2006 IEEE International
Symposium on Signal Processing
and Information Technology

# Implementation of High Data Rate Stream Parsing with Data Aligning Mechanism

Todor Mladenov Mladenov, Fahad Ali Mujahid, Eungu Jung, Dongsoo Har
Department of Information and Communications
Gwangju Institute of Science and Technology
1 Oryong Dong, Buk Gu, 500 712, Gwangju, South Korea
todor@gist.ac.kr, fahad@gist.ac.kr, egjung@gist.ac.kr, hardon@gist.ac.kr

***Abstract***— *Nowadays the HDTV (high definition television) is growing more and more popular. In many cases a compression of this high quality multimedia data is needed for storing or transmitting purposes. In order to preserve good quality, the compressed stream should have high data rate. It is also common case to combine several such streams, representing different programs, into one single multimedia stream. Here comes the need of high data rate multiplexing and demultiplexing. To operate with high data rate stream, the clock frequency, the word width or both have to be increased. For cost efficient FPGA implementation the wider word width is preferred, which allows smaller in logic and slower in speed FPGA to be used. In this paper the issues following this choice are revealed and a data aligning implementation technique is proposed.*

***Keywords***— *data aligning, FPGA, demultiplexing, MPEG-2*

## I. INTRODUCTION

At the beginning of its existence, HDTV standard did not attract the expected attention around the world. Today, with the development of the digital entertainment marked and due to the increased public demands, high definition television seems soon to dominate the world wide television market. Along comes the necessity of compressing those high data rate streams for broadcasting or storing. A multimedia compression standard is chosen depending on the particular requirements but in all cases a multiplexing and demultiplexing of audio and video streams is needed. This paper proposes a new technique for simplification of any multimedia stream demultiplexing and any header parsing implementation. This technique, called data aligning, will be presented in the context of MPEG-2 standard. The paper is organized as follows. Section II presents a brief overview of MPEG-2 standard. Systems Layer's implementation is discussed in section III with focus on the proposed new technique. Section IV describes the device on which all tests were conducted. Finally, section V concludes this paper.

## II. MPEG-2 OVERVIEW

### *A. MPEG-2*

MPEG-2 standard consists of five parts: video compressing and decompressing part, audio compressing and decompressing part, systems layer, simulation mechanism and software part. The first three are the main and they are shown on Fig.1. MPEG-2 Simulation part specifies how tests can be designed to verify whether bitstreams and decoders meet the standard requirements. The MPEG-2 software simulation part consists of software simulating the above mentioned three principle parts.

On Fig.1 the video part is represented by a generalized block diagram of MPEG-2 codec for nonscalable video coding. Initially this standard is intended for coding of interlaced video with standard TV resolution and good quality. The output bitstream ranges from 4 to 9 Mbit/s. Later the scope of MPEG-2 video part is revised to include video of higher resolution, like HDTV, at higher bitrates.

The audio part of MPEG-2 allows coding of multichannel audio signals in a forward (MPEG-2 multichannel audio decoder can decode MPEG-1 mono or stereo audio signal) and backward (MPEG-1 stereo decoder

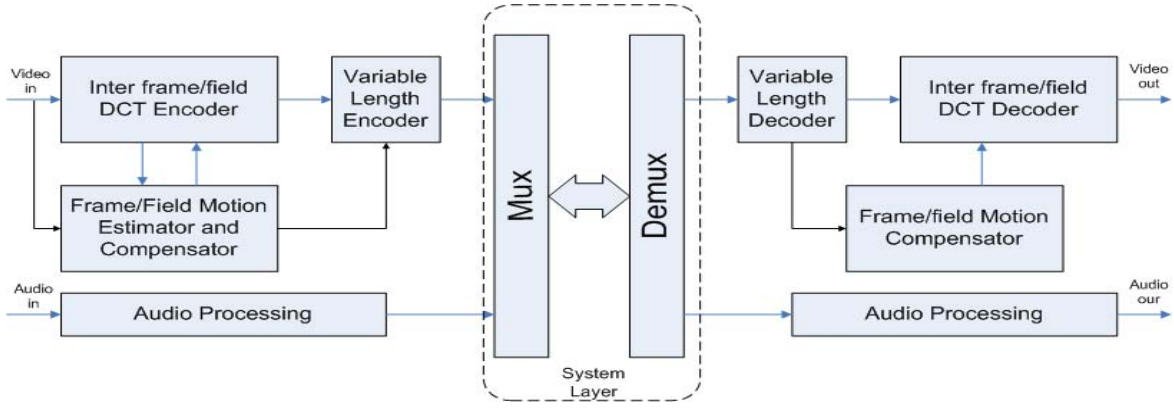



Fig.1 MPEG-2 Overview block diagram

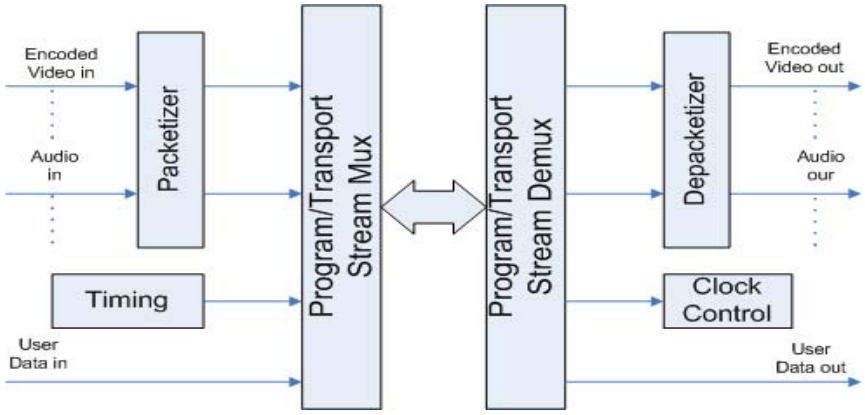



Fig.2 MPEG 2 Systems Layer block diagram

can reproduce a meaningful downmix of the original five channels from MPEG-2 audio bitstream) compatible and not compatible manners with MPEG-1.

MPEG-2 standard is more generic than MPEG-1 and thus intended for a variety of audio-video applications. Systems layer of MPEG-2 is designed to improve error resilience and to be able to carry multiple programs simultaneously without requiring them to have common time base. Furthermore MPEG-2 Systems Layer supports ATM networks and still preserves compatibility with MPEG-1. Further details about MPEG-2 could be found in [1].

*B. Systems Layer*

The MPEG-2 Systems Layer defines two types of streams: Program Stream and Transport Stream. In general Program Stream provides compatibilities to the MPEG-1 systems layer's stream but uses a modified syntax compared to it and has new functions to support advanced functionalities. Program stream decoders typically employ long and variable length packets. Those types of packets are well suited for software based processing and error-free environments – when the compressed data are to be stored. The packet size can be as long as 64 Kbytes. Some of the various features, which Program Stream supports, are: scrambling of data; assigning priorities to packets; copyright mark; trick modes like fast forward and fast reverse; etc.

Transport Stream offers robustness necessary for transporting data over noisy channels and the ability to combine multiple programs into a single stream. The Transport Stream packets have fixed length of 188 bytes. That makes them suitable for implementation of error detection and error correction schemes. Therefore, as its name imply, Transport Stream is used for delivering compressed audio and video over error-prone communication channel. The standard itself does not define an error correction or detection scheme, it just provides means for using one. In Transport Stream multiple programs with independent time bases could be multiplexed in a single stream.

As it is shown on Fig.2 one of the main blocks of MPEG Systems Layer is the Packetizer. It is responsible of forming a basic data structure that is common to the organization of both the Program Stream and Transport Stream, and is called Packetized Elementary Stream (PES) packet. PES packets are generated by packetizing (putting header, information for the length of the packet, etc.) the continuous streams of compressed data, generated by video and audio encoders, called Elementary Streams (ES). Comprehensive description of the main features of MPEG-2 Systems Layer could be found in [2] and [3].

## III. IMPLEMENTATIONS FOR SYSTEMS LAYER

*A. Multipexer*

For implementation of the multiplexing part of the MPEG-2 Systems Layer it is a common situation only a reduced subset of all the features to be needed. Furthermore in most of the cases only one of either Transport Stream or Program Stream is required. In general the MPEG-2 Systems Layer is a byte oriented standard. For high data rate implementation either the frequency of operation could be increased, the word width made wider or both together. The choice mainly depends on the targeted technology and device. When the application is to be programmed on a FPGA matrix, common choice is to increase the word width. ASIC implementation gives more freedom and targets different market segment but still the design's functionality is usually verified on a FPGA device.

The implementation tests performed, showed that the increase of the word width in the MPEG-2 multiplexer doesn't make the design more complex. Contrary it requires less logic after synthesis. The wider word width reduces the number of states and the control logic, which the state machine needs, providing this effect. At most, the various byte stuffing techniques given by the standard has to be used so the packet to be aligned. For the test, a simple five

**Table 1.** Implementation Comparison

| Area ( logic elements) | Multiplexer | Demultiplexer | Total |
|---|---|---|---|
| 8 bit word width | 1486 | 2412 | 3898 |
| 32 bit word width | 1079 | 1749 | 2828 |

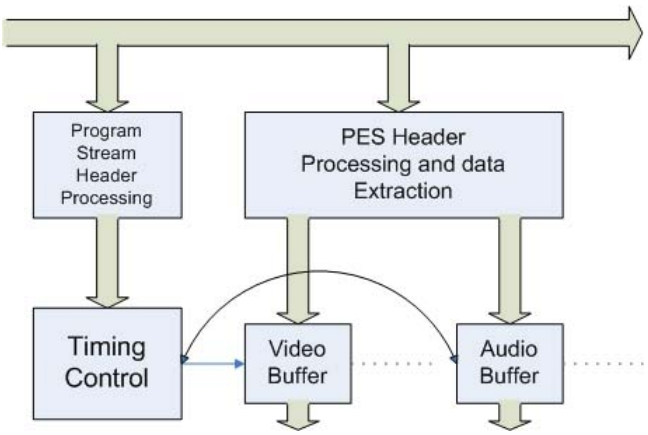



Fig.3 Program Stream Demultiplexer

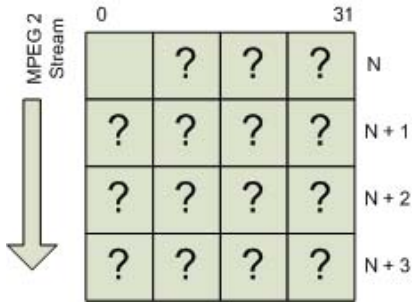
state machine was used, which assigns a four byte value to a register in every state. It requires 45 logic elements on the device which parameters are given in section IV. If the states are extended to 20 and in every state one byte value is assigned to the same register the required logic elements are 63. Therefore a shorter state machine requires less logic. Furthermore in the first case the state machine HDL description becomes shorter and more "readable". Empirical proves for the last claim could not be pointed out as far as it is more close to a philosophical quantity. But it is still a quite logical conclusion.

*B. Demultiplexer*

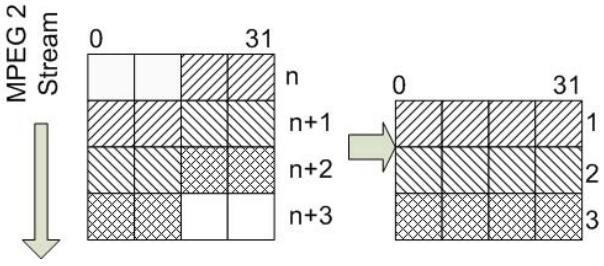
When a demultiplexer or decoder of any type of receiver is to be implemented the resulting device structure is much more complex than the corresponding transmitting (encoding, multiplexing) part. Two main reasons come to support this claim. A demultiplexer is expected to be able to parse all kind of multiplexed streams, regardless of the various functions, which could be incorporated into the stream. As a second reason could be pointed that, in general headers have many fields, which have often variable lengths. The multiplexer has to encode those lengths with the same length in every packet, or if they differ, those variations are less than all possible ones and also known in advance. For the demultiplexer those fields have lengths bounded only by the standard specifications. This paper aims to point out those implementation issues and propose an efficient mechanism (in terms of less logic elements, synthesis time, clock cycles, etc.) for their implementation.

*C. Program Stream Demultiplexer*

Fig. 3 represents a block diagram of a Program Stream demultiplexer with regard to the main performed functions. As it was already mentioned the Program Stream packet can have a variable and significant (up to 64 Kbytes) length. For a demultiplexer, which simply aims to extract the audio and video streams and present them to the video and audio decoders, two operations are needed: a correct time extraction from the Program Stream packets' header and the actual data extraction from the Packetized Elementary Stream packets. Therefore on Fig.3 two independently operating state machines are monitoring the MPEG-2 data stream and they are locking to every Program Stream header and every PES packet. They don't share any data or control signals. When the input MPEG-2 stream has high data rate (as it is the case of compressed HDTV program), it can be parsed either byte by byte at a higher frequency or for example four bytes at a time at four times lower frequency. For FPGA implementation high speed is not affordable at a normal price and a wide word processing is preferred. Why do we want to use an FPGA is a question related more to economics and stays beyond the scope of this paper. As it is expected the complexity of the generated circuit may grow higher but Table 1 shows that as a whole, the design becomes less complex due to the simpler parsing state machines. The tests are conducted on the device, described in section IV.




(a). "All case" decoding




(b). Data aligning mechanism

Fig.4 Stream parsing

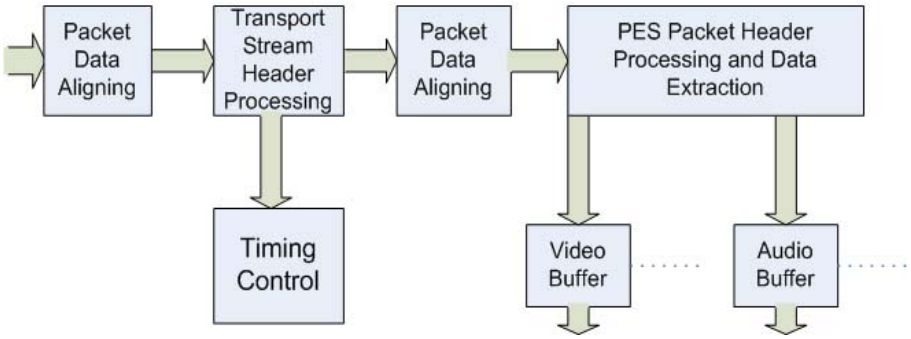



Fig. 5 Transport Stream demultiplexer

In every header the length and the possible number of fields are known in advance. The main difficulty is in the fact that those fields' length can vary in boundaries, which although a priori known, have quite many variations in general. When the stream is parsed byte by byte with the information from the flags in the beginning of the header already extracted, every byte's meaning and exact position into the stream is well known. MPEG-2 and almost every other data stream is generally byte oriented.

Fig.4 shows a four byte word width organization of MPEG 2. It is suitable for data storing on a hard disk through a PCI interface, because it is working with 32 or 64 bit word. As it is pointed on Fig.4 (a) at the point of design of the header processing state machine, it is unknown which flags are selected and which fields are present in the header. Therefore a thorough description of all possible cases is required which grows in complexity with every new word. With the variable length of a field, the next one could start at any of those four bytes, which additionally increase the complexity of parsing.

A simple, yet not reported to the best of my knowledge solution is the data alignment technique. For every field, which is to be parsed with a header parsing state machine, a different state can be designed. That state will always expect the data to start from the first byte in the multiplexed and multiple byte long stream word. This assumption will greatly simplify the HDL description especially when the word width grows wider and the possible positions at which the new field may start become more and more. But this premise will be useful only if another two case state machine always realign the stream data at the end of a field and in the beginning of a new one or at the beginning of a header. The first state will detect the beginning of the desired field and pipeline it so the second state can produce aligned data. Depending on the decoded flags, the data extraction procedure will not differ from field to field and from field's length to field's length. The process of data realigning is depicted on Fig.4 (b) with a simple example. All blocks with a pattern different from white color belong to the same field. The pattern only changes to show how the new words will be formed. This aligning technique can be applied to fields with any length and in the worst case it will cause one clock delay. For practical implementation up to eight bytes word width is reasonable, since the extended PCI interface can operate with 64 bit word width and also it is the highest widely used processing word organization in the computer world.

*D. Transport Stream Demultiplexer*

Fig. 5 shows a block diagram of a MPEG-2 Systems Layer Transport Stream demultiplexer. As the Transport Stream packet length is always fixed to 188 bytes the Transport Stream header and PES packet header parsing is better to be done not by two independently working state machines like it was proposed for Program Stream but by two sequentially working ones. After the first one locks to the packet and extracts the necessary data from the header the data is aligned to the pay load of the packet and send to the second data extracting state machine. Together with all the functions that the data aligning state machine is again performing here it also can work as a connecting buffer between the 'Transport Stream header processing' block and the 'PES packet header processing and data extraction' block. Furthermore the data aligning mechanism, which stays the same in all those different cases, performs the Transport Stream locking function to keep the MPEG-2 demultiplexer synchronous with the processed data stream.

## IV. SIMULATION AND TEST

A byte by byte parsing implementation and four byte word width implementation of MPEG 2 Systems layer were tested on Altera Stratix family FPGA with the following parameters: 25660 logic elements; 1 944 576 total RAM bits; 10 DSP blocks; 80 embedded multipliers; 6 PLLs

**Table 2.** Speed Comparison

| Implementation type | Number of cycle (average) | Speed improvement |
|---|---|---|
| 8 bits word | 59 | 1 |
| 32 bits word | 20 | 2.95 |

Table 2 shows the average number of cycles needed for adding a header in multiplexer and parsing it in demultiplexer. The same test was performed for 8 bits word and 32 bits word. The difference is not multiple of four because aligning adds one clock delay when used. The data shows that the four bytes implementation is almost 3 times faster. As a weak side of a wide word width solution can be pointed out that not many middle class computer systems can support data at rate let's say 64 MHz and 32 bit word under the most common operating systems.

## V. CONCLUSION

This paper aims to bring a useful stream parsing solution to the HDL developing public. Its functionality was demonstrated over MPEG-2 standard but it can be applied to any similar standard or data parsing application.

## VI. ACKNOWLEDGEMENT

This work has been supported in part by the Center for Distributed Sensor Network at GIST, in part by the GIST Technology Initiative (GTI), in part by the MIC, Korea, under the ITRC support program supervised by the IITA" (IITA-2005-C1090-0502-0029), and in part by the Korea Science and Engineering Foundation (KOSEF) through the Ultrafast Fiber-Optic Networks (UFON) Research Center at Gwangju Institute of Science and Technology (GIST).

## REFERENCES

[1] B. G. Haskell, A. Puri, and A. N. Netravali, "Digital Video: an Introduction to MPEG-2", Kluwer academic publishers Boston/Dordrecht/London, fifth edition, 1999

[2] P. A. Sarginson, "MPEG-2: Overview of Systems Layer", Research and development department, BBS, 1996

[3] P.A. Sarginson, "MPEG-2: A Tutorial Introduction to the Systems Layer", The Institute of Electrical Engineering, 1995