# Integrating the MPEG-2 subsystem for digital television

by R. E. Anderson
E. M. Foster
D. E. Franklin
R. S. Svec

**An MPEG-2 subsystem consisting of the transport demultiplexor, audio decoder, and video decoder is described, along with the supporting processor and memory subsystem. This subsystem is directly applicable to set-top box designs used in a digital television broadcast environment. The advantages of an integrated MPEG-2 subsystem consisting of cores from the IBM Blue Logic library are discussed. The integrated architecture supports a shared memory address space implemented using the PowerPC® local bus (PLB) standard high-speed bus. The resulting memory-management improvements for on-screen display (OSD), decoder rate buffers, audio and video clip data, and transport system data are illustrated. The real-time memory bandwidth and latency requirements for processing an MPEG-2 stream also have an impact on the architecture of the subsystem. Additional enhancements are provided for channel changes, time-base changes, error handling, and enhanced processing in the transport by adding more specialized buses. Programming effort is reduced because the software has less responsibility managing data movement through the system and because the same programming method is used to control all of the MPEG-2 functions.**

## Introduction

The continuing improvement in silicon performance and compression algorithms has made it possible to capture, store, and deliver data in digital form that was previously analog at each of these stages. Musical compact disks are an example of the early success of this trend. With the standardization of audio and video data compression techniques, it is now commercially viable to deliver television digitally.

Digital television provides many of the same advantages that the compact disc provides for audio. Quality is improved because noise is reduced, providing cleaner video and audio. Also, with compression, it is possible to deliver multiple programs over the same bandwidth as one analog program, providing viewers with more choices. And since the stream is delivered digitally, additional services can be added for data broadcasting and interactivity.

Digital television can be distributed using adapted versions of current analog systems including satellite, terrestrial, and cable. A satellite system is illustrated in **Figure 1**. The audio and video components of a program are compressed at the source and time-multiplexed with



0018-8646/98/$5.00 © 1998 IBM

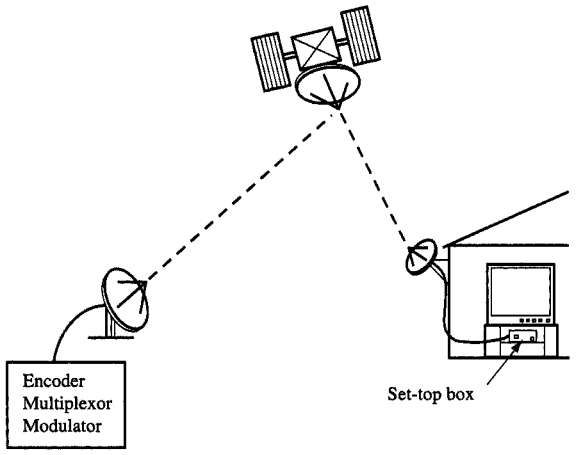


**Figure 1**

Digital television satellite distribution example.

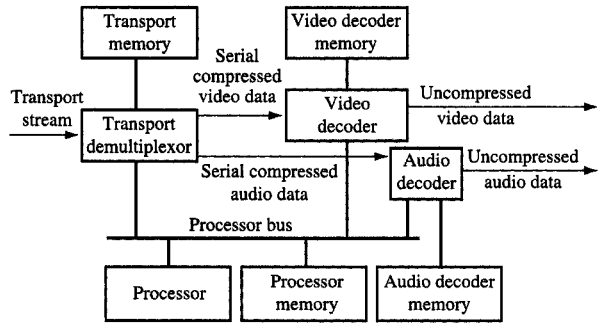


**Figure 2**

Typical MPEG-2 subsystem.

other programs as well as with system information needed to recreate the original program. The digital multiplex is modulated and transmitted to the satellite, which broadcasts the signal to an antenna on the subscriber's home that is connected to a receiver, or set-top box, which usually sits near the television. The set-top box demodulates the signal to recover the multiplexed digital streams, extracts the program of interest, and decodes the compressed audio and video for presentation.

There are several possible variations of this system. The signal can be broadcast using traditional terrestrial techniques, or through the coaxial cable currently installed in many homes. In addition, the receiver may be a stand-alone (set-top) box as described above, or it may in time be integrated directly into the television.

To facilitate the development of interoperable components from different manufacturers, the MPEG[1]-2 international standard[2] was developed; it specifies the syntax of encoded audio and video as well as the requirements for time-multiplexing several programs and other data into a single digital stream [4, 5]. IBM offers several discrete MPEG-2 silicon solutions that are in use today [6].

This paper focuses on the challenges and the advantages of integrating the MPEG-2 cores into a single chip.

## Basics of an MPEG-2 subsystem

In general, the MPEG-2 subsystem[3] of a digital television receiver consists of three main functions: a transport demultiplexor, an audio decoder, and a video decoder, as shown in **Figure 2**. The input to the subsystem is a transport stream that contains all of the components necessary to decode and play one or more programs. Specifically, for each program there is at least one compressed video stream, one compressed audio stream, and associated system data to allow navigation and synchronization of the output. There may also be additional system information, ranging from program guides to conditional access controls to data downloads for interactive applications.

**Figure 3** shows a typical example of the bandwidth allocation of a transport stream at a given point in time. This stream carries seven programs that contain both audio and video and four programs that are audio only. Programs 1 and 5 contain two audio streams, allowing the program to be shown in two languages simultaneously. One important observation is that since the video portion of a program occupies most of the transport stream bandwidth, it is relatively inexpensive to have multiple audio components for a single program or audio-only programs, as shown in the graph.

The empty packets provide a means of padding the stream out to a fixed data rate and also create dynamic headroom for when one or more of the programs requires additional bandwidth for a short period of time. The bandwidth allocation in the transport stream can change over time; another graph taken at a different time would be slightly different.

- *Transport demultiplexor*

The transport demultiplexor, commonly referred to as the transport, monitors the stream to establish packet

[1] MPEG**—Moving Picture Experts Group.
[2] Formally known as ISO/IEC 13818 Information Technology—Generic Coding of Moving Pictures and Associated Audio Information Standard [1–3].
[3] This paper focuses on processing "transport streams," as referenced in [1]. The system described also supports "program streams," which contain only one program, using the host processor.

boundaries so that the data fields can be processed. Then, based on directions from the host application, the transport filters the packets from the incoming transport stream into the video and audio streams for a given program, and also extracts the system data. The audio and video compressed data streams are sent to the audio and video decoders, respectively. The system information is sent to the processor memory for use by the host in constructing navigation tables, conditional access information, and other applications such as electronic program guides.

There are several data rates which define the performance of an MPEG-2 subsystem (see **Table 1**). The transport data rate is the most variable across different systems. No maximum transport data rate is specified to take advantage of improvements in communications mediums. The audio and video data for a single program are forwarded to the appropriate decoder. Limits are placed on the audio and video data rates so that the processing speed and buffer sizes for the audio and video decoders are bounded.

• *Audio decoder*

The audio decoder accepts the compressed audio data stream from the transport. It processes the stream to remove the necessary control and synchronization information from the packet header, and then decodes the data to reconstruct digital audio samples. On the basis of timing information in the packet header, the uncompressed samples are forwarded to a digital-to-analog converter (not shown) for playback through the television.

• *Video decoder*

Much like the audio decoder, the video decoder accepts compressed video data from the transport, processes the stream to remove control and synchronization information, and then decodes the data to reconstruct digital video frames (or fields). On the basis of timing information, the uncompressed frames are forwarded to a video digital encoder (not shown), which performs digital-to-analog conversion and encodes the signal into PAL or NTSC formats for playback on a television. Though it is not an MPEG function, video decoders typically also include the ability to mix graphic images, or bitmaps, with the video as a way to communicate with the viewer through menus and options. This is commonly referred to as on-screen display (OSD).

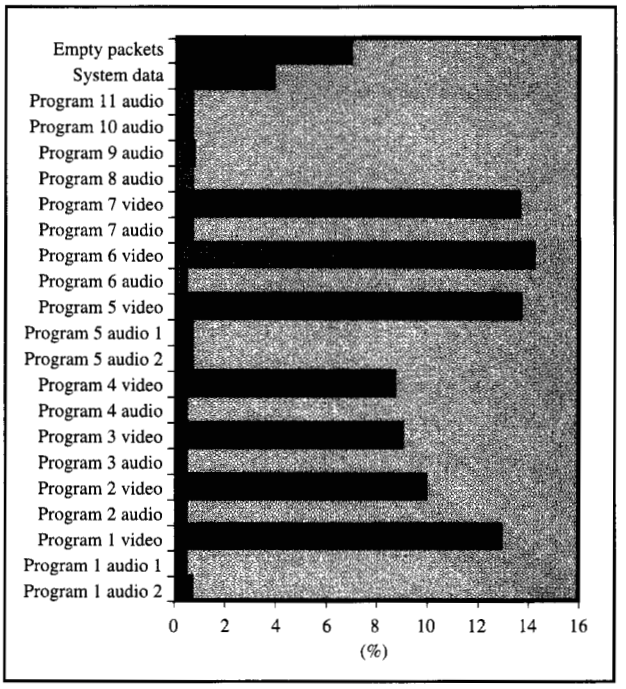


**Figure 3**

Multiple programs in a transport stream.

## How an MPEG-2 subsystem works

In applications such as the satellite system described earlier, transport streams are broadcast or "pushed" to the receiver device. Since there is no means to regulate the delivery rate, a stream of data must be processed in real time as it arrives. The MPEG-2 transport stream includes embedded clock references that are extracted by the transport to control the clock rate of the MPEG-2 subsystem. This ensures that the decoders consume the incoming data at a rate that prevents the buffers from overflowing or underflowing. In the case of overflow, data are lost, and the viewer will hear this as garbled audio or see it as a corrupted video frame. If the decoder consumes

**Table 1** MPEG-2 data rates.

| *Data stream* | *Typical* | *Maximum* | *Supported by IBM MPEG-2 cores* |
|---|---|---|---|
| Transport stream | 22–60 Mb/s | N/A | 100 Mb/s |
| Encoded video stream for standard television (MP@ML) | 3–6 Mb/s | 15 Mb/s | 15 Mb/s |
| Encoded audio stream (MPEG-2 or multichannel Dolby digital) | 192–384 Kb/s | 640 Kb/s | 640 Kb/s |

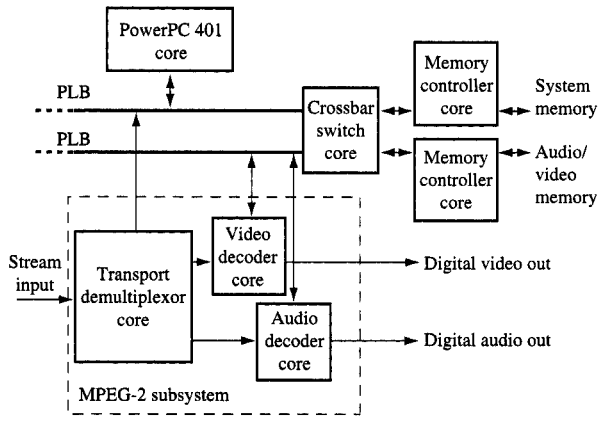


**Figure 4**

MPEG-2 subsystem and processor bus architecture.

data too quickly, it will be necessary to repeat the audio or video information until new data arrive. The definition of the buffer models and clock models in the MPEG-2 specification makes it possible for any compliant MPEG-2 stream to be decoded and displayed by any compliant MPEG-2 decoder. For the consumer, cost is reduced because many manufacturers can participate in the MPEG-2 market.

- *Initialization*

At start-up, the transport synchronizes with the incoming stream by finding the packet boundaries. It is configured by the host application to capture a series of tables, program-specific information, that provide the basic program-mapping information necessary to determine which programs are in the multiplex and what packets make up a given program. When the tables are available, the application processes the information and constructs menus or option panels that are sent to the viewer using the video decoder OSD function. The viewer selects a program of interest with the remote control. The application translates the selection into the appropriate configuration information for the transport. The transport then filters and forwards the required audio and video streams to the decoders.

Some systems also employ a conditional access subsystem that captures additional information in the stream and uses special techniques to determine whether the viewer is entitled to see a given program and whether there is a charge associated with it. The conditional access subsystem can be either a discrete module such as that described in the DVB common interface specification [7] or partially integrated in the MPEG subsystem. Since there are a large number of implementation variations, it is not described further in this paper, but additional details can be found in [8–10].

- *Clock references*

The MPEG-2 specification assumes that the amount of time delay from the point at which the transport stream is generated to the point at which it is received is constant for the entire stream. Clock references for a given program, reflecting the current value of the encoder clock, are inserted into the transport stream as it is created. Since the transport stream is delivered with constant delay, the program clock references (PCRs) can be extracted as the stream arrives at the receiver to construct a delayed version of the source clock. This information can be used to control the frequency of the receiver's local clock, the system time clock (STC), so that it matches the source. As a result, a program can be decoded and played at the same rate it was encoded, preventing buffer overflow and underflow problems. As part of program selection, the packets containing clock references from the encoder for a given program are identified and processed.

- *Audio/video playback and synchronization*

Once configured, the audio and video decoders search the data sent by the transport to establish audio frame and video sequence boundaries respectively, and to extract the control information describing what parameters to use for decoding the data. The decoders also extract the timing information [presentation time stamps (PTSs)] that indicates when a reconstructed frame should be output. Since all of the time stamps are referenced from the encoder clock, it is necessary only for the receiver's local clock to be an approximate copy of the encoder clock as described earlier. This ensures that the audio and video portions of the program are synchronized with each other.

When all of the necessary information is available, the decoders decompress the respective streams and send the output to the television.

## Integrating the MPEG-2 subsystem

The integrated MPEG-2 subsystem of this paper consists of three cores, as shown in **Figure 4**: transport, audio decoder, and video decoder. While not part of the MPEG-2 subsystem, the processor in the integrated design plays an important role in controlling the operation of the system, including the MPEG-2 subsystem. Each of the cores, including the IBM PowerPC 401* processor which is used to support the host application, is available in IBM's Blue Logic core library.[4]

[4] The IBM Blue Logic web site is *http://www.chips.ibm.com/technology/bluelogic*.

All of the cores in the MPEG-2 subsystem were designed to work together as a unit but are also usable as independent functions. This partitioning allows cores to be used for applications other than digital television. It also allows each core to be adapted to new requirements without affecting the remaining cores.

- *Making cores work together*

In a typical MPEG-2 subsystem (Figure 2) several buses are required to interconnect memory and the functional units. Integrating the MPEG-2 functions onto a single chip required minimizing the number of external interfaces as well as redefining the communication structure between units. The integrated solution reduces the number of memory subsystems from four to two. The bus architecture is also influenced by the data bandwidth required for each function as well as by any data used by more than one core.

The PowerPC 401 core processor supports several buses for connecting other cores to the processor [11]. The PowerPC* local bus (PLB) and the device control register (DCR) bus are provided for the MPEG-2 cores so that they can be used in multiple applications. Two additional buses are defined for the MPEG-2 cores. The first is a simple unidirectional bus to deliver data from the transport to the decoders; the second, illustrated later, provides the 401 the ability to assist the transport in processing the transport stream.

The crossbar switch core provides nonblocking access to system memory attached to each memory controller core. Normally the transport and the processor access the top system memory shown in Figure 4 using the top PLB bus. Both the audio and video decoders typically access the lower audio/video memory shown in the figure using the lower PLB bus. The crossbar switch allows devices on either PLB bus to access either memory, allowing a flat memory space to be created for the whole system. This flexibility allows memory to be allocated as needed for a particular set-top box implementation, or it can be changed dynamically. One example of the latter is that the audio/video memory can be used temporarily to download a system software update before it is moved into a flash memory.

- *High-speed memory access*

One concern with integration of the MPEG-2 cores is to ensure adequate bandwidth for each to meet its real-time requirements. For example, if the bandwidth available for the video presentation is insufficient, the viewer will see a corrupted picture. While the audio decoder has lower bandwidth requirements, its latency requirement is still critical to play the audio program without noticeable disruptions. The approximate bandwidth requirement for each function is shown in **Table 2**.

**Table 2** Estimated memory bandwidth requirements for MPEG-2 subsystem.

| *Function* | *Estimated bandwidth* (MB/s) |
|---|---|
| Video decode and display | 70–120 |
| Video on-screen display | up to 27 |
| Audio decode and presentation | 1–5 |
| Transport system data | up to 3 |

While some of the memory systems under consideration could provide a maximum of 180 MB/s per memory port, in practical use their performance is limited by access overhead and refresh. To maintain the low latency requirement, the usable bandwidth is limited further. On the basis of the bandwidth estimates given above, two memory subsystems were required. The bandwidth for the video decoder and the memory requirements of the 401 processor dictated that they be generally assigned to different memory subsystems. Since the transport delivered system data for the processor, it was assigned to the same memory subsystem. The audio decoder was assigned to the same memory subsystem as the video decoder, since it has similar latency requirements and that memory subsystem could be tuned for low latency.

This implementation is shown in Figure 4. There are two memory subsystems connected to the cores through a zero-latency crossbar switch. While a core typically uses its assigned memory subsystem, it can easily access the other one. The memory subsystems operate concurrently, providing the sum of their bandwidths.

This design accomplishes the concurrent and low-latency demands of all of the units and allows maximum flexibility of usage by the integrator. With this flexibility, it is possible to improperly allocate usage of memory, causing bandwidth overload; thus, there must be proper planning during system software design so that minimal memory sizes can be used to keep the cost down.

- *Processor access to registers*

The transport and decoders have a number of registers used by the host application to configure and control their operation. In general, these registers are accessed infrequently, and the data throughput and timing are not critical. The PowerPC architecture provides a specialized bus standard for device control, known as the device control register (DCR) bus, which is ideally suited to this purpose. The bus forms a daisy chain between cores, as shown in **Figure 5**; it is relatively simple to implement, requiring very little overhead within each core. The 32-bit-wide data bus matches the 401's register size. A simple acknowledge protocol allows timing flexibility within each core.

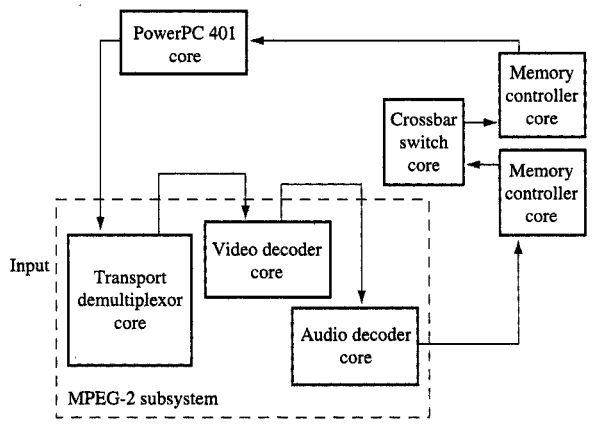


Figure 5

DCR bus interconnect.

The MPEG cores are also designed such that the DCR addresses can be configured by the chip integrator. For example, the video decoder core occupies a block of 64 addresses, but the base address of this block can be specified to start anywhere within the DCR address space as long as it is at a 64-word boundary. This allows flexibility in designing chips using multiple or different cores, since the address map can be adjusted to suit the integrator's requirements.

Not only has the DCR bus simplified hardware development, but the programming model for the cores is now consistent. This reduces the learning curve for developing device drivers.

- *Dedicated buses*

Integrating cores onto one chip removes pin-count constraints between cores and thus allows specialized buses to be defined between cores. The transport delivers a stream of data to each of the two decoders, as shown in Figure 4. Since the data at this point are byte-oriented, a byte-wide unidirectional bus was defined between the transport and each decoder. It uses a simple request/acknowledge protocol in which the decoder requests data and the data are transferred each time the transport acknowledges a valid byte on the bus. One bus is used for each decoder, making the cores easily adaptable to different configurations.

## Advantages of the integrated MPEG-2 subsystem

The different functions of the MPEG-2 subsystem were designed independently in many previous set-top boxes. Since the functions were often in different packages, the designs were also limited by the number of I/O pins available to interconnect them. In the previous sections, a solution to this problem was described in the selection of the bus architectures used to interconnect the MPEG-2 cores on a single chip.

A shared flat memory architecture was defined to conform to the new bus architecture. This allows any core to address any memory location in the system. Each MPEG-2 core defines several memory regions, which can be located anywhere in the full address space. This allows the number of memory subsystems to be reduced, providing a number of system-level advantages.

Performance is increased by reducing data movement in the system. System cost is reduced because memory regions can be more efficiently packed, reducing the number of memory components required for the total system. Device driver development is simplified because the overhead of data management is reduced.

The following subsections describe the specific advantages gained when two or more cores use the same data.

- *Decoder and transport memory*

The MPEG decoders require a significant portion of 2 MB of memory to properly decode and play back the reconstructed audio and video frames. The amount of memory depends on the audio and video format being decoded (e.g., NTSC vs. PAL) and the configuration of the decoders. In addition, the decoders also require a significant amount of memory for rate buffers to store the compressed data, which are used to account for the differences in the amount of data used to construct a new frame and out-of-order reference frames. A typical video rate buffer is 300–350 KB in size, but may be larger, especially at higher compressed data rates.

In designs without a shared-memory architecture, these storage requirements drove a minimum of 2 MB of memory to be allocated solely to the decoders. Unfortunately, the amount of memory available from this 2 MB for rate buffers is not always sufficient to sustain higher compressed data bit rates needed to support the PAL format. To alleviate this situation, some prior implementations included a second large buffer, managed by the transport, for storing data before delivery to the audio and video decoders. This caused the data to be buffered twice, which added complexity to the system and increased data movement.

As mentioned previously, the video decoder also supports an on-screen display (OSD), which allows detailed graphic images to be overlaid on the video. The OSD function supports multiple regions on the screen and varying color depths from four colors for a color-table region to a full 16 million colors for a direct-color region.

The memory for OSD further increases the memory requirements for the video decoder. This requirement can result in the addition of another 2 MB of memory for the decoders; however, only a quarter to a third of the additional 2 MB is actually needed. Memory granularity results in wasted memory space and increased system cost.

A shared-memory architecture allows the rate buffers, OSD memory, and other decoder memory regions to be allocated more efficiently, since those regions can be allocated elsewhere in memory. Overall memory bandwidth requirements are reduced, since the second memory buffer managed by the transport is eliminated and the compressed data are stored only once.

- *Decoder and 401 shared memory*

The MPEG audio and video decoders and the host processor can also access each other's memory space with the shared-memory architecture. This architecture simplifies support for OSD and for playing audio and video clips from memory.

One common use for the OSD is to display a set of menu choices. The viewer uses the remote control to navigate through the menu. Since memory for the decoder and the processor is shared, the host application can construct the desired images and then instruct the decoder to display them directly without moving them into a different memory region, as would be necessary in a nonshared-memory design. A full-size OSD image with 256 colors per pixel is approximately 414 KB in size. When this is added to the frame buffer, rate buffer, and various other space required for a video decoder, the total memory requirement easily exceeds 2 MB. Without shared memory, this results in the need for another (or larger) memory component dedicated to the decoder. The same memory-granularity problem described above for the transport also exists for the decoder memory. By sharing memory, the OSD image can be placed in the system memory pool and, depending on the available space, may not require any additional memory components.

If animation of the OSD is required, the saving is even more significant. Animation is accomplished by switching between alternate OSD images and updating the image which is not being displayed. Depending on the speed of the animation and image size, resolution, and other properties, the space and bandwidth required can be significant. Animating the full-size OSD image previously mentioned at 10 frames per second, the decoder would require 828 KB of memory to hold the two OSD images, along with the 414 KB required in the system memory to hold the image being updated by the application. This can be reduced to just the 828 KB in a shared-memory system, since the application can directly modify the image which is not being displayed. Slightly more than 8 MB/s of memory bandwidth can be saved in a shared-memory system, since an extra read and write operation is saved by not transferring the updated image from system to decoder memory.

Shared memory also allows the decoder to play streams and clips that have been loaded by the application or transport without having to move the data a second time into the compressed-data-rate buffer. Specifically, the audio and video decoders can be instructed to pull compressed data directly from a buffer space loaded by the application, and send the data directly to the decode engine. This buffer space is substituted for the decoder's normal rate buffer, and indeed may be located in the same memory area previously occupied by the rate buffer. Using this method, fixed-length sound or video clips can be played seamlessly from memory, or a running stream of data can be played from a source other than the transport, such as fixed storage. Again, this reduces the memory space and bandwidth requirements compared to the double-buffering scheme inherent in a nonshared architecture.

Savings in memory space and bandwidth are similarly achieved in other areas, such as handling data to be inserted into the vertical blanking interval (VBI), user data in the video stream, etc. The effect is not as significant, however, owing to the smaller amount of data being handled.

- *Shared transport and 401 memory*

The transport stream contains system data used by the application to control the MPEG subsystem. While the decoders accept and process the audio and video data streams, processing of data delivered to the application is delayed. In a typical MPEG-2 subsystem (Figure 2), the system data are first delivered to the transport's memory and then later copied by the processor or accessed through the transport. In both cases, access by the processor is less efficient because the data must be moved first or the processor accesses must wait for the memory arbitration scheme implemented by the transport.

In the integrated design, the transport can deliver the system data directly to the processor's memory space and provide interrupts to indicate when data are available. Eliminating the additional data movement decreases the memory bandwidth requirement and simplifies memory management for the device drivers.

- *Embedded handshakes*

Integrating the functional blocks of the MPEG-2 subsystem allows additional controls to be used to coordinate interaction between blocks. These embedded controls can be used to simplify driver development and to reduce latency and uncertainty. An override capability is

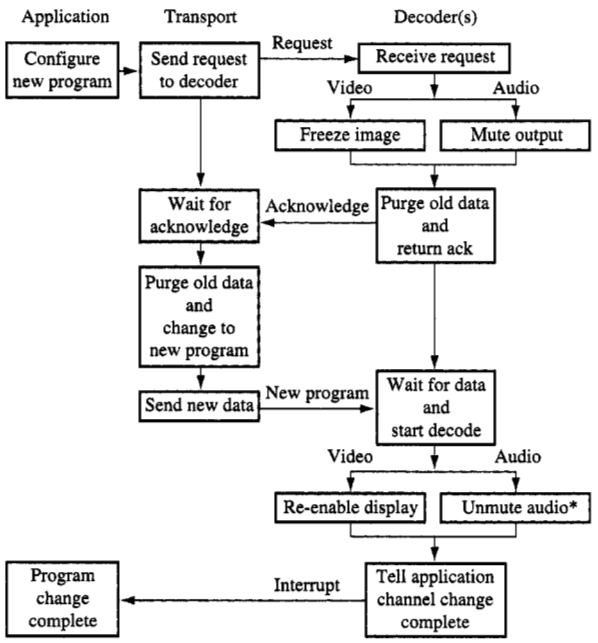


**Figure 6**

Channel change flow.

also included for existing applications that require a more typical interface.

- *Changing channels*

In traditional television receivers, the time required to change channels is minimal, because the audio and video data can be presented immediately after the tuner changes frequencies. The time to change programs in a digital system is inherently longer, because the new stream must be acquired, stored, decoded, and then presented. The time required to change channels ranges from a quarter second to three seconds depending on the implementation and the stream. As the channel-change time increases, it becomes noticeable to the viewer.

As a further complication, stopping the old stream abruptly can leave noticeable artifacts. Instead, the old stream must run until it reaches a valid stopping point, and then the new stream can begin. This adds to the amount of time required for the channel change.

In the integrated design, channel changes are coordinated directly between the transport and the decoders, minimizing the number of processor interrupts and also the viewer-perceived delay. Using viewer input, the application configures the transport for the new program selection. The transport signals the decoders that the stream will be changing and waits for an acknowledgment from the decoders indicating that they have successfully ended the old stream by muting the audio and freezing the video on a valid frame. The transport then flushes all old data from its buffers, and acquires and forwards the new data stream. The last step is to signal the host application that the process is completed. This process is illustrated in **Figure 6**.

- *Time changes*

The MPEG-2 system-layer syntax permits discontinuities in the clock references that are sent from the encoder during a given program as long as they are appropriately flagged in the stream. These changes in the time base can be either a by-product of the encoding and multiplexing process, or intentionally constructed as a means of inserting local content. For example, the local station replaces the audio, video, and time-base information with that for a commercial, and then switches back to the original program.

Receivers must handle time changes seamlessly, assuming that the remainder of the stream information does not change. The time-base change will be reflected in shifts in both the system time clock and the audio and video presentation time stamps. Since decoders buffer a variable amount of compressed data prior to decoding and presentation, there will be a period of time in which the STC and PTS are no longer correctly associated. Consequently, synchronization must be disabled while the time base changes.

Typically, the transport function detects the change in the time base and interrupts the host application, which then disables synchronization and updates the STC. However, there is uncertainty as to when to re-enable synchronization, since it must be after all of the compressed data in the decoder's buffer that are associated with the old time base have been consumed.

In the integrated solution, a flag is passed from the transport to the decoder marking the first byte of data that is associated with the time change. The decoder notes the position in the data stream and disables synchronization. When the flagged data byte is later ready to be decoded, the decoder notifies the application, which updates the time clock and re-enables synchronization. The result is that synchronization is only disabled for a small, definable period of time, eliminating the potential for video hangs because of using the wrong time bases (see **Figure 7**).

- *Hiding errors and data loss*

The MPEG-2 transport specification is built around small (188-byte) packets to minimize the potential amount of corrupted data when an error occurs. It also includes flags and counters to aid in detecting stream errors that might occur. If bad or missing data are allowed to propagate

through the system, it can result in noticeable artifacts in the television output (picture corruption, audio noise) or in corrupted data being sent to the system. Since decoding is performed in real time and the stream data cannot be rebroadcast, errors must be concealed directly by the MPEG functions without intervention from the application.

In the integrated design, the transport detects several stream errors, including a direct signal from the demodulation device, error indicators in the stream, continuity errors indicating missing packets, and table section errors. In the case of audio and video data, the transport removes the erroneous information and flags the error to the decoders to indicate the point in the stream where the error occurred. The decoders can then use this information to conceal the error without artifacts by repeating the last valid audio or video frame. Table data are also monitored for the correct section length and CRC check. Failing sections are not delivered to memory, and the application can be notified.

In addition, the transport can detect large gaps in the incoming data resulting from data loss and flag this error condition to the decoders so that they can reach a valid stopping point in the stream.

- *Extending the transport function*

The transport core is fully implemented in hardware to meet the real-time requirements for processing transport streams in a cost-effective implementation. Since adding additional connections between cores can be done with minimal impact in an integrated environment, the transport includes a special port that allows the PowerPC processor to control transport packet processing within the transport core. This allows the transport function to be extended to meet future requirements. These requirements may include unique table section and stream filtering and additional parsing requirements for fields currently not extracted by the hardware.

The 401 core has two on-chip memory (OCM) buses that are used to connect to the instruction and data cache units, as shown in **Figure 8**. The cache units in turn connect to the PLB bus for access to memory. Figure 8 shows that the transport core also attaches to the data OCM bus, allowing the processor to access the transport's internal data buffer as if it were an extension of the data cache. A dedicated interrupt (not shown) is provided that allows the processor to be called when special processing is needed.

The ability to extend the transport function using software running on the processor is referred to as *transport assist*. The transport-assist processor can be the same processor used to control the rest of the system, or a second processor can be dedicated for this function. The transport-assist software is intended to be independent of the transport drivers and the application, which facilitates

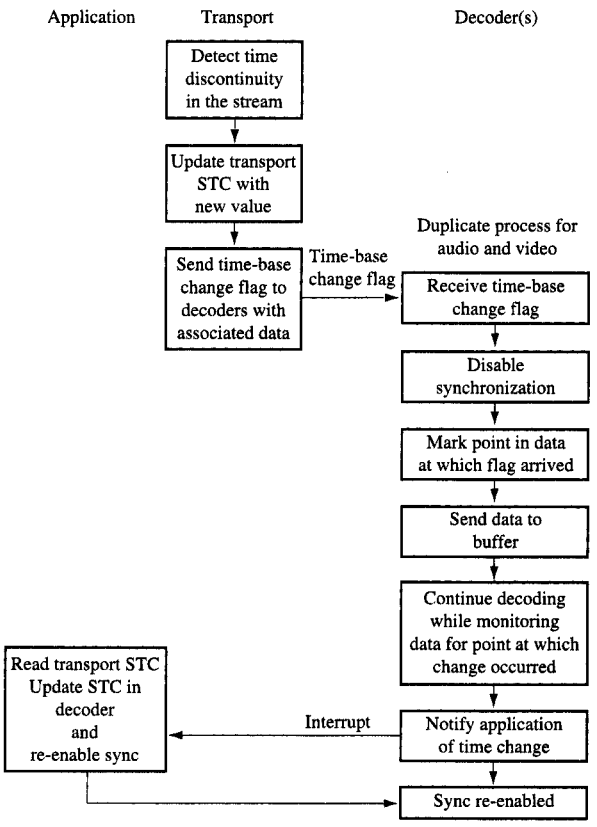


Figure 7

Time-base change flow.

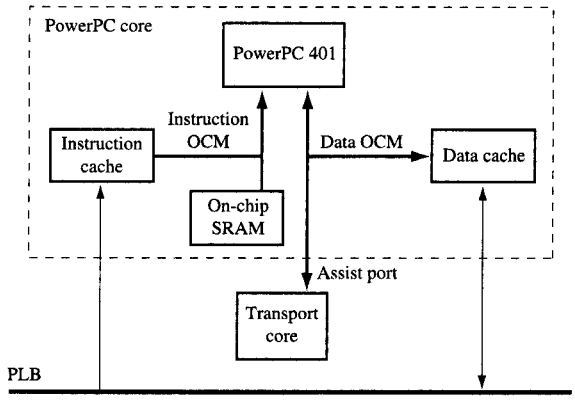


Figure 8

On-chip memory interface for the transport-assist feature.

adding a dedicated transport-assist processor if necessary. A signaling mechanism is included in the transport design to allow the two processes, which may reside on different processors, to communicate with each other as though the assist function were an extension of the transport hardware.

The SRAM shown in Figure 8 contains transport-assist processor (TAP) software to ensure rapid handling of assisted packets. In applications which do not need the TAP function, the SRAM can be used for other critical code.

The TAP software running on the processor configures special registers in the transport to hold given packet types under certain conditions and issue the TAP interrupt to the processor, indicating the need for additional processing. The TAP interrupt-handling routine can read and modify the packet data and the associated information tags for that packet. The interrupt handler removes the hold on the packet by updating one of the information tags. The modified packet continues through the transport and is processed according to the updated information tags.

Streams that are not configured for assistance continue moving through the transport without interaction with the TAP functions. As a result, only packets that require assistance use processor resources. The TAP function also provides the ability to do software upgrades to the transport function in the field.

## Summary

The integrated MPEG-2 subsystem is implemented using components from the IBM Blue Logic core library. Several issues were addressed while designing the subsystem to ensure interoperable and reusable cores that can be used together as a group as well as independently in other applications. This was achieved in part by using standardized bus interfaces wherever possible.

With the cores integrated on the same chip, the cost for adding signals interconnecting macros is very low, allowing the addition of specialized interfaces between the cores. The specialized interfaces between the transport demultiplexor and the decoders provide improved handling of errors, channel changes, and time-base changes. The OCM bus interface between the transport and the processor provides the ability to extend transport processing for future applications.

There are also several advantages that can be capitalized on by developing the cores as a subsystem rather than a collection of discrete functions. A shared-memory model was implemented to allow any core to access memory anywhere in the system. The shared-memory model provides sufficient bandwidth to meet both MPEG decoding and processor requirements, in certain cases reducing the bandwidth demands associated with nonintegrated solutions. Additionally, memory allocation is made more efficient, which may result in fewer memory components being required by the overall system and hence reduced cost.

The overall benefit of integration is lower-cost digital television applications, which are cost-competitive with their analog predecessors. Customers benefit from a wider range of television services, which are also of higher quality.


## Acknowledgments

The authors wish to thank the IBM Digital Video Products management team for supporting the development of the MPEG-2 cores. We would also like to thank all of the members of the set-top box development project in Raleigh, North Carolina, Endicott, New York, Burlington, Vermont, and Haifa, Israel. They influenced the design of the MPEG-2 cores and many of the ideas presented here.


*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Moving Picture Experts Group.

## References


1. "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Systems," *ISO/IEC 13818-1*, First Edition, April 1996.
2. "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video," *ISO/IEC 13818-2*, First Edition, April 1996.
3. "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Audio," *ISO/IEC 13818-3*, First Edition, April 1996.
4. Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg, and Didier J. LeGall, *MPEG Video Compression Standard*, Chapman & Hall Publishers, New York, 1997.
5. Barry G. Haskell, Atul Puri, and Arun N. Netravali, *Digital Video: An Introduction to MPEG-2*, Chapman & Hall Publishers, New York, 1997.
6. *http://www.chips.ibm.com*.
7. "Common Interface Specification for Conditional Access and Other Digital Broadcasting Decoder Applications," *CENELEC Specification, EN 50221*, Ed. 1, February 1997.
8. J.-L. Giachetti, V. Lenoir, A. Codet, D. Cutts, and J. Sager, "A Common Conditional Access Interface for Digital Video Broadcasting Decoders," *IEEE Trans. Consumer Electron.* **41**, No. 3, 836–841 (1995).
9. G. Rossi, "Conditional Access to Television Broadcast Programs: Technical Solutions," *ABU Tech. Rev.*, pp. 3–12 (September–October 1996).
10. D. Van Schooneveld, "Standardization of Conditional Access Systems for Digital Pay Television," *Philips J. Res.* **50**, No. 1/2, 217–225 (1996).
11. Will Remaklus, "On-Chip Bus Structure for Custom Core Logic Designs," *IBM MicroNews* **3**, No. 3, Third Quarter 1997, which can be found at the following web site: *http://.www.chips.ibm.com/techlib/micronews/vol3_no3/article/12.html*.



*Received February 2, 1998; accepted for publication July 9, 1998*

**Richard E. (Andy) Anderson** *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (andya@us.ibm.com).* Mr. Anderson graduated from MIT with a B.S. in electrical engineering and an M.S. in electrical engineering and computer science in 1987. His previous work included human interface development for Point of Sale terminals at NCR Corporation. While at NCR, he was co-inventor of an issued patent, co-wrote a conference paper, and co-authored an article. He joined IBM in 1992 to design and support microprocessors. For the last two years he has worked on the IBM MPEG-2 transport demultiplexor and is a co-inventor for 13 patent applications in this area.

**Eric M. Foster** *IBM Research Division, Endicott, New York 13760 (emfoster@us.ibm.com).* Mr. Foster received a B.S. in mechanical engineering from Clarkson University in 1982 and an M.S. in electrical engineering from the State University of New York at Binghamton in 1987. He joined the Federal Systems Division of IBM in 1982, and over time he worked on the mechanical design of large shipboard computers, module packaging design, manufacturing process development, and the electrical analysis of packaging noise effects. In 1989, he moved to the Microelectronics Division of IBM, where he worked on the design and characterization of high-performance packaging for optoelectronic data links. In 1993 he joined the Digital Video Products group, which is focused on the development of MPEG products; he has worked on set-top box network interface design, PC-based evaluation cards, and MPEG-2 transport architecture. Mr. Foster is currently working on integrated chip architecture and MPEG subsystem design, and has recently moved to the IBM Research Division. He has taught short courses on packaging and optoelectronics, published several internal and external papers, and filed fifteen patents. He is a member of the IEEE Consumer Electronics Society.

**Dennis E. Franklin** *IBM Research Division, Endicott, New York 13760 (dfrankli@us.ibm.com).* Mr. Franklin graduated from Rochester Institute of Technology with a B.S.E.E. in 1970, joining IBM the same year. He worked for ten years in software development doing systems performance analysis and telecommunications development; he managed test organization as well as the CMS component of VM/370. Moving to the hardware group, he worked with intermediate processor performance as well as integrated controller design/analysis. In the past five years, Mr. Franklin has been involved with digital video, working on MPEG decoding and set-top design. He is currently working on architecture for the integration of multiple components needed for consumer video products.

**Ronald S. Svec** *IBM Research Division, Endicott, New York 13760 (svec@us.ibm.com).* Mr. Svec graduated from SUNY/Buffalo with a B.S. in electrical engineering in 1980 and received an M.S. in computer engineering from Syracuse University in 1983. His previous work included hardware development for IBM midrange processors and channel-attached peripherals. He has been involved with MPEG since 1992, and has contributed to the development of numerous IBM decoder and encoder products.

[[Page 806 is blank]]