MULTIMEDIA SIGNAL PROCESSING SYSTEM

BACKGROUND OF THE INVENTION

TECHNICAL FIELD

The invention relates to the time shifting of television broadcast signals. More particularly, the invention relates to the real time capture, storage, and display of television broadcast signals.

DESCRIPTION OF THE PRIOR ART

The Video Cassette Recorder (VCR) has changed the lives of television (TV) viewers throughout the world. The VCR has offered viewers the flexibility to time-shift TV programs to match their lifestyles.

The viewer stores TV programs onto magnetic tape using the VCR. The VCR gives the viewer the ability to play, rewind, fast-forward and pause the stored program material. These functions enable the viewer to pause the program playback whenever he desires, fast forward through unwanted program material or commercials, and to replay favorite scenes. However, a VCR cannot both capture and play back information at the same time.

One approach to solving this problem is to use several VCRs. For example, if two video tape recorders are available, it might be possible to Ping-Pong between the two. In this case, the first recorder is started at the beginning of the program of interest. If the viewer wishes to rewind the broadcast, the second recorder begins recording, while the first recorder is halted, rewound to the appropriate place, and playback initiated. However, at least a third video tape recorder is required if the viewer wishes to fast forward to some point in time after the initial rewind was requested. In this case, the third recorder starts recording the broadcast stream while the second is halted and rewound to the appropriate position. Continuing this exercise, one can quickly see that the equipment becomes unwieldy, unreliable, expensive, and hard to operate, while never supporting all desired functions. In addition, tapes are of finite length, and may potentially end at inconvenient times, drastically lowering the value of the solution.

The use of digital computer systems to solve this problem has been suggested. U.S. Pat. No. 5,371,551 issued to Logan *et al.*, on 6 December 1994, teaches a method for concurrent video recording and playback. It presents a microprocessor controlled broadcast and playback device. Said device compresses and stores video data onto a hard disk. However, this approach is difficult to implement because the processor requirements for keeping up with the high video rates makes the device expensive and problematic. The microprocessor must be extremely fast to keep up with the incoming and outgoing video data.

It would be advantageous to provide a multimedia signal processing system that gives the user the ability to simultaneously record and play back TV broadcast programs. It would further be advantageous to provide a multimedia signal processing system that utilizes an approach that decouples the microprocessor from the high video data rates, thereby reducing the microprocessor and system requirements, which are at a premium.

SUMMARY OF THE INVENTION

The invention provides a multimedia signal processing system. The invention utilizes an easily manipulated, low cost multimedia storage and display system that allows the user to view a television broadcast program with the option of instantly reviewing previous scenes within the program. In addition, the invention allows the user to store selected television broadcast programs while the user is simultaneously watching or reviewing another program.

A preferred embodiment of the invention accepts television (TV) input streams in a multitude of forms, for example, analog forms such as National Television Standards Committee (NTSC) or PAL broadcast, and digital forms such as Digital Satellite System (DSS), Digital Broadcast Services (DBS), or Advanced Television Standards Committee (ATSC). Analog TV streams are converted to an Moving Pictures Experts Group (MPEG) formatted stream for internal transfer and manipulation, while pre-formatted MPEG streams are extracted from the digital TV signal and presented in a similar format to encoded analog streams.

The invention parses the resulting MPEG stream and separates it into its video and audio components. It then stores the components into temporary buffers. Events are recorded that indicate the type of component that has been found, where it is located, and when it occurred. The program logic is notified that an event has occurred and the data is extracted from the buffers.

The parser and event buffer decouple the CPU from having to parse the MPEG stream and from the real time nature of the data streams. This decoupling allows for slower CPU and bus speeds, which translates to lower system costs.

The video and audio components are stored on a storage device. When the program is requested for display, the video and audio components are extracted from the storage device and reassembled into an MPEG stream. The MPEG stream is sent to a decoder. The decoder converts the MPEG stream into TV output signals and delivers the TV output signals to a TV receiver.

User control commands are accepted and sent through the system. These commands affect the flow of said MPEG stream and allow the user to view stored programs with at least the following functions: reverse, fast forward, play, pause, index, fast/slow reverse play, and fast/slow play.

Furthermore, the invention incorporates a versatile system architecture that makes it possible to provide the invention in a variety of configurations, each adapted to receive input signals from a different source. At the highest level, the system board comprises an input section and an output section, in which the output section includes the core functional components. Across all configurations, the output section remains substantially the same, incorporating the three core components either as three discrete chips or as a chipset, while the input section varies according to the signal type and the source. In this way, several configurations are provided, each one requiring only minor modifications to the system board. The system architecture thus simplifies the design and manufacturing challenge presented by producing units to serve different markets, such as digital satellite, digital cable and analog cable.

The core components of the output section of the invention include: a CPU having the primary function of initializing and controlling the remaining system hardware components, an MPEG-2 decoder/graphics subsystem, in communication with the CPU, primarily responsible for decoding transport streams delivered from the input section, and a media manager, in communication with the MPEG-2 decoder/graphics subsystem, having a variety of functions, including media processing, high-speed transport output and miscellaneous I/O functionality. The invention further includes a transport stream interface between the input section and output sections, several memory components, one or more mass storage devices for storage of the separate audio and video components of the input signal, and a system bus for the transfer of data between the various system components of the invention. Other aspects and

advantages of the invention will become apparent from the following detailed description in combination with the accompanying drawings, illustrating, by way of example, the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block schematic diagram of a high level view of a preferred embodiment of the invention according to the invention;

Fig. 2 is a block schematic diagram of a preferred embodiment of the invention using multiple input and output modules according to the invention;

Fig. 3 is a schematic diagram of an Moving Pictures Experts Group (MPEG) data stream and its video and audio components according to the invention;

Fig. 4 is a block schematic diagram of a parser and four direct memory access (DMA) input engines contained in the Media Switch according to the invention;

Fig. 5 is a schematic diagram of the components of a packetized elementary stream (PES) buffer according to the invention;

Fig. 6 is a schematic diagram of the construction of a PES buffer from the parsed components in the Media Switch output circular buffers;

Fig. 7 is a block schematic diagram of the Media Switch and the various components that it communicates with according to the invention;

Fig. 8 is a block schematic diagram of a high level view of the program logic according to the invention;

Fig. 9 is a block schematic diagram of a class hierarchy of the program logic according to the invention;

Fig. 10 is a block schematic diagram of a preferred embodiment of the clip cache component of the invention according to the invention;

Fig. 11 is a block schematic diagram of a preferred embodiment of the invention that emulates a broadcast studio video mixer according to the invention;

Fig. 12 is a block schematic diagram of a closed caption parser according to the invention;

Fig. 13 is a block schematic diagram of a high level view of a preferred embodiment of the invention utilizing a VCR as an integral component of the invention according to the invention.

Fig. 14 is a block schematic diagram of a high level view of a system architecture according to the invention;

Fig. 15 is a block schematic diagram of an output section of the system of Figure 14 according to the invention;

Fig. 16 is a block schematic diagram of a first version of an input section of the system of Figure 14, adapted to receive an analog signal according to the invention

Fig. 17 is a block schematic diagram of a second version of an input section of the system of Figure 14, adapted to receive a digital satellite signal according to the invention;

Fig. 18 is a block schematic diagram of a third version of an input section of the system of Figure 14, adapted to receive a digital cable signal according to the invention;

Fig. 19 is a block diagram of a first embodiment of the system of Figure 14 according to the invention;

Fig. 20 is a block schematic diagram of a second embodiment of the system of Figure 14 according to the invention;

Fig. 21 is a block schematic diagram of a third embodiment of the system of Figure 14 according to the invention; and

Fig. 22 is a block schematic diagram of a system for processing media stream data across multiple channels, in parallel according to the invention.

DETAILED DESCRIPTION OF THE INVENTION

The invention is embodied in a multimedia signal processing system. A system according to the invention provides a multimedia storage and display system that allows the user to view a television broadcast program with the option of instantly reviewing previous scenes within the program. The invention additionally provides the user with the ability to store selected television broadcast programs while simultaneously watching or reviewing another program and to view stored programs with at least the following functions: reverse, fast forward, play, pause, index, fast/slow reverse play, and fast/slow play.

Referring to Fig. 1, a preferred embodiment of the invention has an Input Section 101, Media Switch 102, and an Output Section 103. The Input Section 101 takes television (TV) input streams in a multitude of forms, for example, National Television Standards Committee (NTSC) or PAL broadcast, and digital forms such as Digital Satellite System (DSS), Digital Broadcast Services (DBS), or Advanced Television Standards Committee (ATSC). DBS, DSS and ATSC are based on standards called Moving Pictures Experts Group 2 (MPEG2) and MPEG2 Transport. MPEG2 Transport is a standard for formatting the digital data stream from the TV source transmitter so that a TV receiver can disassemble the input stream to find programs in the multiplexed signal. The Input Section 101 produces MPEG streams. An MPEG2 transport multiplex supports multiple programs in the same broadcast channel, with multiple video and audio feeds and private data. The Input Section 101 tunes the channel to a particular program, extracts a specific MPEG program out of it, and feeds it to the rest of the system. Analog TV signals are encoded into a similar MPEG format using separate video and audio encoders, such that the remainder of the system is unaware of how the signal was obtained. Information may be modulated into the Vertical Blanking Interval (VBI) of the analog TV signal in a number of standard ways; for example, the North American Broadcast Teletext Standard (NABTS) may be used to modulate information onto lines 10 through 20 of an NTSC signal, while the FCC mandates the use of line 21 for Closed Caption (CC) and Extended Data Services (EDS). Such signals are decoded by the input section and passed to the other sections as if they were delivered via an MPEG2 private data channel.

The Media Switch 102 mediates between a microprocessor CPU 106, hard disk or storage device 105, and memory 104. Input streams are converted to an MPEG stream and sent to the Media Switch 102. The Media Switch 102 buffers the MPEG stream into memory. It then performs two operations if the user is watching real time

TV: the stream is sent to the Output Section 103 and it is written simultaneously to the hard disk or storage device 105.

The Output Section 103 takes MPEG streams as input and produces an analog TV signal according to the NTSC, PAL, or other required TV standards. The Output Section 103 contains an MPEG decoder, On-Screen Display (OSD) generator, analog TV encoder and audio logic. The OSD generator allows the program logic to supply images which will be overlayed on top of the resulting analog TV signal. Additionally, the Output Section can modulate information supplied by the program logic onto the VBI of the output signal in a number of standard formats, including NABTS, CC and EDS.

With respect to Fig. 2, the invention easily expands to accommodate multiple Input Sections (tuners) 201, 202, 203, 204, each can be tuned to different types of input. Multiple Output Modules (decoders) 206, 207, 208, 209 are added as well. Special effects such as picture in a picture can be implemented with multiple decoders. The Media Switch 205 records one program while the user is watching another. This means that a stream can be extracted off the disk while another stream is being stored onto the disk.

Referring to Fig. 3, the incoming MPEG stream 301 has interleaved video 302, 305, 306 and audio 303, 304, 307 segments. These elements must be separated and recombined to create separate video 308 and audio 309 streams or buffers. This is necessary because separate decoders are used to convert MPEG elements back into audio or video analog components. Such separate delivery requires that time sequence information be generated so that the decoders may be properly synchronized for accurate playback of the signal.

The Media Switch enables the program logic to associate proper time sequence information with each segment, possibly embedding it directly into the stream. The time sequence information for each segment is called a time stamp. These time stamps are monotonically increasing and start at zero each time the system boots up. This allows the invention to find any particular spot in any particular video segment. For example, if the system needs to read five seconds into an incoming contiguous video stream that is being cached, the system simply has to start reading forward into the stream and look for the appropriate time stamp.

A binary search can be performed on a stored file to index into a stream. Each stream is stored as a sequence of fixed-size segments enabling fast binary searches because

of the uniform timestamping. If the user wants to start in the middle of the program, the system performs a binary search of the stored segments until it finds the appropriate spot, obtaining the desired results with a minimal amount of information. If the signal were instead stored as an MPEG stream, it would be necessary to linearly parse the stream from the beginning to find the desired location.

With respect to Fig. 4, the Media Switch contains four input Direct Memory Access (DMA) engines 402, 403, 404, 405 each DMA engine has an associated buffer 410, 411, 412, 413. Conceptually, each DMA engine has a pointer 406, a limit for that pointer 407, a next pointer 408, and a limit for the next pointer 409. Each DMA engine is dedicated to a particular type of information, for example, video 402, audio 403, and parsed events 405. The buffers 410, 411, 412, 413 are circular and collect the specific information. The DMA engine increments the pointer 406 into the associated buffer until it reaches the limit 407 and then loads the next pointer 408 and limit 409. Setting the pointer 406 and next pointer 408 to the same value, along with the corresponding limit value creates a circular buffer. The next pointer 408 can be set to a different address to provide vector DMA.

The input stream flows through a parser 401. The parser 401 parses the stream looking for MPEG distinguished events indicating the start of video, audio or private data segments. For example, when the parser 401 finds a video event, it directs the stream to the video DMA engine 402. The parser 401 buffers up data and DMAs it into the video buffer 410 through the video DMA engine 402. At the same time, the parser 401 directs an event to the event DMA engine 405 which generates an event into the event buffer 413. When the parser 401 sees an audio event, it redirects the byte stream to the audio DMA engine 403 and generates an event into the event buffer 413. Similarly, when the parser 401 sees a private data event, it directs the byte stream to the private data DMA engine 404 and directs an event to the event buffer 413. The Media Switch notifies the program logic via an interrupt mechanism when events are placed in the event buffer.

Referring to Figs. 4 and 5, the event buffer 413 is filled by the parser 401 with events. Each event 501 in the event buffer has an offset 502, event type 503, and time stamp field 504. The parser 401 provides the type and offset of each event as it is placed into the buffer. For example, when an audio event occurs, the event type field is set to an audio event and the offset indicates the location in the audio buffer 411. The program logic knows where the audio buffer 411 starts and adds the offset to find the event in the stream. The address offset 502 tells the program logic where the next

event occurred, but not where it ended. The previous event is cached so the end of the current event can be found as well as the length of the segment.

With respect to Figs. 5 and 6, the program logic reads accumulated events in the event buffer 602 when it is interrupted by the Media Switch 601. From these events the program logic generates a sequence of logical segments 603 which correspond to the parsed MPEG segments 615. The program logic converts the offset 502 into the actual address 610 of each segment, and records the event length 609 using the last cached event. If the stream was produced by encoding an analog signal, it will not contain Program Time Stamp (PTS) values, which are used by the decoders to properly present the resulting output. Thus, the program logic uses the generated time stamp 504 to calculate a simulated PTS for each segment and places that into the logical segment timestamp 607. In the case of a digital TV stream, PTS values are already encoded in the stream. The program logic extracts this information and places it in the logical segment timestamp 607.

The program logic continues collecting logical segments 603 until it reaches the fixed buffer size. When this occurs, the program logic generates a new buffer, called a Packetized Elementary Stream (PES) 605 buffer containing these logical segments 603 in order, plus ancillary control information. Each logical segment points 604 directly to the circular buffer, *e.g.*, the video buffer 613, filled by the Media Switch 601. This new buffer is then passed to other logic components, which may further process the stream in the buffer in some way, such as presenting it for decoding or writing it to the storage media. Thus, the MPEG data is not copied from one location in memory to another by the processor. This results in a more cost effective design since lower memory bandwidth and processor bandwidth is required.

A unique feature of the MPEG stream transformation into PES buffers is that the data associated with logical segments need not be present in the buffer itself, as presented above. When a PES buffer is written to storage, these logical segments are written to the storage medium in the logical order in which they appear. This has the effect of gathering components of the stream, whether they be in the video, audio or private data circular buffers, into a single linear buffer of stream data on the storage medium. The buffer is read back from the storage medium with a single transfer from the storage media, and the logical segment information is updated to correspond with the actual locations in the buffer 606. Higher level program logic is unaware of this transformation, since it handles only the logical segments, thus stream data is easily managed without requiring that the data ever be copied between locations in DRAM by the CPU.

A unique aspect of the Media Switch is the ability to handle high data rates effectively and inexpensively. It performs the functions of taking video and audio data in, sending video and audio data out, sending video and audio data to disk, and extracting video and audio data from the disk on a low cost platform. Generally, the Media Switch runs asynchronously and autonomously with the microprocessor CPU, using its DMA capabilities to move large quantities of information with minimal intervention by the CPU.

Referring to Fig. 7, the input side of the Media Switch 701 is connected to an MPEG encoder 703. There are also circuits specific to MPEG audio 704 and vertical blanking interval (VBI) data 702 feeding into the Media Switch 701. If a digital TV signal is being processed instead, the MPEG encoder 703 is replaced with an MPEG2 Transport Demultiplexor, and the MPEG audio encoder 704 and VBI decoder 702 are deleted. The demultiplexor multiplexes the extracted audio, video and private data channel streams through the video input Media Switch port.

The parser 705 parses the input data stream from the MPEG encoder 703, audio encoder 704 and VBI decoder 702, or from the transport demultiplexor in the case of a digital TV stream. The parser 705 detects the beginning of all of the important events in a video or audio stream, the start of all of the frames, the start of sequence headers - all of the pieces of information that the program logic needs to know about in order to both properly play back and perform special effects on the stream, *e.g.* fast forward, reverse, play, pause, fast/slow play, indexing, and fast/slow reverse play.

The parser 705 places tags 707 into the FIFO 706 when it identifies video or audio segments, or is given private data. The DMA 709 controls when these tags are taken out. The tags 707 and the DMA addresses of the segments are placed into the event queue 708. The frame type information, whether it is a start of a video I-frame, video B-frame, video P-frame, video PES, audio PES, a sequence header, an audio frame, or private data packet, is placed into the event queue 708 along with the offset in the related circular buffer where the piece of information was placed. The program logic operating in the CPU 713 examines events in the circular buffer after it is transferred to the DRAM 714.

The Media Switch 701 has a data bus 711 that connects to the CPU 713 and DRAM 714. An address bus 712 is also shared between the Media Switch 701, CPU 713, and DRAM 714. A hard disk or storage device 710 is connected to one of the ports of the Media Switch 701. The Media Switch 701 outputs streams to an MPEG video decoder 715 and a separate audio decoder 717. The audio decoder 717 signals

contain audio cues generated by the system in response to the user's commands on a remote control or other internal events. The decoded audio output from the MPEG decoder is digitally mixed 718 with the separate audio signal. The resulting signals contain video, audio, and on-screen displays and are sent to the TV 716.

The Media Switch 701 takes in 8-bit data and sends it to the disk, while at the same time extracts another stream of data off of the disk and sends it to the MPEG decoder 715. All of the DMA engines described above can be working at the same time. The Media Switch 701 can be implemented in hardware using a Field Programmable Gate Array (FPGA), ASIC, or discrete logic.

Rather than having to parse through an immense data stream looking for the start of where each frame would be, the program logic only has to look at the circular event buffer in DRAM 714 and it can tell where the start of each frame is and the frame type. This approach saves a large amount of CPU power, keeping the real time requirements of the CPU 713 small. The CPU 713 does not have to be very fast at any point in time. The Media Switch 701 gives the CPU 713 as much time as possible to complete tasks. The parsing mechanism 705 and event queue 708 decouple the CPU 713 from parsing the audio, video, and buffers and the real time nature of the streams, which allows for lower costs. It also allows the use of a bus structure in a CPU environment that operates at a much lower clock rate with much cheaper memory than would be required otherwise.

The CPU 713 has the ability to queue up one DMA transfer and can set up the next DMA transfer at its leisure. This gives the CPU 713 large time intervals within which it can service the DMA controller 709. The CPU 713 may respond to a DMA interrupt within a larger time window because of the large latency allowed. MPEG streams, whether extracted from an MPEG2 Transport or encoded from an analog TV signal, are typically encoded using a technique called Variable Bit Rate encoding (VBR). This technique varies the amount of data required to represent a sequence of images by the amount of movement between those images. This technique can greatly reduce the required bandwidth for a signal, however sequences with rapid movement (such as a basketball game) may be encoded with much greater bandwidth requirements. For example, the Hughes DirecTV satellite system encodes signals with anywhere from 1 to 10Mb/s of required bandwidth, varying from frame to frame. It would be difficult for any computer system to keep up with such rapidly varying data rates without this structure.

With respect to Fig. 8, the program logic within the CPU has three conceptual components: sources 801, transforms 802, and sinks 803. The sources 801 produce buffers of data. Transforms 802 process buffers of data and sinks 803 consume buffers of data. A transform is responsible for allocating and queuing the buffers of data on which it will operate. Buffers are allocated as if "empty" to sources of data, which give them back "full". The buffers are then queued and given to sinks as "full", and the sink will return the buffer "empty".

A source 801 accepts data from encoders, *e.g.,* a digital satellite receiver. It acquires buffers for this data from the downstream transform, packages the data into a buffer, then pushes the buffer down the pipeline as described above. The source object 801 does not know anything about the rest of the system. The sink 803 consumes buffers, taking a buffer from the upstream transform, sending the data to the decoder, and then releasing the buffer for reuse.

There are two types of transforms 802 used: spatial and temporal. Spatial transforms are transforms that perform, for example, an image convolution or compression/decompression on the buffered data that is passing through. Temporal transforms are used when there is no time relation that is expressible between buffers going in and buffers coming out of a system. Such a transform writes the buffer to a file 804 on the storage medium. The buffer is pulled out at a later time, sent down the pipeline, and properly sequenced within the stream.

Referring to Fig. 9, a C++ class hierarchy derivation of the program logic is shown. The TiVo Media Kernel (Tmk) 904, 908, 913 mediates with the operating system kernel. The kernel provides operations such as: memory allocation, synchronization, and threading. The TmkCore 904, 908, 913 structures memory taken from the media kernel as an object. It provides operators, new and delete, for constructing and deconstructing the object. Each object (source 901, transform 902, and sink 903) is multi-threaded by definition and can run in parallel.

The TmkPipeline class 905, 909, 914 is responsible for flow control through the system. The pipelines point to the next pipeline in the flow from source 901 to sink 903. To pause the pipeline, for example, an event called "pause" is sent to the first object in the pipeline. The event is relayed on to the next object and so on down the pipeline. This all happens asynchronously to the data going through the pipeline. Thus, similar to applications such as telephony, control of the flow of MPEG streams is asynchronous and separate from the streams themselves. This allows for a simple logic design that is at the same time powerful enough to support the features described

previously, including pause, rewind, fast forward and others. In addition, this structure allows fast and efficient switching between stream sources, since buffered data can be simply discarded and decoders reset using a single event, after which data from the new stream will pass down the pipeline. Such a capability is needed, for example, when switching the channel being captured by the input section, or when switching between a live signal from the input section and a stored stream.

The source object 901 is a TmkSource 906 and the transform object 902 is a TmkXfrm 910. These are intermediate classes that define standard behaviors for the classes in the pipeline. Conceptually, they handshake buffers down the pipeline. The source object 901 takes data out of a physical data source, such as the Media Switch, and places it into a PES buffer. To obtain the buffer, the source object 901 asks the down stream object in his pipeline for a buffer (allocEmptyBuf). The source object 901 is blocked until there is sufficient memory. This means that the pipeline is self-regulating; it has automatic flow control. When the source object 901 has filled up the buffer, it hands it back to the transform 902 through the pushFullBuf function.

The sink 903 is flow controlled as well. It calls nextFullBuf which tells the transform 902 that it is ready for the next filled buffer. This operation can block the sink 903 until a buffer is ready. When the sink 903 is finished with a buffer (*i.e.*, it has consumed the data in the buffer) it calls releaseEmptyBuf. ReleaseEmptyBuf gives the buffer back to the transform 902. The transform 902 can then hand that buffer, for example, back to the source object 901 to fill up again. In addition to the automatic flow-control benefit of this method, it also provides for limiting the amount of memory dedicated to buffers by allowing enforcement of a fixed allocation of buffers by a transform. This is an important feature in achieving a cost-effective limited DRAM environment.

The MediaSwitch class 909 calls the allocEmptyBuf method of the TmkClipCache 912 object and receives a PES buffer from it . It then goes out to the circular buffers in the Media Switch hardware and generates PES buffers. The MediaSwitch class 909 fills the buffer up and pushes it back to the TmkClipCache 912 object.

The TmkClipCache 912 maintains a cache file 918 on a storage medium. It also maintains two pointers into this cache: a push pointer 919 that shows where the next buffer coming from the source 901 is inserted; and a current pointer 920 which points to the current buffer used.

The buffer scheme can be implemented using a memory pool where each buffer is allocated on demand by a memory manager. The buffers are linked together by next

buff pointers in a linked list 918. As buffers are released, they a freed back into the memory pool and are available to be allocated to other classes or tasks within the system. The push pointer 919 points to the last buffer in the linked list while the current pointer 920 points to the current buffer used.

The buffer that is pointed to by the current pointer is handed to the Vela decoder class 916. The Vela decoder class 916 talks to the decoder 921 in the hardware. The decoder 921 produces a decoded TV signal that is subsequently encoded into an analog TV signal in NTSC, PAL or other analog format. When the Vela decoder class 916 is finished with the buffer it calls releaseEmptyBuf.

The structure of the classes makes the system easy to test and debug. Each level can be tested separately to make sure it performs in the appropriate manner, and the classes may be gradually aggregated to achieve the desired functionality while retaining the ability to effectively test each object.

The control object 917 accepts commands from the user and sends events into the pipeline to control what the pipeline is doing. For example, if the user has a remote control and is watching TV, the user presses pause and the control object 917 sends an event to the sink 903, that tells it pause. The sink 903 stops asking for new buffers. The current pointer 920 stays where it is at. The sink 903 starts taking buffers out again when it receives another event that tells it to play. The system is in perfect synchronization; it starts from the frame that it stopped at.

The remote control may also have a fast forward key. When the fast forward key is pressed, the control object 917 sends an event to the transform 902, that tells it to move forward two seconds. The transform 902 finds that the two second time span requires it to move forward three buffers. It then issues a reset event to the downstream pipeline, so that any queued data or state that may be present in the hardware decoders is flushed. This is a critical step, since the structure of MPEG streams requires maintenance of state across multiple frames of data, and that state will be rendered invalid by repositioning the pointer. It then moves the current pointer 920 forward three buffers. The next time the sink 903 calls nextFullBuf it gets the new current buffer. The same method works for fast reverse in that the transform 902 moves the current pointer 920 backwards.

A system clock reference resides in the decoder. The system clock reference is sped up for fast play or slowed down for slow play. The sink simply asks for full buffers faster or slower, depending on the clock speed.

With respect to Fig. 10, two other objects derived from the TmkXfrm class are placed in the pipeline for disk access. One is called TmkClipReader 1003 and the other is called TmkClipWriter 1001. Buffers come into the TmkClipWriter 1001 and are pushed to a file on a storage medium 1004. TmkClipReader 1003 asks for buffers which are taken off of a file on a storage medium 1005. A TmkClipReader 1003 provides only the allocEmptyBuf and pushFullBuf methods, while a TmkClipWriter 1001 provides only the nextFullBuf and releaseEmptyBuf methods. A TmkClipReader 1003 therefore performs the same function as the input, or "push" side of a TmkClipCache 1002, while a TmkClipWriter 1001 therefore performs the same function as the output, or "pull" side of a TmkClipCache 1002.

Referring to Fig. 11, a preferred embodiment that accomplishes multiple functions is shown. A source 1101 has a TV signal input. The source sends data to a PushSwitch 1102 which is a transform derived from TmkXfrm. The PushSwitch 1102 has multiple outputs that can be switched by the control object 1114. This means that one part of the pipeline can be stopped and another can be started at the user's whim. The user can switch to different storage devices. The PushSwitch 1102 could output to a TmkClipWriter 1106, which goes onto a storage device 1107 or write to the cache transform 1103.

An important feature of this apparatus is the ease with which it can selectively capture portions of an incoming signal under the control of program logic. Based on information such as the current time, or perhaps a specific time span, or perhaps via a remote control button press by the viewer, a TmkClipWriter 1106 may be switched on to record a portion of the signal, and switched off at some later time. This switching is typically caused by sending a "switch" event to the PushSwitch 1102 object.

An additional method for triggering selective capture is through information modulated into the VBI or placed into an MPEG private data channel. Data decoded from the VBI or private data channel is passed to the program logic. The program logic examines this data to determine if the data indicates that capture of the TV signal into which it was modulated should begin. Similarly, this information may also indicate when recording should end, or another data item may be modulated into the signal indicating when the capture should end. The starting and ending indicators may be explicitly modulated into the signal or other information that is placed into the signal in a standard fashion may be used to encode this information.

With respect to Fig. 12, an example is shown which demonstrates how the program logic scans the words contained within the closed caption (CC) fields to determine starting and ending times, using particular words or phrases to trigger the capture. A stream of NTSC or PAL fields 1201 is presented. CC bytes are extracted from each odd field 1202, and entered in a circular buffer or linked list (using a memory allocation scheme as described above) 1203 for processing by the Word Parser 1204. The Word Parser 1204 collects characters until it encounters a word boundary, usually a space, period or other delineating character. Recall from above, that the MPEG audio and video segments are collected into a series of fixed-size PES buffers. A special segment is added to each PES buffer to hold the words extracted from the CC field 1205. Thus, the CC information is preserved in time synchronization with the audio and video, and can be correctly presented to the viewer when the stream is displayed. This also allows the stored stream to be processed for CC information at the leisure of the program logic, which spreads out load, reducing cost and improving efficiency. In such a case, the words stored in the special segment are simply passed to the state table logic 1206.
One skilled in the art will readily appreciate that although a circular buffer is specifically mentioned in areas above, a linked list using a memory pool allocation scheme, also described above, can be substituted in its place.

During stream capture, each word is looked up in a table 1206 which indicates the action to take on recognizing that word. This action may simply change the state of the recognizer state machine 1207, or may cause the state machine 1207 to issue an action request, such as "start capture", "stop capture", "phrase seen", or other similar requests. Indeed, a recognized word or phrase may cause the pipeline to be switched; for example, to overlay a different audio track if undesirable language is used in the program.

Note that the parsing state table 1206 and recognizer state machine 1207 may be modified or changed at any time. For example, a different table and state machine may be provided for each input channel. Alternatively, these elements may be switched depending on the time of day, or because of other events.

Referring to Fig. 11, a PullSwitch is added 1104 which outputs to the sink 1105. The sink 1105 calls nextFullBuf and releaseEmptyBuf to get or return buffers from the PullSwitch 1104. The PullSwitch 1104 can have any number of inputs. One input could be an ActionClip 1113. The remote control can switch between input sources. The control object 1114 sends an event to the PullSwitch 1104, telling it to switch. It will switch from the current input source to whatever input source the control object selects.

An ActionClip class provides for sequencing a number of different stored signals in a predictable and controllable manner, possibly with the added control of viewer selection via a remote control. Thus, it appears as a derivative of a TmkXfrm object that accepts a "switch" event for switching to the next stored signal.

This allows the program logic or user to create custom sequences of video output. Any number of video segments can be lined up and combined as if the program logic or user were using a broadcast studio video mixer. TmkClipReaders 1108, 1109, 1110 are allocated and each is hooked into the PullSwitch 1104. The PullSwitch 1104 switches between the TmkClipReaders 1108, 1109, 1110 to combine video and audio clips. Flow control is automatic because of the way the pipeline is constructed. The Push and Pull Switches are the same as video switches in a broadcast studio.

The derived class and resulting objects described here may be combined in an arbitrary way to create a number of different useful configurations for storing, retrieving, switching and viewing of TV streams. For example, if multiple input and output sections are available, one input is viewed while another is stored, and a picture-in-picture window generated by the second output is used to preview previously stored streams. Such configurations represent a unique and novel application of software transformations to achieve the functionality expected of expensive, sophisticated hardware solutions within a single cost-effective device.

With respect to Fig. 13, a high-level system view is shown which implements a VCR backup. The Output Module 1303 sends TV signals to the VCR 1307. This allows the user to record TV programs directly on to video tape. The invention allows the user to queue up programs from disk to be recorded on to video tape and to schedule the time that the programs are sent to the VCR 1307. Title pages (EPG data) can be sent to the VCR 1307 before a program is sent. Longer programs can be scaled to fit onto smaller video tapes by speeding up the play speed or dropping frames.

The VCR 1307 output can also be routed back into the Input Module 1301. In this configuration the VCR acts as a backup system for the Media Switch 1302. Any overflow storage or lower priority programming is sent to the VCR 1307 for later retrieval.

The Input Module 1301 can decode and pass to the remainder of the system information encoded on the Vertical Blanking Interval (VBI). The Output Module 1303 can encode into the output VBI data provided by the remainder of the system. The

program logic may arrange to encode identifying information of various kinds into the output signal, which will be recorded onto tape using the VCR 1307. Playing this tape back into the input allows the program logic to read back this identifying information, such that the TV signal recorded on the tape is properly handled. For example, a particular program may be recorded to tape along with information about when it was recorded, the source network, etc. When this program is played back into the Input Module, this information can be used to control storage of the signal, presentation to the viewer, etc.

One skilled in the art will readily appreciate that such a mechanism may be used to introduce various data items to the program logic which are not properly conceived of as television signals. For instance, software updates or other data may be passed to the system. The program logic receiving this data from the television stream may impose controls on how the data is handled, such as requiring certain authentication sequences and/or decrypting the embedded information according to some previously acquired key. Such a method works for normal broadcast signals as well, leading to an efficient means of providing non-TV control information and data to the program logic.

Additionally, one skilled in the art will readily appreciate that although a VCR is specifically mentioned above, any multimedia recording device (*e.g.*, a Digital Video Disk-Random Access Memory (DVD-RAM) recorder) is easily substituted in its place.

Turning now to Figure 14, a schematic block diagram of a top-level view of the invented system architecture is provided. In general, a system board 1400 embodying the invention includes an input section 1401 that accepts an input signal from one of a variety of sources. As described below, the input section 1401 is provided in different versions, each adapted to accept input from a different source. The output section 1402 includes a CPU 1403, which largely functions to initialize and control operation of the various hardware components of the invention. As mentioned above, the CPU is decoupled from the high data rates of the video signal, thus reducing processor requirements. An MPEG-2 transport stream decoder/graphics subsystem 1404 accepts a transport stream delivered from the input section 1401 over a transport stream interface 1406. The transport stream decoder/graphics subsystem 1404 communicates with the CPU 1403 by means of a host bus 1408. While the transport stream decoder/graphics subsystem serves a variety of functions, described in detail below, its primary function is decoding of the transport stream received from the input section, and outputting the decoded stream as a video signal to a television set (not shown).

The output section further includes a media manager 1405. While the media manager provides a number of functions, its major function is that of a bridging element between system components, due to the number and type of I/O functions it incorporates. For example, the media manager includes an IR receiver/transmitter interface to couple with the handheld remote control by which a user operates the invention. Furthermore, the media manager serves an important media processing function. As previously indicated, the transport signal is both routed to the MPEG-2 decoder and saved to the storage device by the media manager. The media manager 1405 communicates with the MPEG-2 transport stream decoder/graphics subsystem 1404 by means of a system bus 1407. A preferred embodiment of the invention uses a PCI bus as the system bus. Advantageously, the output section is partitioned as three discrete chips: the CPU, the MPEG-2 decoder/graphics subsystem and the media manager. The simplicity of this partitioning arrangement enables a substantially reduced per-unit cost by dramatically reducing the time and budget required for initial design and development. Additionally, those skilled in the art will appreciate that the output section may also be provided as a single chip or chipset.

Figure 15 shows the output section 1402 in greater detail. It will be appreciated that the output section encompasses the core components of the invention, the CPU 1403, the MPEG-2 decoder/graphics subsystem 1404, and the media manager 1405. The CPU 1403 functions primarily to run the system software, as well as middleware and application software. The system software includes the OS (Operating System) kernel and the device drivers. The system software operates to initialize and control the various hardware components of the system. A more detailed description of the function of the CPU has been provided above. Almost all data movement in the system is based on DMA transfers or dedicated high-speed transport interfaces that do not involve the CPU. While a variety of RISC processors would be suitable for use in the invention, the current embodiment employs a VR5432 CPU, manufactured by NEC Corporation of New York NY, that provides a 64-bit MIPS RISC architecture with a 32K instruction cache and 32K data cache, running at 202 MHz clock frequency. The CPU is connected with the MPEG-2 decoder/graphics subsystem 1404 by means of a system bus 1407.

An MPEG-2 decoder/graphics subsystem 1404, such as, for example, the BCM7020, supplied by Broadcom Corporation of Irvine CA can be considered the central component of the output section 1402. In fact, the MPEG-2 decoder/graphics subsystem 1404 incorporates a number of important components, including, but not limited to:

- a host bridge;

- a memory controller;
- an MPEG-2 transport de-multiplexer;
- at least one MPEG-2 decoder;
- an audio/video decoder;
- a PCI bridge;
- a bus controller;
- a modem interface; and
- a SMARTCARD interface.

As described above, the transport stream generated by the input section 1401 is fed into one of the transport interfaces 1406, whereupon it is demultiplexed into separate audio and video packet elementary streams (PES). These streams are then stored on the hard drive 1505 and played back through the outputs 1504. The transport stream demultiplexer included in the MPEG-2 decoder/graphics subsystem 1404 is responsible for the demultiplexing operation. Prior to being played back, the audio and video packet streams are retrieved from the hard drive and reassembled into a transport stream. The transport stream is then decoded to a video signal. The MPEG-2 transport stream decoder included in the component 1404 is responsible for decoding the MPEG-2 transport stream. The component 1404 also includes a graphics engine for generating high-quality on-screen displays, such as interactive program guides. The output side of the component 1404 provides several outputs; including S-video, audio, SPDIR (Stereo Paired Digital Interface), CVBS (Composite Video Baseband Signal). Additionally, a SMARTCARD interface 1503, and a modem port 1506 is provided to which a modem 1519 is interfaced. The SMARTCARD interface supports up to two SMARTCARD readers. More will be said about the SMARTCARD functionality below.

The output section 1402 further includes a memory element 1501, under the control of the OS kernel. The system software provides a single device driver interface that enables all other device drivers to allocate contiguous memory buffers typically used for DMA (Direct Memory Access). The memory element is preferably SDRAM (Synchronous Dynamic Random Access Memory), preferably at least 32 MB. However, other memory configurations are entirely within the spirit and scope of the invention. Furthermore, as will be described below, the invention may include other memory elements that are not under the control of the OS kernel.

A flash PROM (Programmable Read-only Memory) 1502 contains the boot code that initializes the system board state prior to booting the OS kernel, either from a hard drive or over a TCP/IP network connection. In addition to performing basic system startup

tasks such as memory test and POST (Power-On Self Test), the PROM 1502 also serves as a key component in the physical architecture of the system by ensuring that neither the PROM itself nor the OS kernel it is booting have been tampered with. This is accomplished by computing digital signatures over the PROM code as well as the OS kernel image.

As previously indicated, the media manager 1405, connected to the MPEG-2 decoder/graphics subsystem 1404 by means of the PCI bus 1407, performs a bridging or mediating function between many of the hardware components of the system, notably the CPU 1403, the hard disk or storage device 1505, and memory 1501. The media manager 1405 provides this function by virtue of the assortment of interfaces and I/O devices integrated within the media manager. In the preferred embodiment of the invention, the media manager is implemented in an ASIC (Application Specific Integrated Circuit). However, the media manager could also be implemented in a programmable logic device, or it could also be composed of discrete devices. The media manager 1405 integrates at least the following:

- an IDE host controller, with data encryption;
- a DMA controller;
- IR receiver/transmitter interface;
- multiple UART's (Universal Asynchronous Receiver/Transmitter);
- multiple I^2C (Inter-IC) buses;
- multiple GPIO's (General Purpose I/O's);
- a PCI bus arbiter;
- an MPEG-2 media stream processor;
- a PCM (Pulse Code Modulation) audio mixer;
- a high-speed transport output interface;
- a fan speed control; and
- front panel keyboard matrix scanner.

As shown in Figure 15, the media manager includes a thermocouple 1507 for monitoring system temperature. The thermocouple is interfaced with the media manager through one of the I^2C buses 1508. In turn, fan speed is controlled by the system software, based on input from the thermocouple, through the fan control 1510 controlling the fan 1509, to maintain the system at an optimal operating temperature.

As previously described, the media manager also mediates the transfer of media streams between the CPU 1403, memory 1501, and the hard drive 1505. This is accomplished through the action of the media stream processor and the high-speed transport output interface mentioned above.

A secure micro controller, such as, for example, an AT90S3232C supplied by ATMEL Corporation of San Jose CA, 1511 is interfaced with the media manager ASIC 1405 through one of the UART's 1512. Preferably, the micro controller 1511 is one specifically designed for cryptographic applications such as encryption and authentication. In addition to providing a master key for disk encryption as described below, the micro controller also contains a private key unique to each unit that is created randomly during manufacturing. Once written into the component, the key cannot be read out and can only be used to respond to authentication challenges.

As shown, up to two hard drives 1505 are provided for storage of recorded video programming. As described above, the IDE host controller is integrated on the media manager ASIC 1405 and provides a disk encryption feature that can be applied to either disk drive on a per-transfer basis. The micro controller, as described above, generates, encrypts and decrypts a master key for disk encryption purposes.

An RS232 port 1514 interfaces with another of the UART's 1513. A front panel navigation cluster 1516 is interfaced with the media manager ASIC through one of the GPIO's 1515. An IR receiver and transmitter 1518 are interfaced with the media manager ASIC through an IR receiver/transmitter interface 1517. The IR receiver assembly is mounted in the front panel navigation cluster, described in greater detail below, behind a transparent window. It receives a modulated signal from a handheld remote control and outputs the signal as is to the media manager ASIC, which either dispatches it to the CPU for further processing or provides a pass-through path to the IR transmitter 1518.

A real-time clock (not shown) is interfaced with the media manager through one of the I^2C ports. Because the invention is intended for use as a personal video recorder, in which the user is able to program the system in advance to record selections at specified times, a real-time clock is a fundamental requirement.

As previously described, the input signal is accepted by an input section 1401 passed to the output section 1402 as an MPEG-2 transport stream. The input section is provided in one of several configurations, according to the type of source originating the signal. By providing an input section 1401 individualized to source type, while keeping the output section the same across all versions, it is possible to produce units in various configurations with only minor modifications to the system board. In this way, the scale of the manufacturing challenge posed by producing units to serve different markets is considerably reduced. Referring now to Figure 16, an input section 1401a adapted to

accept analog signals is shown. In the preferred embodiment, the analog input section accepts analog signals in a variety of formats: composite video, NTSC, PAL, SECAM or S-video.

In the case of NTSC signals, a tuner/RF demodulator 1601, such as the TMDH-2 supplied by ALPS Electric, of San Jose CA, sets the signal to the desired channel. Preferably, the tuner assembly incorporates the tuner, an RF demodulator and an RF bypass into the same component. The tuner assembly is controlled over the I^2C bus port exposed by the media manager ASIC 1405.

A multi-standard sound processor 1603, such as a MSP4448G, supplied by Micronas Semiconductor of Freiburg, Germany accepts analog audio input from the composite audio connectors or the tuner/RF demodulator 1601. Additionally, it accepts digital audio input over an I^2S bus from the media manager ASIC 1405.. The resulting audio signal is output to an MPEG encoder 1604 over the I^2S bus.

The decoder 1602, an NTSC/PAL/SECAM video decoder, such as, for example a SAA7114H video decoder, supplied by Philips Semiconductor, of Eindhoven, the Netherlands, accepts input from either the tuner/RF demodulator 1601, the composite video inputs or the S-video input and converts it into the CCIR 656 (Comité Consultatif International des Radiocommunications, recommendation 656) digital format for input to an MPEG-2 encoder 1604, such as, for example a BCM7040, supplied by BROADCOM.

The MPEG-2 encoder 1604 accepts input from the NTSC/PAL/SECAM video decoder 1602 and the audio input previously mentioned and produces an MPEG-2 transport stream as the output. In the preferred embodiment of the invention, the encoder 1604 is programmed to multiplex the audio and video inputs into a constant bitrate (CBR) MPEG-2 transport stream. However, in order to conserve disk space, it is also possible to program the encoder 1604 to produce a variable bit rate (VBR) stream. Subsequently, the transport stream is delivered to the decoder 1404 over the transport interface 1406 for demultiplexing and further processing. The input section 1401a further includes a memory element 1605 that is not under the control of the OS kernel. Figure 19 provides a block schematic diagram of a system board 1900 incorporating the input section 1401a and the output section 1402. As shown, the MPEG-2 encoder is connected to the MPEG-2 decoder/graphics subassembly 1404 as a client on the PCI bus 1407.

A variation (not shown) of the analog front end includes a secondary input via an additional set of composite audio/video and/or S-video connectors for content originating from camcorders or VCR's. Additional hardware and software support is necessary in order for the variation to be fully enabled.

Turning now to Figure 17, an input section 1401b is shown adapted to accept a digital satellite signal. The digital satellite input section 1401b accepts input from dual satellite receivers 1701. Demodulators 1702 demodulate the incoming QPSK (quadrature phase shift keying) to yield a transport stream. Because the satellite transport stream is not fully MPEG-2 compliant, the MPEG-2 decoder/graphics subassembly 1404 must have the capability of decoding either type of stream. Thus, the transport stream is passed to the output section 1402 via the transport interface 1406 without any further modification or processing. Figure 20 provides a block diagram of a system board 2000 incorporating the input section 1401b.

Referring to Figure 18, an input section 1401c designed to accept either digital or analog cable input is shown. The input section accepts input from one or more RF coaxial connectors 1801, 1802 in both digital and analog format. The analog portion functions similarly to that of the analog input section 1401a. The video signal is decoded by dual NTSC decoders 1602. The audio is processed by dual multi-standard sound processors 1603 and the resulting output is fed to dual MPEG-2 encoders. It should be noted that, in the current version of the input section, each component is provided in duplicate. The digital cable signal is routed to dual demodulators 1803. Depending on the cable signal modulation, the demodulators may be either or both of QAM (quadrature amplitude modulation) and QPSK, either with or without DOCSIS (Data Over Cable Service Interface Specification) and/or DAVIC (Digital Audio Visual Council) support. As shown, the digital signal demodulators have associated with them a memory element 1804 that is controlled independently of the OS kernel. Figure 21 provides a block diagram of a system board 2100 incorporating the digital cable input section 1401c. As in the previous versions, transport streams are passed to the output section 1402 via the transport interface 1406. The digital cable input section 1401c is connected to the MPEG-2 decoder/graphics subsection 1404 as a client on the PCI bus.

As previously described, the invention is intended to be used as a PVR (Personal Video Recorder), in which a user may view a selected video stream in real-time, or they may view a recorded video stream, examining the video stream by taking advantage of such features as rewind, pause, play, stop, slow play, fast forward, and the like. Furthermore, controls are provided for selecting programming to be recorded and for

specifying additional recording parameters. To that end, the invention includes user control interfaces. Primarily, user interaction with the invention is by way of a battery-powered, handheld IR remote control. Activating the various controls by the user causes a modulated IR beam to be emitted and received by the PVR. The IR receiver/transmitter system and interface have been previously described in detail. However, an alternate embodiment of the invention provides an RF-enabled remote control, receiver/transmitter and interface, either instead of or in addition to the IR driven remote control.

In addition to the remote control, the user may interact with the invention by means of a navigation cluster, comprising buttons or keys, on a front panel of the unit. Advantageously, the navigation cluster substantially duplicates the functions of the remote control. Thus, the navigation cluster permits control of the invention, even if the remote control is lost, or stolen, or needs the batteries replaced. As described above, an interface for the navigation cluster is provided on the media manager ASIC.

As previously indicated, the system board supports SMARTCARD functionality. A SMARTCARD reader is accessible through a slot provided on the front panel of the invention. The SMARTCARD slot is intended for use in commerce applications where user authentication is required for billing purposes, such as pay-per-view programming, music sales, merchandise sales and the like.

The invention is produced using conventional manufacturing techniques well known to those skilled in the art of microelectronics design and manufacturing.

As described above, the media manager ASIC includes a media stream processor. Conventionally, media stream processors have been only able to process a single channel, providing a serious bottleneck to the system's throughput. Related, commonly owned applications have described multi-channel media processors that eliminate this bottleneck. Additionally, conventional media stream processors have had to be in the data path of the stream they are processing. Such a requirement necessitates that the processor be integrated on the system board in a manner that would make it very difficult to upgrade the media stream processor without replacing the system board. It would be a great advantage to provide a system independent device to upgrade a PVR's media stream processor capability from single-channel to multi-channel, which could be flexibly incorporated with existing hardware. To that end; the invention provides a system-independent, multi-channel media stream processor 1000. As Figure 22 shows, the multi-channel media stream processor includes:

- a system interface 2201;

- a media stream identifier 2202;
- a media stream processor core 2203;
- a multi-channel state engine 2204; and
- a media stream identification generator 2205.

The system interface 2201 serves as a completely passive, slave client on the system bus, not interfering in any way with data transfer, merely observing or "sniffing" the bus. While the remaining components of the invented media stream processor are system-independent, the system interface 2201 may be tailored to a specific system, or it may be adapted to connect to several different systems, either by means of hardwired elements, or through the use of programming switches. In the case of a unique or proprietary system, the system interface can be placed to observe on the memory bus instead, owing to the fact that hardware and protocols on memory buses are nearly universally uniform. The system interface provides a connection by which the media processor may observe the system bus.

System data is sent to the media stream identifier 2202, which distinguishes media streams from other data, in order to identify data that needs to be processed. The media stream identifier uses information such as source and destination addresses, which in most systems are hardwired signals, to identify media streams.

As media streams are identified, the media stream identification generator 2205 tags media stream data objects so that they may be associated with their respective media streams. Following tagging, the media stream data is routed to the media stream processor core 2203, where it is processed in parallel, rather than in a single channel. By processing the media streams in this manner, it is possible to achieve a four to eightfold increase in throughput.

In the case of multiple media streams, the multi-channel state engine 2204 saves the state of the media processor when a different media stream identification is presented, indicating that the media stream has switched. When the original media stream is again presented, the state is reloaded and processing of the original stream is resumed.

The resulting process is saved to a media data structure. Such data structures are commonly known. As each stream is processed, it is sent to system memory as needed.

While the multi-channel media stream processor has been described herein as an upgrade device, it also could be incorporated into a new system as the media

processor. It provides the advantage of being easily incorporated into a system architecture without requiring major retooling of the system board.

The multi-channel media stream processor may be implemented using discrete components or in a programmable logic device, using known methods of programming such devices.

Although the invention has been described herein with reference to certain preferred embodiments, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the spirit and scope of the present invention. Accordingly, the invention should only be limited by the Claims included below.

CLAIMS

1. A system for processing a media stream across several channels simultaneously, comprising:

means for observing a data stream on a data bus;

means for identifying media streams within said data stream;

means for associating media stream data objects with their respective media streams;

a multi channel media stream processor, wherein said media processor processes media stream data across a plurality of channels, in parallel; and

means for monitoring and saving state of said processor as said processor switches from an original media stream to a next media stream, wherein, if said processor switches back to said original stream, a state associated with said original stream is reloaded.

2. The system of Claim 1, wherein said means for observing said data stream comprises a system interface, said system interface comprising a passive, slave client on said bus, wherein said system interface observes said data stream without interfering with data flow.

3. The system of Claim 2, wherein said system interface is individualized to a particular system type, said individualization being accomplished by one of: programmable switches and hardwiring.

4. The system of Claim 2, wherein said data bus is one of: a system bus and a memory bus.

5. The system of Claim 1, wherein said means for identifying a media stream comprises a media stream identifier, wherein said media stream distinguishes media streams from the remainder of said data stream according to source and destination addresses.

6. The system of Claim 1, wherein said means for associating media data objects with their respective media streams comprises a media identification generator, said media identification generator assigning tags to media stream data objects, so that any data object is associated with its stream of origin.

7. The system of Claim 1, wherein said means for monitoring and saving state of said processor comprises a multi-channel state engine, said state engine monitoring media stream

identifiers, and saving said processor state, said saved state comprising a first state, when a media stream identifier associated with said next media stream is associated.

8. The system of Claim 7, wherein said state engine reloads the first state if a media stream identifier associated with said first state is presented.

9. The system of Claim 1, further comprising a media stream data structure, said processed media stream being saved to said data structure and routed to system memory as needed.

10. The system of Claim 1, wherein said system is implemented in a programmable logic device.

11. A method of processing a media stream across several channels simultaneously, comprising the steps of:

observing a data stream on a data bus;

identifying media streams within said data stream;

associating media stream data objects with their respective media streams;

processing media stream data across a plurality of channels, in parallel; and

monitoring and saving a media processor state as said processor switches from an original media stream to a next media stream; and

reloading state associated with said original stream if said processor switches back to said original stream.

12 The system of Claim 11, wherein said step of observing said data stream comprises the steps of:

providing a system interface, said system interface comprising a passive, slave client on said bus; and

said system interface observing said data stream without interfering with data flow.

13. The system of Claim 12, wherein said system interface is individualized to a particular system type, said individualization being accomplished by one of: programmable switches and hardwiring.

14. The system of Claim 12, wherein said data bus is one of: a system bus and a memory bus.

15. The system of Claim 11, wherein said step of identifying a media stream comprises the steps of:

distinguishing media streams from the remainder of said data stream according to source and destination addresses.

16. The method of Claim 11, wherein said step of:

associating media data objects with their respective media streams comprises:

assigning tags to media stream data objects, so that any data object is associated with its stream of origin.

17. The method of Claim 11, wherein said step of monitoring and saving said processor state comprises the steps of:

monitoring media stream identifiers; and

saving said processor state, said saved state comprising a first state, when a media stream identifier associated with said next media stream is associated.

18. The method of Claim 17, wherein said step of monitoring and saving said processor state further comprises:

reloading the first state if a media stream identifier associated with said first state is presented.

19. The method of Claim 11, further comprising the steps of:

saving said processed media stream to a media data structure; and

routing to system memory as needed.

20. The method of Claim 11, said method implemented by means of a programmable logic device.