TECHNICAL REPORT

ISO/IEC TR 14496-9

Third edition
2009-02-01

Information technology — Coding of audio-visual objects —

Part 9: Reference hardware description

Technologies de l'information — Codage des objets audiovisuels —

Partie 9: Description de matériel de référence

Reference number
ISO/IEC TR 14496-9:2009(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 14496-9, which is a Technical Report of type 3, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This third edition cancels and replaces the second edition (ISO/IEC TR 14496-9:2008) which has been technically revised.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimized reference software for coding of audio-visual objects* [Technical Report]

— *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*

— *Part 9: Reference hardware description* [Technical Report]

— *Part 10: Advanced Video Coding*

— *Part 11: Scene description and application engine*

— *Part 12: ISO base media file format*

— *Part 13: Intellectual Property Management and Protection (IPMP) extensions*

— *Part 14: MP4 file format*

— *Part 15: Advanced Video Coding (AVC) file format*

— *Part 16: Animation Framework eXtension (AFX)*

— *Part 17: Streaming text format*

— *Part 18: Font compression and streaming*

— *Part 19: Synthesized texture streaming*

— *Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*

— *Part 21: MPEG-J Graphical Framework eXtension (GFX)*

— *Part 22: Open Font Format*

— *Part 23: Symbolic Music Representation*

— *Part 24: Audio and systems interaction*

— *Part 25: 3D Graphics Compression Model*

Introduction

The main goal of this Technical Report is to facilitate a more widespread use of the MPEG-4 standard.

Design methodologies of the EDA industry have evolved from schematics to Hardware Description Languages (HDLs) to address the needs of the vast number of gates available on a single device. The increased number of gates allowed more elaborate algorithms to be deployed but also required a shift in design paradigm to handle the complexity created. Through HDLs, more complicated systems could be designed faster through the enabling technology of synthesis of the HDL code towards different silicon technologies where trade offs could be explored. Now the EDA industry again faces challenges where HDLs may not provide the level of abstraction needed for system designers to evaluate system level parameters and complexity issues. There have been a number of tool investigations under way to address this problem. Profiling tools aid in exposing bottlenecks in an abstract way so that early design decisions can be made. C to gates tools allow a C based simulation environment while also enabling direct synthesis to gates for hardware acceleration.

In conclusion, it is the aim of this Technical Report to enable more widespread use of the MPEG-4 standard through reference hardware descriptions and close integration with MPEG-4 Part 7 Optimized Reference Software. Additionally, it is aimed that exposure to such a platform will enable a more systematic way to investigate the complexity of new codecs and open up the algorithm search space with an order of magnitude more compute cycles.

Information technology — Coding of audio-visual objects —

Part 9:
Reference hardware description

1 Scope

This part of ISO/IEC 14496 specifies descriptions of the main video coding tools in hardware description language (HDL) form. Such alternative descriptions to the ones that are reported in ISO/IEC 14496-2, ISO/IEC 14496-5 and ISO/IEC TR 14496-7 correspond to the need of providing the public with conformant standard descriptions that are closer to the starting point of the development of codec implementations than textual descriptions or pure software descriptions. This part of ISO/IEC 14496 contains conformant descriptions of video tools that have been validated within the recommendation ISO/IEC TR 14496-7.

2 Copyright disclaimer for HDL software modules

Each HDL module shall be accompanied by the following copyright disclaimer, to be included in each HDL module and all derivative modules:


```
/*************************************************************************

This software module was originally developed by
    <Family Name>, <Name>, <email address>, <Company Name>

    (date: <month>,<year>)

and edited by: <Family Name>, <Name>,<email address>

This HDL module is an implementation of a part of one or more MPEG-4
tools(ISO/IEC 14496).

ISO/IEC gives users of the MPEG-4 free license to this HDL module or
modifications thereof for use in hardware or software products claiming
conformance to the MPEG-4 Standard.

Those intending to use this HDL module in hardware or software products are
advised that its use may infringe existing patents.

The original developer of this HDL module and his/her company, the subsequent
editors and their companies, and ISO/IEC have no liability for use of this HDL
module or modifications thereof in an implementation.

Copyright is not released for non MPEG-4 Video conforming products.

<Company Name> retains full right to use the code for his/her own purpose, assign
or donate the code to a third party and to inhibit third parties from using the
code for non MPEG standard conforming products.
```

```
This copyright notice must be included in all copies or derivative works.

Copyright (c) <year>.

Module Name:  <module_name>.vhd

Abstract:


Revision History:

*********************************************************************/
```

3 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply:

AV	Audio-Visual
DCT	Discrete Cosine Transform
IDCT	Inverse Discrete Cosine Transform
HDL	Hardware Description language
ISO	International Organization for Standardization
MPEG	Moving Picture Experts Group
Verilog	A Hardware Description Language
VHDL	VHSIC high speed Hardware Description Language
SAD	Sum of Absolute Differences
MAC	Multiply ACcumulate
MAD	Minimum Absolute Difference
SIMD	Single Instruction Multiple Data
DA	Distributive Arithmetic
EDA	Electronic Design and Automation
IEEE	Institute of Electrical and Electronic Engineers
IMEC	Interuniversity Micro Electronic Center
EPFL	École Polytechnique Fédérale de Lausanne

4 HDL software availability

The HDL and System C software modules described in this part of ISO/IEC 14496 are available within the zip file containing this Technical Report. Each module contains a separate directory structure for the source code with a readme.txt file explaining the top level and all files to be included for simulation and synthesis.

5 HDL coding format and standards

5.1 HDL standards and libraries

As the IEEE has several HDL coding standards that are commonly used in hardware reference code (i.e. VHDL1076-1987, VHDL 1164-1993, Verilog 1364-2000, Verilog 1364-1995), the modules constituting this part

of ISO/IEC 14496 are made of the latest IEEE standard possible at the time of coding for all reference HDL code. As the IEEE has provided libraries to assist in the use of HDL, only IEEE standard libraries are needed to use the HDL code.

Custom libraries which are specific to the vendor's (Silicon) base library elements are used only if they are freely available for synthesis and simulation and are provided in an accompanying module version of the submitted HDL code using the standard libraries mentioned above.

5.2 Conditions and tools for the synthesis of HDL modules

As there are many choices commercially for HDL synthesis and HDL simulation software tools, specific synthesis or simulation libraries that are used for reference HDL code are properly documented. The same code that is used to synthesize towards an implementation is also used to perform HDL behavioral simulation of the MPEG-4 tool. The code is properly documented with respect to the synthesis and simulation tool (and version) that has been used to perform the work. HDL module codes with multiple synthesis and simulation tools are also possible. In the event a source code modification must be made to support an additional synthesis or simulation tool, an additional source code is provided with proper documentation.

5.3 Conformance with the reference software

HDL reference code provides sufficient test bench code and documentation on how it is conformant with respect to the reference software. To the extent possible, bit and cycle true models are provided which can be used directly in the reference software code for verification. In the case that the reference HDL code is derived from other languages such as: C, C++, System C, Java, it is recommended that that this code and information on the methodology used to generate HDL should be provided to improve verification of conformance of the HDL code.

6 Integrated Framework supporting the "Virtual Socket" between HDL modules described in Part 9 and the MPEG Reference Software (Implementation 1).

6.1 Introduction

The aim of this clause is to document the framework developed by Xilinx Research Labs for the integration of HW modules with the MPEG-4 reference software. The purpose of this virtual socket framework is to create an abstraction between the specific physical layer and specific software driver library to facilitate a reusable hardware/software co-design environment. By acting as an intermediary between specific physical layer bus protocols, the hardware accelerator designer can focus on the acceleration algorithm rather than lower level interface protocols.

The framework of the Virtual Socket allows for 31 addressable hardware accelerators to be present in a single device (see Figure 1). Each specific hardware accelerator will be assigned a bit of the 32-bit hardware identification register and these bit locations shall be assigned to particular MPEG development teams (see Figure 2 for an example containing two accelerators at slots 1 and 6). If an accelerator socket is not present then its bit in the identification register will be de-asserted. Unassigned sockets will also be de-asserted indicating no accelerator is present. In the event that hardware accelerator designers wish to put further identification of their socket they may do so by allocating further identification registers within their socket's assigned register space.

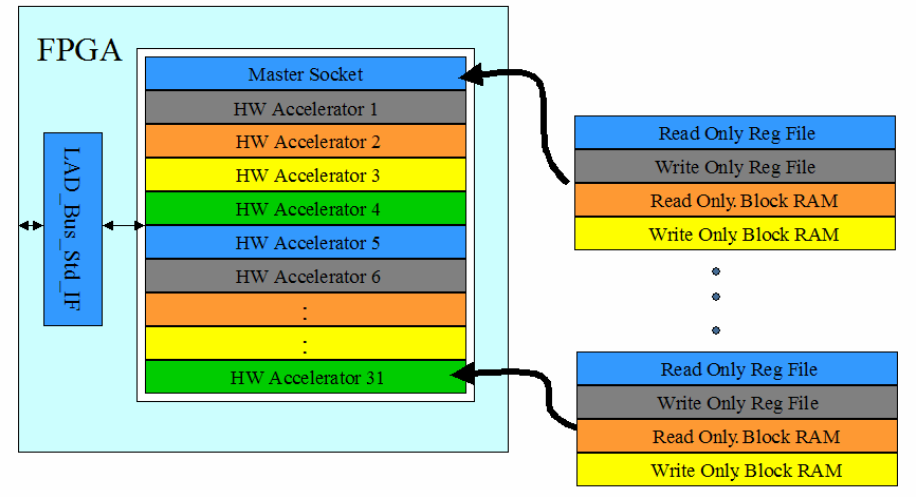

Figure 1 — Block Diagram of Virtual Socket Platform.

Figure 2 — Example 32-Bit Hardware Identification Register.

6.2 Addressing

The virtual socket provides four strobes that indicate what region of the memory space, register or memory, has been accessed as well as the type of operation, write or read. Although a 16-bit is provided to each socket, the least significant nine bits are only necessary to address within the 512 word assigned memory region. The Virtual Socket API uses macros that assist the software designer in transferring data to and from memory locations.

6.3 Memory Map

Table 1 — Memory Mapping for Register File Allocation.

	Register Read-Only			Register Write-Only	
Socket #	Begin	End		Begin	End
Master	0000	01FF		4000	41FF
1	0200	03FF		4200	43FF
2	0400	05FF		4400	45FF
3	0600	07FF		4600	47FF
4	0800	09FF		4800	49FF
5	0A00	0BFF		4A00	4BFF
6	0C00	0DFF		4C00	4DFF
7	0E00	0FFF		4E00	4FFF

8	1000	11FF		5000	51FF
9	1200	13FF		5200	53FF
10	1400	15FF		5400	55FF
11	1600	17FF		5600	57FF
12	1800	19FF		5800	59FF
13	1A00	1BFF		5A00	5BFF
14	1C00	1DFF		5C00	5DFF
15	1E00	1FFF		5E00	5FFF
16	2000	21FF		6000	61FF
17	2200	23FF		6200	63FF
18	2400	25FF		6400	65FF
19	2600	27FF		6600	67FF
20	2800	29FF		6800	69FF
21	2A00	2BFF		6A00	6BFF
22	2C00	2DFF		6C00	6DFF
23	2E00	2FFF		6E00	6FFF
24	3000	31FF		7000	71FF
25	3200	33FF		7200	73FF
26	3400	35FF		7400	75FF
27	3600	37FF		7600	77FF
28	3800	39FF		7800	79FF
29	3A00	3BFF		7A00	7BFF
30	3C00	3DFF		7C00	7DFF
31	3E00	3FFF		7E00	7FFF

Table 2 — Memory Mapping for the Block RAM Allocation.

	Memory Read-Only			Memory Write-Only	
Socket #	**Begin**	**End**		**Begin**	**End**
Master	8000	81FF		C000	C1FF
1	8200	83FF		C200	C3FF
2	8400	85FF		C400	C5FF
3	8600	87FF		C600	C7FF
4	8800	89FF		C800	C9FF
5	8A00	8BFF		CA00	CBFF
6	8C00	8DFF		CC00	CDFF
7	8E00	8FFF		CE00	CFFF
8	9000	91FF		D000	D1FF
9	9200	93FF		D200	D3FF
10	9400	95FF		D400	D5FF
11	9600	97FF		D600	D7FF
12	9800	99FF		D800	D9FF
13	9A00	9BFF		DA00	DBFF
14	9C00	9DFF		DC00	DDFF
15	9E00	9FFF		DE00	DFFF
16	A000	A1FF		E000	E1FF
17	A200	A3FF		E200	E3FF
18	A400	A5FF		E400	E5FF
19	A600	A7FF		E600	E7FF
20	A800	A9FF		E800	E9FF
21	AA00	ABFF		EA00	EBFF

22	AC00	ADFF		EC00	EDFF
23	AE00	AFFF		EE00	EFFF
24	B000	B1FF		F000	F1FF
25	B200	B3FF		F200	F3FF
26	B400	B5FF		F400	F5FF
27	B600	B7FF		F600	F7FF
28	B800	B9FF		F800	F9FF
29	BA00	BBFF		FA00	FBFF
30	BC00	BDFF		FC00	FDFF
31	BE00	BFFF		FE00	FFFF

Table 1 and Table 2 show the memory mapping for the 31 hardware sockets in the virtual socket platform. Note in Figure 1 that the memory is allocated into four distinct sections: 1) read-only register file; 2) write-only register file; 3) read-only block RAM; and 4) write-only block RAM. The allocation size for each type of memory for every HW socket is 512 bytes.

6.4 Hardware Accelerator Interface

Figure 3 shows a typical block diagram of a hardware accelerator socket. Note that input and output block RAMs are provided for input and output data while important flags are mapped to the register file sections, such as start and finish flags.

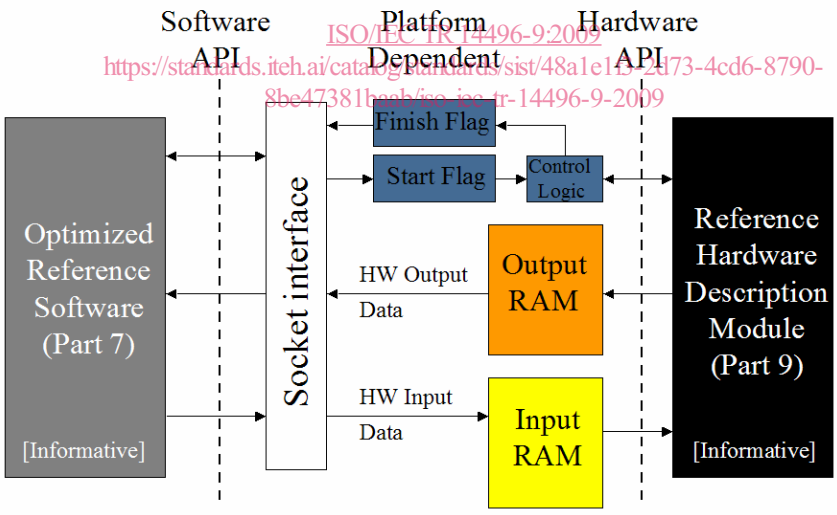

Figure 3 — Block Diagram of Typical Hardware Accelerator.

When a hardware socket is selected for a particular transaction, one of its strobes will be asserted. It is up to the user's particular socket designs whether register or memory regions will be treated differently, however in most cases their behaviour may be identical. The necessary signals to interface to the virtual socket with respect to the hardware accelerator socket are shown in Table 3 below.

Table 3 — Hardware Accelerator Socket Interface.

Signal	Length	Direction*	Polarity	Description
Globals	**<2>**			
Clk	1	Input	R	hardware accelerator socket clock
global_reset	1	Input	H	global reset
Strobes	**<4>**			
strobe_reg_read	1	Input	H	read-only register space selected
strobe_reg_write	1	Input	H	write-only register space selected
strobe_ram_read	1	Input	H	read-only memory space selected
strobe_ram_write	1	Input	H	write-only memory space selected
Write Signals	**<50>**			
write_addr	16	Input		write address
data_in	32	Input		data to write into socket
write_valid	1	Input	H	data_in is valid
write_rdy	1	Output	H	socket available to take more write data
Read Signals	**<49>**			
read_addr	16	Input		read address
data_out	32	Output		data for read operation
strobe_out	1	Output	H	data_out has requested data
External Memory Manager	**<92>**			
ZBT_ReadEmpty	1	Input	H	read fifo is empty
ZBT_WriteFull	1	Input	H	write fifo is full
ZBT_ack_job	1	Input	H	job to memory manager accepted
ZBT_wf_grant	1	Input	H	write fifo access granted
ZBT_rf_grant	1	Input	H	read fifo access granted
ZBT_ReadData	32	Input		data read from external memory
ZBT_issue_job	1	Output	H	issue job to memory manager
ZBT_rwb	1	Output	H	job is read = '1' or write = '0'
ZBT_popfifo	1	Output	H	retrieve word of data from read fifo
ZBT_pushfifo	1	Output	H	place data onto write fifo
ZBT_addr	19	Output		address to access data to in external memory
ZBT_dpush	32	Output		data to send to external memory

The user may optionally connect their device to the external memory manager that allows access to either the ZBT SRAM or DDR DRAM (see subclause 6.4.2). The *block move* and *external block move* example VHDL modules demonstrate the basic interface to the virtual socket (see subclause 6.5). Hardware socket designers are strongly encouraged to read this section and use them as building blocks for their own sockets.

6.4.1 Transferring Data To/From a Socket

When a socket detects a write access to it, it should check to see if the write_valid signal is asserted. This signal will indicate that the data present on Data_In is valid and ready to be processed by the socket. Whenever the user is capable of taking data from the virtual socket interface it should drive its write_rdy signal high. This will bring new data to it from the interface. Register or Memory writes to a socket may be multiple words. The write_rdy signal provides a flow control mechanism back to the virtual socket interface. Below are two waveforms demonstrating example writes to register and memory space.