FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 22, no. 3, December 2009, 371-383

Multiplexing H.264 Video with AAC Audio Bit Streams

Harishankar Murugan, K. R. Rao, and Do Nyeon Kim

Abstract: This paper has developed a multiplexing-demultiplexing system for H.264 video stream and AAC audio stream that is based on the MPEG-2 framework and uses frame numbers as timestamps. It also provides lip sync after decoding the video and audio bit streams.

Keywords: H.264 video, AAC audio, multiplexing/demultiplexing.

1 Introduction

WITH better quality and less bandwidth requirement, digital television transmission has already replaced analog television transmission in a big way. With the advent of HDTV, transmission schemes are aiming at transmitting superior quality video with provision to view both standard format and wide screen (16:9) format along with one or more audio streams per channel. Digital video broadcasting (DVB) in Europe and advanced television systems committee (ATSC) [1–3] in North America are working in parallel to achieve high quality video and audio transmission. Choosing the right video codec and audio codec plays a very important role in achieving the bandwidth and quality requirements. H.264, MPEG-4 part-10, or AVC [4–15] achieves about 50% bit rate savings as compared to earlier standards [16]. H.264 provides the tools necessary to deal with packet losses in packet networks and bit errors in error-prone wireless networks. These features make this codec the right candidate for using in transmission.

Manuscript received on October 31, 2009.

Harishankar Murugan has been with NVIDIA Corp. since 2007. (e-mail: harishankar.murugan@gmail.com). K. R. Rao is with Dept. of Electrical Eng., University of Texas at Arlington, 416 Yates Street, UTA, Box 19016, Arlington, Texas 76019, USA (e-mail: rao@uta.edu). Do Nyeon Kim is with Dept. of Electrical Eng., University of Texas at Arlington, 416 Yates Street, UTA, Box 19016, Arlington, Texas 76019, USA (e-mail: dnkim@uta.edu).

Advanced audio coding (AAC) [17–21] is a standardized lossy compression scheme for audio, used in MPEG–2 [17], and MPEG–4 [22,23]. This codec showed higher coding efficiency and superior performance at both low and high bit rates, as compared to MP3 and AC3.

The H.264 video and AAC audio coded bit streams need to be multiplexed in order to construct a single stream. The multiplexing process mainly focuses on splitting the individual streams into small packets, embedding information to easily realign the packets, achieving lip sync between the individual streams and providing provision to detect and correct bit errors and packet losses. In this paper, the process of encoding video and audio streams, multiplexing the compressed streams followed by demultiplexing, decoding and synchronizing the individual streams during playback, is explained in detail.

1.1 Factors to be Considered for Multiplexing and Transmission

- Split the video and audio coded bit streams into smaller data packets.
- Maintain buffer fullness at demultiplexer without buffer overflow or underflow.
- Detect packet losses and errors.
- Send additional information to help synchronize audio and video.

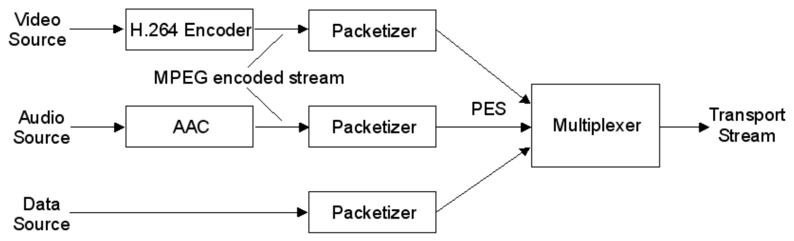

Fig. 1. Two layers of packetization.

1.2 Packetization

According to MPEG–2 systems [4], [24–27], two layers of packetization process (Fig. 1) are carried out. First layer of packetization yields packetized elementary stream (PES) and the second layer yields transport stream (TS). This second layer is what is used for transmission. Multiplexing takes place after the second layer of packetization, just before the transmission.

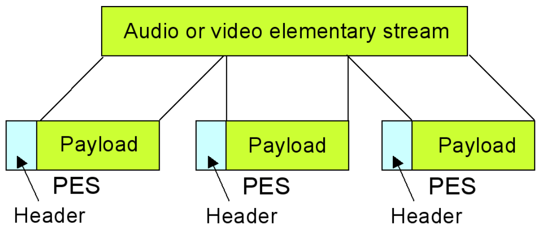

Fig. 2. PES from elementary stream.

2 Packetized Elementary Stream

The packetized elementary stream (PES) packets are obtained by encapsulating coded video, coded audio, and data elementary streams (Fig. 2). This forms the first layer of packetization. The encapsulation on video and audio data is done by sequentially separating the elementary streams into access units. Access units in case of audio and video elementary streams are audio and video frames respectively. Each PES packet contains data from one and only one elementary stream. PES packets may have a variable length since the frame size in both audio and video bit streams is variable. The PES packet consists of the PES packet header followed by the PES packet payload. The header information distinguishes different elementary streams, carries the synchronization information in the form of timestamps and other useful information.

2.1 PES Packet Header

The PES header consists of 3 bytes of start code with a value of `0x000001` followed by 2 bytes of stream ID and 2 bytes of packet length. PES packet length field allows explicit signaling of the size of the PES packet (up to 65,536 bytes) or, in the case of longer video elementary streams, the size may be indicated as unbounded by setting the packet length field to zero. Finally, last 2 bytes of the header are used for timestamps. The frame number of the corresponding audio and video frames in the PES packet is sent as timestamp information in the header.

2.2 Frame Number as Timestamp

he proposed method uses frame number as timestamps. Both H.264 and AAC bit streams are composed of data blocks sorted into frames. A particular video bit stream has a constant frame rate during playback specified by *frames per second* (fps). So, given the frame number, one can calculate the time of occurrence of this frame in the video sequence during playback as follows.

$$\text{Playback Time} = \frac{\text{Frame Number}}{\text{fps}}. \tag{1}$$

AAC compression standard [17] defines each audio frame to contain 1,024 samples. The audio data in the AAC bit stream can have any discrete sampling frequency between 8 – 96 kHz. The frame duration increases from 1/96 ms to 1/8 ms. However, the sampling frequency and hence the frame duration remains constant throughout a particular audio stream. So, the time of occurrence of the frame during playback is as follows.

$$\text{Playback Time} = \frac{1,024 \times \text{Frame Number}}{\text{Sampling Rate}}. \tag{2}$$

Thus from (1) and (2) we can find the time of playback by encoding the frame numbers as the timestamps. In other words, given the frame number of one stream, we can find the frame number of the other streams that will be played at the same time as the frame of the first stream. This will help us synchronize the streams during playback. This idea can be extended to synchronize more than one audio stream with the single video stream like in the case of stereo or programs with single video and multiple audio channels. The timestamp is assigned in the last 2 bytes of the PES packet header. This implies that timestamp can carry frame numbers up to 65,536. Once the frame number exceeds this in case of long video and audio streams, the frame number is rolled over. The rollover takes simultaneously on both audio and video frame numbers as soon as either one of the stream crosses the maximum allowed frame number. This will not create a conflict at the demultiplexer during synchronization due to the fact that the audio and video buffer sizes are much smaller than the maximum allowed frame number. So, at no point of time there will be two frames in the buffer with the same timestamp. This method has the following advantages over using clock samples as timestamps.

- Less complex and more suitable for software implementation.
- Saves the extra PES header bytes used for sending the program clock reference (PCR) information periodically.
- No synchronization problem due to clock jitters or inaccurate clock samples.
- No propagation of delay between audio and video due to drift between the master clocks at the transmitter and receiver.

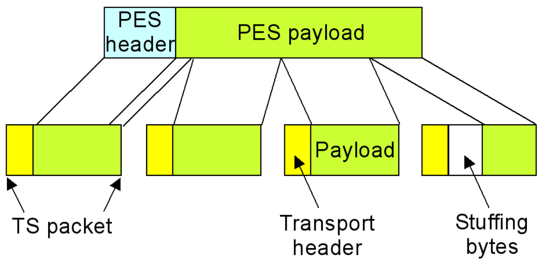

Fig. 3. TS packet formation from PES packet.

3 Transport Stream

The second layer of packetization forms a series of packets called as *transport stream* (TS). These are fixed length sub divisions of the PES packets with additional header information. These packets are multiplexed together to form a transport stream carrying more than one elementary stream. A TS packet is 188 bytes in length and always begins with a synchronization byte of `0x47`. This structure of the packet was originally chosen for compatibility with ATM systems [14]. However there are some applications where more bytes are added at the end to accommodate error correction data like Reed-Solomon or CRC error check data. There are a few constraints to be met while forming the transport packets:

- Total packet size should be of fixed size (188 bytes).
- Each packet can have data from only one PES.
- PES header should be the first byte of the transport packet payload.
- Either PES packet is split, or stuffing bytes are added, if the above constraints are not met.

The encapsulation of PES packets to form TS packets is shown in Fig. 3.

3.1 TS Packet Header

The TS packet normally consists of a 3-byte header. However, if adaptation field is present, then an additional byte is added to the header and 184 bytes are available for payload including the adaptation field. The header structure of the TS packet with the description of the syntax is as follows.

- Payload unit start indicator: Single bit flag set to indicate presence of PES header in the payload.

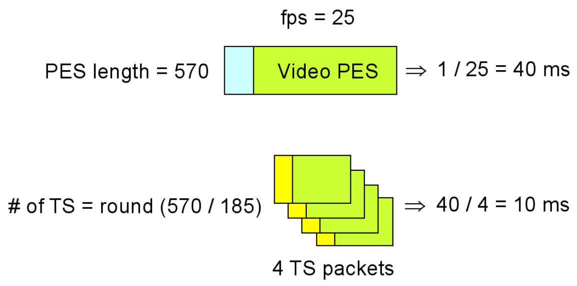

Fig. 4. Calculation of playback time of a TS packet.

- Adaptation field control: Single bit flag to indicate presence of any data other than PES data in payload.
- PID (Packet Identifier): Each elementary stream has a unique 10-bit PID. Some are reserved for null packets or PSI (Program Specific Information). Sequence parameter set and picture parameter set are sent as PSI at frequent intervals. Null packets are used to maintain constant bit rate.
- Continuity counter: 4-bit rolling counter which is incremented by 1 for each consecutive TS packet of the same PID. To detect packet loss.
- Payload byte offset: If adaptation field control bit is '1', byte offset value of the start of the payload or the length of adaptation field is mentioned here.

3.2 Proposed Multiplexing Method

The final transmission stream is formed by multiplexing the TS packets of the various elementary streams. The number of packets allocated for a particular elementary stream during transmission, plays an important part in avoiding buffer overflow or underflow at the demultiplexer. If more number of video TS packets is sent as compared to audio TS packets, then at the receiver there might be a situation when video buffer is full and is overflowing whereas audio buffer does not have enough data. This will prevent us from starting a playback and will lead to loss of data from the overflowing buffer.

In order to prevent such a scenario, timing counters are employed at the multiplexer. Each elementary stream has a timing counter, which gets incremented when a TS packet from that elementary stream is transmitted. The increment value depends on the playback time of the TS packet. The playback time of each PES can be calculated since the frame duration is constant in both audio and video elementary streams. By finding out how many TS packets are obtained from a single PES packet, the playback time of each TS packet can be calculated (Fig. 4). The elementary stream whose counter has the least timing value is always given preference in packet allocation. This method will make sure that at any point of time, the difference in the fullness of the buffers, in terms of playback time is less than the playback time of one TS packet. This is never more than the duration of a single frame and is typically in milliseconds.

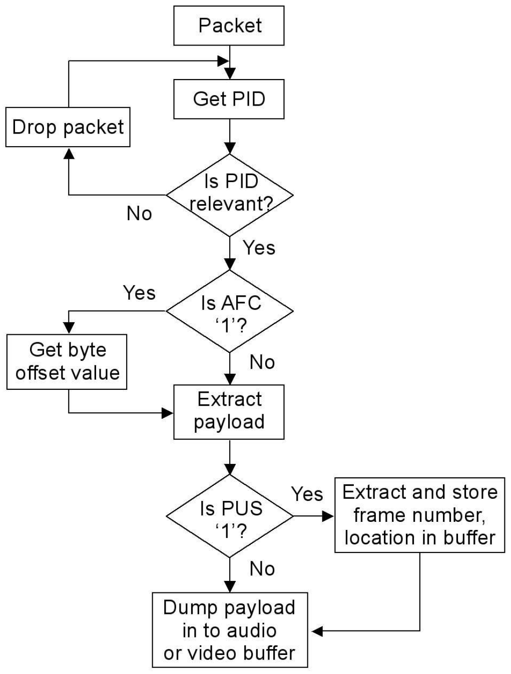

Fig. 5. Flowchart of the demultiplexer.

4 Demultiplexing

The flowchart of the demultiplexer algorithm is given in Fig. 5. Upon receiving the packets, their corresponding PID values are extracted. If the packet has any PID value that is not relevant to the multimedia program that is being recovered, the packet is dropped and the next packet is analyzed. All the TS packets from other programs or null packets are eliminated at this stage. This is done to prevent using the resources on unwanted data. Once the packet has been identified to be a required one, further analysis of the packet is carried out. The '*adaptation field control*' (AFC) bit is checked to see if any data other than the elementary stream data is present in the packet. If yes, then the essential data is recovered from the payload starting from the byte obtained by reading the '*byte offset value*' from the header. The remaining data is rejected if it is filled with stuffing bytes. The data is identified to be audio data or video data through the PID value and is redirected to the appropriate buffer.

The packet is also analyzed to check if the ‘*payload unit start*’ (PUS) bit is set. If it is set, then the PES header is present in the packet. The header information is read to recover the frame length and timestamp, which is the frame number. The frame number and location of the frame in the data buffer are stored in a separate buffer. This process is continued until one of the elementary stream buffers is full.

In order to detect packet losses, 4-bit continuity counter value is continuously monitored for each PID separately, to check if the counter value increases in sequence. If not, a packet loss is declared, and the particular frame in the buffer, which is involved in the loss, is marked to be erroneous. In some transmission schemes, retransmission of the packet is requested to correct the error. If that is not possible, the frame is skipped during playback to prevent any stall in the decoder. If some error correcting codes are sent along with the packet, then the error correction process is completed before the data are put in the buffer. The buffer fullness at the demultiplexer for various buffer size criteria is shown in Table 1. This table clearly shows that the audio and video buffer content has nearly the same amount of playback time, irrespective of the buffer size or the TS start packet during demultiplexing. The difference is in milliseconds. This enables us to playback the content as soon as one of the buffers gets filled and starts refilling the buffer with new content.

Table 1. Buffer fullness at demultiplexer (1 B = 1 byte = 8 bits).

Test criteria / Buffer details	Video buffer 100 kB	Video buffer 600 kB	Video buffer 600 kB
Start TS packet number	1	1	3,850
End TS packet number	802	4,341	7,928
Video buffer fullness (kB)	100	600	600
Audio buffer fullness (kB)	33	14	11
Number of video frames	43	211	173
Number of audio frames	82	397	326
Video buffer content playback time (s)	1.72	8.44	6.92
Audio buffer content playback time (s)	1.75	8.47	6.95

4.1 Synchronization and Playback

Once the elementary stream buffer is full, the content is ready to be played back for the viewer. The audio bit stream format, i.e., audio data transport stream (ADTS), enables us to begin decoding from any frame. However, the video bit stream does not have the same kind of sophistication. The decoding can start only from the anchor frames, which are the instantaneous decoder refresh (IDR) frames. As explained earlier, IDR frames are forced during the encoding process at regular intervals. So, we first search the video buffer from the top to get the first occurring IDR frame. Once this is found, the timestamp or the frame number is obtained for that IDR frame. Then audio stream is aligned accordingly to achieve synchronization. This is done by calculating the audio frame number that would correspond to the IDR frame in terms of playback time, as follows.

$$\text{Audio Frame Number} = \frac{\text{Video Frame Number} \times \text{Sampling Rate}}{1{,}024 \times \text{fps}}. \tag{3}$$

If the frame number calculated is a non-integer value, then the value is rounded off and the corresponding frame is taken. If the calculated frame number is not found in the buffer, next IDR frame is searched and a new audio frame number is calculated. Once video and audio frame numbers are obtained, the location of these frames is looked up in the buffer and the block of data from this frame to the end of the buffer is taken and sent to the decoder for playback. The buffer is then emptied and the incoming data is filled in the buffer and the process is repeated. In order to

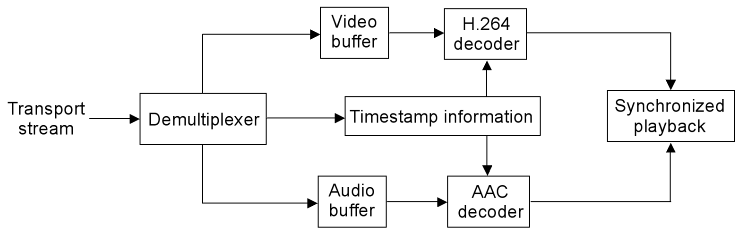

Fig. 6. Demultiplexer block diagram.

have a continuous playback, the block of data from the first IDR frame to the last IDR frame in the buffer is played back and during this playback the next set of data buffering takes place in the background. This process continues and the program is continuously played for the viewer. The maximum error due to rounding the audio frame number to an integer is equal to 0.5 times the duration of an audio frame. This is determined by the sampling frequency of the audio stream. For most sampling frequencies the audio frame duration is not more than 64 ms so the maximum error is about 30 ms. The MPEG–2 systems standard allows a maximum time difference between audio and video streams of 40 ms [27]. So the rounded value is just taken as the required frame number. However, if the maximum possible delay due to round off error is more than 40 ms, then the first IDR frame is repeated which makes up for 40 ms (25 fps). If the audio leads video, then the previous audio frame is taken. This reduces the delay between audio and video to less than 40 ms. Once synchronized, the delay remains constant from there on throughout the playback time. This possible delay is very small and is not perceptible. Still, if the round off error leads to delay of more than 40 ms, then the next IDR frame in the buffer is searched and that new frame number is used for synchronization. Once the buffer is full and the synchronized frames are calculated, the audio and video contents are decoded and played back (Fig. 6).

5 Results

The compression achieved, using H.264 video codec and AAC audio codec, in the proposed method is more than that of the MPEG–2 movie files (.mpg) or AVI files (Table 3). This is true even with the packet header overheads included. The bit rate required for transmission depends heavily on the bit rates chosen by the audio and video encoders. The final bit rate can be varied to suit the transmission channel, by changing the rate control parameter in the H.264 encoder.

Table 2. Observed synchronization delay on test clip.

Start TS packet number	Synchronized frame numbers chosen		Video frame playback time (s)	Audio frame playback time (s)	Delay (ms)	Visual delay
	Video	*Audio*				
120	18	34	0.72	0.725	5.22	Not perceptible
2,000	102	191	4.08	4.074	5.97	Not perceptible
6,000	282	529	11.28	11.284	3.57	Not perceptible
23,689	1,308	2,453	52.32	52.322	2.49	Not perceptible

Some random TS packets are chosen to begin the demultiplexing process and the results of the synchronization process are shown in Table 2. The delay between audio and video is less than 6 ms in the observed cases. This delay remains constant throughout the playback time once synchronized. The details of the video clip used and those of the multiplexed stream are given in Table 3.

6 Conclusions

This paper has developed an effective transport stream that would carry multiple elementary streams and deliver a synchronized multimedia program to the receiver. This is achieved by adopting 2 layers of packetization of the elementary stream followed by a multiplexing procedure. The proposed multiplexing procedure enables the user to start demultiplexing from any TS packet and achieve synchronized playback for any program. There is also provision for error detection and correction after receiving the packets. These are absolutely essential for video broadcasting

Table 3. Results of multiplexed stream on test clip (1 B = 1 byte).

Test clip details	*Clip 1*	*Clip 2*
Duration of clip (s)	125	56
Audio frequency (Hz)	48,000	11,025
Video frame rate (fps)	25	25
YUV file size (kB)	468,168	150,075
WAVE file size (kB)	25,624	957
H.264 file size (kB)	6,566	4,551
AAC file size (kB)	2,066	149
Number of TS packets	50,577	26,872
Total transport stream size (kB)	9,508	5,051
Compression ratio with header overhead	51.93	29.90
Bit rate for transmission (kbps)	76.06	90.20
MPEG-2 movie (*.mpg) file size (kB)	15,666	12,036

applications. Usage of H.264 video bit stream and AAC audio bit stream helps transmit high quality audio-video at less bit rates. This meets all the basic requirements to transmit a high quality multimedia program.

6.1 Future Research

The implementation of the proposed algorithm is directed towards multiplexing two elementary streams that would constitute a single program. However, it can be easily modified to include more streams and can be used to transmit multiple programs together. For television broadcasting applications, error resilience is an important factor. With minor additions to the code, some robust error correction codes can be integrated into the transport packets, to make them more suitable for such applications.

For software contact `harishankar.murugan@gmail.com`.

References

[1] B. J. Lechner *et al.*, “The atsc transport layer, including program and system information protocol (psip),” *Proc. of the IEEE*, vol. 94, pp. 77–101, Jan. 2006.

[2] R. Hopkins, “United states digital advanced television broadcasting standard,” in *SPIE/IS&T Photonics West*, vol. CR61, San Jose, CA, Feb. 1996, pp. 220–226.

[3] Y. Wu *et al.*, “Scanning the issue – special issue on global digital television: Technology and emerging services,” *Proc. of the IEEE*, vol. 94, pp. 5–7, Jan. 2006.

[4] *MPEG–2*, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information ISO/IEC 13 818-1, 2005, part 1: Systems.

[5] *MPEG–4*, Information Technology - Coding of Audio-Visual Objects ISO/IEC JTC1/SC29 14 496-10, 2005, part 10: Advanced Video Coding.

[6] A. Puri, X. Chen, and A. Luthra, "Video coding using the H.264/MPEG-4 AVC compression standard," *Signal Processing: Image Communication*, vol. 19, pp. 793–849, Oct. 2004.

[7] T. Wiegand *et al.*, "Overview of the h.264/avc video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560–576, July 2003.

[8] M. Fieldler, "Implementation of basic h.264/avc decoder," Chemnitz Univ. of Technology, Chemnitz, Germany, June 2004, seminar Paper.

[9] R. Linneman, "Advanced audio coding on FPGA," Master's thesis, Brisbane, Australia, Oct. 2002, BS Honours Thesis.

[10] J. Watkinson, *The MPEG Handbook*, 2nd ed. Oxford; Burlington, MA: Elsevier/Focal Press, 2004.

[11] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*. West Sussex, UK: John Wiley & Sons, 2003.

[12] P. D. Symes, *Digital Video Compression*. New York, USA: McGraw-Hill, 2004.

[13] C. Wootton, *Practical Guide to Video and Audio Compression: From Sprockets and Rasters to Macro Blocks*. Oxford: Focal, 2005.

[14] MPEG official website. [Online]. Available: http://www.mpeg.org

[15] H.264 software JM (10.2). [Online]. Available: http://iphome.hhi.de/suehring/tml/

[16] S. K. Kwon, A. Tamhankar, and K. R. Rao, "Overview of H.264 / MPEG–4 Part 10," *J. Visual Communication and Image Representation*, vol. 17, pp. 186–216, Apr. 2006.

[17] *MPEG–2, Advanced Audio Coding, AAC*, International Standard IS 13818-7 ISO/IEC JTC1/SC29 WG11, 1997.

[18] M. Bosi and R. E. Goldberg, *Introduction to Digital Audio Coding and Standards*. Boston, MA: Kluwer, 2003.

[19] K. Brandenburg, "MP3 and AAC explained," in *Proc. AES 17th Int'l Conf. on High Quality Audio Coding*, Florence, Italy, Sept. 1999.

[20] Alternative AAC software. [Online]. Available: http://www.psytel-research.co.yu

[21] FAAC and FAAD AAC software. [Online]. Available: http://www.audiocoding.com

[22] *Generic Coding of Moving Pictures and Associated Audio Information – Part 3: Audio, MPEG-2*, International Standard IS 13818-3, Information Technology ISO/IEC JTC1/SC29 WG11, 1994.

[23] H. Kalva *et al.*, "Implementing multiplexing, streaming, and server interaction for mpeg–4," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 1299–1311, Dec. 1999.

[24] P. V. Rangan, S. S. Kumar, and S. Rajan, "Continuity and synchronization in MPEG," *IEEE J. Sel. Areas in Commun.*, vol. 14, pp. 52–60, Jan. 1996.

[25] D. K. Fibush, "Timing and synchronization using mpeg-2 transport streams," *SMPTE Journal*, vol. 105, pp. 395–400, July 1996.

[26] Z. Cai *et al.*, “A RISC implementation of MPEG-2 TS packetization,” in *Proc. of IEEE Conf. on High Performance Computing (HPC–Asia)*, Beijing, China, May 2000, pp. 688–691.

[27] *Generic Coding of Moving Pictures and Associated Audio Information – Part 4: Conformance Testing, MPEG–2*, International Standard IS 13818-4, Information Technology ISO/IEC JTC1/SC29 WG11, 1998.