Data Communication between GUI and PLL device using TCP/IP embedded system interface

Amay Prabhakar Kamat
Dept. of Computer Science and Applications
Reva University
Bengaluru, India
Amayprabhakarkamat@gmail.com

Abhiroop Biswas
Electronic Warfare Department
Tata Power Strategic Electronics Division
Bengaluru, India
abiswas@tatapowersed.com

Abstract- **Application Development for embedded system and GUI with same software IDE enhances cross-platform knowledge and ensures less dependency on different teams for implementation of hardware and software logics separately. The idea of the project is to develop a GUI interface along with embedded application for FPGA to transmit and receive data from a PLL front end device.**

The objective of this paper is to send a frequency word from a GUI to a PLL device interfaced with FPGA and to receive the data lock status from PLL on GUI. Here PC is connected to the FPGA through Ethernet interface and FPGA is connected to the PLL front end through RS-232 Serial Interface. This process ensures and proves that we are able to do faster data transaction from a GUI to PLL Front-end device in shorter time and in more secured manner over TCP/IP 1 Gigabit LAN interface.

Keywords— GUI, PLL(Phase Locked Loop), RS-232 Serial Interface, FPGA, IDE(Integrated Development Environment).

I. INTRODUCTION

Here Zynq (ZC702) Xilinx 32-bit System on-a-Chip (SOC) development board is used. Along with a Programmable Logic unit it has dual ARM Cortex A9 processor (Samsung S3C6410), 1 GB DDR Memory and various peripheral for data communication. The ARM processor facilitates Qt based embedded application over Linux OS.

The purpose of this paper is to develop a Qt based GUI application to send single frequency(1 – 500) MHz word from GUI to a PLL front end device. Once the PLL receives the command from GUI, it locks to that particular frequency. The Data Lock status is transmitted back to the GUI. Zynq ZC702 board is used to help the data transaction happen over a 1 Gigabit LAN interface with TCP/IP protocol. Qt based embedded application thread is developed for the Zynq device to ensure TCP/IP communication.

This paper focuses on the process and complex points in the application development of GUI and embedded threads using Qt IDE. Typically, programming, digital logic design and computer architecture are prerequisites for the more advanced embedded systems or microprocessor design that is the focus of this paper.

For software development in the embedded systems industry, the C/C++ family of languages is still used in the large majority of new SOC designs. System on-a-Chip (SOC) 32-bit devices contains reduced instruction set computer (RISC) processor with volatile memory, non-volatile flash memory, and a wide assortment of standard I/O interfaces, all on a single chip. According to annual industry surveys of embedded designers, 70% of new designs now utilize an operating system (OS), and 59% include networking. Now that a single-chip microcontroller already contains the processor, memory, and numerous I/O interfaces with built in hardware controllers, it is appropriate to use a higher level of abstraction in such a course.

This process helps in faster data transaction speed and an increased focus on networking. The Following course of paper would explain all the methodologies separately and primarily focus on their specific role which helped in developing the final outcome.

II. SYSTEM PLAN AND TOPOLOGY

Zynq ZC702 has Petalinux development tool to develop Linux based application over ARM. Petalinux uses C/C++ threads over a pre-built hardware logic to perform various embedded tasks. These C/C++ threads when built with Petalinux SDK are also called Linux drivers. We can cross-compile Qt IDE with Petalinux SDK and write Qt threads for Linux drivers. Here a Linux driver is developed to automate data transfer from ARM Processor to the PLL using an RS 232 interface achieving the combination of Qt and the Linux system programming.

The Embedded system is mainly divided into PS (Processing System) and PL (Programmable Logic) where PS controls all signal transaction with PLL front-end. All processor logics, Linux drivers and APIs used for automated process are loaded onto PS ARM. Zynq PS contains ARM processor, DDR

Memory, Multiplex I/O, SD Card interface. ARM processor is connected to several memory controllers through 64-bit AXI-bus interface and is externally connected to PLL device through RS 232 Interface.

GUI Application on the PC select a particular frequency from frequency range 1 MHz to 500 MHz and send a 8 bit ASCII converted binary data onto the Zynq PS through Ethernet TCP/IP interface and the same binary data is send through ARM Processor of Zynq PS to the PLL front-end systems through RS-232 interface. Once the frequency is locked in PLL, it sends back an 8 bit PLL lock status frequency word to FPGA over RS-232 and the FPGA forwards the same status word over TCP/IP to GUI.

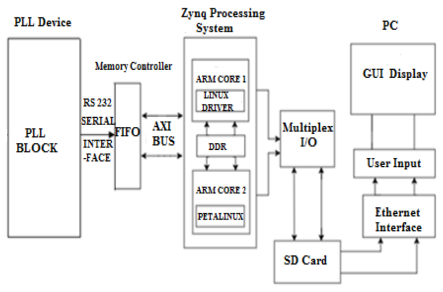


Figure (a): Block diagram of the proposed system

III. SOFTWARE DESIGN OF ARM PROCESSOR

A. Embedded Linux Operating System

The Embedded Linux (Petalinux) is a SDK platform provided by Xilinx to develop real time application threads for Zynq ZC702 board Figure (b). In boot mode from SD card the Linux driver developed in petalinux has to complete three tasks. Boot loader transplant, Linux kernel transplant and root filing system transplant. Boot loader is a small program which loads operating system kernel into memory and transfer control to it. The main role is initializing hardware equipment (including I/O, the special function register), establishing the memory space map and bringing the environment of the system's hardware and software to an appropriate state. The Linux operating system's kernel should be able to provide good support to the ARM processor and manage most of components which connect to the periphery of the controller. The embedded Linux kernel only requires providing support to the hardware which will be used; therefore we may cut the kernel according to the practical application.

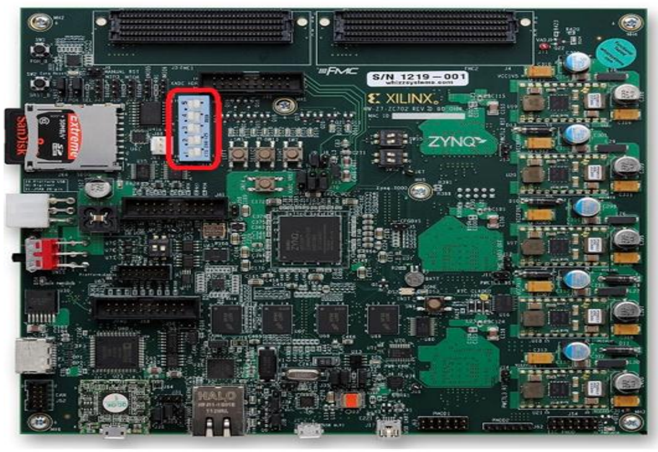


Figure (b): Zynq (ZC702) Development Board

B. Application design

The system adopts Linux as the operating system of the ARM's main controller. The work needed to be done is: the porting of Linux in ARM board, the programming of the serial driver used to read values from Ethernet interface and display the values on GUI window. The implementation of GUI application is for displaying PLL Lock status. The system adopts graphical user interface based on QT IDE and establishes a user interface to optimize the human-computer interaction environment.

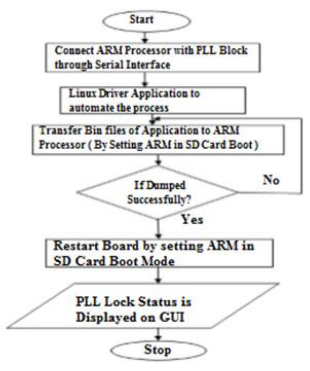
C. SD Card as a Storage Medium

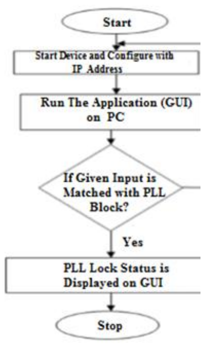
The data management of an FPGA is performed by using an SD Card as a Storage medium. The Embedded Linux Operating System creates a Bootable file and Boots it over a SD Card and hence the process of acquiring data from an Application takes place via the ARM Processor on the embedded device. ARM Processor send control commands to PLL device over the serial communication protocol, which returns the data according to the commands. The data is acquired and updated in a fixed time interval managed by GUI Application itself. However, when the frequency of data acquisition is high, the data in the embedded database will be drastically increased and there is a need to clear the historical data, or the access performance of the system will gradually decline. The Advantage of using SD Card is that the data is always secured and also there is least possibility for loss of data during processing the system.

IV. THE DESIGN OF GUI AND DEVICE DRIVERS

The design of GUI for embedded systems is different from that of traditional data computing class software. Since the embedded systems is resource-constrained, the propose mode of the GUI for the conventional PC and also the memory consumption of which is relatively large can take up more CPU time, which is not suitable for embedded systems.

In this paper, Qt IDE is used for embedded Linux and GUI application development. It should be capable of fully satisfying the constraint of embedded system resources. As Qt uses C++ as its programming language, it can implement hybrid and multi-threaded programming with Linux C. The header files include both Qt-API library and Linux system calls libraries. Writing of Linux system calls can be done as part of the slots functions which can respond to specific signals in order to achieve the combination of Qt / Embedded and Linux-C.
Certainly, to achieve reading and writing of a data for a specific device file, there should be device drivers which give reading and writing operation interface functions. Therefore, we require the process of preparing, configuring and modifying of the drivers of ARM. The Application use Qt to complete GUI on the ARM machine to achieve the graphical display of data collected by showing its Data Lock status for PLL devices. This paper focuses on elaborating the design of the Linux drivers onto the application in the system. The workflow of dumping Linux driver applications to automate process into Arm Processor is shown in flowchart 1. The workflow of running a Qt Application on PC is shown in flowchart 2.


Start
Connect ARM Processor with PLL Block through Serial Interface
Linux Driver Application to automate the process
Transfer Bin files of Application to ARM Processor (By Setting ARM in SD Card Boot)
If Dumped Successfully?
No
Yes
Restart Board by setting ARM in SD Card Boot Mode
PLL Lock Status is Displayed on GUI
Stop


Start
Start Device and Configure with IP Address
Run The Application (GUI) on PC
If Given Input is Matched with PLL Block?
Yes
PLL Lock Status is Displayed on GUI
Stop

Flowchart 1: For Dumping Flowchart 2:
For Running GUI
Linux Driver
Application on PC
Application into
ARM Processor

With the help of Qt Creator, the programmer can quickly develop relevant GUI Application by adjusting the size and position on that Application. It also includes buttons, labels and textboxes. By using labels we are displaying the IP Address and Port number of the embedded device to be connected. Buttons are used to Transmit/Receive Data. Textboxes are used to display the PLL Lock status.

V. WORKFLOW OF THE SYSTEM

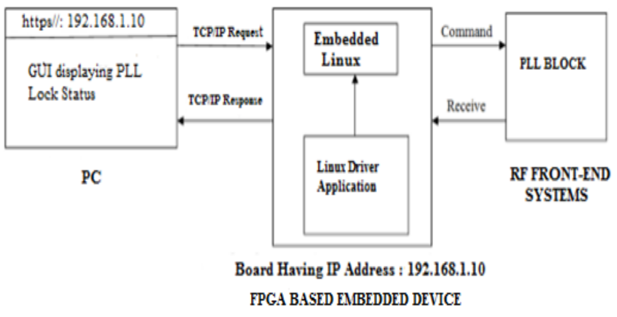


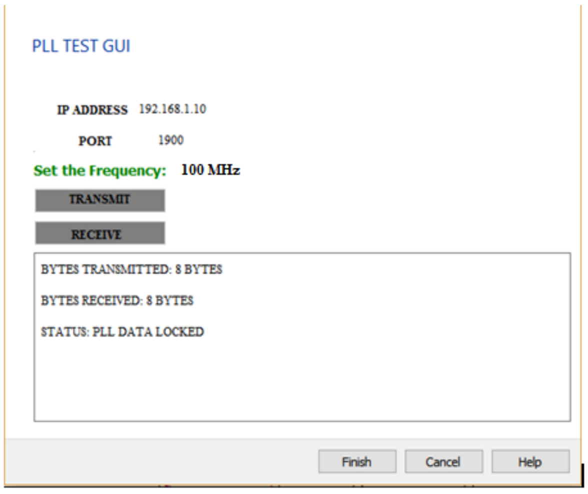
Figure (c): Workflow of proposed system

As shown in figure (c), FPGA receives all the data values from the GUI Application where the connection is established between PC and the FPGA over a Gigabit LAN using a TCP/IP Protocol. At First, The Board's IP address and port is set in the GUI, where PC acts as a client and the embedded device acts as a server. A peer-to-peer centric platform is created between two devices for communication. Secondly, an ASCII to binary converted data logic is written in the application to transfer 8 bytes of data and send single frequency selection from a GUI application to a PLL block as an input and is sent to the embedded device. The Embedded Linux operating system and the Linux drivers automate the process for the free flow of data and send the data over RS 232 serial interface to the PLL registers present in the PLL block. If the 8 byte data which was transmitted by application is received by the PLL Register, it locks to that particular frequency in the PLL block and a displays message stating 'PLL Data Locked' in the application Screen or else if the transmitted data is unable to Lock the Frequency on PLL block then a display message pops up stating 'Unable to match data' in the application screen.

VI. EXPERIMENTAL RESULT

For implementing the Application based on FPGA device, we have prepared two methods; At First, we dumped the Linux (Petalinux) operating system into the ARM Processor and created an Embedded Application Development Environment (Qt Creator) using Open CV Library. Second, testing the application for desktop environment was done with cross-compiling the application with the FPGA. Linux drivers were written to automate all information from ARM and send to the PLL Block using RS 232 Serial Interface.
By creating GUI application on Qt creator, we have to build this application and run the built application. After all this process, main window of PLL Test GUI Application appears as shown in figure (c) and figure (d) where two test cases are tested.

Test Case 1:


PLL TEST GUI

IP ADDRESS 192.168.1.10

PORT 1900

Set the Frequency: 100 MHz

TRANSMIT

RECEIVE

BYTES TRANSMITTED: 8 BYTES

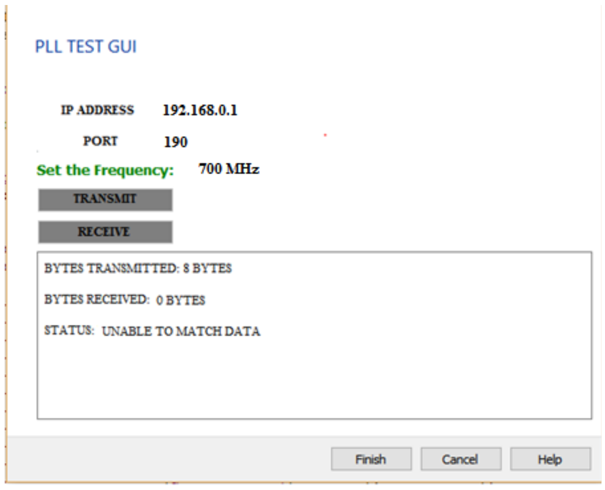
BYTES RECEIVED: 8 BYTES

STATUS: PLL DATA LOCKED

Figure (d): PLL Data Locked successfully

Figure (d) depicts a Test Case which shows the PLL Lock status of data from a GUI to PLL. If all the connections are correct and there is no physical damage of any components or devices. If the frequency selection is made within range to lock the PLL block, then this test case is successful with the desired output.

Test Case 2:


PLL TEST GUI

IP ADDRESS 192.168.0.1

PORT 190

Set the Frequency: 700 MHz

TRANSMIT

RECEIVE

BYTES TRANSMITTED: 8 BYTES

BYTES RECEIVED: 0 BYTES

STATUS: UNABLE TO MATCH DATA

Figure (e): PLL Data Unmatched successfully

Figure (e) depicts a Test Case which shows the PLL Lock status of data which fails to lock the data at the PLL block. Reasons for failure might be incorrect IP address and port name and also physical damage of any components or devices. If the frequency selection is out of range to lock the PLL then the bytes are not received at PLL block hence this test case is successful with the desired output.

The research work presented in this paper is aimed at creating applications for faster data transfer techniques by designing and implementing embedded system.

VII. CONCLUSION

This paper is for creating Lab Applications to test individual components of hardware devices such as PLL devices. Building a Qt/embedded application to implement the various data transfer techniques and to achieve cross-platform knowledge of both hardware and software components is the main aim of this paper. It also develops an intelligent Lab system with ARM Processor as its major controller, elaborating the complex points of the development of the GUI applications based on Qt IDE and Linux drivers for automated process and quick transmission of data through Ethernet interface in the project. We can develop this project in industrial sectors also will have better performance and broader market prospect.

VIII. ACKNOWLEDGMENT

Firstly I would like to thank REVA UNIVERSITY for giving me an opportunity to take up this project as part of curriculum. I would like to thank Tata Power Strategic Electronics Division, for giving me the opportunity to work in this Project. I would like to thank my project guide Prof. Pinaka Pani R (Asst. Professor in department of MCA, School of computer Science and Applications), Reva University, India.
This work was supported by my superior Mr. Abhiroop Biswas (D1, Electronic Warfare Department), Tata Power SED, Bangalore , India and I express my sincere appreciation to him for constant encouragement during the preparation of the manuscript.

IX. REFERENCES

[1] Zynq-7000 AP SoC - Implementing a Host PC GUI for Communication with Zynq Tech Tip, Online Reference Documentation [http://www.wiki.xilinx.com/].
[2] Jacek W, "Embedded Internet technology in process control devices," IEEE Internet Computing, Vol. 34, 2000.
[3] Zynq Qt and Qwt Base Libraries-Build Instructions, Online Reference Documentation [http://www.wiki.xilinx.com/Zynq+Qt+and+Qwt+Base+Libraries-Build+Instructions].
[4] ug1144-petalinux-tools-reference-guide.pdf. Online Reference Documentation [support documentation and software manuals for using petalinux2014_4].
[5] Wang Tianmiao, Publication: Embedded System Design and Case Development. Beijing: Tsinghua University Press, 2002.
[6] Yun Sin-quan, Lu Qiang, Qian Pei-del. One implementation of Linux application based on Qt / Embedded [J]. Computer Application and Software, 2006, 23 (2): 105-107.

[7] Chen Kun, Chen Yun-qiu, Liu Xin. Application design based on Qt / Embedded and embedded Linux [J]. Computer and Digital Engineering, 2009,37 (1).
[8] K. Ricks, D. Jackson, and W. Stapleton, ―An embedded systems curriculum based on the IEEE/ACM model curriculum,‖ IEEE Trans Educ., vol. 51, no. 2, pp. 262–270, May 2008.
[9] EE Times Group, New York, NY, ―2010 embedded market study,‖ Apr. 19, 2010.
[10] Embedded Linux Development Guide, Online Reference Documentation [https://reference.digilentinc.com/_media/digilent_embedded_linux_guide.pdf].
[11] Quick Start Guide, Online Reference Documentation [Zynq-7000 All Programmable SoC ZC702 Evaluation Kit].
[12] Qt for Embedded Linux, Online Reference Documentation [http://doc.qt.io/qt-5/embedded-linux.html].
[13] Linux Drivers, Online Reference Documentation [http://www.wiki.xilinx.com/Linux+Drivers].
[14] PetaLinux Tools User Guide, Application Development Guide, Online Reference Documentation [ug981-petalinux-application-development-debug.pdf].
[15] Real-Time Linux , Online Reference Documentation [http://www.wiki.xilinx.com/Real-Time+Linux].