CHAPTER 18

Operating System Support for SOPC Design

Embedded operating systems can provide a wide variety of services to support application developers. The DE boards and Nios II soft-core processor support several embedded operating systems including µClinux, which is shown here driving a VGA monitor.

18 Operating System Support for SOPC Design

Many electronic devices that contain a processor now require complex software that needs support for multitasking, synchronization of tasks, a wide range of I/O devices, scheduling and buffering of I/O operations, memory management, graphics displays, file systems, and/or networking. Developing custom code (like was done in Chapter 16) that can provide all of these services is an expensive and time consuming task. For example, one recent cell phone design contained over five million lines of code. Few projects will have the time and funding needed to develop all of this code entirely on their own. In cases such as this, it makes economic sense to use an existing operating system. An operating system (OS) can provide a wide array of features and services including those listed above. Software developers are more productive when an operating system is present since they can work at a higher level of abstraction by using the operating system's Application Programming Interface (API) calls to access the services provided by the OS.

The development time and costs saved more than offsets the licensing fees for the operating system. The typical commercial embedded OS license fees run only a few dollars per device and several open source operating systems are free. Some very simple low-end devices might not need an OS, but complexity constantly increases with each new generation of devices.

The traditional desktop operating systems require a memory management unit (MMU) that provides hardware support for virtual memory addressing. This allows the OS to provide a kernel memory address space and a user memory address space. An MMU requires extensive processor hardware and the current FPGA soft-core processors do not contain an MMU. Specially designed embedded operating systems that have been targeted for small MMU-less devices are available[11]. They use only a single linear memory address space. An embedded OS typically requires less processing power and has a smaller memory footprint than a desktop OS. It also is likely to support booting from flash memory, produce ROMable code (i.e., generates code that can run from ROM memory), and to have I/O device drivers for the I/O devices that are more commonly found in small devices. A C/C++ compiler is typically provided with the OS.

For these reasons, most complex embedded devices use an existing embedded operating system. Embedded operating systems typically are developed largely in C/C++ and normally come bundled with a C/C++ compiler, assembler, and debugging tools to assist designers in developing application programs and testing the device. Embedded system development tools must also support program execution using code stored in non-volatile memory such as ROM or Flash memory.

Figure 18.1 shows the response embedded designers gave in a 2006 survey to the question "what languages do you use to develop embedded systems?" The

[11] Embedded Linux System Design and Development by P. Raghavan, Amol Lad, and Sriram Neelakandan, 2005.

C family of languages is clearly used for the majority of embedded systems development. For assembly language, the response indicated that around one third of embedded systems designers still have to use assembly language for at least some small portion of their designs. Other studies have indicated that assembly language is typically less than five to ten percent of the total code in current devices.

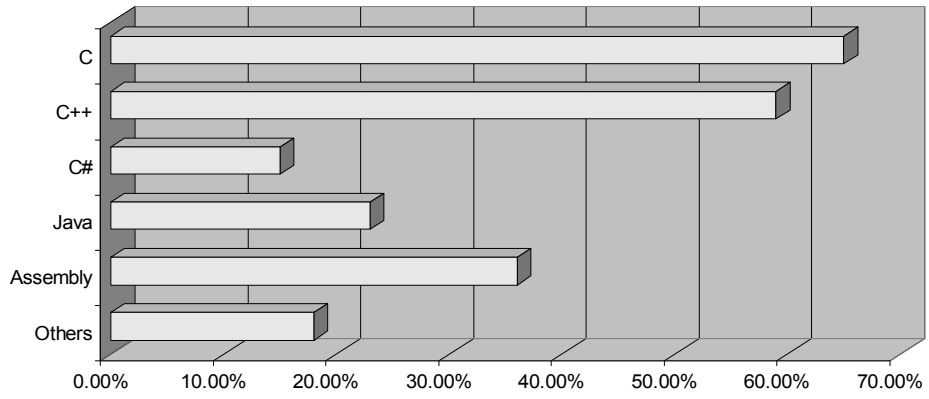

Figure 18.1 Programming languages used to develop embedded devices. [12]

Figure 18.2 indicates that 70% of new embedded devices contain an operating system. In those devices with an operating system, the most popular choice is an off-the-shelf commercial operating system. Commercial operating systems and commercial distributions of open source operating systems may have better OS development tools, a wider selection of device drivers, and an experienced consulting staff available to help with development. The real economic value of these services should not be overlooked when choosing an embedded OS for your project. One recent study claims that the total product development cost can actually turn out to be higher in some cases for an open source OS when software development time, salaries, and other required license fees are all included.[13]

The embedded open source community is continuing to grow. When using an open source operating system, the costs and development time can be minimized by carefully selecting a platform (i.e., the processor and peripheral devices) that has a mature supporting code base within the open source community.

[12] Language Survey data is from the 2006 annual embedded market survey conducted by EETimes and Embedded Systems Design Magazine (www.embedded.com). Articles published throughout the year discuss the results of the annual survey and examine the implications and trends found in the data.

[13] The Total Cost of Development study is available at www.embedded-forecast.com. This study surveyed several embedded product development projects to compare costs when using Open Source versus a Commercial OS.

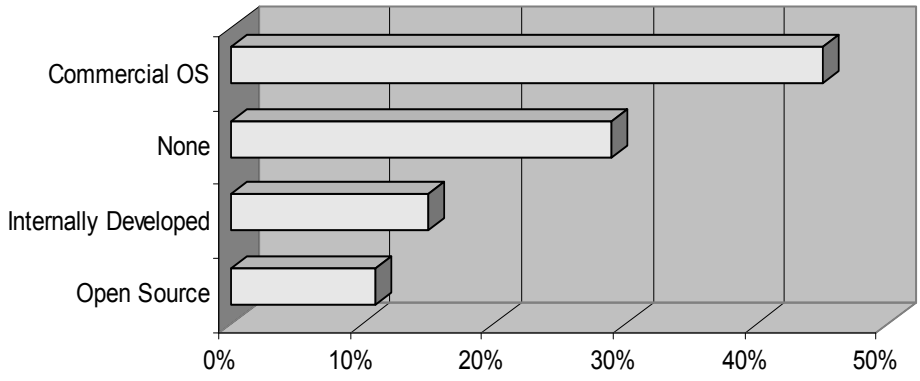

Figure 18.2 Real-time OS kernels used in new embedded designs.[14]

18.1 Nios II OS Support

To support SOPC-based designs, a number of commercial third-party and open source operating systems have been ported to the processor cores running on current FPGAs. Several embedded operating systems are supported on the Nios II soft-core processor as seen in Table 18.1. An assortment of both open source and commercial operating systems are available. Many can be developed using the Nios II IDE. OS development often lags a bit behind new hardware developments, so the OS may require an older version of the IDE tools. Since the Nios II processor does not include MMU hardware, virtual memory addressing is not supported. An MMU-less embedded operating system is required that uses flat memory addressing.

Most of the operating systems in Table 18.1 claim to be Real-Time Operating Systems (RTOS) . An RTOS is an OS that has a fixed upper bound on the interrupt latency and service time. A real-time system must respond to external events (i.e., interrupts) in a limited amount of time or the system will fail. There are two classes of real-time systems soft real-time and hard real-time. In soft real-time systems, critical tasks get priority and their response rate typically meets the timing constraint but is not guaranteed to always respond within the defined limit. A typical soft real-time example is a multimedia player. The player could occasionally skip a video frame or audio sample and a user might not even notice as long as it ran correctly the vast majority of the time. A hard real-time system, however, must guarantee a specified response rate or the system will fail. The average response time in a hard real-time OS is typically in the range of 0.5ms to 10ms. Some hard real-time systems must also

[14] OS Survey data is from the 2006 annual embedded market survey conducted by EETimes and Embedded Systems Design Magazine (www.embedded.com).

guarantee a narrow margin of response time variation. For example, a response rate that varies more than 10% can also cause some systems to fail.[15]

Traditional desktop operating systems such as Windows and most Linux distributions are not hard real-time OSs, since they can occasionally have very long interrupt response times that are well outside the range currently expected in an RTOS. In an RTOS, the OS kernel code and user application code cannot disable interrupts for long periods of time. To convert a traditional OS to an RTOS normally requires rewriting all of the kernel code with this feature in mind.

Table 18.1 OS support for the Nios II Processor

OS	RTOS	OS Type	Company Name	Nios II IDE Plug-in
eCos	Yes	Open Source	eCosCentric	-
Euros RTOS	Yes	Commercial	Euros	-
Erika Enterprise	Yes	Commercial	Evidence	Yes
ThreadX	Yes	Commercial	Express Logic	Yes
Nucleus Plus	Yes	Commercial	Mentor Graphics	-
MicroC/OS-II[16]	Yes	Commercial	Micrium	Yes
embOS	Yes	Commercial	Segger	-
osCAN[17]	Yes	Commercial	Vector Informatik	-
µClinux	-	Open Source	Microtronix	Yes
µClinux	-	Open Source	Community Supported (based on Microtronix port)	-

eCos, MicroC/OS-II, and µClinux are three of the more popular OS choices available for the Nios II processor. These will be briefly examined in the next three sections followed by a more detailed look at running the µClinux kernel on the DE board.

18.2 eCos

eCos (embedded Configurable operating system) is an open source, royalty-fee RTOS designed for embedded systems and applications which need only one process with multiple threads. The OS can be can be customized for application requirements to deliver the best possible run-time performance and minimize hardware needs. It is programmed in the C programming language and has compatibility layers and APIs for POSIX and µITRON[18]. eCos was designed

[15] Based on the definition and timing that was adopted by the Open, Modular, Architecture Control (OMAC) user group: A hard real-time system is a system that would fail if its timing requirements were not met; a soft real-time system can tolerate significant variations in the delivery of operating system services like interrupts, timers, and scheduling

[16] Included with the Nios II Embedded Design Suite, but licensed separately by Micrium.

17 OSEK/VDX compliant. OSEK/VDX is an open standard of the automotive industry.

[18] More eCos information and examples can be found in Programming Embedded Systems, Second Edition With C and GNU Development Tools by Michael Barr and Anthony Massa.

for devices with memory size in the tens to hundreds of kilobytes and with real-time requirements.

eCos was developed by Cygnus Solutions and was later bought by Red Hat. In 2002, Red Hat ended development of eCos and the developers then formed, eCosCentric, in order to continue development and provide support for eCos. Red Hat agreed to transfer its eCos copyrights to the Free Software Foundation in October 2005. eCosCentric has a version of their eCosPro distribution that has been ported to the Nios II processor.

18.3 µC/OS-II

µC/OS-II is a highly portable, scalable, preemptive, real-time, multitasking kernel for microprocessors and microcontrollers. µC/OS-II is written in C[19]. Since its introduction in 1992, µC/OS-II has been used in a wide array of products including cell phones, climate controls, audio/video processors, credit card processing units, and electrical instrumentation.

µC/OS-II can manage up to 255 tasks and provides services such as semaphores, mutual exclusion semaphores, event flags, message mailboxes, message queues, task management, fixed-size memory block management, and time/timer management.

µC/OS-II has been ported to the Nios II processor (Altera usually refers to it as MicroC/OS-II). Altera distributes MicroC/OS-II in the full commercial version of the Nios II EDS software and supports the Nios II port of the MicroC/OS-II kernel. Examples of MicroC/OS-II programs are included with the Nios II EDS software.

The license for the Nios II MicroC/OS-II port is available from Micrium. Micrium offers free OS licensing for universities and students. A MicroC/OS-II reference manual and tutorial are available on the DVD and from Altera.

As seen in Figure 18.3, the MicroC/OS-II kernel operates on top of the hardware abstraction layer (HAL) system library for the Nios II processor. By using the HAL, programs based on MicroC/OS-II are more portable to other Nios II hardware systems and are somewhat flexible with respect to hardware changes. MicroC/OS-II programs can use all HAL services and use the HAL APIs described earlier in this text.

[19] The internal architecture of µC/OS-II is described in "µC/OS-II, The Real-Time Kernel" by Jean J. Labrosse.

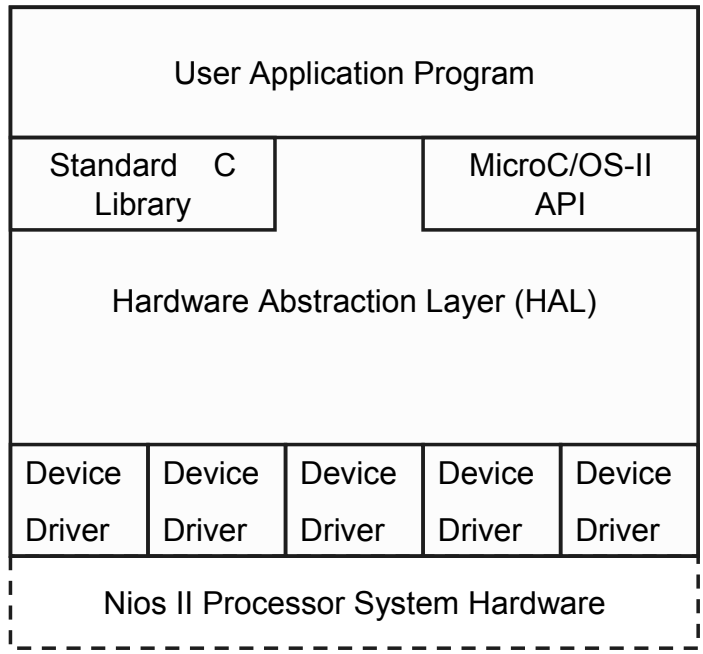

Figure 18.3 MicroC/OS II System Architecture.

18.4 µClinux

The original "micro controller" Linux, µClinux, was derived from the Linux 2.0 kernel and intended for microcontrollers like the Nios II processor that do not have a Memory Management Unit (MMU). µClinux was first ported to the Motorola MC68328 DragonBall Integrated Microprocessor. The first target system to successfully boot was a PalmPilot device in 1998. Currently, µClinux includes Linux kernel releases for 2.0, 2.4, and 2.6 as well as a collection of user applications, libraries and tool chains.

After releasing the initial version, a developer community soon sprung up to extend work to newer kernels and other microprocessor architectures. In 1999, support was added for the Motorola (Freescale) ColdFire family of embedded microprocessors and the ARM processor. Recently, it has been ported to the Nios II processor.

The µClinux project community continues to develop patches and supporting tools for using Linux on microcontrollers. µClinux has support for several architectures and forms the basis of many embedded products. Examples include network routers, gateways, security cameras, DVD and MP3 players, VoIP phones, scanners, and card readers. The NiosForum (www.niosfourm.org) supports an open source version of µClinux that has been ported to the Nios II Processor.

18.5 Implementing the µClinux on the DE Board

To illustrate the benefits of operating systems in embedded systems, the next few sections will step the reader through the design and implementation of a Nios II system running µClinux on a DE board.

There are two main distributions of µClinux that work on the Nios II processor. The first distribution is controlled and released by Microtronix (www.microtronix.com). They provided the original kernel code modifications to support the Nios II processor. Their distribution also integrates into the Nios II IDE design environment. However, as of the publication date of this book, Microtronix's latest version, release 1.4, does not work with Nios II 7.1. Release 1.4 was originally designed for Nios II 5.0 and there are reports of users on the Nios Community Forum (www.niosforum.com) successfully using it in Nios II 6.0. Microtronix distributions are freely available for download on the Nios Community Forum under Downloads. (To view the Downloads section, you must be a registered user of the forum's website.) The primary advantage of this distribution is its tight integration into the Nios II IDE and build environment; however, Microtronix releases tend to lag behind Altera's software releases.

A second distribution of µClinux has been developed and released on the http://nioswiki.jot.com/WikiHome/ website by a group of developers who frequent the Nios Community Forum. These developers took Microtronix's distribution and updated it with support for the latest Linux kernel release, additional device drivers, and a larger collection of libraries and user applications. Instead of integrating µClinux developing and compilation into the Nios II IDE, this distribution includes uClinux-dist, a Linux-based build environment. You must have access to a computer running the Linux operating system to install, customize, and build this distribution of the µClinux kernel.

The remainder of this chapter will focus on this second distribution of µClinux, which will be referred to as µClinux for Nios II. While full details of the development of a custom µClinux kernel for the DE boards is beyond the scope of this book, a pre-built kernel is provided for the reader's use and exploration. Readers who want to modify the kernel image should consult the http://nioswiki.jot.com/WikiHome/ website, which includes the uClinux-dist distribution as well as the necessary patches for Nios II freely available for download. In addition, this website has a number of tutorials and application notes available that step through the customization of various devices and services in µClinux (including information specific to the DE2 board). Additional help is available in the Nios Community Forum (www.niosforum.com).

18.6 Hardware Design for µClinux Support

When compiling µClinux for Nios II, the Nios II peripheral template file (.ptf) for the specific Nios II processor configuration being used must be provided. The .ptf file contains the processor and peripheral details for a given Nios II configuration including the IRQ mappings, memory address ranges for the

memory and I/O peripherals, and a list of the names and types of the peripherals selected in the SOPC Builder. During the µClinux build process this file is used to generate a nios2_system.h file that contains C definitions for all of the peripheral names, addresses, IRQs, etc. The kernel and device driver source code includes this file and relies on it to access and configure the peripherals. Many of the device drivers use a hardcoded name for the peripheral being accessed. Thus, designing a Nios II system to be compatible with µClinux for Nios II requires that specific names be used for the peripherals in SOPC Builder. Failure to properly name devices (or edit the device driver files to match your Nios II system) will result in a loss of functionality within µClinux. A Nios II reference design for use with µClinux for Nios II is available on the DVD in the /DE*x*/chap18 directory (for the DE1 and DE2 boards only). This system was developed in a method similar to that in Chapter 17. However, specific peripheral names and configuration settings along with a number of additional peripherals were used to make this Nios II system compatible with µClinux for Nios II. The final SOPC Builder configuration is shown in Figure 18.4.

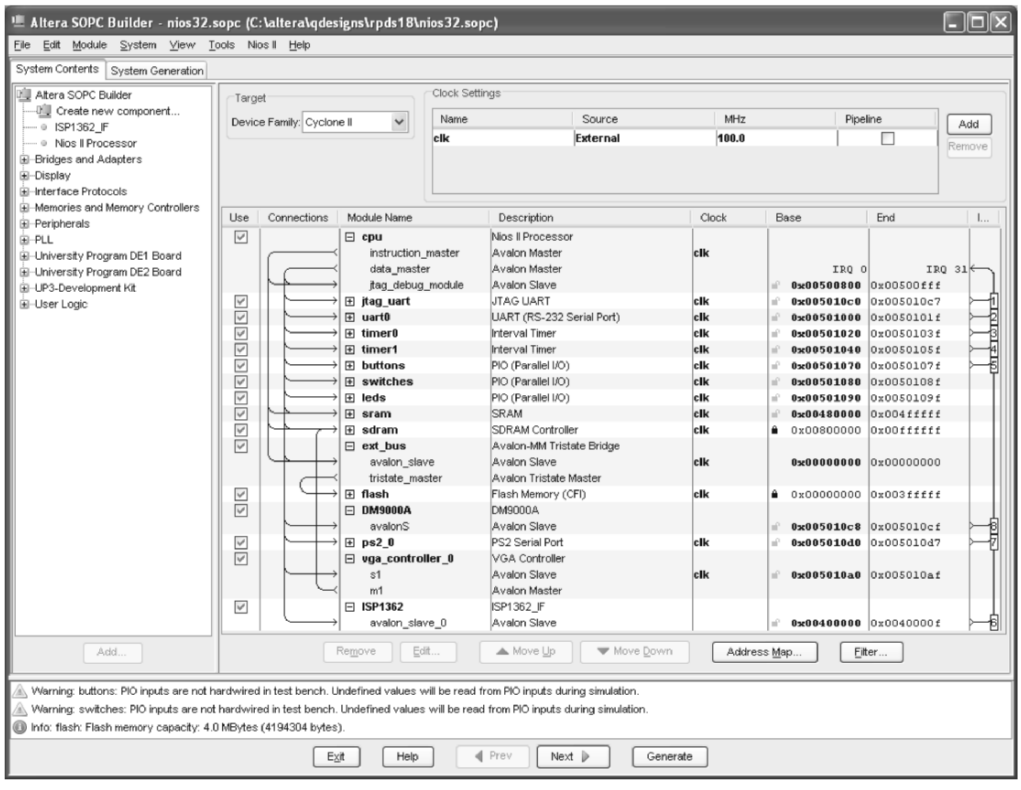

Figure 18.4 This is the completed µClinux compatible Nios II design in SOPC Builder.

18.7 Configuring the DE Board

Configuring the DE board to run µClinux requires two steps. The FPGA must be configured to implement the Nios II processor system, and then the µClinux kernel image must be downloaded into SDRAM. Both configuration steps can be accomplished via the **Nios II 7.1 Command Shell**, which is available on the Windows Start menu **Altera ⇨Nios II EDS 7.1 ⇨Nios II 7.1 Command Shell**. The Command Shell window is a Cygwin environment so it will behave similarly to a Linux shell. Change directories to get to the folder where the reference design for Chapter 18 resides. For example, if you copied the chap18 folder from the DVD to the C:\altera\qdesigns\ directory, then you would need to type the following commands:

```
cd c:
cd altera/qdesigns/chap18/complete
```

Before configuring the FPGA, verify that the DE board is powered on and a USB cable is connected between your PC and the USB Blaster port on the DE board. Configure the FPGA with the .sof file provided with the reference design by running the following command:

```
nios2-configure-sof rpds18.sof
```

If you have multiple JTAG cables or are programming a board with multiple FPGAs on it, then you will need to specify some of the optional parameters for the nios2-configure-sof command. Run **`nios2-configure-sof --help`** to view additional information about the command-line options for this program.

The reference µClinux kernel provided here supports a VGA monitor, a PS/2 keyboard or mouse, USB HID devices (i.e., keyboards, mice), USB memory sticks, Ethernet communication, and serial UART console terminal. (Not all of these devices are available on the DE1 board.) Attach a 9-pin serial cable between the DE board and your PC. Use HyperTerminal or another terminal program (BPS: 115200, Data Bits: 8, Parity: None, Stop Bits: 1, Flow Control: None) to attach to the µClinux console. The console will allow you to view the boot-up sequence and interact with the µClinux kernel via a command prompt. Attach a VGA monitor and PS/2 keyboard to the respective ports on the DE board. If you are using a DE2 board, you may also attach a USB mouse and Ethernet cable to the respective ports on the DE2 board. Because there is only one PS/2 port, it is recommended that you attach a keyboard to the PS/2 port and a mouse to the USB port or vice versa. A USB hub can be used to if you need to attach multiple USB devices to the DE2 board.

Once you have connected the appropriate cables, the µClinux kernel image must be downloaded into the SDRAM chip and processor started executing at the proper memory address. Type the following command to do both of these actions at once:

```
nios2-download -g zImage
```

An illustration of the Nios II Command Shell after the complete sequence of commands have executed is shown in Figure 18.5. When the kernel image download is complete, the processor will immediately begin executing at memory address 0x00D00000. μClinux will start booting immediately and the series of messages shown in Figure 18.6 should appear in your terminal window. (Some messages will vary for the DE1 board because device drivers for peripherals not available on the DE1 board will not load.)

```
/cygdrive/c/altera/71/nios2eds/examples
[SOPC Builder]$ cd c:
/cygdrive/c
[SOPC Builder]$ cd altera/qdesigns/rpds18/complete/
/cygdrive/c/altera/qdesigns/rpds18/complete
[SOPC Builder]$ nios2-configure-sof rpds18.sof
Searching for SOF file:
in .
  rpds18.sof

Info:
*******************************************************************
Info: Running Quartus II Programmer
Info: Command: quartus_pgm --no_banner --mode=jtag -o p;rpds18.sof
Info: Using programming cable "USB-Blaster [USB-0]"
Info: Started Programmer operation at Wed Aug 01 14:11:26 2007
Info: Configuring device index 1
Info: Device 1 contains JTAG ID code 0x020B40DD
Info: Configuration succeeded -- 1 device(s) configured
Info: Successfully performed operation(s)
Info: Ended Programmer operation at Wed Aug 01 14:11:28 2007
Info: Quartus II Programmer was successful. 0 errors, 0 warnings
    Info: Allocated 51 megabytes of memory during processing
    Info: Processing ended: Wed Aug 01 14:11:28 2007
    Info: Elapsed time: 00:00:09
/cygdrive/c/altera/qdesigns/rpds18/complete
[SOPC Builder]$ nios2-download -g zImage
Using cable "USB-Blaster [USB-0]", device 1, instance 0x00
Pausing target processor: OK
Initializing CPU cache (if present)
OK
Downloaded 1775KB in 22.6s (78.5KB/s)
Verified OK
Starting processor at address 0x00D00000
/cygdrive/c/altera/qdesigns/rpds18/complete
[SOPC Builder]$
```

Figure 18.5 The sequence of commands and output that appear when programming the FPGA via the Nios II Command Shell is shown here.

```
Uncompressing Linux... Ok, booting the kernel.

Linux version 2.6.19-uc1 (tyson@tyson-ubuntu) (gcc version 3.4.6) #125
PREEMPT Tue Jul 31 14:10:46 EDT 2007

uClinux/Nios II
Altera Nios II support (C) 2004 Microtronix Datacom Ltd.

setup_arch: No persistant network settings signature at 003F0000
Built 1 zonelists.  Total pages: 2032
Kernel command line:
PID hash table entries: 32 (order: 5, 128 bytes)
Console: colour dummy device 80x25
Dentry cache hash table entries: 1024 (order: 0, 4096 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Memory available: 5044k/8192k RAM, 0k/0k ROM (1971k kernel code, 1084k
data)
...
fb0: Altera frame buffer device, using 600K of video memory
altps2 : base 805010d0 irq 7
io scheduler noop registered
io scheduler deadline registered (default)
NIOS serial driver version 0.0
ttyS0 (irq = 2) is a builtin NIOS UART
<DM9KS> I/O: 805010c8, VID: 90000a46
usbmon: debugfs is not available
driver isp1362-hcd, 2005-04-04
isp1362-hcd isp1362-hcd: ISP1362 Host Controller
isp1362-hcd isp1362-hcd: new USB bus registered, assigned bus number 1
isp1362_hc_reset:
...
Welcome to
          ____ _  _
         /  __| ||_|
    _   _| |  | | _ ____  _   _  _  _
   | | | | |  | || |  _ \| | | |\ \/ /
   | |_| | |__| || | | | | |_| |/    \
   |  ___\____|_||_|_| |_|\____|\_/\_/
   | |
   |_|

For further information check:
http://www.uclinux.org/
...
Sash command shell (version 1.1.1)
/>
```

Figure 18.6 This is an abbreviated version of the messages that print out in the console terminal when µClinux is booting.

18.8 Exploring µClinux on the DE Board

Once µClinux has booted properly, you can use console window on your PC (connected to the FPGA via a serial cable) or the terminal window display on the VGA monitor and a PS/2 keyboard attached to the DE2 board to explore µClinux. Standard Linux commands such as ls, cd, more, ps, kill, free, and stty all work. In particular, type **`ls -al`** to get a listing of the files and directories on the root filesystem (being implemented in SDRAM). Because the root filesystem, program memory, and data memory are all utilizing the single 8MB SDRAM chip on the DE board, this implementation is very tight on memory. Before running a new command, be sure to run the **`free`** command to check how much memory is available. If too many processes are running simultaneously, the kernel will run out of memory and crash. Table 18.2 contains a summary of useful commands and programs are supported in this µClinux kernel image. Additional commands are available in the /bin directory and via the busybox application (type **`busybox`** for more information).

Table 18.2 Selected Supported Commands in µClinux

Command/Program	Description
ls	List the files and folders in the current directory.
ps	List information about the currently running processes including the PID number for each process.
kill	Stops a running process. The PID for the process that will be halted must be specified.
free	Lists the amount of used and available memory.
ping	Starts a network utility that can be used to test network communication.
ifconfig	Displays and sets network configuration settings.
dhcpcd	Starts a DHCP client daemon that will automatically acquire an IP address and other the network settings from a DHCP server.
ftpd	Starts an FTP server daemon that can be used to access files on the µClinux system from over the network.
telnetd	Starts a telnet server daemon that can be used to access a µClinux console terminal over the network.
boa	Starts a lightweight web server on the µClinux system. The root web page is stored in /home/httpd/.
mount	Used to mount/access different filesystems including USB memory sticks.
nano-X	Starts a windowing environment in µClinux.
nanowm	Starts a window manager that runs in the nano-X environment.
nxterm	Starts a terminal program in nano-X.
ntetris	Starts a graphical version of the Tetris game in nano-X.
nxview	Displays a JPEG image in a window in nano-X.

In additional to the standard Linux shell commands and typical user applications, a custom application targeting the DE boards has been added to

this kernel image. The source code for this user µClinux application is shown in Figure 18.7. When run, it will display an 8-bit counter on eight of the red LEDs on the DE board. Start the program by typing **`flash_leds &`** in the terminal window.

```
#include <stdio.h>
#include <unistd.h>
#include "nios2_system.h"

int main( void )
{
   int i;
   volatile unsigned *leds = ((volatile unsigned *)(na_leds));

   for(i=0; i<256; i++)
   {
      *leds = i;
      sleep(1);
   }
   return 0;
}
```

Figure 18.7 This is the source code for a user program that has been included in the µClinux kernel image included on the DVD.

18.9 PS/2 Device Support in µClinux

The µClinux kernel image provided on the DVD includes support for a PS/2 keyboard or mouse. The DE boards only have one PS/2 port on them, so only one PS/2 device can be used at a time. Attaching a PS/2 keyboard or mouse to the port will be detected by the µClinux PS/2 device driver, and several debug messages will print out in the console terminal window when a new device is recognized. The PS/2 devices are plug-and-play meaning that one can be removed and another device can be connected on the fly without having to reboot the kernel or processor.

18.10 Video Display in µClinux

The µClinux kernel image provided on the DVD supports the Video output on the DE board. This version of the kernel image starts the nano-X windowing environment along with the nanowm window manager and nxterm window terminal application during the boot process. The turquoise background color of the window manager and the terminal window should be visible on the VGA monitor when µClinux completes booting. There are several nano-X applications included in the kernel image for exploring the graphical capabilities of µClinux. From a terminal window, try running **`nxclock`** (a

graphical clock application), **ntetris** (a nano-X version of the Tetris game), or **nxview** (a JPEG image preview application). Be sure to monitor the available memory as additional processes are started.

18.11 USB Devices in µClinux *(DE2 Board Only)*

The µClinux kernel image provided on the DVD includes support for USB host functionality using the ISP1362 USB controller IC on the DE2 board. USB devices such as keyboards and mice can be supported through the Human Interface Devices (HID) drivers in µClinux. Plug a USB mouse or keyboard into the USB port on the DE2 board and notice the debug messages that display in the console terminal window. A USB mouse will control the cursor in the nano-X window manager, and a USB keyboard will allow you to enter text into the nano-X terminal window program or control the ntetris game.

Support for USB storage devices (i.e., USB memory sticks or flash drives) is also included in this µClinux kernel image. Insert a USB flash drive into the USB slot. Several debug messages will print out on the console terminal window indicating that the USB storage device is recognized and accessible as device **/dev/sda1**. To access files on the USB storage device, the filesystem must first be mounted and mapped to a directory in the root filesystem. If the USB storage device is using a FAT filesystem (most USB flash drives that are used with Windows PCs use a FAT filesystem), type the following commands:

```
mkdir /mydrive
mount –t vfat /dev/sda1 /mydrive
```

Once these commands have been executed, the contents of the USB flash drive will be available in the **/mydrive** directory. When you are finished with the USB flash drive, be sure to unmount the drive before removing it from the USB port. To unmount the drive, run the following command:

```
umount /mydrive
```

18.12 Network Communication in µClinux *(DE2 Board Only)*

The µClinux kernel image provided on the DVD includes a number of network utilities and applications. The **ifconfig** command is the primary utility used to configure the network settings such as MAC address and IP address. All network devices must have a unique hardware address known as a MAC address. The DE2 board has a software configurable MAC address, thus it is up to you to ensure that a unique address is used. To assign a MAC address, use the following command, where XX:XX:XX:XX:XX:XX is the unique number you are assigning:

```
ifconfig eth0 hw ether XX:XX:XX:XX:XX:XX
```

To use a DHCP server to automatically configure the remaining network settings, run the following commands:

```
ifconfig eth0 up
 dhcpcd &
```

The command **`ifconfig -a`** will display the network settings. If a valid IP address is displayed after the label **inet addr**, then the DE2 board is successfully communicating on the network. To start any network server services on the DE2 board, you must first run **`inetd &`**. Inetd is a server daemon that monitors and manages all ports and internet services. Once the inetd process is running, you can start one or more of the server services such as **`boa`** (web server), **`telnetd`** (telnet server daemon), or **`ftpd`** (file transfer protocol server daemon). Be sure to monitor the available memory as additional processes are started.

ALL SOURCE FILES FOR THIS NIOS II µCLINUX REFERENCE DESIGN CAN BE FOUND ON THE DVD IN THE \DE*x*\CHAP18 DIRECTORY.

18.13 For additional information

This chapter has provided a brief overview of embedded operating and the port of µClinux for the Nios II processor. Additional information about µClinux can be found at the can be found at the official µClinux website (www.uclinux.org). Additional information about the µClinux port for Nios II can be found at the µClinux Wiki (http://nioswiki.jot.com/WikiHome/) and the Nios Community forum (http://www.niosforum.com). More information about the other embedded operating systems discussed in this chapter are available at http://www.micrium.com/, http://ecos.sourceware.org/, http://www.rtos.com/, http://www.microtronix.com, http://www.mentor.com/, http://www.segger.com/, http://www.vector-informatik.de/, and http://www.euros-embedded.com/.

18.14 Laboratory Exercises

1. In Figure 18.4, notice that the clock frequency is set to 100 MHz. To handle the computational needs of an operating systems, the clock rate was increased from the 50 MHz clock used in Chapter 17. Changing the clock frequency required several changes in the PLL settings that generate the processor and memory clocks. Open the Quartus II project for Chapter 18 that is provided on the DVD. What are the frequencies of the three clock signals being generated by the PLL block? Why is the phase shift of the SDRAM clock set to 108 degrees instead of the 54 degrees specified in Chapter 17?

2. Load a small (~20KB) JPEG image onto a USB flash drive. Insert the drive into the USB slot on the DE2 board. Mount the drive as discussed in the chapter and use the nxview program to display the image. Make sure that there is sufficient memory available before you run the nxview program.

3. Follow the instructions in the chapter for establishing a network connection on the DE2 board and starting the boa web server. To demonstrate the working system, open a web browser on your PC and view the web page being served by the web server by entering the DE2 board's IP address as the URL in the web browser.

4. Using the ps, kill, and free commands, try stopping and starting the nano-X, nanowm, and other graphical programs one at a time. Record the approximate memory required for each program to run. Create a table that lists the processes and memory requirements for at least five μClinux programs.

5. Using the ps, kill, and free commands, try stopping and starting the dhcdcd, inetd, telnetd, ftpd, and boa programs one at a time. Record the approximate memory required for each program to run. Create a table that lists the processes and memory requirements for these five μClinux programs.

6. Create a new HTML page on your PC and save it to a USB flash drive. Insert the drive into the USB slot on the DE2 board. Mount the drive as discussed in the chapter. Delete the /home/httpd/index.html file using the rm command. Then replace the index.html file with a symbolic link to the HTML page on your flash drive using the ln command. Verify that your system displays the new web page by opening a web browser on your PC and viewing the web page being served by the web server (enter the DE2 board's IP address as the URL in the web browser).

7. Obtain the licenses needed for the MicroC/OS-II Nios II OS port. The license for the Nios II MicroC/OS-II port is available from Micrium (www.micrium.com) and a full commercial license for the Altera tools for schools is available from Altera's University Program (www.altera.com). Micrium also offers free OS licensing for universities and students. Follow the steps in Altera's MicroC/OS-II tutorial (available on the DVD or at Altera's web site) to develop MicroC/OS-II for a Nios II reference design and run an application program on a DE board. Use the Cyclone II reference design and complete all steps up to the point where the board is downloaded. Currently, the older 6.1 version of the Quartus II and Nios II tools are required, but an updated version of the tutorial may be available soon.

8. For a challenging problem, a Reference design is provided with the MicroC/OS-II tutorial for commercial Cypress II FPGA boards. Consult this reference design to aid in modifying your DE Nios II reference design so that it is setup correctly to support MicroC/OS-II. Check Altera's website for newer versions as they become available.

Modify your Nios II DE board design so that the MicroC/OS-II is supported and run a test program on the DE board.

9. For a more challenging problem, port the eCos operating system to a DE Board. It is available free at www.niosforum.com. First, run a simple hello world application using the UART. For the second test, write a multithreaded application with one thread talking to the UART and a second thread blinking the LEDs.