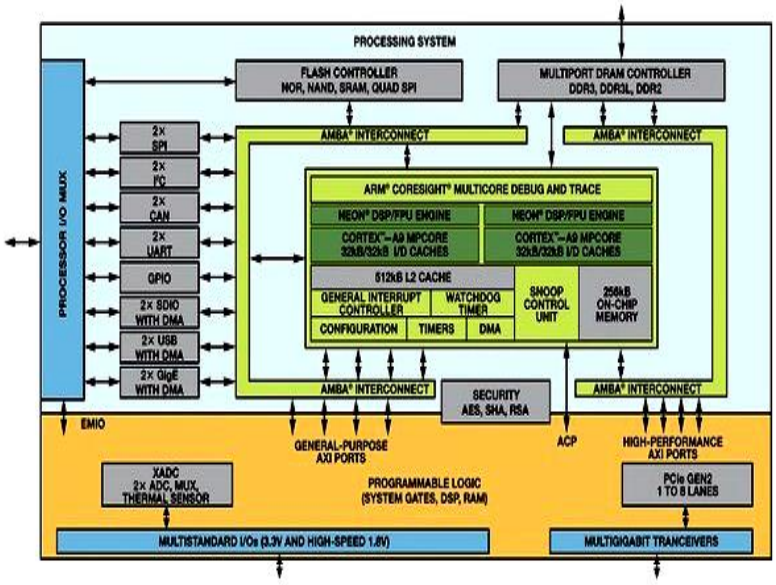

Figure 1. Zynq-7000 System Structure Block Diagram

The application processing unit (APU) consists of two ARM Cortex-A9 processor with a snoop control unit (SCU), which is responsible for maintaining the cache coherency between the two processors. Each processor has its own 32 KB level-one (L1) instruction and data caches, memory management unit (MMU), and separate media processing engine (NEON). L1 caches include two parts: instruction-side cache (I-Cache) and data-side cache (D-Cache). I-Cache is responsible for providing an instruction stream to the Cortex-A9 processor. D-Cache is responsible for holding the data used by the Cortex-A9 processor. The MMU in the ARM architecture involves both memory protection and address translation. The MMU works closely with the L1 and L2 memory systems in the process of translating virtual addresses to physical addresses. NEON is co-processor and extends the Cortex-A9 to provide support for the ARM v7 advanced single instruction multiple data and vector floating-point instruction sets.

The PL is derived from Xilinx 7 series FPGA technology. Advanced high-performance FPGA logic based on real 6-input lookup table (LUT) technology is used to distribute memory. The PL includes many different types of resources including configurable logic blocks, port and width configurable block RAM, DSP slices with a 25×18 multiplier, 48-bit accumulator and pre-adder, a user configurable analog to digital converter, clock management.

3. Embedded Network Video Monitoring System

The whole system design is divided into three steps. First, the Linux operating system is ported to the Zedborad. Secondly, V4L2 read the camera device data and complete the video API interface. Finally, server and client achieve data communication using the TCP protocol.

3.1. Hardware System Structure

The whole video monitoring system mainly includes CMOS camera OV5640, Zedborad, LCD, PC and others peripherals [6][7]. The system block diagram is showed as Figure 2. Client / server structure is adopted by the system. The video data acquired by the remote collection terminal is transmitted to the local client through the network, and the display and analysis of the alarm are carried out in real time.

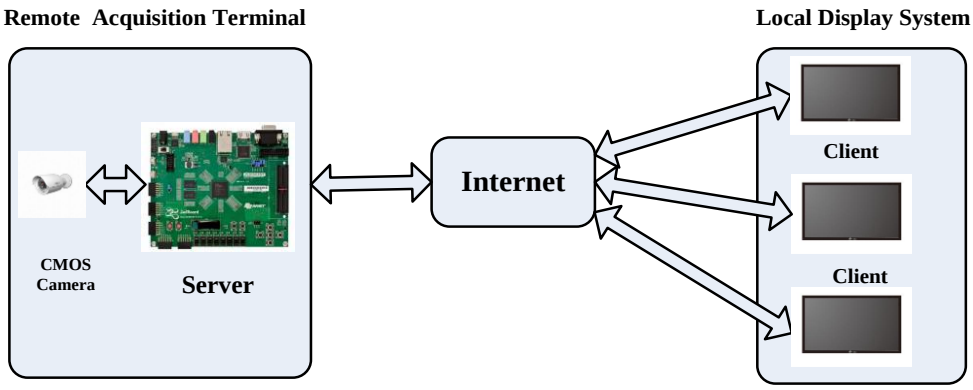

Figure 2. Embedded Network Video Monitoring System Block Diagram

The ZedBoard is an evaluation and development board based on the Xilinx Zynq-7000 Extensible Processing Platform. Combining a dual Corex-A9 Processing System (PS) with 85,000 Series-7 Programmable Logic (PL) cells, it applies special ways to make it suitable for video processing. Omni Vision's OV5640 is used here. It incorporates a 10 bit A/D converter, corresponding to data output interface [0:9]. The output image data format can be 10 bit raw RGB or 8 bit RGB/ YCbCr through internal DSP processing. Peripherals part mainly includes LCD and touch screen interface circuit, JTAG debugging circuit, reset circuit, the power supply circuit, RS232 serial interface circuit and so on.

3.2. Linux Operating System Porting

Embedded Linux is the use of a Linux operating system in embedded systems. Unlike desktop and server versions of Linux, embedded versions of Linux are designed for devices with relatively limited resources. The ARM Cortex-A9 processor used in Xilinx Zynq All Programmable SoCs support embedded Linux. In order to use the ZedBoard as a desktop computer with a monitor, keyboard and mouse, the following items are required:

- A monitor capable of displaying VESA-compliant 1024x768 @ 60Hz
- An analog VGA cable for the monitor
- A USB keyboard
- A USB mouse
- A USB hub

Detailed Linux system migration process steps can be referred to the literature [8].

3.3. The Basic Principle of V4L2

V4L is a range of interface functions provided by Linux for video devices applications, the image data can be read out from video equipment by using API functions [9]. V4L2 is an upgraded version of V4L, which is used to capture images, video and audio data API interface under the Linux operating system. V4L2 specification not only defines the common API, image format, input method, but also defines a series of interfaces of the Linux kernel driver processing video information.

In the Linux system, all the peripherals are seen as a special file, which is called the device file. Video equipment is also a device file, which can be read and written like other ordinary files. V4L2 supports two ways to capture images: memory mapping and direct reading. The first one is mainly used for the collection of continuous video data, while the second is commonly used in static image data acquisition. Collecting video data of application program by the V4L2 interface is divided into 5 steps:

(a) Opening the video equipment file to initialize the video capture parameters.

(b) Apply a number of video capture frame buffer, and map the frame buffer from the kernel to the user space.

(c) Start video capture.

(d) Driving video data acquisition, video capture application accessing to video data in the frame buffer, and the cycle of continuous video data acquisition

(e) Stop video capture, and shut down the device file.

The flow chart of image acquisition based on V4L2 is as Figure 3.

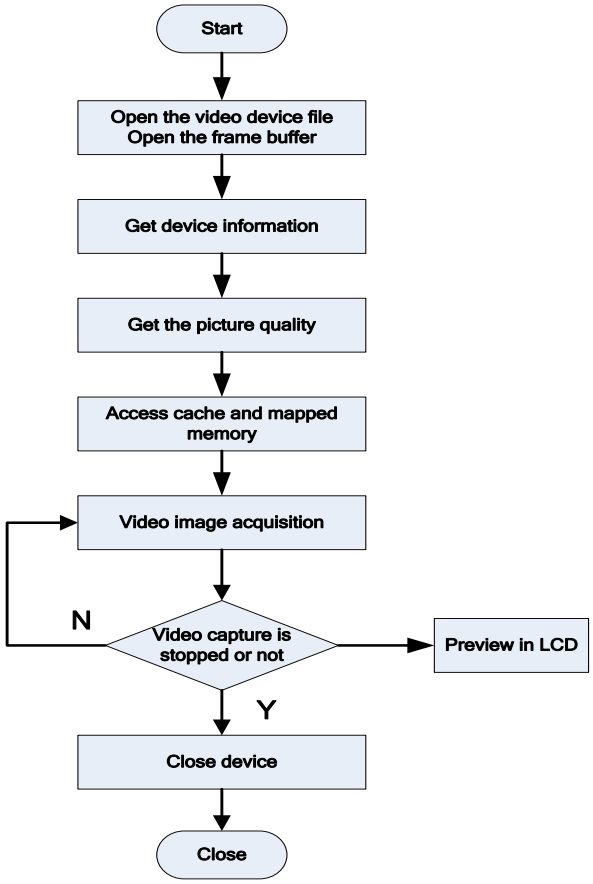

Figure 3. Flow Chart of Image Acquisition via V4L2

In the process of obtaining video data by the V4L2 interface, the interface function ioctl( ) is frequently used. ioctl( ) is a powerful function. It can control the I/O channels of

equipment, set the format of video and frame, and also can inquiry current device properties. Main ioctl function commands are shown as Table 1. V4L2 defines a number of important data structures in the header file. In the process of collecting images, the operation of these data is used to obtain the data of the image. Table 2 gives the explanation of the structure of V4L2 and its function.

Table 1. ioctl( ) Function Command

ioctl Command	Function
VIDIOC_CROPCAP	video information of cutting and scaling capabilities
VIDIOC_G_CROP VIDIOC_S_CROP	read or set the rectangular area currently cropping
VIDIOC_G_FMT VIDIOC_S_FMT VIDIOC_TRY_FMT	read or set data format and format
VIDIOC_G_STD VIDIOC_S_STD	inquiry or choose video standards currently input
VIDIOC_QBUF VIDIOC_DQBUF	read data from buffers or put data back to buffer sequence
VIDIOC_QUERYCAP	query device attribute

Table 2. Commonly Structures and Functions

Structure name	Function
v4l2_requestbuffers	application cache area data of V4L2
v4l2_capability	capacity type description of V4L2
v4l2_standard	video format type of V4L2
v4l2_format	frame format type of V4L2
v4l2_buffer	buffer data structure of V4L2
v4l2_queryctrl	control structure of query of V4L2
v4l2_control	structure of control value of V4L2

3.4. TCP Network Programming Based on QT

Transmission Control Protocol (TCP) is a basic network protocol for data transmission, which is the basis of many Internet protocols. TCP is a reliable protocol for connection and data flow. That is to say, it can make the data on a computer error free to send to other computers on the network. Therefore, when a large amount of data is transferred, the TCP protocol should be selected. TCP protocol programming generally uses the server / client mode [10].

QT provides a QtNetwork module for network programming, and the module provides the TCP network programming for the QTcpSocket class and QTcpServer class [11]. The QTcpSocket class is used to create a TCP connection and data flow exchange. The QTcpServer class is used to write the server program. Figure 4 shows the client / server model for network communications

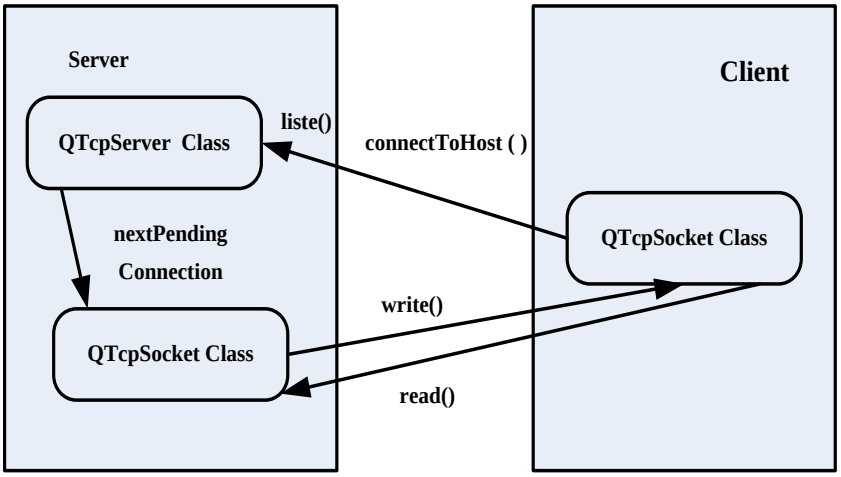

Figure 4. QT Network Programming Block Diagram

A TCP connection must be established to a remote host and port before any data transfer can begin. Once the connection has been established, the IP address and port of the peer are available through QTcpSocket::peerAddress() and QTcpSocket::peerPort(). At any time, the peer can close the connection, and data transfer will then stop immediately. QTcpSocket works asynchronously and emits signals to report status changes and errors, just like QNetworkAccessManager and QFtp. It relies on the event loop to detect incoming data and to automatically flush outgoing data. You can write data to the socket using QTcpSocket::write(), and read data using QTcpSocket::read(). QTcpSocket represents two independent streams of data: one for reading and one for writing.

Since QTcpSocket inherits QIODevice, you can use it with QTextStream and QDataStream. When reading from a QTcpSocket, you must make sure that enough data is available by calling QTcpSocket::bytesAvailable() beforehand. If you need to handle incoming TCP connections (*e.g.*, in a server application), use the QTcpServer class. Call QTcpServer::listen() to set up the server, and connect to the QTcpServer::newConnection() signal, which is emitted once for every client that connects. In your slot, call QTcpServer::nextPendingConnection() to accept the connection and use the returned QTcpSocket to communicate with the client.

Although most of its functions work asynchronously, it's possible to use QTcpSocket synchronously (*i.e.*, blocking). To get blocking behavior, call QTcpSocket's waitFor...() functions; these suspend the calling thread until a signal has been emitted. For example, after calling the non-blocking QTcpSocket::connectToHost() function, call QTcpSocket::waitForConnected() to block the thread until the connected() signal has been emitted.

Synchronous sockets often lead to code with a simpler flow of control. The main disadvantage of the waitFor...() approach is that events won't be processed while a waitFor...() function is blocking. If used in the GUI thread, this might freeze the application's user interface. For this reason, we recommend that you use synchronous sockets only in non-GUI threads. When used synchronously, QTcpSocket doesn't require an event loop.

4. System Testing and Verification

The testing and verification of the whole video monitoring system is divided into two parts: off-line testing and networking testing.

In order to off-line testing, the server and the client are put on a computer test. First, connect the USB camera to your computer, and run the server program. After that, the

client program is run. Enter the host name and port number, click the connection button. The captured video is displayed on the LCD. Program running results are shown in Figure 5.

Figure 5. The Testing Results of Video Monitoring System

Using SD production system mirror, ZedBrod is set to SD boot. ZedBrod is used as a server, and the camera is connected to the board. The server and the client machine IP is set in the same segment. The client application is executed. At the client interface, the IP address and port number of the board are entered. When you click the connect button, the video captured by the camera on the board is transferred to the client, which can be seen in Figure 6.

Figure 6. The Testing Results of Video Monitoring Network Display

5. Conclusion

The real-time network video monitoring system based on embedded technology is very important and becomes a developing direction of the network video with its low price and portability. How to expand the hardware design and software application in the case of meeting the design requirements has become a core issue of modern video monitoring design. Zynq integrate a feature-rich dual-core ARM Cortex-A9 based PS and PL in a single device. This means that the processor and logic can each be used for what they do best, without the overhead of interfacing between two physically separate devices. In this paper, ARM of the Zynq chip is responsible for embedded system structures and high definition video processing, and FPGA is used to design other logic and hardware expansion. Testing results show that the embedded network video monitoring technology based on Zynq not only improves the quality of the image, but also has better real-time performance and scalability.

Acknowledgments

We would like to thank Professor HE for stimulating discussions with respect to the topic of this paper. Moreover, we greatly appreciate the reviewers' comments that lead to an improved presentation of the results. These works were supported by Undergraduate Training Programs for Innovation and Entrepreneurship of Inner Mongolia Autonomous Region, and Undergraduate Training Programs for Innovation and Entrepreneurship of Jining Normal University.

References

[1] S. Park, H. Yoon, J. Kim, "A cross-layered network-adaptive HD video streaming in digital A/V home network: channel monitoring and video rate adaptation", IEEE Transactions on Consumer Electronics., vol. 52, no. 4, **(2006)**, pp. 1245-1252.
[2] H. A, S. Guo, P. Liu, "Study on Embedded Software Applied to Vehicle Intelligent Monitoring System", Proceedings of the 2th International Conference on Instrumentation, Measurement, Computer, Communication and Control, Harbin, China, **(2012)**.
[3] K. Hammoudi, N. Ajam, M. Kasraoui, F. Dornaika, K. Radhakrishnan, K. Bandi, Q. Cai and S. Liu, "Design, implementation and simulation of an experimental processing architecture for enhancing real-time video services by combining VANET, cloud computing system and onboard navigation system", Proceedings of the 5th International Conference on Pervasive and Embedded Computing and Communication Systems, Loire Valley, France, **(2015)**.
[4] Xilinx, "Zynq-7000 All Programmable SoC Technical Reference Manual". [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.
[5] Yang Nie, Zhenhuan Ma, Lili Jing, "Research on the Design of Multi-Core Embedded System Based on Microblaze", International Journal of Control & Automation., vol. 8, no. 12, **(2015)**, pp. 425-434.
[6] J. Joshi, K. Jain and Y. Agarwal, "CVMS: Cloud based vehicle monitoring system in VANETs", Proceedings of International Conference on Connected Vehicles and Expo, Shenzhen, China, **(2015)**.
[7] X. Li, L. Pang and Y. Liu, "Design and implementation of video frame extractor for television broadcasting monitoring system", Proceedings of International Conference on Signal Processing, Communications and Computing, Zhejiang, China, **(2015)**.
[8] L. Qishuai, L. Yanting and W. Di, "Practical Design Guide for Xilinx ZYNQ SoC Running Embedded Linux An Approach Compatible with ARM Cortex-A9", Edited Sheng Dongliang, Tsinghua University Press, Bei Jing, vol. 10, **(2014)**, pp. 155-169.
[9] L. Yinli, Y. Hongli, Z. Pengpeng, "The implementation of embedded image acquisition based on V4L2", Proceedings of International Conference on Electronics, Communications and Control, Zhejiang, China, **(2011)**.
[10] Y. Qun and Z. Jianbo, "Development of remote video monitoring system based on TCP/IP", Proceedings of the 10 International Conference on Computer Science & Education, Cambridge, UK, **(2015)**.
[11] B. Jianping, W. Junfeng, Y. Fengxin and B. Jing "Advanced Programming Qt", Edited Wang Jingdong, Electronics Industry Pub, Bei Jing, vol. 4, **(2011)**, pp. 87-120.

Authors

Yang Nie, he received the MS degree from Wuhan Research Institute of Posts &Telecommunications in 2009. He is currently working toward the PhD degree in the Digital Engineering Center of Communication University of China. He is a lecturer of Jining Normal University, China. His research interest includes embedded system design, high-performance DSP algorithms and VLSI architectures.

Xiaofei Yin, he is an undergraduate in the Physics Department of Jining Normal University, China. His major is electronic information science and technology, and the direction is embedded system design. His research interests include embedded system design and FPGA digital system design

Yu Hu, he is an undergraduate in the Physics Department of Jining Normal University, China. His major is electronic information science and technology, and the direction is embedded system design. His research interests include embedded system design.

Hanbin Xu, he is an undergraduate in the Physics Department of Jining Normal University, China. His major is electronic information science and technology, and the direction is embedded system design. His research interests include embedded system design and FPGA digital system design

Ruofei Yan, he is an undergraduate in the Physics Department of Jining Normal University, China. His major is electronic information science and technology, and the direction is embedded system design. His research interests include embedded system design.