Signal acquisition and processing system based on zynq dual core

Jingjia Tan[1,a], Lesheng He[1] and Jun Wang[1]

[1]*School of information, Yunnan University,650500 Kunming,China*

Abstract. In order to speed up the acquisition and processing of signal, this paper has developed a signal acquisition and processing system based on zynq platform. Based on the ARM Cortex-A9 dual-core and editable logic unit architecture of Zynq AP SoC platform, this paper implements a fully functional signal acquisition and processing system by software and hardware collaborative design. ARM0 is the main processor that controls system and shared resources. ARM1 is the slave processor. ARM1 is responsible for receiving the data converted by the AD7606 analog-to-digital chip. The data is sent to the Hamming window function IP core created under vivado HLS through the AXI bus. After the data is processed by Hamming window function, it is sent to ARM1 again through AXI bus. OCM acts as the shared memory for ARM0 and ARM1 communication. The Linux system runs on ARM0. The processed data is sent to the upper computer through ethernet through UDP protocol. Utilizing the architecture of the Zynq platform, the system efficiency is improved, and the stability of the system is ensured, so that the FPGA can enter the field of embedded systems.

1 Introduction

In terms of signal acquisition and processing, all-programmable zynq-7000 SoC embedded system design hardware platform has many advantages over traditional embedded system hardware platform. All-programmable zynq-7000 SoC embedded system design hardware platform has obvious advantages in performance, cost and system power consumption. Not only reduces the number of electronic components used in embedded system design, but also significantly reduces the size of the board. It not only can resolve the contradiction between product performance and volume, but also make signal processing faster and more functional.

2 Introduction to Zynq-7000 SoC

The Zynq-7000 series is based on the Xilinx fully programmable scalable processing platform architecture. That integrates ARM's dual-core ARM Cortex-A9 multi-core processing system (PS) and programmable logic (PL) system on a single chip. The PL part is the traditional FPGA, which can easily customize the peripheral circuit IP. At the same time, the architecture is based on the latest high-performance, low-power 28nm, high-k metal gate process. This ensures that the device has lower power consumption than its cortex-a9 dual-core processor while running with high performance[1]. Each cortex-a9 processor core has its own NEON, which can implement 128-bit SIMD coprocessors and VFPv3. The PS part mainly includes application processing unit (APU), memory interface, I/O peripherals and interconnection of internal modules. The PL part adopts FPGA technology to expand functions to meet specific functional requirements.

3 Overall design

The system uses zedboard development board as the hardware platform. It uses vivado 2015.2 software design circuit. The vivado HLS 2015.2 software is created Hamming window function IP. The ad7606 module is used acquisition data. The PC software adopts visual studio 2017 development platform. The overall framework is shown in Figure 1.

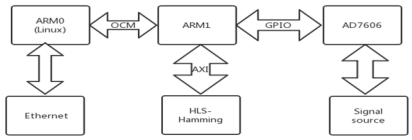

Figure 1. Overall system design.

4 Hardware circuit design

Add IP cores under vivado, then perform validate design, run synthesis, run implementation, and generate bitstream. Finally, it generates a bit stream file under vivado software. The SDK software is imported into the bit stream file. The overall hardware circuit is shown in Figure 2.

a Corresponding author: 1131841611@qq.com

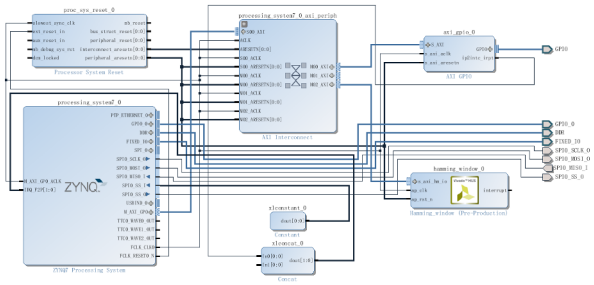

Figure 2. Hardware circuit diagram.

5 Software design

The software part is mainly to improve the system functions and realize the important link of human-computer interaction. The software design mainly includes configuring the dual-core boot under the SDK, making the Boot.bin file required for the SD card boot and the elf program running the dual-core. The vivado HLS software is written the C++ source code about the Hamming window IP core. This system must creat kernel files and device tree files for SD card startup. The ubuntu system is compiled u-boot and file system. Finally, the upper computer application is written under the visual studio 2017 software.

5.1 Configuring dual core

The Zynq AP SoC contains two Cortex-A9 processors[2]. There are two different modes of operation, one is symmetric multiprocessor mode (SMP) and the other is asymmetric multiprocessor mode (AMP). This design uses AMP mode. Linux system runs on ARM0 in AMP mode, and bare metal system runs on ARM1. The chip initialization starts from ARM0. ARM0 loads U-Boot, and ARM1 loads the bare metal mode elf program. ARM0 and ARM1 each occupy separate DDR space. ARM0 uses an address space of 0x00100000 to 0x001FFFFF. ARM1 uses an address space of 0x00200000 to 0x002FFFFF. After all booting is completed, the two ARMs will communicate by setting and resetting the same address of the OCM. The specific configuration process is as follows: Import the bit stream file into the SDK, first import the sdk_repo file package into the SDK software library, and then establish the FSBL application project. The SDK software is created two applications for ARM0. One uses UDP to transfer data and the other reads OCM. The SDK software is created project for ARM1. The ARM1 project includes two programs. One is to write a program for dual-core communication, and the other is to write an AD7606 driver.

5.2 Creating a Hamming window

5.2.1 Hamming window

The Hamming window[3] is an improved raised cosine window. The improved raised cosine window is more concentrated in the main lobe. The energy of the main lobe is about 99.96%, and the peak amplitude of the flap is 40 dB. Adding the Hamming window function can intercept the infinitely long sequence, take a piece of data for analysis each time, and then take a piece of data and analyze it again. The time domain form of the Hamming window function can be expressed as:

$$w_{Hm}(n) = \left[0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)\right] R_N(n)$$

Its spectral function W_Hm (e^jω) is:

$$W\_Hm\ (e\hat{}j\omega\ ) = 0.54W\_R\ (e\hat{}j\omega\ ) - 0.23W\_R\ (e\hat{}j(\omega - 2\pi/(N-1))\ ) - 0.23W\_R\ (e\hat{}j(\omega + 2\pi/(N-1))\ )$$

Its amplitude function W_{Hmg} (ω) is:

$$W_{Hmg}(\omega) = 0.54W_{Rg}(\omega) + 0.23W_{Rg}\left(\omega - \frac{2\pi}{N-1}\right) + 0.23W_{Rg}\left(\omega + \frac{2\pi}{N-1}\right)$$

5.2.2 HLS

The advantage of HLS is that it can transform the C/C++ language into our hardware description language [4-5]. It allows more developers who do not understand the hardware description language to invest in the development of FPGA. Some excellent C algorithms can be converted into hardware description languages using HLS. This quality is better than manual design. It can save man power and material resources. Use the high-level synthesis tool Vivado HLS[4-5] to write C++ source code about on the Hamming window function and create a hardware-accelerated IP core. Then code synthesis and code optimization. Finally, it generates RTL. Through comparison between Figure 3 and Figure 4, it can be seen that after code optimization, the total number of clock cycles is changed from 5002 before optimization to 1007, and the total number is significantly reduced. It effectively improves the speed of the algorithm processing data.

Summary

Latency		Interval		
min	max	min	max	Type
5001	5001	5002	5002	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	0	14
FIFO	-	-	-	-
Instance	4	2	196	180
Memory	1	-	0	0
Multiplexer	-	-	-	11
Register	-	-	50	-
Total	5	2	246	205
Available	280	220	106400	53200
Utilization (%)	1	~0	~0	~0

Figure 3. Before optimization.

Summary

Latency		Interval		
min	max	min	max	Type
1006	1006	1007	1007	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	-	-
FIFO	-	-	-	-
Instance	4	24	196	180
Memory	2	-	0	0
Multiplexer	-	-	-	13142
Register	-	-	1199	-
Total	6	24	1395	13322
Available	280	220	106400	53200
Utilization (%)	2	10	1	25

Figure 4. After optimization.

5.3 UDP protocol

UDP is the abbreviation of User Datagram Protocol. The Chinese name is User Datagram Protocol[6]. It is a connectionless transport layer protocol in the OSI (Open System Interconnection) reference model. That provides transaction-oriented simple and unreliable information transfer service. UDP(User Datagram Protocol) transmission is similar to IP transmission exactly. The UDP protocol can be seen as an interface exposed by the IP protocol at the transport layer. The main function of the UDP protocol is to compress network data traffic into the form of data packets. A typical data packet is a unit of transmission of binary data. The first 8 bytes of each packet are used to contain header information, and the remaining bytes are used to contain specific transmission data. There is no concept of a port in the IP protocol. The IP protocol carries the transmission of IP addresses to IP addresses, which means a dialogue between the two computers. However, multiple communication channels are required in each computer, and multiple communication channels are assigned to different processes. A port represents such a communication channel. The UDP protocol implements the port so that the packet can be sent to a port based on the IP address. Using UDP socket programming can realize connectionless communication based on TCP/IP protocol. It is divided into two parts: server and client. The main implementation process is shown in Figure 5.

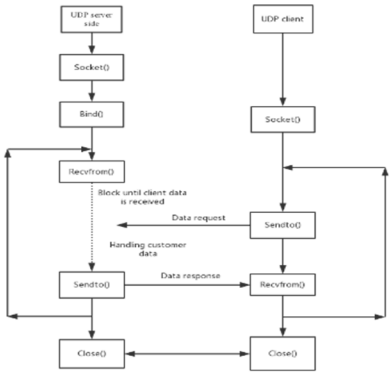

Figure 5. UDP protocol.

5.4 Creating an SD card image file

Firstly, download the uboot source under the ubuntu system, and generate the u-boot.elf file by using the following command: make zynq_zed_config, make, mv u-boot u-boot.elf. After downloading the linux kernel file, generate a uImage file by using the following command: make Xilinx_zynq_defconfig, make menuconfig, make uImage LOADADDR=0x00008000. Then, in order to make the device tree file, this design must enter the previously downloaded linux kernel directory in the dts file directory then modify the contents of the zynq-zed.dts file as shown in Figure 6. Then the terminal is entered dtc –I dts -O dtb -o devicetree.dtb zynq-zed.dts. The device tree file is generated by using this command. Next, the PC is opened the SDK software and clicked Create Zynq Boot Image in the Xilinx Tools directory. Firstly, the SDK software is loaded the amp_fsbl.elf file in the pop-up dialog box, then loaded the bit stream file generated under vivado software, and loaded the u-boot generated by the ubuntu compiler. Finally, the SDK software is loaded the app_cpu1.elf file. The Boot.bin[7-8] file is generated by clicking Create Image in the create zynq boot image box. The SD card is copied the compiled boot.bin file, uImang file, devicetree.dtb file, the uramdisk.image file system officially created by Xilinx, the read OCM application file and the UDP application file.

```
memory@0 {
        device_type = "memory";
        reg=<0x0 0x18000000>  ;
};

chosen {
        bootargs = "console=ttyPS0,115200 maxcpus=1 root=/dev/ram rw earlyprintk";
        linux,stdout-path=&uart1;
        stdout-path=&uart1;

};
```

Figure 6. Device tree configuration dual core.

6 Test results

6.1 Verify SD card startup file

The computer is connected to the development board through a serial port line. The computer runs the SecureCRT software. The serial port printing information is shown in Figure 7. The PC is configured IP so that the computer can communicate with the development board through the network cable. The SecureCRT software is entered the following command: ifconfig eth0 192.168.1.118, ./mnt/rwmen.elf 0xfffffff0 0x18000000, ./mnt/UDP.elf, so that the entire system starts running.

```
RAMDISK: gzip image found at block 0
mmc0: new high speed SDHC card at address e624
mmcblk0: mmc0:e624 SU8G 7.40 GiB
 mmcblk0: p1 p2
platform ci_hdrc.0: Driver ci_hdrc requests probe deferral
VFS: Mounted root (ext2 filesystem) on device 1:0.
devtmpfs: mounted
Freeing unused kernel memory: 216K (40646000 - 4067c000)
Starting rcS...
++ Mounting filesystem
FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
++ Setting up mdev
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting ssh daemon
random: sshd urandom read with 1 bits of entropy available
rcS Complete
zynq>
```

Figure 7. Serial port print information.

6.2 Verify the correctness of the data collected by the system

In order to verify the correctness of the data collected by the system, the IP of hamming window was removed in hardware project. The results are shown in Figure 8 and Figure 9.

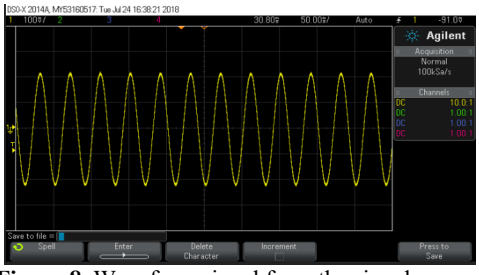

Figure 8. Waveform signal from the signal source.

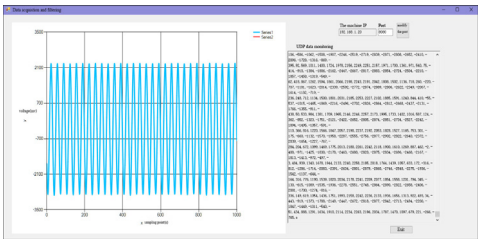

Figure 9. Host computer receiving display.

6.3 Verify the correctness of the data processed by Hamming window

The Hamming window IP core is re-added to the project. The AD7606 input channel and the output channel of sine wave from signal source is connected by signal lines. The result of the Hamming window processing data is shown in Figure 10. The signal frequency is changed to 100Hz, the data display result is shown in Figure 11.

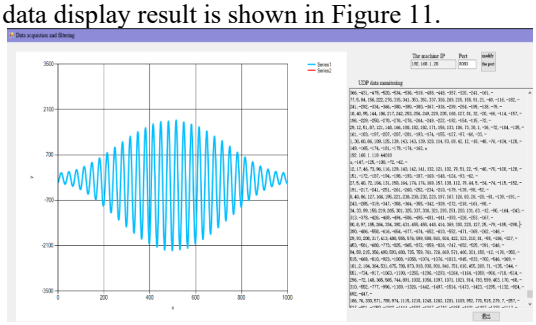

Figure 10. Host computer display Hamming window processing data results.

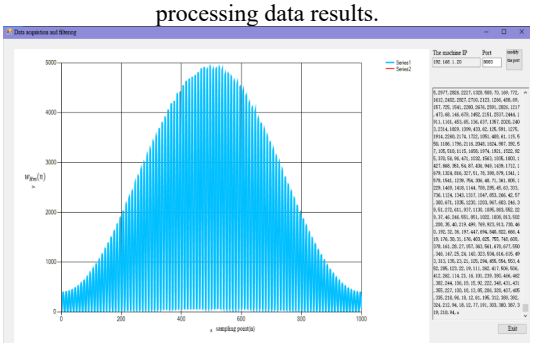

Figure 11. Shows the result at a frequency of 100Hz.

7 Conclusion

This paper introduces the construction of embedded signal acquisition system and processing scheme on Zynq APSoC platform in detail. It provides basic ideas for other embedded design schemes. The Zynq-based signal acquisition and processing system is realized by the software and hardware collaborative design method. This paper mainly introduces the hardware and software co-design, dual-core configuration and the creation process of the files required for the development platform to boot from the SD card. The experimental results verify that the system configuration is correct. The signal processing adds to the hamming windows created under HLS software so that the signal processing speed is faster and quicker. It provides a solution for processing complex signals in the future. Using the high-level synthesis tool, the migration time of high-level code description algorithms to FPGA hardware code is greatly reduced. This design can be applied to the processing and analysis of infinitely long audio signals and biomedical signals and other more complex signals. It has a good man-machine interface, a wide range of universality and practical application, and a broad market prospect.

Acknowledgement

The financial support of this research by the National Natural Science Foundation of China, under Grant No. U1631121 is greatly appreciated.

References

1. B. He, *Xilinx Zynq-7000 Embedded system design and implementation*, **1**, 7(2016).
2. J. McDougall, Simple AMP Bare Metal System running on both cortex-a9 processors, https://www.xilinx.com/support/documentation/application_notes/xapp1079-amp-bare-metal-cortex-a9.pdf, (2014)
3. Y. Li, S. Yang, X. Sun, *Digital signal processing*, **7**,179(2008)
4. X. Peng, T. Zhang, Application of edge detection hardware acceleration based on Vivado HLS, J. AET, **5**, 1(2017)
5. J. Duarte et al, Fast inference of deep neural networks in FPGAs for particle physics, J. Inst, **13**,13-20(2018)
6. P. Födisch et al, A synchronous Gigabit Ethernet protocol stack for high-throughput UDP/IP applications, J. Inst, **11**,2-9(2016)
7. J. Lu, Practical guide for collaborative design of embedded system software and hardware, **9**, 123-127(2015)
8. A. Svetek et al，The Calorimeter Trigger Processor Card: the next generation of high speed algorithmic data processing at CMS, J. Inst, **11**,2-3(2016)