Is It Really Possible to Play DVD-quality Media while executing Linux Applications on the same Low Cost Processor?

Russell Rivin
Analog Devices, Inc.
One Technology Way
Norwood, MA 02062
USA
(781) 461 – 4092
Russell.Rivin@Analog.com

Rajesh Mishra
Analog Devices, Inc.
One Technology Way
Norwood, MA 02062
USA
(781) 461 - 3558
Rajesh.Mishra@Analog.com

In the last couple of years consumer audio/video products have moved from closed single format (ex. MPEG2/AC-3) to closed single format video with multi-format audio (ex. AC-3, MP3, WMA, etc.). In the next few years, these products will need to support a whole host of new audio, video and image formats as they become more open network connected appliances. These products include: networked Personal Video Recorders (PVR), Digital Media Adapters (DMA), Internet Protocol based Television (IPTV), Networked Digital Receivers, IP based Video On Demand players (IP-VOD) and IP Set Top Boxes (IP-STB).

Although the devices have a lot in common, each has a different set of additional requirements. For example: an IP-VOD player needs to communicate with a specific media server, or a PVR needs to be able to place broadcast content onto a storage device, and later stream from that device.

Common requirements

All of these products need to stream compressed audio/video over a network, process the stream containers, decode the streams, and present a synchronized audio/video output to the consumer. All these tasks must be done in hard real time. Failure to perform these tasks in real time, cause an unpleasant and unacceptable user experience. To achieve this level of performance on a low cost processor, such as an Analog Devices Blackfin®, the SW architecture must achieve a very high level of hardware efficiency.

Since these devices can be connected to the internet, many different codecs will be required. Also, digital rights management (DRM) is a key enabler to network based media appliances since the value of the content must be protected. This content protection must be proven before content owners will allow it to be available via an open network.

It is impossible to release a product with all the Audio and Video codecs, DRM schemes, media server protocols, and audio/video transport capabilities built in, so it is critical that these network connected devices be field upgradeable.

To allow the products to be widely deployed and accepted they must allow an OEM to customize the "look and feel" of the device. The "look and feel" is very important to OEM's so that a device fits in with the rest of their product line. Another critical characteristic is that these products must be affordable. To accomplish this, system design must utilize the available low cost processor in a highly efficient manner.

IP-STB Example

An IP-STB is generally connected to an ISP's WAN that has media servers directly connected.

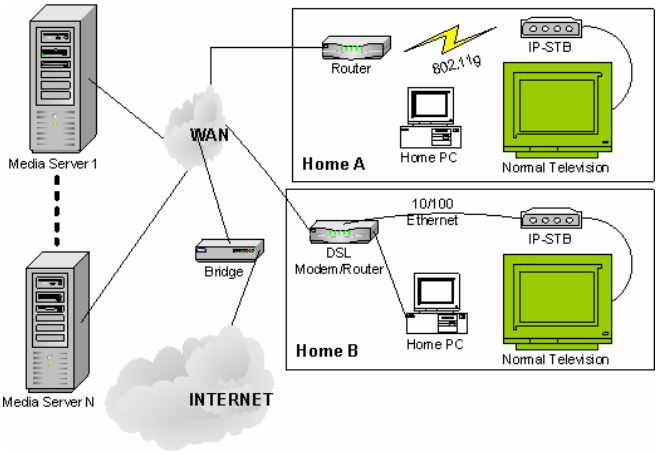

Figure 1 - IP-STB use model

The Internet is connected into the ISP's WAN via a bridge or firewall device. This configuration requires that the IP-STB and the specific media

server understand exactly how to communicate with each other.

IP-STB's also need to be able to browse the internet and the media server's program guide. This requires the IP-STB to integrate a standard web browser, including many of the capabilities of a desktop browser (i.e. Flash® player, scalable fonts, progressive JPEG codec, cookies, etc.). Embedded browsers are widely available, but generally require an OS, such as Linux.

The ability to use the IP-STB as a karaoke machine is a common request from many regions of the world. Since the IP-STB has all the interfaces, Audio, Video, and Graphics, adding the SW and User interface is a fairly straightforward Linux application.

Architecture

The basic architecture of a single chip Application & Multimedia processor requires that both hard real time deadlines are met, while in the same system allowing an application to have access to all the resources it needs. When a web browser is running, it should get close to 100% of the processor resources to give the user a satisfying experience.

When a user requests to watch a video from the media server, they will not tolerate pauses in the video, obvious dropped frames, or sound problems. Thus, when decoding a media stream, whatever level of processor resources needed will be applied to decoding and presenting the audio and video.

To accomplish this, we have two separate scheduling domains, a custom hard real time scheduler that is specifically designed to support the real time requirements of streaming multimedia decoders (called "Streaming Multimedia Kernel" or SMK), and Linux as a convenient widely available application scheduler.

Taken all together, the functions scheduled by SMK make up a virtual hardware device. That is, to the application running on top of Linux, the device looks like any other hardware audio/video decoder peripheral.

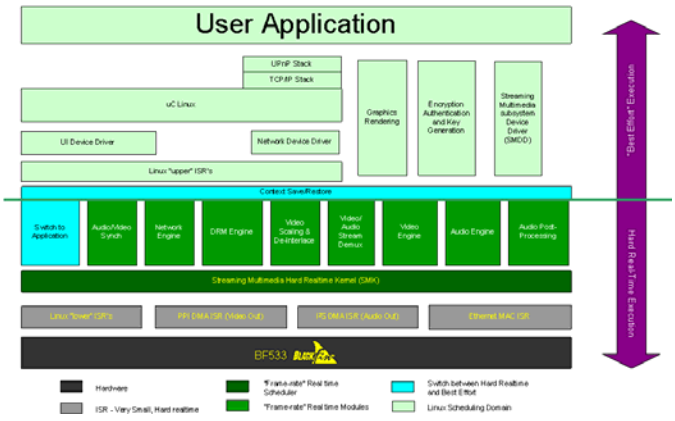

Figure 2 - Streaming Multimedia System

Streaming Multimedia "Virtual Hardware"

The Streaming Multimedia System (SMS) guarantees presentation of video media at the requisite display rate (e.g., 30 frames of D1 resolution per second for television output) and audio media at the output sample rate (e.g., 48 KHz for DVD quality audio). These hard real time constraints are met by the entire system as a whole, not by individual software components (i.e., the media codec software). As illustrated in Figure 3, the system delivers the flow of data to the DACs at their output rates by ensuring that all elements in the data path operate congruously to meet this requirement, including the Streaming Multimedia: Network Engine (SMNE), Transport Engine (SMTE), Audio Engine (SMAE) and Video Engine (SMVE), and the Audio and Video DAC drivers.

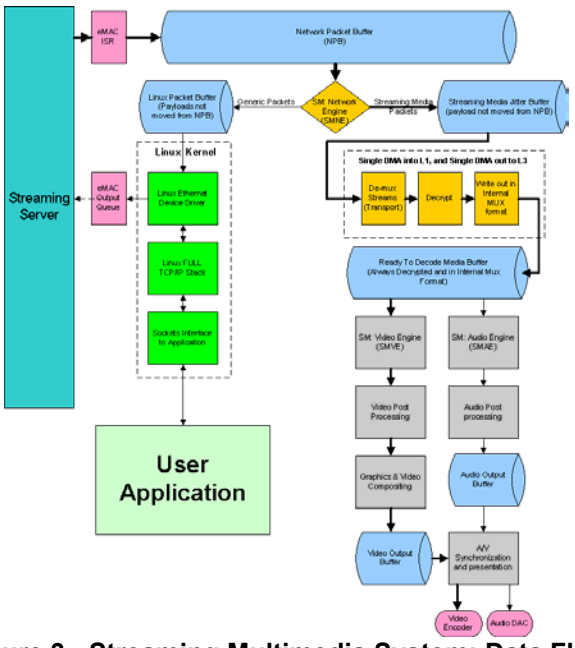

Figure 3 - Streaming Multimedia System: Data Flow

Network Packet Transport

SMNE is a reduced implementation of the network layer of the TCP/IP protocol stack, required to allow timely and efficient delivery of streaming media IP packets to the SMTE from the network device. SMNE caters especially to media streams encapsulated in IP packets, because: it determines which IP packets belong to the media stream as soon as they arrive from the network device (e.g. Ethernet controller), and it forwards only the designated media stream packets to the SMTE in the most efficient way. This scheme is different from traditional TCP/IP implementations in which network packet data must be processed by several layers before the streaming media application can process it. The results of the SMNE-based network transport mechanism are: streamed media packets are in the network stack for a much shorter period of time (lower latency – faster response time), a lot more IP packets can be processed per unit time to allow transport for even high bit-rate media streams (higher throughput), and system resources are used more efficiently resulting in lower processor, memory, and I/O consumption.

Video Processing

Video processing generally takes much more processing time than audio processing. For some codec types, this video processing time can exceed more than the allotted time to output a video frame (e.g., 33 milliseconds for 30 Hz interleaved video output) or a sequence of video frames. In these special cases, the data path provides the appropriate level of buffering to compensate for this brief shortage of resource.

Audio Processing

Audio processing must guarantee lossless output at the sample level as compared to video, since the human ear is much more sensitive to sound than the eye is to motion video. Because of this physiological bias, the audio output generally serves as the "master" clock in a multi-media system. Although audio processing takes less time than video, the audio output buffers must always be kept non-empty to guarantee lossless output. Conversely, the video output may, at the worst case, drop frames if its buffer levels cannot be maintained.

Audio/Video Synchronization

The system maintains two types of media synchronization: intra-stream sync – synchronization within audio or video stream between source and sink, and inter-stream sync – synchronization between both audio and video streams. Intra-stream synchronization requires locking the source clock derived from the presentation timestamps within the media packets to the output clocks. Inter-stream synchronization requires use of the audio output clock as the "master" system clock, for reasons cited above.

Managing memory bandwidth

Video codecs require megabytes of memory for both compressed and decompressed video data, even for standard definition television resolutions. On-chip memory in low-cost processors is usually offered in kilobytes - orders of magnitude below what is required. This requires allocation of external system memory, typically SDRAM. Although SDRAM offers a large amount of memory at a low cost, access time is several orders of magnitude above that of on-chip static memory. Low-cost processors usually provide two facilities to reduce this access time: instruction and data cache memory and direct-memory-access (DMA).

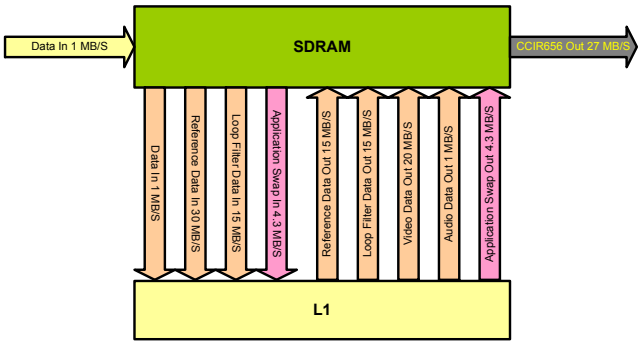

Figure 4 – Video Codec Memory Bandwidth Requirements

Figure 4 describes some characteristics of memory accesses during video processing. The video codecs operate mostly on 2D regions of fixed block sizes. These 2D blocks reside in memory in both linear (e.g., compressed and reference data) and raster (e.g., video output data) forms.

Data caches typically perform well when data accesses are performed in linear fashion. The performance, however, is not optimal when these accesses are non-linear, such as for raster 2D blocks. For this reason, some processors such as the Blackfin offer two strides in the memory-to-memory DMA engines to allow 2D blocks to be transported between external and internal memories. Because DMA is independent of

processor execution, it can be pipe-lined with processor execution to allow the most optimal processing of 2D data regions.

Aside from video processing, other software and hardware components in a multimedia system require use of DMA engines, including system peripherals, audio codecs, network stack, transport layer, and application-level software. Hence, the DMA controllers and internal buses are resources shared by all of these system components and must be managed. Some low-cost processors, such as Blackfin, provide facilities to allow system software to manage these resources, including DMA priority assignment, traffic control, and bus arbitration policy. SMS utilizes all of these facilities for optimum system performance.

Managing Processor utilization

Multimedia processors, such as Blackfin, value low cost over extremely high clock rates. Unlike higher cost general-purpose processors, multimedia processors are designed with "just enough" performance. This lack of excessive MIPS emphasizes the need to manage processor execution as efficiently as possible. Since code execution is based on execution context (interrupt servicing and task processing), two factors are responsible for execution efficiency: interrupt priority assignment and task scheduling.

Interrupt sources range from peripherals and DMA controllers (hardware interrupts) to code (software interrupts and exceptions). The variety of peripherals in a multimedia system is made even more complex by the persistent use of DMA. The interrupt frequency and service time determine the priority of the interrupt. For example, audio interrupts occur as frequently as every microsecond to process each sample from an AC'97 controller operating at 48 KHZ. The AC'97 interrupt service routines also consume significant processor cycles. Because of these characteristics, the AC'97 interrupt may be assigned a much higher priority than other interrupt sources

SMK executes the audio and video codecs at a higher priority level than all other tasks and processes in the system. All other tasks, including the overlying operating system (e.g., uCLinux), are considered to execute in the "slack" period, when audio and video codecs do not require the processor. When the multimedia codecs are executing for prolonged periods (for example, when the video codec has to process successive frames of high motion), execution of these non-critical tasks are delayed accordingly. Consequently, these tasks follow a "best effort" scheduling policy, whereas the multimedia codecs execute as interrupt-driven, "hard" real time tasks. Hence, the response time of the multimedia codecs is directly proportional to interrupt latency; the response time of the "best-effort" tasks is dependant upon the instantaneous requirements of the multimedia codecs.

Managing I/O and Memories

Since SM accesses arbitrary regions in the full processor address space, it utilizes and manages the following system resources: fast on-chip SRAM, slow off-chip SDRAM, memory-mapped peripherals, and memory management unit. As all of these resources may be used system-wide, SM is responsible for controlling access to these shared system resources.

Since the size of on-chip memory makes it a valuable system resource, SM reserves most of it for use under its own discretion. This reservation is made statically during system initialization, along with allocation of reserved areas in off-chip memories. These reserved regions in the processor address space are supplemented by all address spaces occupied by peripherals. The memory-mapped I/O regions that belong to the peripherals controlled by SM are also reserved (ex. memory-mapped audio and video output peripherals). All other memory-mapped regions that are not reserved by SM are left to the discretion of the overlying operating environment (e.g., uCLinux).

Although many desktop processors provide an on-chip memory management unit (MMU) with virtual memory support, they are not widely used in the multimedia processor market. In multimedia processors a memory protection and cache management unit is much more beneficial. Virtual memory adds non-determinism to the execution environment. The Blackfin processor provides Cache Protection Look aside Buffers (CPLBs) to support deterministic multimedia applications. This hardware unit allows attributes to be assigned to memory regions, or "pages". These pages are designated as either cached or non-cached, with an access mode (user-mode only, supervisor-mode only, or both), and size. CPLBs are separately available for both instruction and data memories, to allow independent management of both spaces. The SM manages on-chip (and off-chip) memories by statically allocating these resources, it also controls memory access by managing the CPLBs. CPLB management includes: assignment of page attributes, page replacement, and cache operation.

uC Linux

The standard micro-C Linux kernel requires certain modifications, albeit small, in order to interoperate with the Streaming Multimedia System (SMS). These modifications fortunately are only needed in the machine-dependant portion of the Linux kernel, credited to the elegant design of the operating system. In addition to these minimal core kernel changes, several new kernel modules are needed to interface to the SMS.

Core Kernel Requirements

Kernel Bootstrap

The standard uCLinux kernel assumes little, if nothing at all, of the machine state during core kernel initialization. Consequently, it initializes the entire machine state for the kernel run-time environment, including kernel stack, core machine registers, interrupt and exception vectors, and memories. However, in the multimedia framework of the SMS, the run-time environment has already been initialized for the most part and only certain interrupt and exception vectors and portions of memory need to be set up. The multimedia framework also supplies the kernel boot parameters as standard string-based command-line options.

Exception Handling

The standard uCLinux kernel has its own exception vector table, that has kernel entry points for handling hardware and software exception events. The SMS now handles all of these events. Only those exceptions that need to be delivered to the Linux kernel are system calls. This is performed through a Deferred Exception Handler (DEH), which executes as a Linux exception handler. The DEH executes at a higher execution level than the kernel, possible through software interrupt priority assignment on the Blackfin architecture.

Interrupt Handling

The uCLinux kernel manages two types of interrupts: the core timer interrupt and peripheral interrupts.

The core timer serves as the system clock, whose value is held by the global kernel variable “jiffies”. In order to maintain reliable system response time and synchronization, this timer should have marginal variance. SMS automatically guarantees this low level of clock jitter by fixed hardware assignment in the Blackfin processor of the core timer interrupt at the highest priority level.

Peripheral interrupts are only managed for those devices controlled by the uCLinux OS. All others are handled in the SMS. All Linux device interrupts share a common hardware interrupt priority level for two reasons: the standard uCLinux kernel has a common interrupt dispatcher that is invoked when any hardware interrupt event occurs, and all interrupt vectors are managed in the SMS.

In order to maintain low interrupt latency throughout the multimedia system, timer and Linux device interrupt service routines execute in on-chip instruction memory. Since this memory is a valuable system resource, these routines need to be minute in code size and execution time.

Memory Allocation

The standard uCLinux kernel consists of following types of memory sections: init/exit text and data, run-time text and read-only data, run-time read/write kernel data, ramdisk area, and uninitialized kernel data (bss). The kernel reclaims the init sections for heap and processes. All of these regions need not be contiguous in physical memory, and so are assigned in a fragmented fashion by the SMS. This assignment is made statically in the linker description file used during both the kernel build and SMS build.

SMS Interface

Linux and its applications must be able to communicate with the SMS for several reasons: sending media control and other commands to the SMS, transmitting and receiving network packets on the same interface as the SMS, making DMA requests, and passing “raw” audio data for mixing with audio codec data.

SM Command Device Driver

A character-mode device driver is required for uCLinux applications to communicate with the SMS. This communication may include sending media control commands (e.g., PAUSE, STOP, PLAY, etc.), triggering SM functions (e.g., Display Graphics Overlay function), and message passing (e.g., setting media parameters and modes).

VHNE Device Driver

A uCLinux Ethernet device driver module is required for receiving packets from the network interface and delivering them to the Linux TCP/IP network stack. This driver interfaces directly to the

SMNE, which forwards all packets to uCLinux other than those destined for SM and transmits packets from uCLinux applications on the network interface.

SMDE Device Driver

The uCLinux kernel file-systems, device drivers, and applications may utilize the memory-to-memory DMA controller to facilitate data movement. A kernel module is required to communicate with the SM DMA Engine (SMDE) to make these DMA requests and receive end-of-DMA notifications. Just as the other shared peripherals (eg. network interface), no guarantees are made when the requests will be serviced; that is, the requests are processed “best-effort”.

SMAE Device Driver

A sound driver is required to supply audio data to be mixed with the audio codec output within the SMS. This data may originate from WAV files or compressed formats at the Linux kernel or application level. This driver also interfaces directly to the SMS to supply the data and control the mixing operation.

Application framework

From the application developers viewpoint, the system must have all the functionality available on a standard embedded microprocessor system. The interaction between the SMS and the application must be well defined and the debug environment must have the capability to efficiently integrate a user application with the media decoder. To implement a complete system the application will require; multiple processes and threads, inter-process communications and synchronization, Linux device drivers for all SMS capabilities, shared data and program objects, memory management functionality, and a debug server (such as GDB server).

Conclusion

The days of implementing media-centric products using two separate processors: a general purpose applications processor, and a media processor, are coming to a close. By using our Streaming Multimedia System (SMS) software framework, we are able to deliver full DVD quality multimedia decode of a variety of different compression formats (WM, MPEG2, MPEG4, JPEG, etc.). All this on a single low cost 600MHz Blackfin. This same Blackfin also runs uCLinux and applications that utilize both the uCLinux OS capabilities and our SMS “Virtual Hardware.” This network connected, field upgradeable system makes this pure software approach uniquely suited to keep up with the accelerating rate of change in this new category of network connected media processing client devices.