EMBEDDED LINUX ON FPGA INSTRUMENTS FOR CONTROL INTERFACE AND REMOTE MANAGEMENT*

B.K. Huang, Dept. of Physics, Durham University, Durham and CCFE, Abingdon, Oxfordshire, UK
G. Naylor, G. Cunningham, CCFE, Abingdon, Oxfordshire, UK
O. Goudard, ESRF, Grenoble, France
J. Harrison, Merton College, Oxford, UK
R.M. Myers, R.M. Sharples, Dept. of Physics, Durham University, Durham, UK
R.G.L. Vann, York Plasma Institute, Dept. of Physics, University of York, Heslington, York, UK

Abstract

FPGAs are now large enough that they can easily accommodate an embedded 32-bit processor which can be used to significantly extend the capability of an FPGA system. Running embedded Linux gives the user many more options for interfacing to their FPGA-based instrument, and in some cases this enables removal of the middle-person PC. It is possible to manage the instrument directly by ModBus (130 Hz poll rate and 6.6ms latency) and StarkStream (42 MB/s transfer and 83 MB/s raw throughput), and use these protocols as a low level layer to communicate with EPICS or TANGO. Additionally it is possible to use the embedded processor to facilitate remote updating of firmware which, in combination with a watchdog and network booting ensures that full remote management over Ethernet is possible.

INTRODUCTION

Embedded Linux is a very powerful tool on FPGA instruments. The reason for this is that the embedded processor is directly coupled to the FPGA logic thus giving full control using Linux. By using high level tools it is possible to create very complex systems that do not require VHDL or verilog code to be written. With a soft processor all of the logic is embedded in the FPGA, thus abstracting the design from the hardware and making it more portable helping to prevent system obsolescence.

This paper demonstrates the versatility of embedded Linux combined with two protocols: i) a low latency UDP protocol called StarkStream, ii) the widely used Modbus TCP PLC protocol. Also discussed is the strategy for inclusion to complex control systems.

EMBEDDED LINUX

Many devices now incorporate embedded Linux. The main reason for this is access to a large amount of open source software supporting a large number of hardware devices. Embedded Linux has existed on FPGA devices for at least 5 years with the Xilinx Microblaze processor currently supported in the mainline Linux kernel tree and the Altera Nios II also having a Linux port. Recent developments in the last year include support for the AXI bus which has enabled much faster data rates coupled with the use of DMA enabled drivers thus making embedded Linux more powerful than it has been before.

* This work was part-funded by the RCUK Energy Programme under grant EP/I501045 and the European Communities under the Contract of Association between EURATOM and CCFE.

HIGH LEVEL TOOLS

High level tools are a very significant timesaver in developing FPGA applications and likely justify their cost. They allow access to complex DSP logic blocks such as FFT, CORDICs, etc. We have found that the MATLAB Simulink environment is very good for producing FPGA products as it has a completely open environment in which it is possible to access all the pins of the FPGA and use them in any way desired. For Xilinx FPGAs the tool is called System Generator and for Altera FPGAs the system is called DSP builder. System Generator even allows you to compile a portion of logic code and compile it as a hardware component (PCORE) of the embedded processor. This extremely useful function greatly simplifies the integration with the processor - the result is that this can be done without writing any code but solely through the use of a GUI.

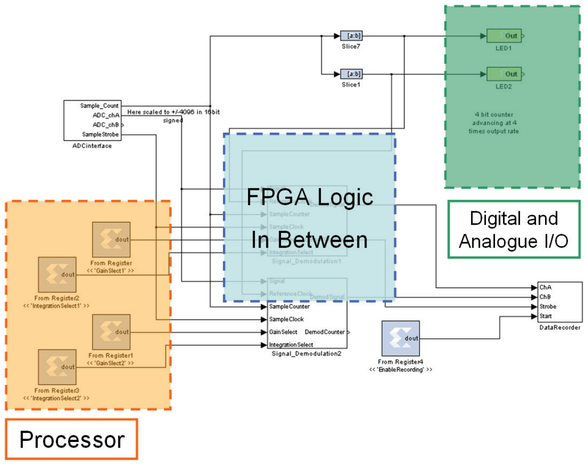

Figure 1: The high level MATLAB Simulink environment used for developing FPGA logic. In this figure the processor logic drives the FPGA logic which in turn drives the digital and analogue I/O. The digital flow can also be reversed to feed back or run completely independently of the processor.

AXI BUS

The AXI bus is an open on-chip bus. It is used by ARM, Xilinx and becoming more widely supported on Altera FPGAs. It is a very fast bus with separate read and write channels thus allowing uninterrupted data flow in both directions. It can be driven at beyond 200 MHz at 256+ bits wide which allows 51.2+ Gbps transceivers.

It is only with the AXI bus that we have been able to achieve high speed on compact systems - achieving up to 10 GB/s in our continuous data streaming application.

STARKSTREAM

StarkStream is a high speed UDP protocol. It runs on embedded Linux, and sits inside a DMA ethernet enabled driver. With interrupts on both the transmit and receive lines it is very quick to respond to sending and receiving input packets. The principle for systems using this protocol is that the FPGA board must respond as quickly as possible to any read/write requests over StarkStream.

Despite the fact that the Microblaze is only running at ~83 MHz it is still able to achieve very high raw throughput (83 MB/s) and continuous data transfer rates (42 MB/s) that may not be expected with a soft processor. The reason for the high speed is an AXI DMA bypassing the processor for memory transfer.

Even faster speeds could be achieved by embedding the UDP packet header around the data to be transmitted. This would require FPGA logic change to add this header (no more than 64 bytes) to each 1kB up to 8kB data payload. But then the consideration has to be that the net throughput rate also depends on the receiving side, which may become the limiting factor.

REMOTE MANAGEMENT

Remote Reboot Control

Remote power control of an FPGA system is crucial for two reasons: i) Many FPGA boards are designed such that the new firmware will only be loaded once a power cycle has been performed. ii) An FPGA system with an embedded processor may have a processor which crashes (as can happen to a standard PC running an operating system). In order to be able to have full control over this we take advantage of the ATX power supply which has two critical elements, a permanent standby low current (~100 mA) supply when the power is off and circuitry for enabling/disabling the power.

Watchdog

An Embedded Linux controlled watchdog with the poll code implemented in the heartbeat section of the kernel ensures that in the event the Linux crashes the system will automatically reboot in 5 minutes.

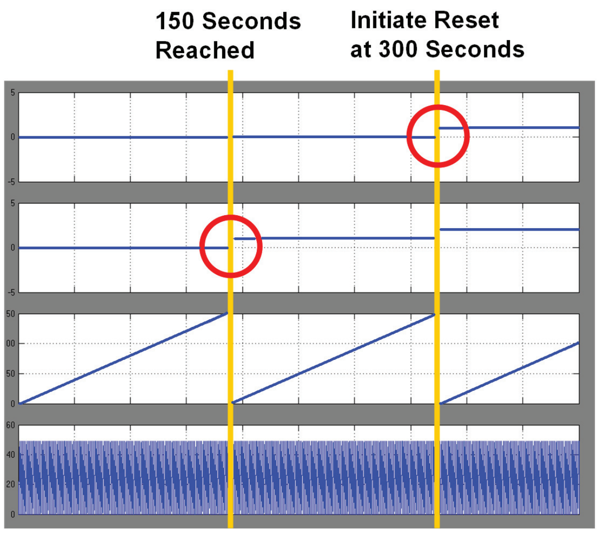

Figure 2: A Simulink simulation of the FPGA watchdog code which shows the counting ramp with a reset/reboot signal initiated after the processor has failed to poll the watchdog for 300 continuous seconds.

REMOTE FIRMWARE UPDATES

The remote firmware update uses the embedded Linux to access the storage area of the firmware (generally flash). Using embedded Linux on the FPGA provides a standard interface for updating FPGA firmware. By relying on the Linux drivers it is not necessary to develop custom drivers thus saving a lot of development time. A standard script is used to write to the specific flash locations to perform the firmware update. The system is then power cycled using the remote reboot control capability.

PLC MODBUS CONTROL

On Linux one can use the Modbus library [1] to create Modbus compatible systems. Modbus is one of the most widely supported PLC communication protocols but other protocols can similarly be adapted to the embedded Linux in a similar way.

Having embedded Linux on an FPGA systems allows it to behave as either a PLC master or slave. This gives a huge amount of flexibility to an FPGA systems adding yet another remote management interface. Since the FPGA system is running embedded Linux there is the possibility to perform other communications (HTTP/SSH/etc) in addition to PLC communications.

Latencies of 6.6ms with 0.38ms deviation (with 995 tests) have been achieved. An error rate of 3.4×10^{-6} errors/write found after approximately 35 million writes. An error is not a catastrophic failure in that the PLC master will just attempt the write again if a failure (such as a timeout) occurs. An average poll rate of 130 Hz is achieved over a period of 72 hours, resulting in a long term polling rate of at least 100 Hz.

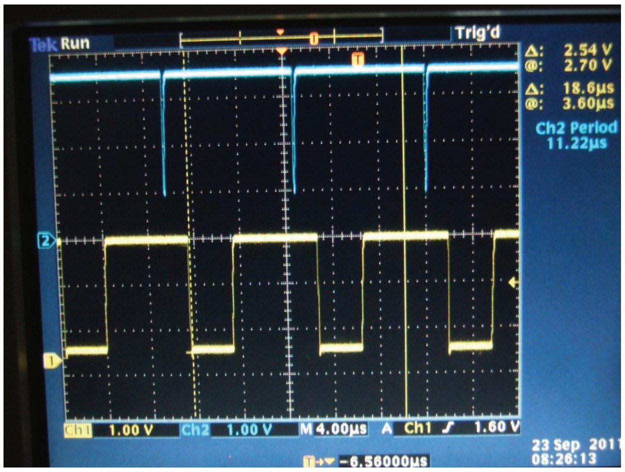

Figure 3: An oscilloscope screenshot of the Modbus FPGA System, with a configurable pulse delay and length controlled by a Schneider Modicon Quantum 140-CPU-651-60 PLC using Modbus.

DEVELOPMENT STRATEGY

EPICS and TANGO

EPICS and TANGO [2] are widely used control systems so developing an FPGA system with this capability would be useful. Some control systems have been designed to communicate over Modbus to EPICS [3, 4, 5] and TANGO [6, 7]. On an FPGA system with a soft processor the limiting factor is the processor speed since EPICS and TANGO are not lightweight. As such future development will initially involve an intermediate protocol (such as Modbus or StarkStream).

Embedded System Development

Using a generic layer between the processor and FPGA logic, it will be easier to create systems that incorporate embedded Linux with minimal effort. The goal is to have a capability to easily “drag-in” embedded Linux and synthesis the system in one step.

CONCLUSIONS

Embedded Linux has been used as a powerful tool for developing more capable FPGA systems using a standard operating system to integrate with custom FPGA logic easily using high level tools such as the MATLAB Simulink environment with the FPGA front end tools (Xilinx System Generator or Altera DSP Builder).

The AXI Bus has been used to achieve fast streaming rates (up to 10GB/s) with memory shared between the FPGA logic and the processor. Remote management has been achieved using embedded Linux to communicate over the StarkStream and Modbus protocols. In future these protocols can be incorporated with standard control systems such as EPICS and TANGO. Full remote system control has been improved by enabling firmware updates and power cycling using embedded Linux and an ATX power distribution board.

ACKNOWLEDGEMENTS

This work was funded by the RCUK Energy Programme under grant EP/I501045 and the European Communities under the contract of Association between EURATOM and CCFE. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

REFERENCES

[1] Stéphane Raimbault. A Modbus library for Linux, Mac OS X, FreeBSD, QNX and Win32. `http://libmodbus.org/`, 2011.

[2] A Götz, E Taurel, J L Pons, P Verdier, J M Chaize, J Meyer, F Poncet, G Heunen, E Buteau, N Leclercq, and M Ounsy. Tango a corba based control system. 2003.

[3] Ha Yu In, Myung-Hwan Chun, Ki Man Ha, Han Yeung Jin, Ha Hwang Woon, Maeng Hyo Jeong, Kang Heung Sik, Tae Kim Do, Sung-Chul Kim, I S Park, Yang Jae Seok, Yong-Sub Cho, and Tae Seol Kyung. Rfq low level RF system for the PEFP 100MeV proton linac. page 3 p, 2004.

[4] P. Gillette, C. Haquin, E. Lcorch, D. Touchard, J.F. Denis, F. Gougnaud, J.F. Gournay, Y. Lussignol, P. Mattei, P. Graehling, J. Hosselet, C. Maazouzi, and C. Olivetto. Preliminary implementation for the new SPIRAL2 project control system. *7th International Workshop on Personal Computers and Particle Accelerator Controls.*

[5] Sang Hoon Nam, Paul Bellomo, Richard Cassel, Seong-Hun Jeong, Kim Sang Hee, Sung-Chul Kim, Raymond Sverre Larsen, Minh N Nguyen, Soung Soo Park, and Jae-Hak Suh. Development of a general purpose power system control board. page 3 p, 2006.

[6] D. Fernández-Carreiras, D. B. Beltran, J. Klora, O. Matilla, R. Montano, M. Niegowski, R. Ranz, A. Rubio, and S. Rubio-Manrique. Alba the plc based protection systems. *Proceedings of ICALEPCS2009, Kobe, Japan*, 2009.

[7] D. Schmied, E. Burtin, J. M. Chaize, M. Hahn, I. Parat, M. Peru, and P. V. Verdier. A different way to survey the ESRF vacuum system. *Proceedings of ICALEPCS2009, Kobe, Japan*, 2009.