An Implementation of Open Source Operating System on Multi-processor System-On-a-Chip

Zhiming Tan, Shibao Zheng, *Member*, IEEE, Peilin Liu, Guixu Lin, and Shimei Yu

Abstract — *Rich requirements from High Definition Television (HDTV) decoder argue the supports of complex architecture and Operating System (OS). As an embedded system, mult-processor System-on-a-Chip (MPSoC) is an ideal implementation of the decoder. Open source OS has many advantages to be ported to embedded fields. This paper describes the detailed process of its porting, including building environment, kernel modification, and debug. The porting specifications are constrained by the resources on the HDTV SoC platform, development period, debug efforts, and system robustness, etc. As an application of the open source OS, our work shows the joint of industry practice and theory research*[1].

Index Terms —High Definition Television (HDTV), Multi-processor System-on-a-Chip (MPSoC), Open source OS

I. INTRODUCTION

High Definition Television (HDTV) decoder System-On-a-Chip (SoC) platform asks a support from a carefully designed, high performance, and robust software system to meet its rich demands, such as video decoding, audio decoding, synchronizing between video and audio frames, program information processing, and system controlling. The software system in such an embedded system commonly consists of Hardware-dependent Software (HdS), APIs, and applications [1]. Operating System (OS) is one of the components in HdS. It is an important role in the software architecture.

Open source OS is attracting more and more enthusiasts with its sparkling features and free sources. Its dominant trend is to apply in embedded systems [2]. Building embedded systems is to use open OS as the kernel and other open source tools as the framework [3]. Its industry application cases cover embedded fields such as Portable Device Assistances (PDAs), mobiles, field devices, network devices, and digital TVs [4].

[1] This work was supported by National High Technology Research and Development Program of P.R. China (863 Program) No. 2003AA1Z1070, and by Program for Changjiang Scholars and Innovative Research Team in University (IRT0426).

Zhiming Tan is with the Institute of Image Communication and Information Processing (IICIP), Shanghai Jiao Tong University (SJTU), Shanghai, P.R. China (e-mail: zhmtan@sjtu.edu.cn).

Shibao Zheng is with the IICIP, SJTU, Shanghai, P.R.China (e-mail: sbzheng@cdtv.org.cn).

Peilin Liu is with SJTU, Shanghai, P.R. China (e-mail: liupeilin@sjtu.edu.cn).

Guixu Lin is with the IICIP, SJTU, Shanghai, P.R. China (e-mail: guixu_lin@sjtu.edu.cn).

Shimei Yu is with the IICIP, SJTU, Shanghai, P.R. China (e-mail: honemle@sjtu.edu.cn).

Besides the basic services as provided by common OS, Open source OS has more advantages to be implemented in the HDTV decoder SoC platform. They are listed as follows:

·zero cost. The open source OS can be copied, distributed, and modified freely [5]. We can port it on the SoC platform without license and royalty fees. The free source code is on the internet or from the processor vendor. Modifying and adapting it to our own platform is the only effort.

· convenient building environment. Kernel building environment has aspects of configuration and cross-compilation. For configuration, most of the modular components of kernel can be selected and configured with a Graphic User Interface (GUI). For cross-compilation, the tools of compiling, linking, and transforming on the host PC make the building of binary image to the target platform feasible. Internet facilitates the availability of the tools.

·rich file systems. File systems in the open source OS have dozens of types, and the number is increasing. One specific (i.e., second extended file system 2 (EXT2)) is mounted as the root file system when the OS is booted. For some embedded system without disk storage, it may be located on a ramdisk in SDRAM. In some circumstance, information must be written to flash on each operation for backup or on power failure for recovery. Journaling Flash File System (JFFS) [6] is a choice to be built on the flash.

·available shell utilities. Several kinds of shells are provided for the open source OS, and BASH [7] is the most popular one. Shell binary images can be wrapped into the file system for upper-level operations, such as copy, dump, chmod, and mkdir.

·open source device drivers. Device drivers constitute a big part in the source codes. Open source gives reusability and inspiration for the development of new device drivers.

·plentiful documentation and references. Readable articles in the documentation directory in source code describe what they are and how they act for some modules in the kernel. Porting guide details and problem solutions can be available on the internet or through mailing lists. All of them provide practicable help when you are in a mess.

HDTV decoder SoC is a multi-processor platform with system CPU, audio CPU, video decoder, and video processor. Porting open source OS on Multi-Processor SoC (MPSoC) is still a joint of theory research and industry practice. System complexity, development period, debug efforts, and system robustness [8] are still pressing the porting with many problems and warnings. As a successful practice and research

Contributed Paper
Original manuscript received May 18, 2006
Revised manuscript received July 3, 2006

of such work, we will give the details of open source OS[2] porting procedure and careful tips on MPSoC.

Platform and OS hierarchy will be illustrated in the following section. Section 3 gives building environment. Kernel modification details including initialization, interrupt, timer, and device drives will be given in section 4. Section 5 describes debug tips. Results will be given in section 6. Section 7 concludes the paper and gives the future work.

II. PLATFORM AND OS HIERARCHY

We only describe what related with the open source OS porting, including hardware architecture and processor considerations. The hierarchy of open source OS is tightly based on the platform.

A. Architecture

The specification of HDTV decoder comes from MPEG-2 standard [9]. Transport Stream (TS) is got from an external tuner and sent to the module of TS Demultiplexer (TSD) via Synchronous Parallel Interface (SPI). The demultiplexed outputs are video Elementary Stream (ES), audio Packet Elementary Stream (PES), and some Program Specific Information (SPI). The video ES is decoded and processed by the video decoder and processor, respectively. The audio PES is decoded by the decode software running on the audio CPU. The system CPU serves as a master for controlling over slave audio CPU and hardware modules. The architecture of HDTV decoder SoC platform is shown in Fig. 1.

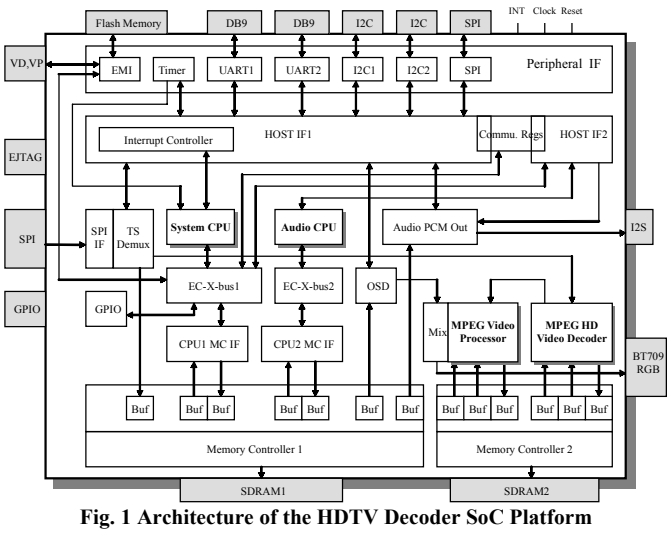

Fig. 1 Architecture of the HDTV Decoder SoC Platform

Two Easy Connect (EC)-X-bus bridges connect between two CPUs and X-buses. Hardware modules connect with X-buses. The register accessing of some modules is put into HOST IF1, 2. System and audio CPUs access memory through Memory Controller1 (MC1), whereas the video decoder and processor access through MC2. We design all the hardware modules except for two CPUs, the video decoder, and the video processor.

B. Processor Core

Two CPUs are chose from the RISC camp[3], with 5-lelvel pipeline, coprocessors, Memory Management Unit (MMU), configurable caches, and EJTAG, etc. They have a precise exception handling mechanism, through which CPU can precisely return to the victim (whose address stored in the specific register) after each exception. Different types of exceptions have their specific entries in the address space, and interrupt is one of them. Some interrupt pins are for hardware (one pin specific for the timer), and some pins for software. The external interrupts have no priorities in hardware, whereas software handlers can define them [10].

C. OS Hierarchy

Porting the open source OS is constrained by specific hardware resources. On the HDTV decoder SoC platform, it mainly involves modifying kernel services, creating device drivers, and changing shell utilities in the RAM disk, etc. The kernel, device drivers, shell, and library constitute an integral embedded OS.

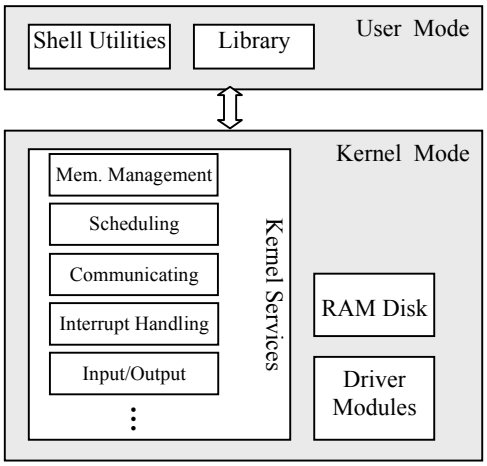

Fig. 2 The Open Source OS Hierarchy

III. BUILDING ENVIRONMENT

Kernel building from source code to binary image is commonly through the way of host-to-target, and needs the support of toolchains.

A. Host-to-target

Three factors decide the way of host-to-target when porting the open source OS. The first is the limitation of resources on the target. Low processor speed, limited storage space, immature hardware, unavailable compiling environment, limited application library, and inconvenient user interface on the target platform make the kernel building really difficult. The second is co-development requirement between hardware and software. The executable binary image should be available

[2] Linux as this case shows.

[3] MIPS is a typical one.

before the hardware maturity to reduce the whole design period. The third is the application-specific processor architecture for each implementation. It needs much time and effort to build different compilation environments for different application systems.

In most situations, host processor type is different from target. Therefore it needs cross-building tools of compilation, link, and format transformation running on the host to get the final executable images.

B. Toolchains

A toolchain is a complete collection of compiler and binutils programs and can be run either as a cross-compiler, or native on the target (if performance allows) [11]. In our context, we define the toolchain as the collection of host OS, cross-compilation tool, link tool, binary format transformation tool (one item of the binutils), uploading tool, flash writing tool, and debugger, etc.

Host OS is the base workbench for the kernel building. Cross-compilation tool translates C and assembly source codes into specific object files, and link tool links them together. Binary format transformation tool transforms the binary kernel image into required format.

Moving kernel image from the host PC to the target platform can be handled through two methods according to target storage types. The first is to upload to SDRAM with ports of USB, net, and UART, etc. The second is to writr to flash with parallel port or specific flash writing tool. Host software utilities are all needed in two situations.

Kernel debugger can be chosen among software and hardware debuggers for C source level or assembly level debug. A software debugger can communicate with the target kernel via UART port 2 (port 1 used for the console). Kernel running information can also be output to the console for debug (via the function printk()).A hardware debugger can connect with EJTAG interface in the processor core.

IV. KERNEL MODIFICATION

Kernel modification is the main effort in the open source OS porting. It mainly consists of module configuration, codes adaptation, and device driver programming [12]. They lie in the OS porting layer.

A. Module Configuration and Selection

The open source kernel is commonly a monolithic entity, with the whole operating system (process management, memory management, file system, and drivers, etc.) contained within one binary image [13]. It benefits the kernel storage on flash and loading into the SDRAM when running. The kernel is also modular. Specific CPU architecture and services are selected and configured into a single image for specific embedded system. A friendly user interface is provided to module selection and configuration. On the HDTV decoder SoC platform, the main modules and services are configured and selected as shown in Fig. 3.

We make a directory of "hdtvsoc" for the DHTV decoder SoC platform, and hook it into the kernel under the item of "Machine selection". The configuration of "Loadable module support" is for the loadable device driver modules. A size of 8,192KB ramdisk (RAM disk, a block of SDRAM memory used as disk storage) is integrated into the kernel image, with EXT2 included as the root file system. The serial ports (UART1, 2) used in the platform are not standard ones. The kernel console is based on the first one and the kernel debugger is based on the second one.

```
Code maturity level options  --->
        [*] Prompt for development and/or incomplete code/drivers
Machine selection  --->
        [*] Support for SJTU HDTVSOC board (EXPERIMENTAL)
Loadable module support  --->
        [*] Enable loadable module support
        [*]     Kernel module loader
General setup  --->
        [*] Generate little endian code
        [*] System V IPC
        [*] Sysctl support
Block devices  --->
        <*> RAM disk support
        (8192)     Default RAM disk size
        [*]     Initial RAM disk (initrd) support
Initrd options  --->
        [*]     Embed root filesystem ramdisk into the kernel
Character devices  --->
        [*] Non-standard serial port support
        [*]     HDTVSOC serial port support
        [*]         Console on HDTVSOC serial port
File systems  --->
        <*> Kernel automounter support
        <*> Kernel automounter version 4 support (also supports v3)
        [*] /proc file system support
        <*> Second extended fs support
Kernel hacking  --->
        [*] Are you using a crosscompiler
```

Fig. 3 Module Configuration and Selection on the HDTV Decoder SoC Platform

B. Platform Dependent Initialization

The initialization process is platform-dependent due to the different module configuration. What specifically involved in the HDTV decoder SoC platform is shown as bold characters in Fig. 4. Board, timer, traps, interrupt, initrd, and console are initialized and setup in *start_kernel*. Because many modules and services are unselected, things done in *init* are few. Loading and initializing the ramdisk is the most time-consuming thing due to the big size of it. The initialization process shows a simple version of call graph [14] in the open source kernel.

C. Interrupt Framework

Interrupt sources (with priority and Interrupt ReQuest (IRQ) number) consist of timer, VD, TSD, UART1, 2, I2C1, 2, audio CPU, PCM out and system CPU. The last two are the interrupt sources to the audio CPU.

The interrupts from HW modules are routed to the top interrupt module, which prioritizes the interrupts, binds some

interrupts into one line and connects to CPU's interrupt pin0 to pin5. Therefore the dispatch framework of the system CPU has two levels. The first level is from sources to the top interrupt module, and the second is from the top interrupt module to the system CPU.

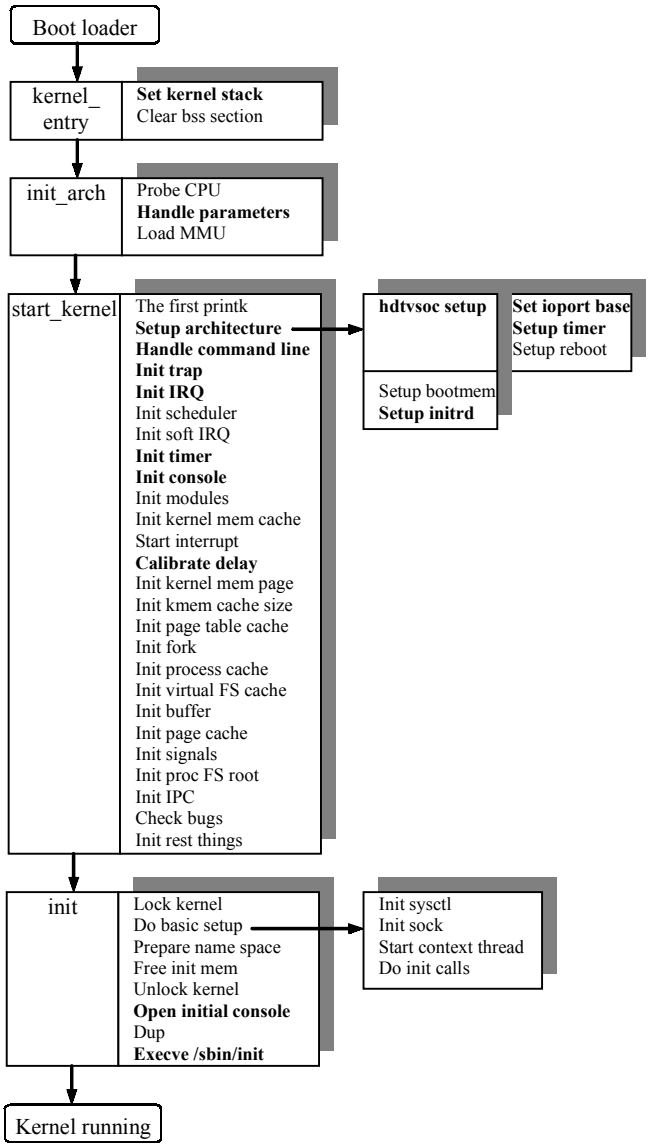

Fig. 4 Kernel Initialization on the HDTV Decoder SoC Platform

Each module can have at most 5 interrupt outputs, and the interrupt controller supports 32 modules. Therefore there can be 160 interrupt sources for the system CPU in all [15]. Interrupt signals to the audio CPU are directly connected with its pins.

All the interrupts to the system CPU are handled in the kernel and interrupts to the audio CPU are handled in the control SW component running on the audio CPU. The codes of IRQ dispatch are implemented in assembly, and hooked into the exception vector in the initializing process [1]. The interrupt framework is shown in Fig. 5.

D. Timer

Timer in the open source kernel provides time slice for task scheduling and interrupt services. It is so important that the kernel can not run without the timer service.

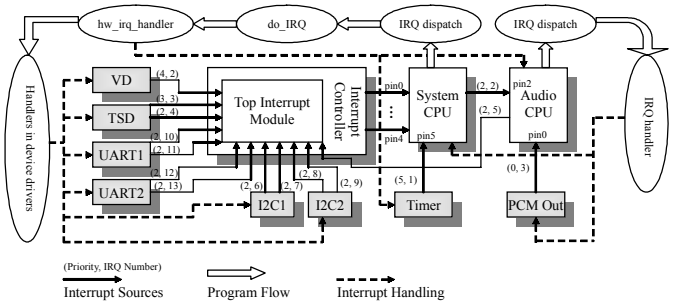

Fig. 5 Interrupt Handling Framework and Services

Count and compare registers constitutes the timer. Count register increments every other clock and compare register maintains a stable value. When the value of count register equals the value of compare register, a timer interrupt is triggered [10]. Timer interrupt is handled via the function of hdtvsoc_timer_interrupt(), in which do_timer() is called.

E. Device Drivers

Device drivers control hardware modules and provide services of configuration and access for the open source kernel. They run in the kernel mode and access the same privileged resources as kernel codes. Two methods are used to implement them: being compiled into the kernel as an integral entity, and being compiled as loadable modules. The initialization phases are different for different compilation methods. The former is in kernel initializing and the latter in kernel running, as shown in Fig. 6. The initialization of device drivers covers things: setting parameters and status, requesting interrupt resources, and requesting memory, etc. The services of device drivers include interrupt service, status control services, and I/O interface services, etc.

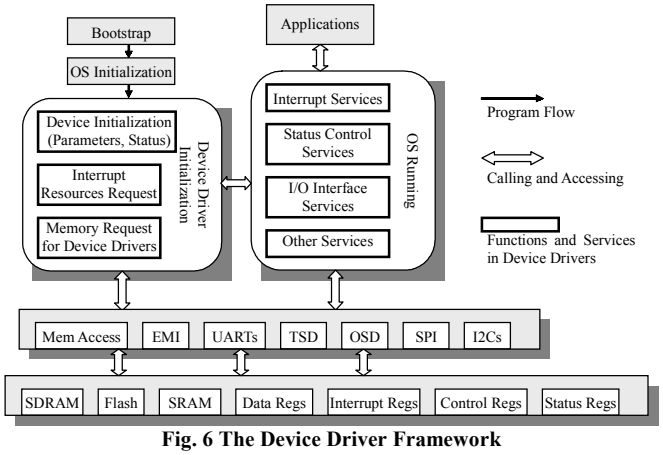

Fig. 6 The Device Driver Framework

In the HDTV decoder SoC platform, device drivers handle two things: setting status (for interrupt registers, interrupt masking registers, status registers, and control registers) and accessing data (for some data holders, such as SDRAM, GPIO, and data registers). Control registers get control messages from CPU, actually from user program. Status registers show the status and action of device modules. From the hardware modules' view point, control registers are input-style while status registers are output-style. Data registers are data holders in the data-transferring paths. Some hardware modules need them to hold the data for next modules. For example, we put data into the GPIO data registers, which can light Light-Emitting Diodes (LEDs). The LED messages can show failure or success signals in the debug procedure [1].

V. DEBUG

The open source kernel debug on the immature hardware platform costs much time and carefulness. Because bugs exist not only in the kernel codes but also in immature hardware modules, we can not decide which fault is to which part without deep investigation. In such circumstance, finding bugs in assembly-level is more efficient than in C source-level because the former is instruction precise.

The processor features pipeline execution flow and precise exception. The exception context can be explicitly traced because the victim and all the following instructions in the pipeline are cancelled when each exception happens [10]. Exceptions are classified into several types for the "cause" register coding, through which we can know whether the exception caused from software or hardware. For example, "address error on load or store" shows an exception that may be caused by illegal reference to kernel address in user mode. It is a software exception. On the other hand, "bus error" may show that something wrong in the address translation or memory transaction. It is a hardware fault.

A debugger connecting with EJTAG interface is a fundamental tool for assembly-level debug [16]. It helps to catch the exception context. On the HDTV decoder SoC platform, Memory Controller 1 (MC1) is an important hardware module in the data path of "peripheral-system CPU-SDRAM". The kernel is a full-featured software component to test if the data path is working well. We find some bugs of MC1 through this method. One of them is improperly supported burst transaction when the caches miss.

VI. RESULT

The open source OS is ported to a verification board of HDTV decoder SoC with two hard processor cores, two FPGAs with 6,000,000 gates, 32MB SDRAM, 8MB flash, and peripheral interfaces, as shown in Fig. 7.

The console is based on UART1, through which the kernel interacts with the user with output information and input commands. The final OS image (including ramdisk, glibc, some shell utilities, and SoC-specific device drivers) has a size of about 2.8MB. We develop a boot loader running on the system CPU to initialize the hardware environment and start up the kernel. The boot loader also triggers the audio CPU to execute its instructions. We develop system control software on the system CPU to control the whole system, and audio control software to control the behavior of the audio CPU.

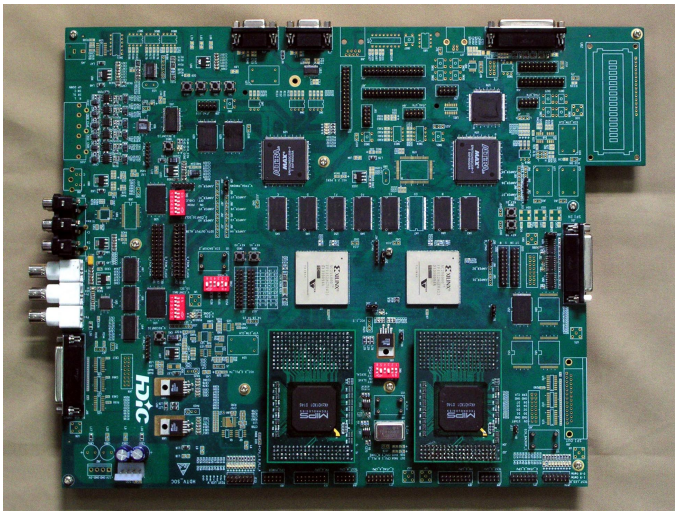

Fig. 7 The FPGA Verification Board

VII. CONCLUSION AND FUTURE WORK

The open source OS has more and more applicable values in embedded system, especially complex MPSoC. Porting the kernel involves investigating hardware constraints, making compilation environment and toolchains, modifying some codes in the OS porting layer, debugging kernel on immature hardware, and co-verification with hardare designers, etc. As a successful experience, this paper has given the open source OS porting details on the HDTV decoder SoC platform.

The open source OS porting is still a joint of industry application and theory research when meeting complex multi-processor system. Firstly, design factors such as hardware resources, design cost, cycle, debug efforts, and system robustness constrain the consideration of the OS porting. For example, as a monolithic kernel, it needs a big ROM storage space in embedded system (for the implementation of HDTV decoder SoC platform, the image size is nearly 3M Bytes). An approach of call graph [14] is used to adapt the kernel to specific embedded application with the least size. However, it needs another porting effort when requirements change. Secondly, the standardization of OS components [17] facilitates auto generation of the kernel, building environment, and debug, etc. Apparently this can accelerate the porting process. Nevertheless, standardization is not an easy thing and the goal of auto generation is still has a long way to achieve [18]. Thirdly, the open source kernel is highly modularized and has a big number of versions, which are ever increasing. Coupling between the kernel modules heavily affects the maintainability of kernel between all these versions [19]. It is a

big problem facing the embedded world for things such as stability [20], maintenance, and updating.

ACKNOWLEDGMENT

We give special thanks to all the team members in the HDTV decoder SoC project for the helpful discussions and contributions.

REFERENCES

[1] Z. Tan, S. Zheng, J. Hu, Y. Chen, and P. Liu, "Design and implementation of the software system on MPSoC: an HDTV decoder case study," unpublished.
[2] D. Geer, "Survey: Embedded Linux Ahead of the Pack", *IEEE Distributed Syst. Online*, Vol. 5, No. 10, pp. 1-6, Oct. 2004.
[3] K. Yaghmour, *Building Embedded Linux Systems*, O'Reilly, 2003.
[4] S. Moon, J. Kim, K. Bae, J. Lee, and D. Seo, "Embedded Linux Implementation on a Commercial Digital TV System", *IEEE Trans. Consumer Electron.*, Vol. 49, No. 4, pp. 1402-1407, Nov. 2003.
[5] *http://www.gnu.org/copyleft/gpl.html*, GNU General Public License.
[6] *http://developer.axis.com/software/jffs/*, JFFS Home Page.
[7] *http://www.gnu.org/software/bash/bash.html*, BASH.
[8] T. Nakajima, M. Sugaya, S. Oikawa, "Operating systems for Building Robust Embedded Systems", *Object-Oriented Real-Time Dependable Systems*, 10th IEEE International Workshop on, 2-4, pp. 211-218, Feb. 2005.
[9] ISO/IEC International Standard 13818-1, "Generic coding of moving pictures and associated audio information: systems," 1996.
[10] MIPS Technologies Inc., *MIPS32 4K™ Processor Core Family Software User's Manual*, Revision 01.17, http://www.mips.com/, Sept. 2002
[11] *http://www.linux-mips.org/wiki/Toolchains*, Toolchains.
[12] J. Sun, "Linux MIPS porting guide," *http://linux.junsun.net/porting-howto/*.
[13] A. Lennon, "Embedding Linux", *IEE Review*, Vol. 47, No. 3, pp. 33-37, May 2001.
[14] C. Lee, Z. Rong, and J. Lin, "Linux Kernel Customization for Embedded Systems by Using Call Graph Approach", *Design Automation Conf.*, Proceedings of the ASP-DAC 2003, 21-24, pp. 689-692, Jan. 2003.
[15] Y. Chen, G. Lin, F. Wang, and Z. Tan, "Multiple MIPS 4Kc core based interrupt controller design and its implementation on HDTV SoC platform", unpublished.
[16] *http://www.fs2.com/*, First Silicon Solutions.
[17] S. Hong, "Embedded Linux Outlook in the PostPC Industry", *Object-Oriented Real-Time Distributed Computing*, Sixth IEEE International Symposium on, 14-16, pp. 37-40, May 2003.
[18] V. J. III Mooney, and D. M. Blough, "A hardware-software real-time operating system framework for SoCs," *Design & Test of Computers, IEEE,* Vol. 19, Issue 6, pp. 44-51, Nov.-Dec. 2002.
[19] S. R. Schach, B. Jin, D. R. Wright, G. Z. Heller, and A. J. Offutt, "Maintainability of The Linux Kernel", *Software*, IEE Proceedings, Vol. 149, No. 1, pp. 18-23, Feb. 2002.
[20] J. Champaign, A. Malton, X. Dong, "Stability and Volatility in the Linux Kernel", *Software Evolution*, Proceedings, Sixth International Workshop on, 1-2, pp. 95-102, Sept. 2003.

Zhiming Tan graduated from Science and Technology of Shannxi University, Xianyang, P.R. China, and received B.S. & M.S. degrees in 2000 and 2003, respectively. He is currently working toward the Ph.D degree in electronic engineering at the Institute of Image Communication and Information Processing (IICIP) of Shanghai Jiaotong University (SJTU), Shanghai, P.R. China. His research interests include embedded software (Linux kernel, and boot loader, etc.), digital television, and video segmentation.

Shibao Zheng (M'01) graduated from Xidian University, Xi'an, P.R. China and received the B.S. and M.S. degrees in 1983 and 1986 respectively. From 1986 to 1999, he was an expert of the national project in HDTV and the chief designer of the ground digital TV equipments in the project of Shenzhou Spaceship. From 2000 to now, he is a professor and doctoral tutor in the IICIP and IC and System Research Center (ICSRC) of SJTU. His general research interests include DTV, ASIC, and multimedia system.

Peilin Liu is a PhD Graduate (1992~1998) from the University of Tokyo majoring in Electronic Engineering and worked there as a Research Fellow in 1999. After that, she was hired as a Senior Researcher for Central Research Institute of Fujitsu, Tokyo (1999~2003). Her research mainly focuses on Multimedia (Audio/Video) Processing, IC Design and High-performance Processor Development. She is now a professor of Department of Electronic Engineering in SJTU.

Guixu Lin received B.S. degrees in 2001 from Harbin Engineering University, Harbin, P.R. China, and M.S. degrees in 2003 from Harbin Institute of technology, Harbin, P.R. China. He is currently working toward the Ph.D degree in electronic engineering at the IICIP of SJTU, Shanghai, P.R. China. His research interests include DTV codec system, Multimedia system design and ASIC implementations.

Shimei Yu received B.S degree in 2004 from Huazhong University of Science and Technology, Wuhan, P.R. China. He is now working toward the M.S degree in electronic engineering at the IICIP of SJTU, Shanghai, P.R. China. His research interests include embedded software, digital television.