The VHDL Based Design of the MIDA MPEG1 Audio Decoder

Andrea Finotello, Maurizio Paolini
CSELT - Centro Studi E Laboratori Telecomunicazioni S.p.A.
Via Guglielmo Reiss Romoli, 274
I-10148 Torino, Italy

Abstract

This paper describes the features and design methodology of MIDA, a MPEG1 integrated audio decoder. MIDA has been almost completely designed using automatic synthesis of VHDL descriptions, and has been implemented using a cell based approach and a 0.7 µm, 2 metal layers CMOS technology. The die area is 95 mm^2. Synthesis tools have also been used for automatic insertion of test structures and automatic test pattern generation.

1. Introduction

This paper details the features and the design methodology of the MIDA[1] (MPEG-I Integrated Decoder of Audio streams) integrated circuit. MIDA is a decoder of audio sequences, encoded in layer I or II of the ISO 11172 (MPEG1) standard [1]. MIDA receives as input a MPEG1 packet audio stream, formatted in 8 or 16 bit words, and outputs the decoded audio stream in serial 16-bit PCM format. The circuit also handles synchronization between audio and video frames by a feedback loop on the audio data presentation frequency. Finally, MIDA handles audio frame synchronization and errors in the audio data stream. Several circuit functions can be programmed by writing into a set of configuration registers.

The target application for MIDA is the M-BIRD MPEG1 audio/video decoder board for IBM compatible PCs, being developed in the framework of the ESPRIT projects M-PLANAR and MAXI.

MIDA has been implemented using a cell based approach and a 0.7 µm, 2 metal layers CMOS technology. The die area is 95 mm^2. The circuit test design has been simplified using design for testability techniques: full scan for all the registers and BIST for all the macrocells.

VHDL and synthesis have played a major role in the MIDA design flow. MIDA has been almost completely implemented by automatic synthesis of VHDL RTL descriptions. Besides, an algorithmic VHDL model of the decoder has been used as reference for verification of both the RTL and the gate level descriptions of MIDA. Finally, synthesis tools have been used for automatic insertion of test structures and automatic test pattern generation.

A brief overview on the structure of the MPEG1 audio code is given in section 2. The chip architecture and the operations performed by each circuit module are described in section 3. The VHDL based design methodology of MIDA is discussed in section 4, together with the relevant design data. Finally, an evaluation of advantages and limits of the design methodology followed is given in section 5.

2. MPEG1 audio code overview

The ISO CD 11172 (MPEG1) standard defines a digital data storage format for moving pictures and associated audio. The MPEG1 encoding format is structured in a set of hierarchical layers for system information, video and audio data.

A MPEG1 system stream consists of one or more multiplexed elementary streams, and is composed of a sequence of *packs*. Each pack consists of a pack layer header, an optional system header, and a sequence of *packets*. The pack layer header is composed of a pack start code, the system clock reference value and the specification of the data arrival rate to the decoder; the system header contains various system-wide limit values and flags. The pack layer is not directly handled by MIDA.

An *audio packet* is a sequence of audio data belonging to the same elementary audio stream. An audio packet is composed of an audio packet start code, a stream identification code, the packet length in bytes, optional buffer size information, optional presentation and decoding time stamps, and a sequence of *audio frames*. The presentation (decoding) time stamp specifies the time at which the first frame in the packet has to be output by (given as input to) the audio decoder. The audio packet layer is the input format for MIDA; MIDA uses the presentation time stamp (PTS) to perform the audio-video synchronization task.

An *audio frame* contains a fixed number of encoded audio PCM samples, and is composed of an audio frame header, an optional error detection checksum for the header, the encoded audio data and optional user-defined

[1] Patent pending

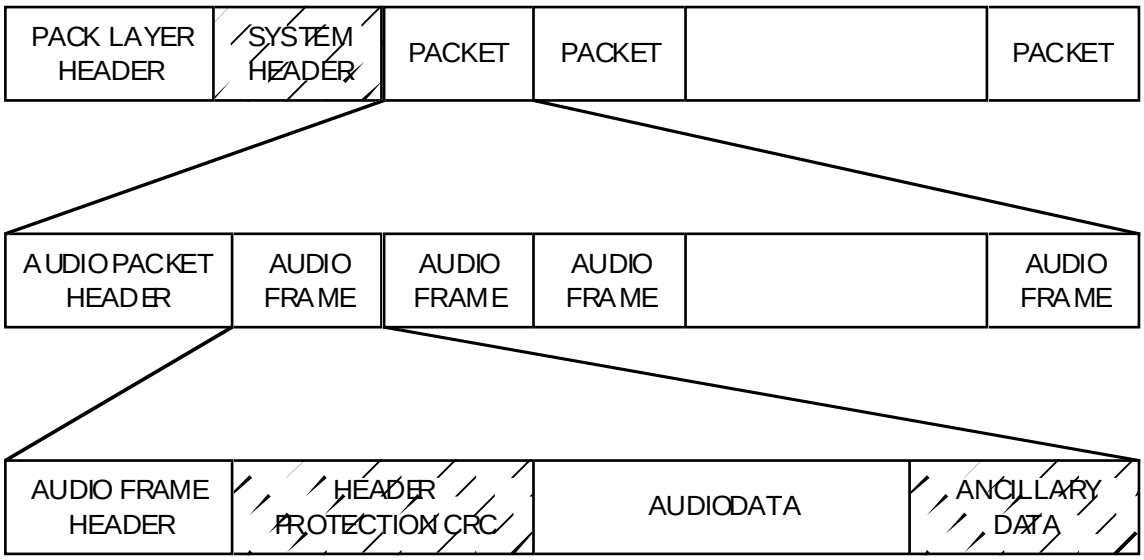

Figure 1 - MPEG1 audio stream organization

data (ancillary data). Three different audio coding layers with increasing complexity and performance are defined by the MPEG1 standard; MIDA handles layers I and II, that are those most widely used.

The audio frame header contains a 12-bit synchronization word and frame related information such as encoding layer, header protection, encoded stream bit rate, output sampling frequency, output mode (mono, stereo, dual channel, intensity stereo) and others. The format of the audio data depends on the selected coding layers and on the output mode.

3. Chip architecture

The internal architecture of MIDA is shown in Figure 2. In this section, the operations performed by each module are detailed.

3.1. External clocks

The MIDA master clock frequency is 24.576 MHz. The master clock is also used to synthesize audio data output frequencies related with 32 KHz and 48 KHz sampling frequencies. A second clock frequency of 22.5958 MHz is used to synthesize audio data output frequencies related with the 44.1 KHz sampling frequency.

3.2. Input data buffer

The input data buffer absorbs the variations in the frequency of the input data. It holds up to 32 8-bit data, and outputs data in a variable format (from 1 to 8 bits), set by the packet audio parser .

3.3. Packet audio parser

The packet audio parser identifies the packet audio level codes in the input streams and extracts from the packet header all the information needed for decoding and synchronization. In detail:

- the packet start code prefix is recognized and used for packet level stream synchronization;
- the stream id value is extracted and compared against the target id set by the user, to verify whether the incoming packet belongs to the selected stream;
- the packet length field is extracted and saved for a later packet consistency check;
- the presentation time stamp PTS is extracted and saved; a PTS found flag is then propagated through the decoder stages, together with the audio data to which the PTS is associated;
- any other information in the packet header is parsed, but is neither used nor saved.

The occurrence of errors in the packet size and structure or in the stream id cause the interruption of the decoder output and the restart of the packet synchronization procedure.

Audio data in the packet are transferred to the audio stream decoder only after a PTS has been found. If no PTS has been found yet, data are discarded because no presentation time can be determined for them.

3.4. Audio stream decoder

The audio stream decoder parses and decodes the audio stream level information. In detail:

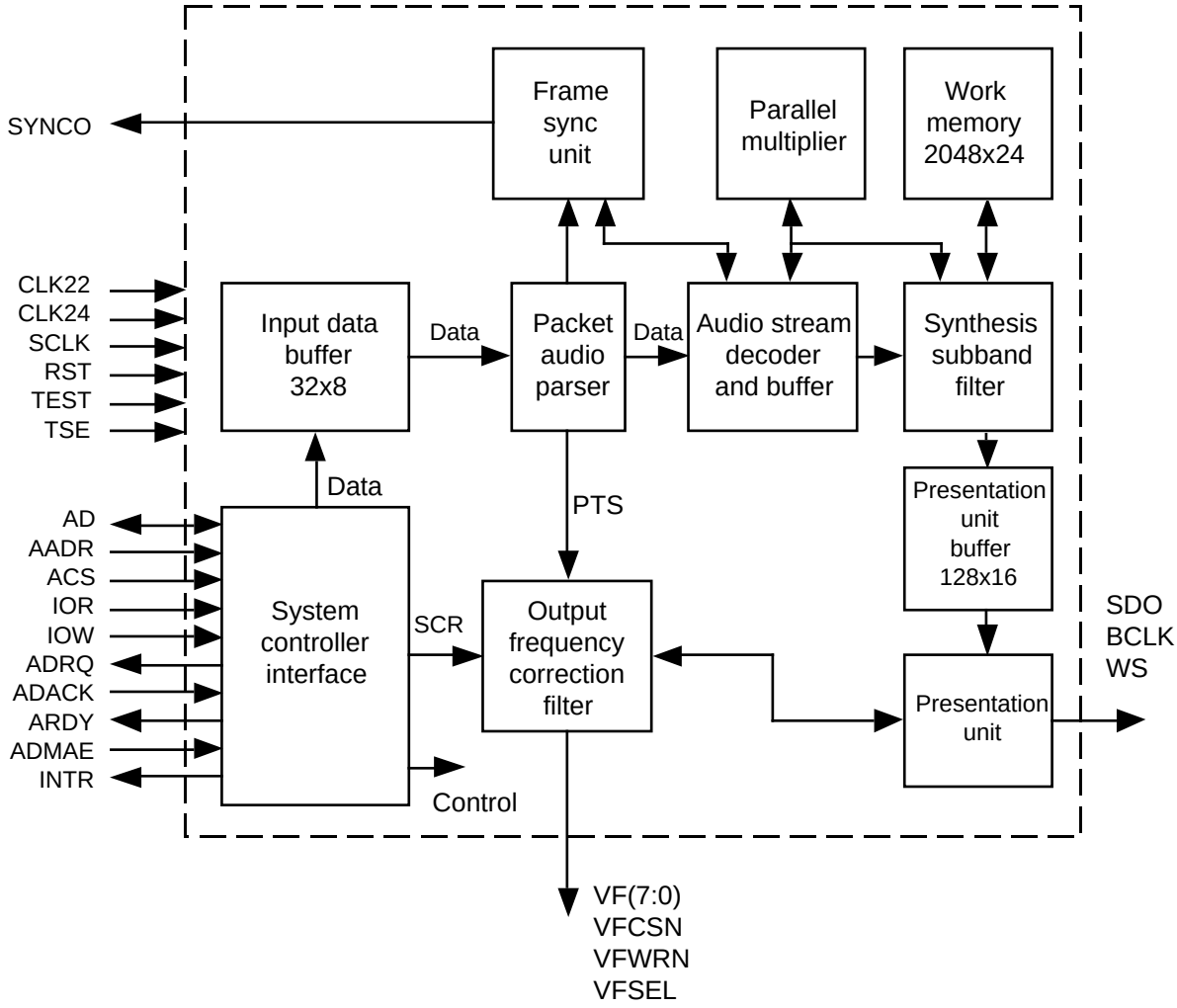

Figure 2 - MIDA internal architecture

• the audio sync word at the beginning of each frame is recognized;

• the frame header is read and the information useful to the decoding process (control word, allocation tables, sample scale factor tables) is saved;

• if the frame header is protected by a CRC, the header consistency is checked;

• the audio subband samples in the frame are uncompressed, dequantized, rescaled and stored into the audio decoder buffer.

Any ancillary data in the frame are discarded.

3.5. Audio decoder buffer

The audio decoder buffer is a paged memory that can store up to 384 23-bit words. The buffer control logic handles the memory paging and the reordering of the data to be transferred to the subband synthesis filter. Both page size and data order depend on the audio stream encoding layer - I or II - and mode - mono, stereo, dual channel or joint stereo. Each memory page is read as a circular buffer.

3.6. Synthesis subband filter

The synthesis subband filter rebuilds the audio data samples from the subband samples, performing the matrixing, windowing and sample reconstruction defined by the standard through a multiply and accumulation algorithm. The filter uses a 24x24 parallel multiplier (shared with the audio stream decoder) and a 2048 word, 24-bit internal work memory. The computed audio samples are stored in the presentation unit buffer.

3.7. Presentation unit buffer

The presentation unit buffer holds up to 128 16-bit audio data samples, organized in pages of 64 samples.

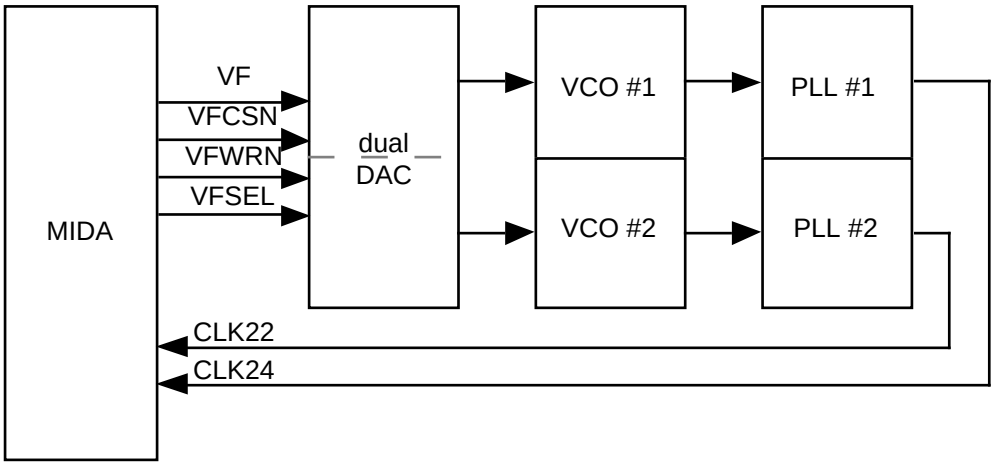

Figure 3- Audio-video synchronization feedback loops

The buffer control logic handles paging and reordering of the audio samples; the sample ordering depends on the audio stream mode .

3.8. Presentation unit

The presentation unit handles the serialization and the output of the decoded audio samples. Samples can be output in two's complement or offset binary format, either in baseband or with a 2x, 4x, 8x oversampling factor. The presentation unit also supplies the output data sampling clock. The audio output can be muted.

3.9. Frame synchronization unit

The frame synchronization unit handles synchronization detection in the input stream, and synchronization recovery in case of decoding errors. This unit also determines the frame length for audio streams encoded in free format. Detected synchronization errors may trigger an interrupt to the external controller.

3.10. Output frequency correction filter

The output frequency correction filter performs two related tasks:

- it controls the timing of the audio output start, by comparing the current value of the System Clock Reference SCR to the PTS value associated to the current decoded audio frame;
- it handles the audio-video synchronization by adapting the frequency of the clock controlling audio output presentation rate so as to minimize the difference between SCR and MIDA internal time.

The MIDA internal time can be determined whenever the first sample of an audio frame with an associated PTS is output and is adapted by a feedback loop on the clock frequency of the external clock controlling the output data presentation. The two feedback loops - one for each clock - are composed of a DAC/VCO/PLL chain controlled by an internal digital filter. The filter outputs 8-bit words in two's complement or offset binary format. If the difference between SCR and internal time falls out of the filter operation range, filtering is suspended and an interrupt is sent to the system controller.

3.11. System controller interface

The system controller interface handles the communication with the system controller and the device programming. The MIDA programming registers allow the user to configure and control the following parameters:

- input data width (8 or 16 bits);
- output data format (offset binary or two's complement);
- output data oversampling factor (1x, 2x, 4x, 8x);
- input data stream id selection;
- audio output mute;
- gain, pole and zero of the output frequency correction filter;
- data format (offset binary or two's complement) of the output frequency correction filter.

Audio data are transferred to MIDA using a DMA-like communication protocol. Data transfer activation and deactivation are controlled by writing command words into the device.

The interface may generate interrupts on occurring events, such as:

- individuation of a PTS in the input stream;
- frame synchronization lock;
- frame synchronization errors;
- start of the output data presentation ;

- other errors in the output frequency correction filter, in the presentation unit or in the synchronization unit.

Each interrupt cause can be masked.

Some status information can be read from the interface: the control word of the last decoded audio frame, the last PTS value found, and information on the activity of some internal modules.

The interface also computes the SCR value using an internal counter externally loadable and driven by the system clock.

4. Circuit design methodology

The MIDA circuit has been designed using a methodology based on automatic synthesis of circuit modules from VHDL RTL descriptions.

At first, a behavioral VHDL description of the audio decoder has been written and simulated. The behavioral simulation results have been used as reference throughout the design process. This model uses floating point arithmetic for internal computations, thus allowing for high simulation throughput. The code size of the algorithmic model is about 7000 lines.

The circuit architecture has then been defined and a synthesizeable VHDL description of every circuit module - excluding the system controller interface, because of the presence of asynchronous parts in it - has been written and verified by simulation. The synthesizeable description size is of about 17000 VHDL lines. The system controller interface has been separately designed using more traditional design methods, i.e. gate level schematic capture and simulation.

A simulation of the complete description has been run and the simulation results have been compared against those obtained from the behavioral simulation. The same set of patterns has been used for both behavioural and RTL simulation; the input packet audio streams have been read from an ASCII file and fed to the model under test by pattern generators written in behavioral VHDL. A format converter – still coded in behavioral VHDL – has been used for translating the floating point output of the behavioral circuit model into 16 bit fixed point integers, to be compared with the RTL model output.

The circuit netlist has then been generated by a bottom-up synthesis process; each module has been separately synthesized with an appropriate set of physical constraints, and an incremental synthesis step has been performed on the complete circuit. The synthesis was performed using the Synopsys VHDL compiler ™ and Design compiler™ software, and the ES2 ECPD07 standard cell library (0.7 μm, 2 metal layers). Macrocells (single and dual port RAMS, parallel multiplier) have been modeled in behavioral VHDL for simulation purposes, and implemented by ES2 module generators. A characterized “black box” model of each macroblock has also been supplied to the synthesis tools so as to ensure the correctness of synthesis constraints (loads, delays) imposed by each block.

The synthesized implementation contains 15428 standard cells, four 1024x12 single port RAMs, three dual port RAMs (128x16, 384x23 and 32x8) and a 24x24 parallel multiplier for a total amount of about 325000 transistors.

The logic level netlist of the circuit has been simulated before and after the layout using the Cadence Verilog™ simulator.

The circuit layout has been realized using the Cadence Preview™ floorplanner and the Cadence CellEnsemble™ placement & routing software. The chip die size is 9,7 mm x 9,7 mm, corresponding to a die area of 95 mm^2.

The circuit test strategy is based on the direct accessibility of internal registers through 29 scan chains, and on BIST of macrocells. The scan chains have been automatically inserted by the Synopsys Test compiler™ software, that has also been used for test pattern generation. The macrocell BIST structures have been supplied by ES2.

MIDA is available in an 84-pin ceramic PGA package.

5. Design methodology evaluation

As stated earlier in this paper, the MIDA design methodology is based on the automatic synthesis of VHDL RTL descriptions. Circuit modules have been described as finite state machines, that have been translated into multilevel logic equations and mapped onto a target cell library by a synthesis tool. Besides, the global chip architecture has been implemented using a schematic editor which gives as output a structural VHDL description of the defined schematic, thus limiting the code writing effort and the possibility of introducing design errors.

This methodology provides many advantages to the design group. The use of both a standard language and a common methodology make communication and information exchange among different designers or partners easier. VHDL also allows to include in the same description and to simulate modules at different abstraction levels (behavioral, RTL, gate), facilitating the top-down design procedure; besides, VHDL descriptions at different abstraction levels can easily share the same test bench. Automatic synthesis allows a fast translation of specifications into logic, increasing the time available for evaluating different design solutions; it also reduces the design dependency from the target technology. Finally, the separation between functionality (described in the VHDL source code) and implementation constraints (separately supplied to the synthesis software) allows an easier circuit debugging and the reuse of modules in different designs.

Beside these advantages, this methodology still has some drawbacks. Firstly, the VHDL simulators available during the development of this design showed out to be not suitable for gate level simulation, and a VHDL model of the gate level library was not available. This forced us to use different simulation environments for RTL and gate level simulations, and to develop a specific set of scripts for translating the patterns into the gate

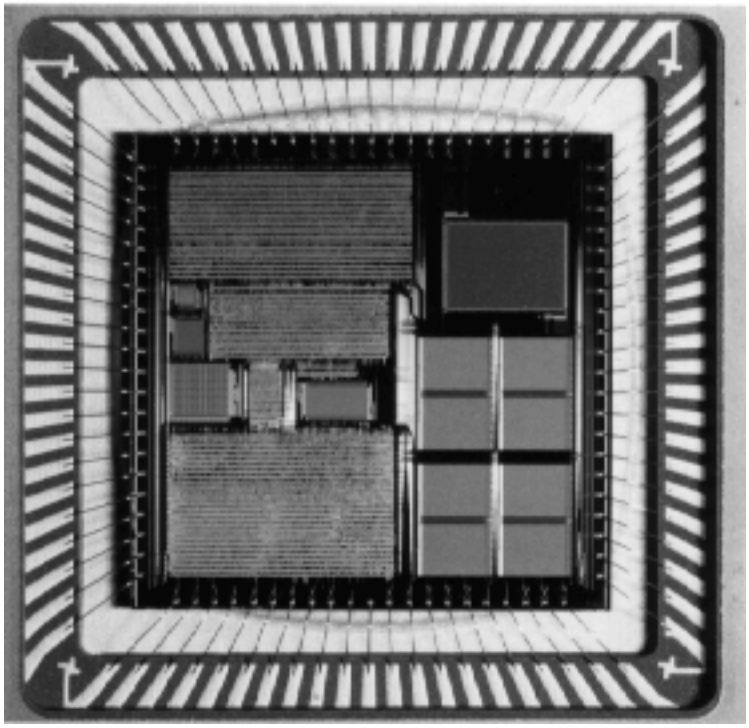

Figure 4 - MIDA die photo

level format. It has to be said that this situation is now changing, because both CAD and silicon vendors are proposing a wide spectrum of solutions to these problems (gate level oriented VHDL simulators, mixed VHDL-gate level simulators, VHDL libraries).

Secondly, the integration of macrocells in the synthesis environment leaves much to be desired. Macrocells could not be inferenced from RTL descriptions, so structural macrocell instances had to be added to the global structural code; that made the code of the modules interfaced with macrocells more complicated. Besides, the module generators did not supply any synthesis model for the generated macrocells; synthesis models had to be developed by hand.

Finally, no automatic feedback from the layout tools to the synthesis tools was possible when this design was completed. That caused an undesirable amount of synthesis and place/route cycles, mostly caused by discrepancies between the actual capacitive load of some connections and the corresponding estimates computed by the synthesis tool. However, synthesis-layout interaction tools that specifically target this class of problems are now becoming available on the market.

6. Conclusions

This paper described the characteristics and the design methodology of the MIDA circuit, an integrated MPEG1 audio decoder.

MIDA has been almost completely designed using automatic synthesis of VHDL descriptions. The synthesis tools have also been used for the automatic insertion of test structures and the automatic test pattern generation.

The VHDL based design methodology used has given good results in terms of both design consistency and productivity, and is currently being extended towards the design and management of a library of parametric synthesizable modules for telecom applications [2]. The ongoing evolution of the synthesis tools, together with the improvement of the support given to VHDL and synthesis users by the silicon foundries, should lead to a rapid increase in popularity of this design methodology.

References

[1] “Information technology - Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s”, ISO/IEC DIS 11172, 1992

[2] Enrico Domenis, Enrica Filippi, Luigi Licciardi, Maurizio Paolini, Maura Turolla, Denis Rouquier, “Fast Prototyping for Telecom Components Using a Synthesizable VHDL Flexible Library”, VLSI '95, Makuhari Messe, Japan, August 1995