Curricular Innovation for Real-Time Embedded Systems Course

Reza Raeisi
Sudhanshu Singh, Graduate Student
California State University, Fresno
Electrical and Computer Engineering Department

Abstract

The objective of this project is to experience and develop rapid prototypes of System-on-chip (SoC) using soft-core processor in the undergraduate laboratory. We will share the experience on a reconfigurable hardware-software co-design environment and µClinux embedded Real Time Operating System (RTOS). A soft core processor is a microcontroller developed in software, which can be synthesized in a FPGA. A soft-core processor approach is ideal for teaching the embedded systems design course in electrical and computer engineering majors. It allows students to customize the exact set of several CPU's peripherals and the interfaces needed for scientific application. The ultimate goal is to disseminate the soft-core processor experience for teaching and learning in the department and enabling students enough flexibility to configure a processor according to their design project needs. The lesson learned from this experience is valued and a new embedded systems design course has been developed in our department.

Introduction

Any device that includes a programmable computer but is not itself a general-purpose computer is termed as an embedded system. It is a special purpose computer which is designed to do certain dedicated functions. FPGA-based embedded system using soft-core processors are adapted more commercially and gained popularity in university teaching. Hence, we are motivated to practice a new pedagogy in learning embedded system where students will have the option of customizing their own peripheral subsystem. Along with this pedagogy we have integrated the µClinux, a Real Time Operating System (RTOS) into classroom/laboratory experience for students to learn embedded systems[1]. Embedded systems are preferred over general computers because of the following advantages:

1. Real-time operation
2. Low manufacturing cost
3. Reliability and security

In general embedded systems are not recognizable as regular computers. Embedded systems typically do not interface with the outside world through familiar personal computer interface devices such as mouse, keyboard and graphic user interfaces. Instead, they interface with the outside world through unusual interfaces such as sensors and other communication links. Most time-constrained resource allocations and task scheduling across spectrum of subsystems such as sensor and actuators processing, communications, CPU, memory and other peripheral devices today are required to use RTOS in order to meet the system response. The practical details of implementing and integrating a RTOS with a soft core processor for interfacing real-time services and data processing through different I/O subsystems are the focus of this paper. Embedded systems are implemented using the following classification of processors:

1. Hard- core processors
2. Soft-core processors

The hard-core processor consumes more power than the soft-core processors and is not flexible. They are also big in size and the possibility of losing hardware connections is higher in hard-core processors. Hard core processors are fast but when we need more data processing, like mathematical operations, data arrays, data sorting, or anything that does not need a quick result we can use soft core processors. Above all, a soft core processor targeting FPGA is flexible because its parameters can be changed at any time by reprogramming the device. [2] There are many different kinds of soft-core processors available in market now days. Some of them are –

1. ARM
2. Xilinx
3. Zylin AS
4. NIOS II

The Altera's NIOS II is available in the Electrical Engineering Department. So we will be concentrating on the NIOS II processor.

NIOS II System

NIOS embedded processors were introduced in 2001 in the electronics industry by Altera as the viable commercial processor specially created for embedded system design in FPGAs. Since then it has been used widely in the industry. It is a 32 bit soft-core processor, which is defined in a hardware descriptive language. It can be implemented in Altera's FPGA devices (DE2) by using the Quartus II CAD system. The soft-core nature of the NIOS II processor lets the system designer specify and generate a custom NIOS II core, tailored for his specific application requirements. [3]

NIOS II processors allow us to-

1. Customize the CPUs, peripherals and the interfaces needed for aforementioned application.
2. Increase performance by implementing Real-time embedded system applications.
3. Lower laboratory cost by not spending additional money on hardware microcontroller board.

The NIOS II processor can be used with a variety of other components to form a omplete system. Altera's DE2 Development and educational board contain several components that can be integrated into a NIOS II system. An example of such system is shown in figure 1. The NIOS II processor has Reduced Instruction Set Computer (RISC) architecture. Its arithmetic and logic operations are performed on operands in the general purpose registers. The data is moved between the memory and these registers by means of Load and Store instructions. The word length of the NIOS II processor is 32 bits. All registers are 32 bits long. The NIOS II architecture uses separate instruction and data buses, which is often referred to as the *Harvard* architecture. [4]

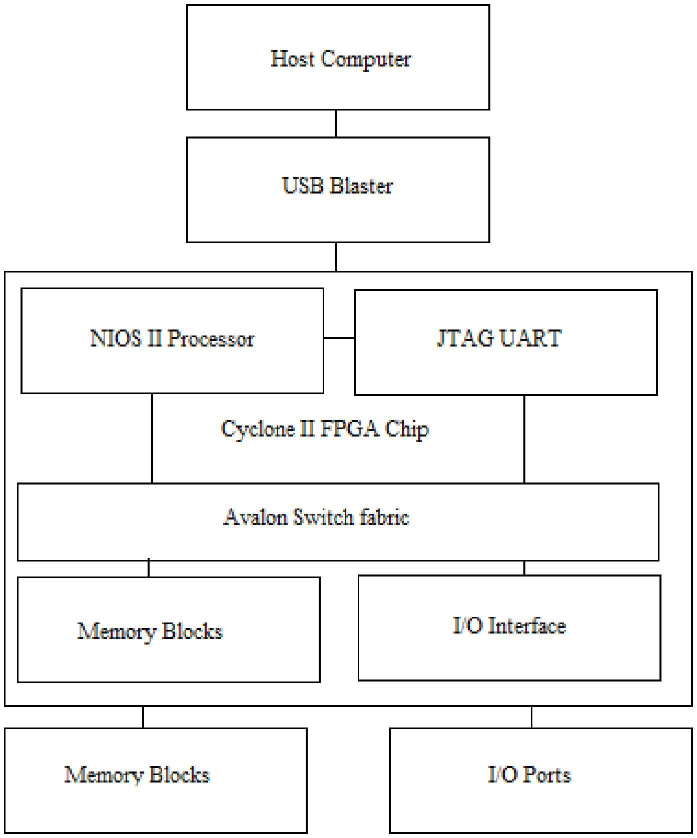


Figure 1-Nios II system implemented on the DE2 board

µClinux

Real-time and embedded systems operate in constrained environments in which computer memory and processing power are limited. They often need to provide their services within strict time deadlines to their users and to the surrounding world. It is these memory, speed and timing constraints that dictate the use of real-time operating systems in embedded software. RTOS kernels hide from application software to the low-level details of system hardware, and at the same time provide several categories of services to theapplication software. These include: task management with priority-based preemptive scheduling, reliable inter task communication and synchronization, non-fragmenting dynamic memory allocation, and basic timer services. [10] The µClinux kernel supports multiple of different CPU platform including Altera's NIOS II architecture. The main advantage of this OS is that it is an open source project and it is smaller than the regular Linux kernels. Most features of Linux kernel are available, like process control, file system, networking, and device drivers. [6]

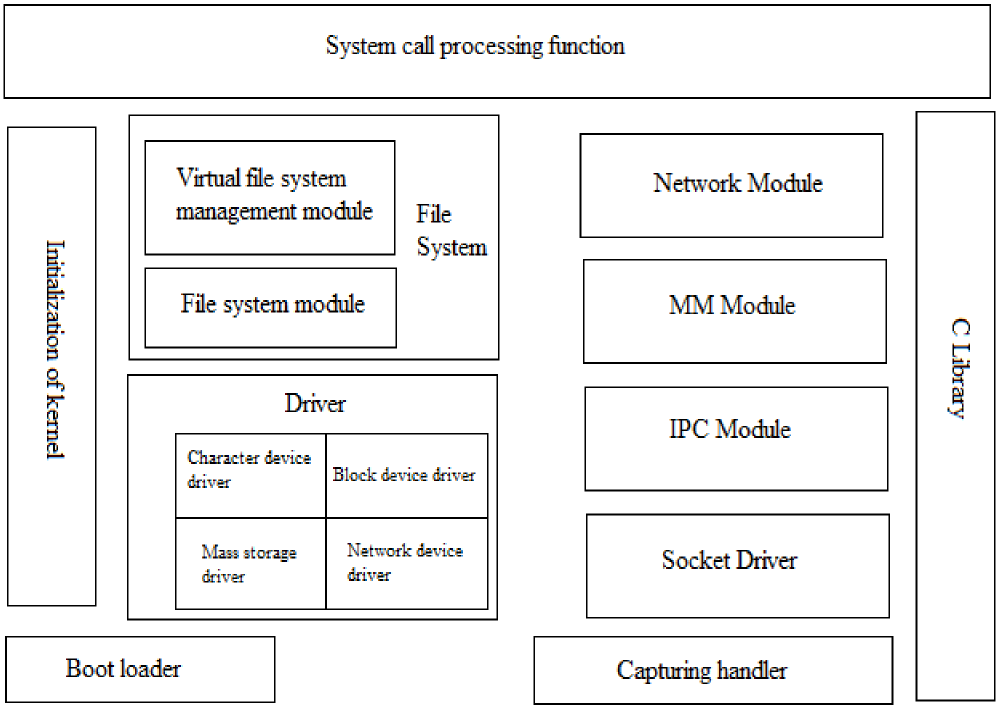


Figure 2- Architecture of µClinux [7]

µClinux Implementation on Soft-Core Processor

Altera's DE2 board is shown in Figure 3. We will use this board as a vehicle for teaching and learning the new embedded systems course. Students will experiment the design and implementation of RTOS using NIOS II soft-core processor that supports flexible memory option and I/O device combination. Design project to deal with real-time task using interrupt programming, connections with I/O devices such as audio, video, USB, network and memory are examples of experiments that are implemented.

NIOS II Integrated Development Environment (IDE) is the standalone program that helps us to accomplish our task of implementing µClinux over the FPGA device. NIOS II IDE 9.0 is the latest version of the software and can be downloaded from the Altera website. The distribution for µClinux can be obtained from http://nioswiki.jot.com/WikiHome/. The Nios community develops and releases the latest kernels according to Altera software release. Because of the licensing issue we have build the uClinux kernel in Linux environment and then transfer the kernel image into the windows to do the rest of our project. Running the µClinux in DE2 board requires two steps. First the FPGA must be configured to implement the NIOS II processor system, and second the µClinux kernels image must be downloaded into SDRAM on the DE2 board. Both configuration steps can be accomplished via the NIOS II 9.0 command shell.

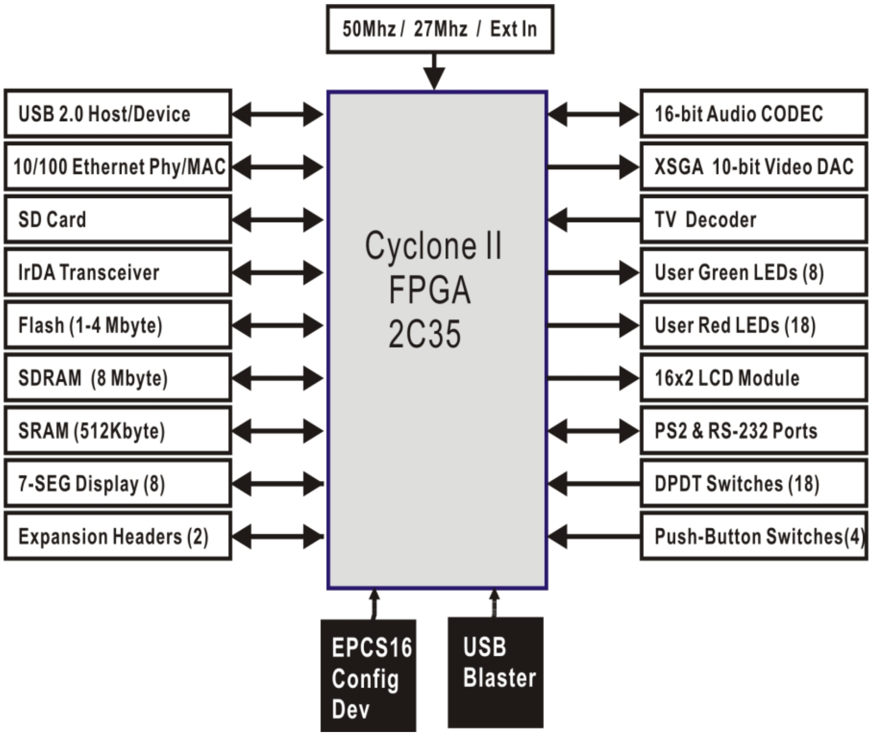
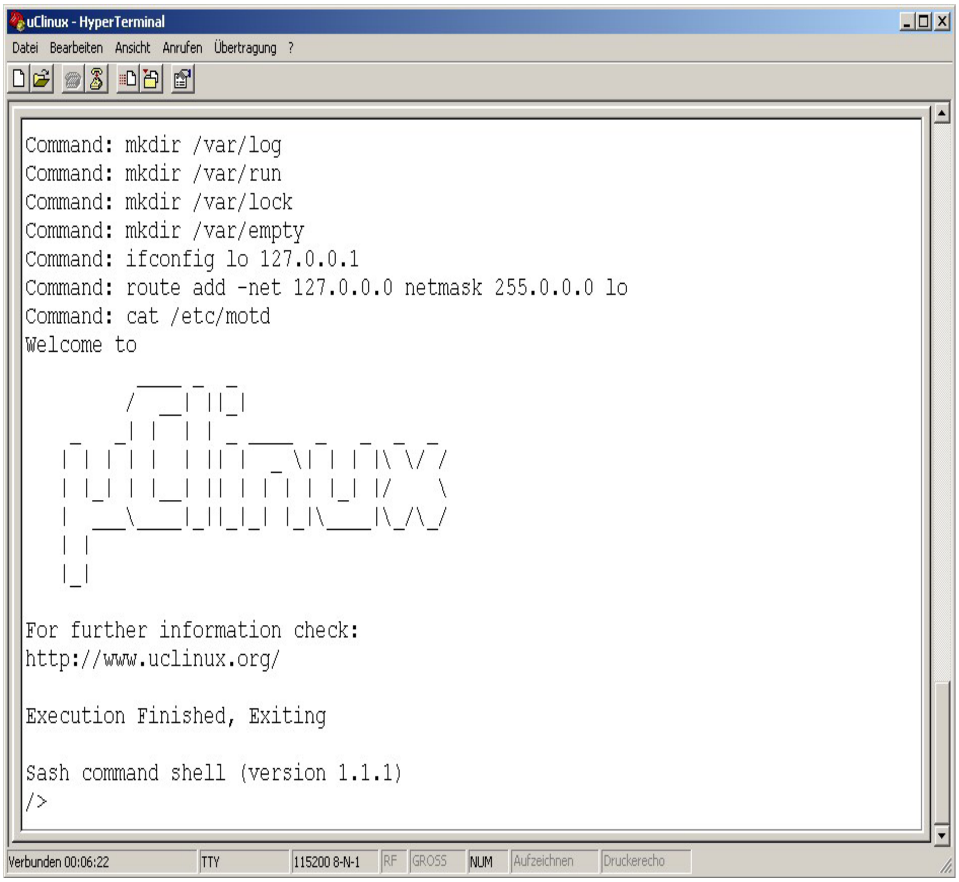


Figure3 -Altera DE board block diagram [2]

Before starting the configuration of the DE board the power cable should be well connected, the DE2 board should be turned ON and USB cable is connected between the PC and the USB blaster port on the DE board. We used an existing NIOS II project from the demonstrations directory of the enclosed DE2 CD-ROM. We have chosen DE2_NIOS_HOST_MOUSE_VGA project. We used *wget* command in Linux terminal to download the µClinux distribution. A basic µClinux kernel image is built uing the *make menuconfig* command. The built image is located on nios-linux/uClinux-dist/image/zimage. Zimage is nothing but a compressed form of a kernel image. The Linux kernel takes care of expanding the image at boot. On Linux systems, vmLinux is a statically linked executable file that contains the Linux kernel in one of the executable file formats supported by Linux, including ELF, COFF and a.out. To configure the FPGA and download the zImage to the processor we will write the following command steps in NIOS II 9.0 command shell.[5]

step 1. Configure the FPGA,
nios2-configure-sof DE2_NIOS_HOST_MOUSE_VGA.sof
step 2. Download and run the kernel image,
nios2-download -g zImage_DE2_NIOS_HOST_MOUSE_VGA_v1.6

After the kernel image is downloaded into the DE2 board, µClinux in *nios2-terminal* is active and ready as shown in Figure 4.



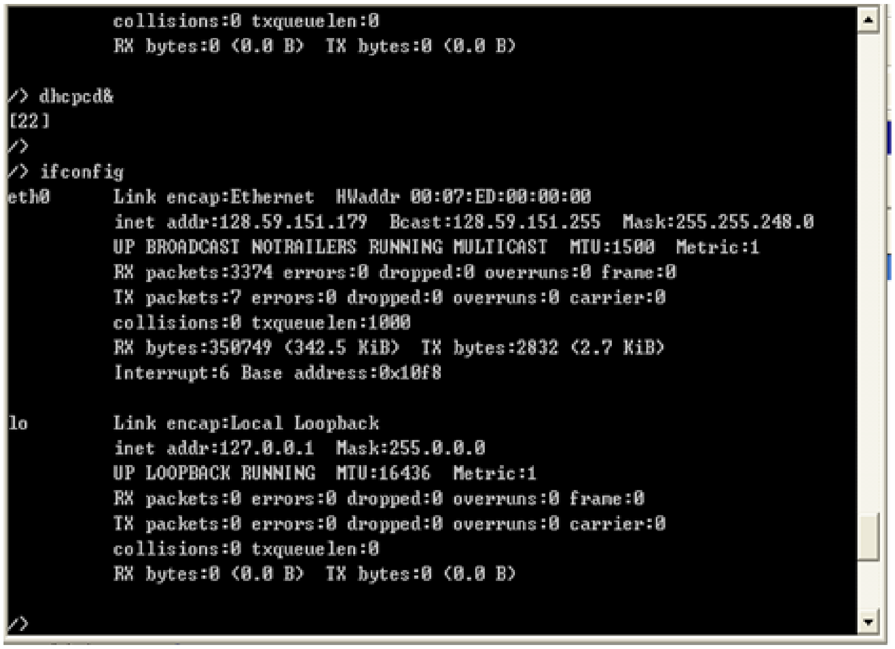
```
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: mkdir /var/empty
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.0.0.0 lo
Command: cat /etc/motd
Welcome to
          ____ _  _
         /  __| ||_|
    _   _| |  | | _ ____  _   _  _  _
   | | | | |  | || |  _ \| | | |\ \/ /
   | |_| | |__| || | | | | |_| |/    \
   |  ___\____|_||_|_| |_|\____|\_/\_/
   | |
   |_|

For further information check:
http://www.uclinux.org/

Execution Finished, Exiting

Sash command shell (version 1.1.1)
/>
```

Figure 4- A basic uClinux kernel image

Application

The µClinux is configured on our NIOS II system and is ready for use. The next step is customizing kernel and adding a user application. We will log into Linux platform and will do the *make menuconfig*. One of the application experiments is to complete the Ethernet interfacing, and connecting our board to the outside world. This will be done by invoking ftp and telnet. We also activate the Ethernet network support while doing the *make menuconfig.*

To check the Ethernet connection we will run the *ifconfig* command. *ifconfig* command allows the operating system to setup network interfaces and allow the user to view information about the configured network interfaces. [6] If a valid IP address is displayed after the label *inet addr* as shown in figure 5, then the DE2 board is successfully communicating on the network.



```
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/> dhcpcd&
[22]
/>
/> ifconfig
eth0      Link encap:Ethernet  HWaddr 00:07:ED:00:00:00
          inet addr:128.59.151.179  Bcast:128.59.151.255  Mask:255.255.248.0
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3374 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:350749 (342.5 KiB)  TX bytes:2832 (2.7 KiB)
          Interrupt:6 Base address:0x10f8

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/>
```

Figure 5- ifconfig result

Furthermore, the NIOS II processor developed system was used to implement variety of experiments. Examples are writing a device driver for the LCD controller interface and I/O and interrupt programming. Figure 6 shows an I/O interface waveform generation of a 70% duty cycle as displayed on an oscilloscope. This experiment was implemented using C Language.

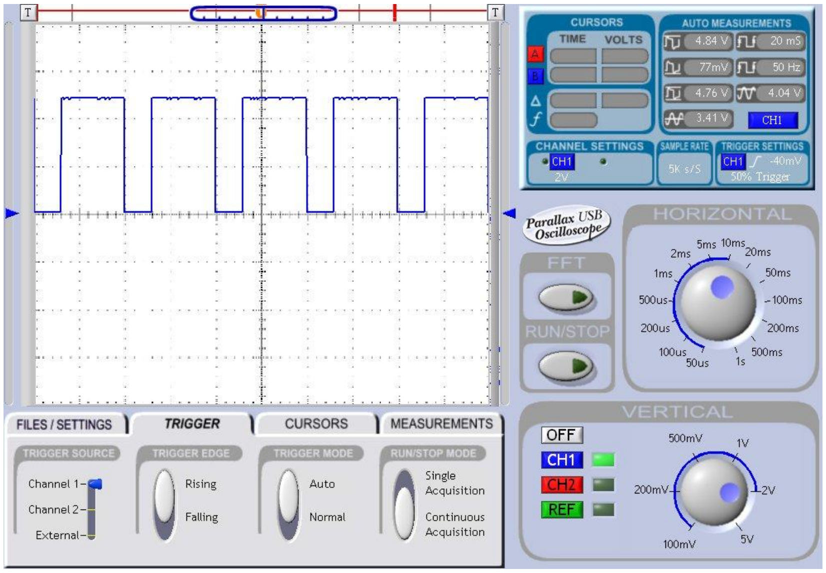


Figure 6- Output in oscilloscope

Conclusion

As a result of this project, a new course ECE 178, embedded systems, course has been developed. This course will demonstrate soft-core processor embedded system which will be implemented on a real time operating system over the FPGA's. During this project, we have also shown how to add applications in the Linux kernels. The Ethernet connection application and some other I/O and interrupt programming has been implemented. The scope of this project is very vast and we have set the platform now for teaching this new approach for learning embedded systems concepts.

Bibliography

1. "Using an FPGA Processor Core and Embedded Linux for Senior Design Projects"- Tyson S. Hall, James O. Hamblen. IEEE International Conference on
   Volume, Issue, 3-4 June 2007 Page(s):33 – 34.
2. "Soft Core Processor Design"- Franjo Plavec, Graduate Department of Electrical and Computer Engineering, University of Toronto
3. Altera Homepage, www.altera.com
4. Introduction to the Altera SOPC Builder- Quartus II Development Software Literature
5. A book on "Rapid prototyping of digital systems"- by J.O Hamblem, T.S Hall and M.D. Furman.
6. "Device drivers and Test application for a SOPC solution with Nios II soft-core processor and μClinux" - Philipp Lutz
7. An Embedded System with uClinux based on FPGA- Zongqing Lu1 , Xiong Zhang2 ,Chuiliang Sun3*1School of Electronic Science and Engineering, Southeast University, Nanjing, JiangSu, 210096, China*
8. "Embedded uClinux, the Altera DE2, and the SHIM Compiler"- Wei-Chung Hsu, David Lariviere, and Stephen A. Edwards