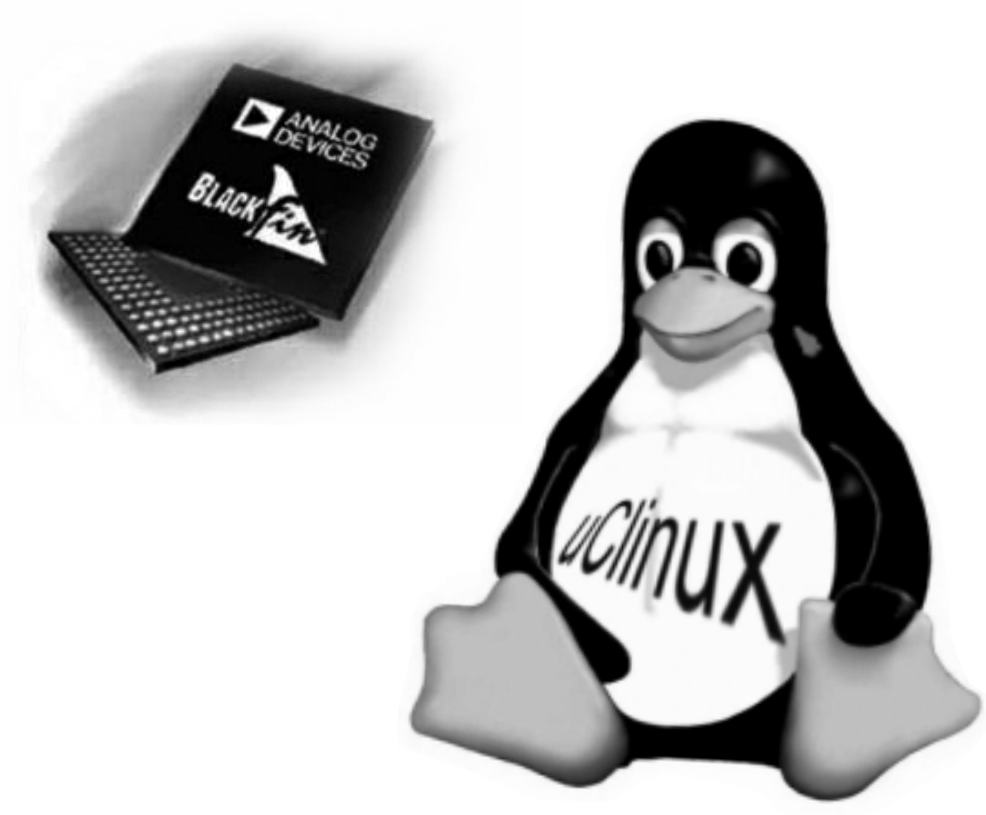
uClinux as an Embedded OS on an Embedded Processor



www.analog.com/processors

ANALOG DEVICES

uClinux as an Embedded OS on an Embedded Processor

Introduction

In the past two years, Linux has become an increasingly popular operating system choice in the development of embedded devices—particularly consumer products, telecommunications routers and switches, Internet appliances, and industrial and automotive applications.

Recently, several large consumer electronics companies announced a collaboration, the Consumer Electronics Linux Forum (CELF), to further develop the Linux platform for use in digital home electronic devices. The founders of CELF (Matsushita Electric, Sony, Hitachi, NEC, Royal Philips Electronics, Samsung, Sharp and Toshiba) are focused on the advancement of Linux as an open source platform for consumer electronics devices. As such, they are actively supporting and promoting the spirit of the open source community (see www.celinuxforum.org for more information).

The advantage of Embedded Linux is that it is a royalty-free, open source, compact solution that provides a strong foundation for an ever-growing base of applications to run on. Linux is a fully functional operating system (OS), with support for a variety of network and file-handling protocols—a very important requirement in embedded systems because of the need to "compute anywhere, anytime." Modular in nature, Linux is easy to slim down by removing utility programs, tools, and other system services that are not needed in an embedded environment. The advantages for companies using Linux in embedded markets are faster time to market and reliability. For those developers, the combination of Analog Devices' Blackfin® Processor and uClinux may be of particular interest. Blackfin Processors combine the DSP computing power and the functionality of microcontrollers, fulfilling the requirements of digital audio, video, and communication applications. The combination of a first-class DSP core with traditional microcontroller architecture on a single chip avoids the restrictions, complexity, and higher costs of traditional heterogeneous multiprocessor systems. Beneath the established peripheral equipment (SPI, UART with IrDa® support, timer, RTC, watchdog and event controller), all members of the Blackfin Processor family provide two serial dual-channel ports (SPORTs)—each supporting four stereo I²S channels with data rates up to 100 MBits/s. Furthermore, the newest members of the Blackfin Processor family (ADSP-BF531, ADSP-BF532, ADSP-BF533, and ADSP-BF561) provide a parallel peripheral interface (PPI) that seamlessly provides connectivity for TFT flat panel displays and video converter (CCIR-656, 27 MHz), or may be used as a parallel interface for AD/DA converters with up to 65 MSPS.

Table 1: The Blackfin Processor Family

Part	Clock Speed (MHz)	MMACs (Max)	Memory (KBytes)	External Memory Bus	PPI	PPI 2.2 Master/ Slave	USB Dev.	UARTs, Timers	Watchdog Timer, RTC	Core Voltage Reg.	Package
ADSP-BF535	350	700	308	32-Bit	No	Yes	Yes	Yes	Yes	1.6	260 PBGA
ADSP-BF531	400	800	52	16-Bit	Yes	No	No	Yes	Yes	1.2	160 Mini-BGA, 169-PBGA, 176 LQFP
ADSP-BF532	400	800	84	16-Bit	Yes	No	No	Yes	Yes	1.2	160 Mini-BGA, 169-PBGA, 176 LQFP
ADSP-BF533	756	1512	148	16-Bit	Yes	No	No	Yes	Yes	1.2	160 Mini-BGA, 169-PBGA
ADSP-BF561	756	3024	328	32-Bit	Yes	No	No	Yes	Yes	1.2	256 Mini-BGA, 297-PBGA

Other package options available

All Blackfin Processors combine a state-of-the-art signal processing engine with the advantages of a clean, orthogonal RISC-like microprocessor instruction set and Single-Instruction, Multiple-Data (SIMD) multimedia capabilities into a single instruction set architecture. The Micro Signal Architecture (MSA) core is a dual-MAC modified Harvard Architecture that has been designed to have unparalleled performance on both audio and video algorithms, as well as standard program flow and arbitrary bit manipulation operations mainly used by an OS.

The ADSP-BF531/BF532/BF533 Blackfin Processors have two large blocks of on-chip memory providing high-bandwidth access to the core. These memory blocks are accessed at full processor core speed. The two memory blocks sitting next to the core, referred to as L1 memory, can be configured either as data or instruction SRAM or cache. When configured as cache, the speed of executing external code from SDRAM is nearly on par with running the code from internal memory. This feature is especially well suited for running the uClinux kernel, which does not fit into internal memory. Also, when programming in C, the memory access optimization can be left up to the core by using cache.

Blackfin Processors are designed in a low power and low voltage design methodology and feature Dynamic Power Management. They fully meet the requirements of current mobile and battery-powered applications, like no other processor in their class. A Blackfin Processor has multiple, highly flexible and independent Direct Memory Access (DMA) controllers that support automated data transfers with minimal overhead impact on the processor core. DMA transfers can occur between the ADSP-BF531/BF532/BF533 processor's internal memories and any of its DMA-capable peripherals. Additionally, DMA transfers can be performed between any of the DMA-capable peripherals and external devices connected to the external memory interfaces, including the SDRAM controller and the asynchronous memory controller.

Several questions this article probes:

- What advantages does Linux provide over other operating systems?
- Why use Linux at all?
- How much does Linux cost?
- Where can I get Linux?
- Why Linux on a DSP?
- Is Linux capable of providing real-time functionality?
- What is the difference between Linux and uClinux?

In 1991 a young Finnish student named Linus Torvalds posted a message to the comp.OS.minix newsgroup:

"Hello everybody out there using minix—I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones."

This unassuming message was the birth of the Linux movement. Since then, Linux has matured with the help of thousands of developers all over the world. Linux has become an operating system that rivals the performance and features of even the most expensive commercial UNIX implementations. Over the years, Linux has been ported to a countless number of processors and platforms, from small SOCs to large mainframe computers.

Linux owes much of its success to the fact that Linus chose the GNU General Public License (GPL), published by the free software foundation, as the software licensing mechanism for his OS. This allows everyone who receives a piece of GPL software the right to demand the source code. This ensures that a GPL program stays eternally free. A common misconception is that an author of a GPL program, or a derivate work, gives up his intellectual property. This is not true. The author does retain his copyrights. The author only provides others a license to use his code under the terms of the GPL. Another misconception is that GPL software cannot be sold. There are a lot of companies that sell GPL software. The only stipulation is that they have to provide their customers with a written offer for the source code. Of course, the code for user space programs is not affected by the kernel license. Therefore, intellectual property contained in an application running on top of Linux does not have to be shared.

The Linux kernel by itself would be somewhat worthless without programs running on top. To fill that gap, Linux generally uses open source and freely available code from the GNU project (www.gnu.org), from the Berkeley Software Distribution (BSD) projects, and from the rest of the open source community. This collection of software is called a "distribution." For Linux there are literally hundreds of distributions available (a sampling can be found at: http://lwn.net/Distributions/). Distributors take the Linux kernel and bundle it with a selection of software from the large pool of open and free software, often mixing it with their own programs to sell or freely provide.

Below are a few examples of the value that such distributions provide to the user:

(David A. Wheeler, June 30, 2001, taken from http://www.dwheeler.com/sloc/)

- It would cost over $1 billion to develop this Linux distribution by conventional proprietary means in the United States (in year 2000; U.S. dollars).
- It includes over 30 million physical source lines of code (SLOC).
- It would have required about 8,000 person-years of development time, as determined using the widely used basic COCOMO model.
- Red Hat Linux 7.1 represents over a 60 percent increase in size, effort, and traditional development costs over Red Hat Linux 6.2 (which was released about one year earlier).

As astonishing as those numbers are, they are already outdated. With today's distributions, those figures could nearly be doubled by now.

Since the beginning of Linux, Linus Torvalds has been the primary developer of the Linux kernel. As such, he decides what developments are worthy to be included. The Linux kernel tree is only one of many, because the kernel licensed under the GPL, and all users are allowed to create their own tree. So there are a lot of different kernel trees from different maintainers, focused on fulfilling many different purposes. When the code in those trees has matured enough, and proved stable and worthy enough, they are sent to Linus for inclusion in his tree. Linus continually releases new kernel versions. Each of these versions get a unique number, such as 2.5.23. After a few years of development time, Linus declares the new kernel stable and gives it an even, major number, such as 2.6.x. After a certain time period, Linus assigns another developer the authority to maintain the new stable kernel, and he starts a new development cycle process. This is an open process that is documented and can be followed by subscribing to the Linux kernel mailing list or reading the development archives online.

One of the special trees mentioned above is the uClinux kernel tree, at www.uclinux.org. This is a port of the Linux kernel designed for hardware without a Memory Management Unit (MMU). While the uClinux kernel patch has been included in the official Linux kernel (www.kernel.org), the most up-to-date development activity and projects can be found at www.uclinux.org.

Patches such as these are used by commercial Linux vendors in conjunction with their additional enhancements, development tools and documentation to provide their customers an easy-to-use development environment for rapidly creating powerful applications on uClinux.

Additionally, www.uclinux.org provides developers with a uClinux distribution that includes three different kernels (2.0.x, 2.4.x, 2.6.x) along with required libraries; basic Linux shells and tools; and a wide range of additional programs such as web server, audio player, programming languages, and a graphical configuration tool. This distribution is more than adequate enough to compile a Linux image for a communication device, like a router, without writing a single line of code in C.

What is the difference between Linux and uClinux?

Since Linux is similar to UNIX in that it is a multiuser, multitasking OS, the kernel has to take special precautions to assure the proper and safe operation of up to thousands of processes from different users on the same system at once. The UNIX security model, after which Linux is designed, protects every process in its own environment with its own address space. Every process is also protected from processes being invoked by different users. Additionally, a Virtual Memory (VM) system has additional requirements that modern CPUs have to fulfill, like dynamic allocation of memory and mapping of arbitrary memory regions into the private process memory.

Some devices, like the Blackfin Processor, do not provide a full-fledged MMU because developers targeting their application to run without the use of an OS do not normally need an MMU. Additionally, MMU-less processors such as the Blackfin are more power efficient and are often significantly cheaper than the alternatives.

To support Linux on such devices, a few trade-offs have to be made:

1. No real memory protection (a faulty process can bring the complete system down)

2. No fork system call

3. Only simple memory allocation

4. Some other minor differences

Memory protection is not a real problem for most embedded devices. Linux is a very stable platform, particularly in embedded devices, where software crashes are rarely observed.

The second point is a little more problematic. In software written for UNIX or Linux, developers often use the fork system call when they want to do things in parallel. The fork call makes an exact copy of the original process and executes it simultaneously. To do that efficiently, it uses the MMU to map the memory from the parent process to

the child and copies only those memory parts to that child it writes. Therefore, uClinux cannot provide the fork system call. It does however provide "vfork," a special version of fork, in which the parent is halted while the child executes. Therefore, software that uses fork system calls has to be rewritten to use either vfork or threads that uClinux supports, because they share the same memory space, including the stack.

As for point number three, there usually is no problem with the malloc support uClinux provides, but sometimes minor modifications may have to be made.

Most of the software available for Linux or UNIX (a collection of software can be found on http://freshmeat.net) can be directly compiled on uClinux. For the rest there is usually only some minor porting or tweaking to do. There are only very few applications that do not work on uClinux, with most of those being irrelevant for embedded applications.

Developing on uClinux

When selecting development hardware, developers should not only carefully make their selection with price and availability considerations in mind, but also look for readily available open source drivers and documentation.

A uClinux Blackfin Processor development environment consists of the GNU Compiler Collection (gcc cross compiler) and the binutils (linker, assembler, etc.) for the Blackfin Processor. Additionally, some GNU tools like awk, sed, make, bash ... plus tcl/tk are needed, although they usually come with the desktop Linux distribution.

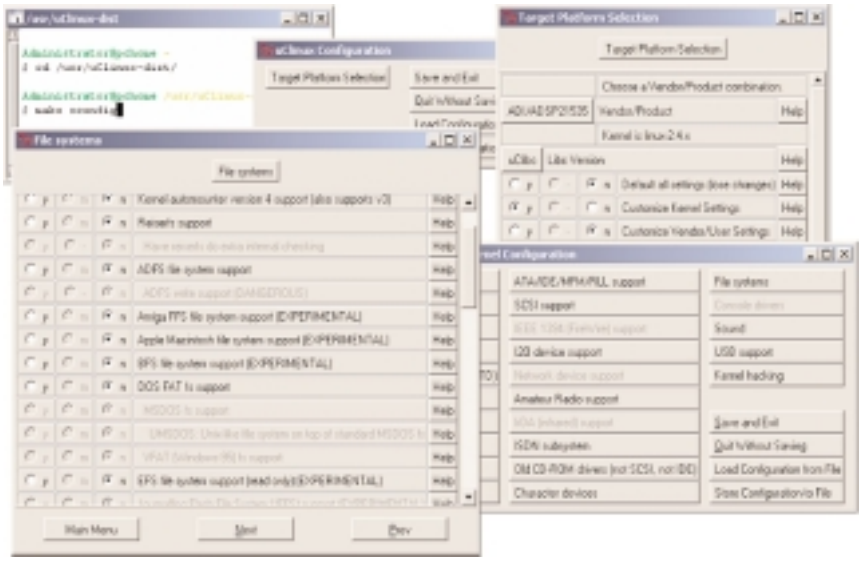
After the installation of the development environment and the decompression of the uClinux distribution, development may start.

First the developer uses the graphical configuration utility to select an appropriate Board Support Package (BSP) for his target hardware. Developers using their own hardware should make themselves comfortable with development on the EZ-KIT Lite™ or STAMP hardware (schematics and production files available at www.blackfin.uclinux.org.) After that, they can start writing their drivers and making a BSP by copying an existing one and modifying a few parameters.

Most of the development work consists of selecting the appropriate drivers and de-selecting kernel features that are not needed for the project in question. A selection of library features and user space programs follows thereafter.

The uClinux distribution provides a wide selection of utilities and programs specially designed with size and efficiency as their primary considerations. One example is busybox (www.busybox.net), a multicall binary, which is a program that includes the functionality of a lot of smaller programs and acts like any one of them if it is called by the appropriate name. If busybox is linked to *ls* (the DOS equivalent of the *dir* command) and contains the *ls* code, it acts like the *ls* command. The benefit of this is that busybox saves some overhead for unique binaries, and those small modules can share common code.

Figure 1: Graphical kernel configuration

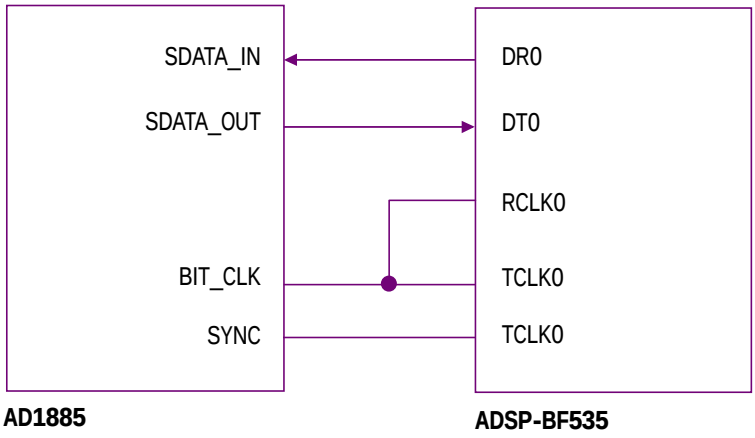


After everything is selected and successfully compiled, there is a Linux kernel and a ramdisc image that can be loaded onto the target hardware with the help of the VisualDSP++®. Once successful, further development can proceed. The next step is to use a serial or network enabled bootloader instead of loading through the JTAG interface. For example, U-boot (http://blackfin.uclinux.org/projects/uboot/) offers a variety of features and can also be used to flash on-board memory devices. An option for those who cannot afford an original Analog Devices in-circuit emulator is the low cost JTAG hardware and software implementation (http://blackfin.uclinux.org/projects/jtagtools/). This can be used to initially flash the bootloader to the target memory. But it is important to note that this workaround does not provide the debugging and emulation capabilities that VisualDSP++ does. Once the kernel is up and running, the free GNU Debugger (GDB) can be used to debug user applications.

The next step would be the development of the special applications for the target device or the porting of additional software. A lot of development can be done in shell scripts or languages like Perl or Python. Where C programming is mandatory, Linux, with its extraordinary support for protocols and device drivers, provides a powerful environment for the development of new applications.

Figure 2 is an example of how easily an AC'97 audio CODEC could be wired to a Blackfin Processor, without the need of additional active hardware components.

Figure 2: AD1885 wiring diagram



Below is an example of a very simple program for reading from this codec assuming an audio AC'97 driver is compiled into the kernel.

```
 main(){
...
fd = open("/dev/dsp", O_RDONLY, 0); //open the audio device
...
int speed = 44100 // 44.1kHz
ioctl(fd, SNDCTL_DSP_SPEED, &speed)// set sample rate
...
read(fd, buffer_rx, number_of_bytes); // read number_of_bytes into buffer
...
close(fd); // close device
}
```

Example: Reading from AC97 CODEC

Why Linux on embedded hardware?

Despite the fact that Linux was not originally designed for use in embedded systems, it has found its way into a lot of embedded devices. Since the release of kernel version 2.0.x and the appearance of commercial support for Linux on embedded processors, there has been a real explosion of new embedded devices that feature the OS. Almost every day there seems to be a new device or gadget that uses Linux as its operating system, in most cases going completely unnoticed by the end users. Today a majority of the available broadband routers, firewalls, access points, and even some DVD players utilize Linux (for more examples see http://www.linuxdevices.org).

Linux and uClinux offer a bevy of drivers for all sorts of hardware and protocols. Combine that with the fact that Linux does not have run-time royalties, and it quickly becomes clear why there are so many developers using Linux for their devices.

But why would anyone use Linux on a DSP?

In the past, DSPs have been used in a lot of applications, including sound cards, modems, telecommunication devices, medical devices, and all sorts of military and other appliances that perform pure signal processing. Those DSP systems were generally designed specifically for those applications and had only basic capabilities in order to meet their tight cost and size constraints. As DSPs have become more powerful and flexible, thereby servicing the more advanced requirements of military, medical, and communication users, they still have lacked the proper capabilities to run advanced operating systems. Those traditional DSPs are very powerful and flexible, but can be rather expensive. They are often found clustered on special signal processing hardware where there is no need to have an operating system like Linux running on the DSP itself. This is generally due to the fact that in those systems the DSP gets its data from some type of additional central processing unit. Therefore only basic system software had to be written for such DSPs.

With the quickly advancing multimedia convergence and the proliferation of multimedia and communication-enabled gadgets, there is now a big market for a new type of DSP. Currently, the most widely used design

for servicing these markets is the combination of a general-purpose processor and a traditional DSP serving as a co-processor. In this scenario, the operating system runs on the host processor, and the signal processing is done on the DSP. This type of dual-processor design is suboptimal due to inefficiencies incurred in cost, power, and size.

There are a few solutions to accommodate the new market demands:

- The use of specially designed ASICs and FPGAs—and the large upfront investment required to develop from scratch or to use and modify some third-party IP.
- The use of special hardware often combined with a general-purpose IP-Core on a SOC (System on Chip), for example, a DVD player on a chip, scanner and digital camera on a chip. These devices are generally limited to the function they were originally designed for.
- The combination of a traditional DSP and a general-purpose IP-Core on a SOC device, where the operating system runs on the IP-Core and the signal processing can be offloaded to the embedded DSP; such an approach has been taken in some wireless LAN chipsets.
- Or finally, the redesign of the DSP to fit the demand of an advanced operating system while preserving the advanced DSP architecture. This approach has been taken by the Blackfin Processor designers—by designing a processor with advanced DSP features around the well-proven Harvard Architecture with a RISC-like instruction set. Such a device is no longer a simple DSP, but rather a powerful processor that will meet the intensive demands of a wide range of communication and multimedia applications. Combined with the capabilities and the power of an operating system like Linux, there are endless possibilities.

Real-time capabilities of uClinux

Since Linux was originally developed for server and desktop usage, it has no hard real-time capabilities like most other operating systems of comparable complexity and size. Nevertheless, Linux—and in particular, uClinux—has excellent so-called “soft real-time” capabilities. This means that while Linux or uClinux cannot guarantee certain interrupt or scheduler latency compared with other operating systems of similar complexity, they show very favorable performance characteristics. If one needs a so-called “hard real-time” system that can guarantee scheduler or interrupt latency time, there are a few ways to achieve such a goal:

- Use another operating system: There are a lot of RTOS systems available to choose from that meet this requirement (VisualDSP++ kernel, Nucleus PLUS, ThreadX, uITRON).
- Provide the real-time capabilities in the form of an underlying minimal real-time kernel such as RT-Linux (http://www.rtlinux.org) or RTAI (http://www.rtai.org). Both solutions use a small real-time kernel that runs Linux as a real-time task with lower priority. Programs that need predictable real time are designed to run on the real-time kernel and are specially coded to do so. All other tasks and services run on top of the Linux kernel and can utilize everything that Linux can provide. This approach can guarantee deterministic interrupt latency while preserving the flexibility that Linux provides.
- Change the Linux kernel to achieve hard real-time interrupt latencies: Bernhard Kuhn is developing a patch for the Linux kernel that could achieve that (http://linuxdevices.com/articles/AT6105045931.html). In the future, it is possible that this could be ported to the uClinux Blackfin tree.

In most cases, hard real time is not needed, particularly for multimedia applications, in which the time constraints are dictated by the abilities of the user to recognize glitches in audio and video. Those physically

detectable constraints that have to be met normally lie in the area of tens of milliseconds—which is no big problem on fast chips like the Blackfin Processor. Stricter timing requirements can be achieved with just a little tweaking or some straightforward changes to the scheduler. In kernel 2.6.x, the new stable kernel release, those qualities have even been improved with the introduction of the new O(1) scheduler and kernel preemption.

Tables 2 and 3 below list the execution times for many popular multimedia and communication algorithms running on the Blackfin Processor without an OS. In most cases, there is enough processing power left so that the scheduler has enough time to handle the low number of processes that normally run on such devices. Therefore, there is no problem in utilizing some programming in languages like Perl, Python, and PHP while having web servers, snmp, ppp, or pppoe, firewall, etc., running while decoding video and audio.

Therefore, there is no need for hard real-time operating systems that lack the advanced features that only such a powerful OS like Linux can provide.

Table 2: Video and audio codecs

Video Codec	Image	MHz Req.	% Loading (of 600 MHz)
MPEG-2 Player	D1 (720 x 480) @ 30 fps	290 MHz	48%
MPEG-4 SP Player	D1 (720 x 480) @ 30 fps	311 MHz	52%
H.264 Player	D1 (720 x 480) @ 30 fps	446 MHz	74%
MPEG-2 Player	CIF (360 x 240) @ 30 fps	73 MHz	12%
MPEG-4 SP Player	CIF (360 x 240) @ 30 fps	78 MHz	13%
H.264 Player	CIF (360 x 240) @ 30 fps	111 MHz	19%
Audio Codec	**Sample Rate**	**MHz Req.**	**% Loading (of 600 MHz)**
WMA ver. 8 Player	48 kHz, 128 kHz	50 MHz	8%
WMA PRO ver. 9 Player		125 MHz	21%
MP3 Player		19 MHz	3%
MP3 Pro Player		70 MHz	12%

Table 3: Speech codecs

Speech Codec	MHz Req.	Instr. Mem. (KBytes)	Data Mem. (KBytes, Common)	Data Mem. (KBytes/Ch.)	% Loading (of 600 MHz)
G.728	27 MHz	14.4	3.0	7.0	4.5%
G.726	6.5 MHz	4.35	0.58	0.12	1.0%
G.729AB	14 MHz	36.0	12.0	3.6	2.3%
G.723.1A	22 MHz	29.0	24.0	2.0	3.6%
DTMF	1.5 MHz	4.2	0.2	0.8	0.3%
G.168 (64 ms Sparse)	8.0 MHz	9.0	1.4	3.0	1.3%
RTP/RTCP/JIB	1.0 MHz	12.0	0.8	2.0	0.2%
AMR	16.0 MHz	49.0	37.8	—	2.6%

Sources for uClinux on the Blackfin Processor

All sources and tools (compiler, binutils) needed to create a working uClinux kernel on the Blackfin Processors can be obtained from http://www.blackfin.uclinux.org. To use the binary rpm you need a PC with a Linux distribution like Red Hat or SuSE.

Developers who cannot install Linux on their Windows® PC, have a few alternatives:

- Buy another PC for use with Linux. This PC could provide Linux services for the entire development staff of a company. With an Xserver on the Windows PC and Samba on the Linux PC, this development environment can be seamlessly integrated with the existing Windows development tools. Samba is open source and available on nearly every Linux installation, and a free Xserver is available within the cygwin environment (http://www.cygwin.com).
- Use Linux on Windows machines: There are several programs available that allow the use of a complete Linux distribution on Windows (2000, XP). Those programs emulate a PC within the Windows OS so that an unmodified guest OS could be executed. Two examples are VMWare or Virtual PC.
- Use a specially ported Linux kernel for the Windows platform, like the coLinux project provides. (www.colinux.org)
- Use a Windows port of the development tools.

For the last item there already exists an out-of-the-box solution that can be downloaded for free from http://www.blackfin.uclinux.org. This port utilizes the cygwin environment and comes with a complete Blackfin uClinux distribution, including all user space applications and a graphical Windows-like installer.

It should be noted that it is a requirement to use at least a Windows 2000 PC (NT may work but is not tested). The drive on which the uClinux distribution is installed should use NTFS (FAT32 works as well with some limitations regarding speed).

Outlook and conclusion

Blackfin Processors offer a superb price performance ratio (800 MMAC @ 400 MHz for less than $5/unit in quantities), advanced power management functions, and small mini-BGA packages. This represents a very low power and space-efficient solution for even the most ambitious projects. The Blackfin's advanced DSP and multimedia capabilities qualify it not only for audio and video appliances, but also for all kinds of industrial, automotive, and communication devices. Another advantage of the Blackfin Processor in combination with uClinux is the availability of a wide range of applications, drivers, and protocols, often as open source or free software. In most cases, there is only a compilation or some minor tweaking necessary to get that software up and running. Combine this with such invaluable tools as Perl, Python, and PHP, and developers have the opportunity to develop even the most demanding feature-rich applications in a very short time frame, often with enough processing power left for future improvements and new features.

The latest kernel 2.6.6 is nearing release, with a new tool-chain in gcc 3.3.3. The Blackfin patch has been merged into the uClinux kernel tree and is available for ADSP-BF531/BF532/BF533 as well as the ADSP-BF535 Blackfin Processors. Due to the fact that Blackfin is a brand new architecture, many other single- and dual-core derivates will follow (www.analog.com/processors/blackfin).

This year there will be a new member of the Blackfin Processor family available with an Ethernet MAC. Some people already started to port uClinux to the new dual-core ADSP-BF561. The idea behind this attempt is to have uClinux running on one core and performing such tasks as high performance real-time video encryption or decoding on the other core.

Links:

www.blackfin.uclinux.org

www.analog.com/blackfin

www.uclinux.org

www.blackfin.org

www.tldp.org

Author:

Dipl.-Ing.(FH), MSc Michael Hennerich

European DSP Applications Engineer (ADI Germany, Munich). Studied electronic and computer engineering, as well as computer-based engineering at Reutlingen University (Germany).

Co Author:

Juergen Hennerich

Studies physics at University of Tuebingen. Longtime member of the
Linux User Group Tuebingen (LUGT). Related to Unix/Linux since the mid-1990s.

Embedded Processing Support

www.analog.com/processors
Email (in the U.S.A.): embedded.support@analog.com
Email (in Europe): embedded.europe@analog.com
Fax (in the U.S.A.): 781.461.3010
Fax (in Europe): 49.89.76903.157

Worldwide Headquarters
Analog Devices, Inc.
One Technology Way
P.O. Box 9106
Norwood, MA 02062-9106
U.S.A.

Tel: 781.329.4700
Fax: 781.326.8703
Toll-free: 800.262.5643 (U.S.A. only)

Analog Devices, Inc. Europe
c/o Analog Devices SA
17-19, rue Georges Besse
Parc de Haute
Technologie d'Antony
F-92182
Antony Cedex, France

Tel: 33.1.46.74.45.00
Fax: 33.1.46.74.45.01

Japan Headquarters
Analog Devices, Inc.
New Pier Takeshiba
South Tower Building
1-16-1 Kaigan,
Minato-ku, Tokyo
105-6891, Japan

Tel: 813.5402.8210
Fax: 813.5402.1063

Southeast Asian Headquarters
Analog Devices, Inc.
RBS Tower, Rm 4501-3
Times Square
One Matheson Street
Causeway Bay
Hong Kong, PRC

Tel: 852.2.506.9336
Fax: 852.2.506.4755

www.analog.com/processors

Printed in the U.S.A.

ANALOG DEVICES