MPEG-4 AVC – H.264 Motion Estimation IP Datasheet

SUMMARY

1 Introduction

1.1 H.264 Overview

The H.264 is also known as MPEG-4 ISO/IEC14496-10 or MPEG-4 / AVC. This standard has been co-developed by JVT group composed by MPEG-ISO/IEC members and VCEG-ITU-T members.

Three profiles have first been defined, each with several levels. A High Profile has been added, and standardisation work is going on.

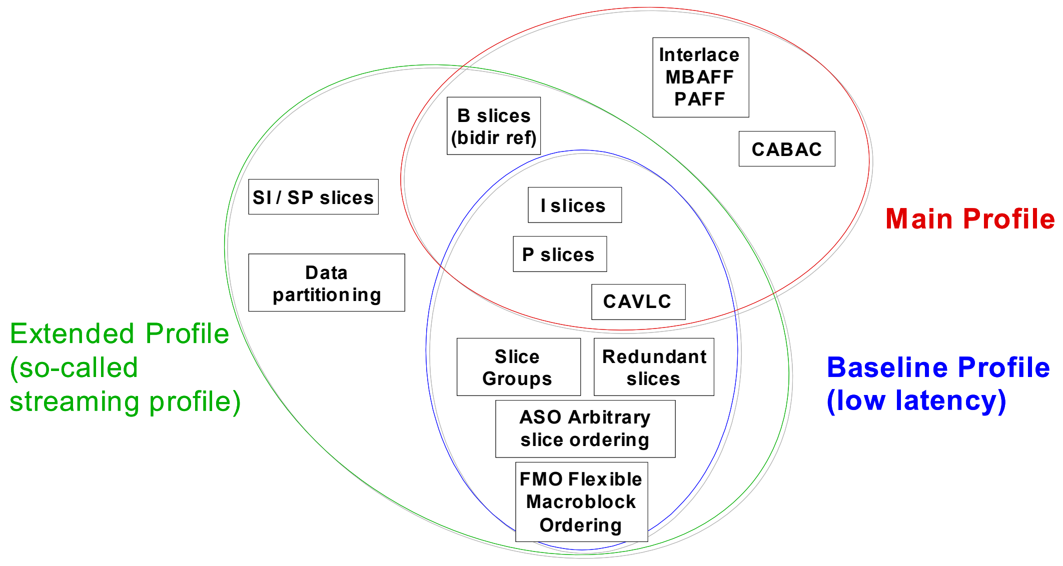


Figure 1- Profiles of MPEG-4 part 10 / AVC H.264

Ateme has developed a full Main Profile Level 1-4 implementation of H.264 encoder and decoder, ensuring Baseline, Extended and High Profiles partial compatibility.

The encoder is available as:

- Software library for x86 (PC),
- VHDL firmware, for embedded equipements.

The decoder is available as software library for x86 and for DSP.

The synoptic for a full encoder is shown below, with the Motion Estimation highlighted.

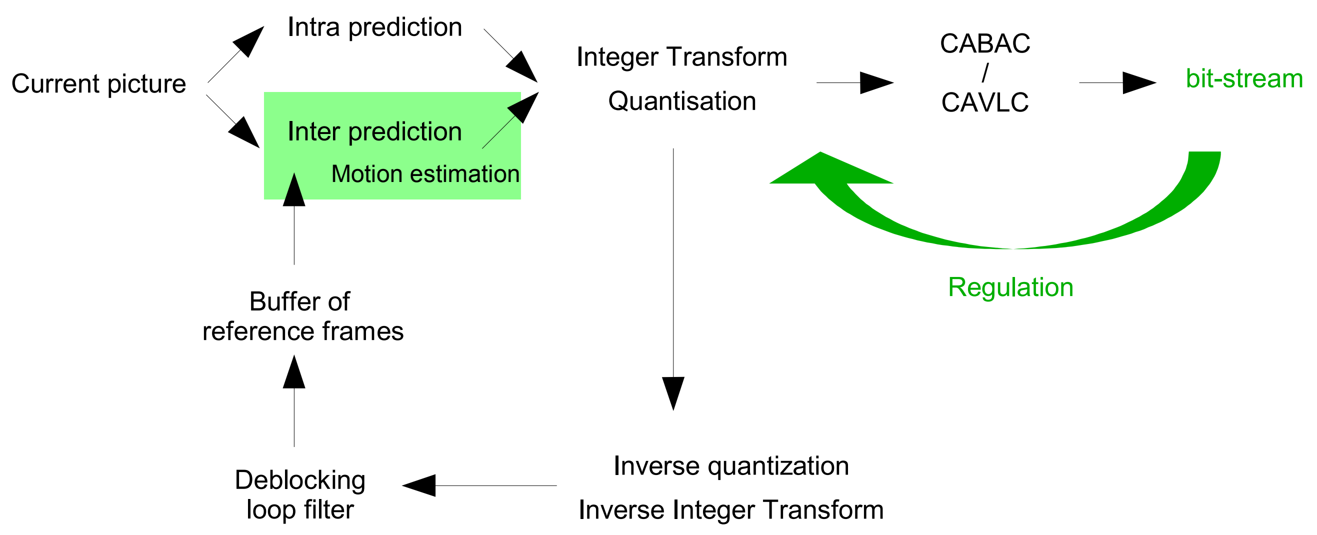


Figure 2- synoptic of MPEG-4 part 10 / AVC H.264 encoder

A video sequence is composed of images. In the H.264 process, an image is split in macroblocks (MB). A macroblock is a 16x16 pixels luminance block and two 8x8 pixels chrominance blocks.

A slice is a collection of macroblocks belonging to a single image.

The H.264 bitstream is structured in NAL units. Each NAL unit contains the information of one slice.

1.2 IP overview

ATEME H.264 Motion-Estimation IP is a hardware module delivering an estimation of the relative motion of each macroblock of a source compared to a reference. The whole picture is processed autonomously by the IP.

This estimation is made in rate-distortion sense using a customizable Lagrangian multiplier.

The neighborhood information necessary for encoding is autonomously managed.

The module is designed to compute full size video sequences. It can be adapted to any format.

1.3 Commercial References

The Motion Estimation IP is available under the following references:

	***SD** – levels 1-3*	***HD** – levels 1-4*
H264_SD-ME	X	
H264_HD-ME	X	Consult us

SD stands for Standard Definition, or Full D1, or 720x480 30 / 29.97 fps in NTSC and 720x576 25 fps in PAL.
This corresponds to a throughput of max 60750 Macroblocks per second.
Level 3 is up to 10 Mbps.

HD stands for High Definition, with 720p 50 / 60, 1080 i 50 / 60, 1080 p 25 / 30.
This corresponds to a throughput of max 367200 Macroblocks per second.

- HD 720p/25-30fps is level 3.1 - up to 14 Mbps
- HD 720p/50-60fps is level 3.2 - up to 20 Mbps
- HD 1080 i&p/25/30fps is level 4 up to 25 Mbps

Other products are available from Ateme:

- CABAC/CAVLC Bitstream encoding IP
- Deblocking filter IP
- Full SD main profile Level 3 encoder IP,
  including video acquisition, decision, regulation, memory management, PCI, etc...

2 Features

- Platform independent design (written in VHDL)
- Various compilation options to adapt to specific needs (bus width, max image size...)
- Motion estimation of all images macroblocks
- Horizontal vectors up to ±112 pixels, vertical vectors up to ±48 pixels
  with 32kB of internal RAM.
- MBAFF support (planned)
- Autonomous context management
- Customizable scalar coefficient
- PAL and NTSC support, and other formats on demand

3 Performances

High throughput performances are obtained with low clock frequency.

In the worst case, the IP needs 434 clock cycles per macroblock.

A clock frequency of 100 MHz leads to 4.34 µs/macroblock, or 230414 macroblocks/s.

A 720x480 image is processed in 5.8ms.

4 Resources

The IP has been optimized, thus for information on footprint and size, Logic Element number, and memory usage, please consult us for your specific target.

5 Architecture

5.1 Principle

The IP uses a generic bus interface for registers and streams.
It is delivered encapsulated with two AHB agents:

1. AHB Slave for registers access
2. AHB Master to access directly data in external memory

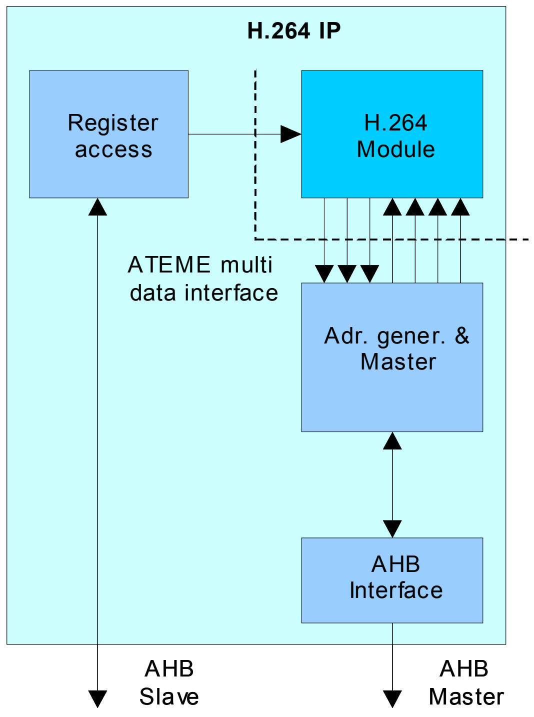


Figure 3- CPU or AHB bus interfaces

The three associated modules:

- Register access,
- Address generation and management of data accesses,
- Master AHB interface,

are delivered in source code with the H.264 module.
Thus the IP can be used either directly, for example by a CPU,
or with all associated modules in a SoC design.

5.2 Hardware interface

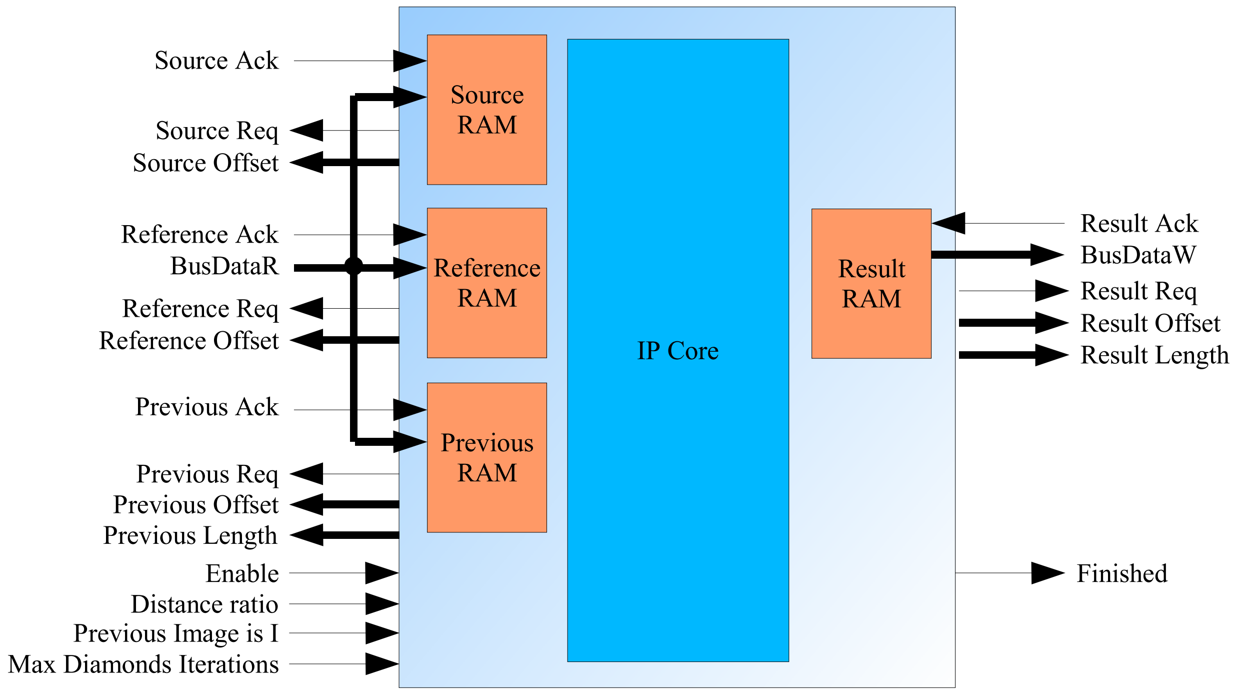


Figure 4- Hardware Interfaces

The IP has a memory for each type of data. They are seen as FIFOs by the system. When the system receives a request, it must fill / read the corresponding memory by the appropriate amount of data *(refer to software interface)*.

The above table lists the IP ports.

Signal name	Dir.	Active	Description
Rstn	I	0	Reset signal
Enable	I	1	Enables processing and buffer requests to be emitted
Clk	I	↑	IP internal clock
PreviousImageIsI	I	1	Flag indicating that the previous image was I.
DistancesRatio	I	-	d(previous-reference)/d(current-reference) in 4.12 format
MaxDiamondIterations	I	-	Maximum number of diamond iterations
ImageWidth	I	-	Width of the image minus 1
ImageHeight	I	-	Height of the image minus 1
ProcessEnd	O	1	Pulse to indicate that the image was processed
BusClk	I	↑	Bus clock
BusDataR	I	-	Read data bus
BusDataW	O	-	Write data bus
SourceReq	O	1	Pulse to request source data
SourceOffsetReq	O	-	Offset of the requested source macroblock
SourceAck	I	1	Write command for source data
ReferenceReq	O	1	Pulse to request reference data
ReferenceOffsetReq	O	-	Offset of the requested reference macroblock
ReferenceAck	I	1	Write command for reference data
PreviousReq	O	1	Pulse to request previous image vectors
PreviousOffsetReq	O	-	Offset of the requested previous line of vectors of previous image
PreviousLengthReq	O	-	Length in int16 of the requested previous image vectors
PreviousAck	I	1	Write command for previous image vectors
ResultReq	O	1	Pulse to request the emission of resulting data
ResultLengthReq	O	-	Length in int32 of the requested result transfer
ResultOffsetReq	O	-	Offset of the requested source macroblock
ResultAck	I	1	Read command for resulting data
LambdaAdd	I	-	Address of the λ table, corresponding to the Qp value
LambdaW	I	1	Write command for the λ table
LambdaDataW	I	-	λ value

xxxxOffsetReq and xxxxLengthReq are valid during the xxxxReq pulse. They provide additional information to the system to which data is transferred.

SourceOffsetReq & ReferenceOffsetReq are the concatenations of macroblock column & macroblock line numbers.

PreviousOffsetReq & ResultOffsetReq are macroblock line numbers.

PreviousLengthReq & ResultLengthReq are numbers of macroblocks.

SourceAck, ReferenceAck & PreviousAck are write signals for their respective memories.

ResultAck is a read acknowledge signal for its memory, meaning that next data will be presented on the cycle following the assertion of ResultAck.

5.3 *Software interface*

Little endian byte arrangement must be used.

Four structures of data are required :

- SourceInfo : Input structure for the Mb of current image,
- ReferenceInfo : input structure for the Mb in reference frame
- PreviousInfo : input structure for the horizontal + vertical vectors of the Mb in previous image
- ResultInfo : Output structure containing resulting horizontal and vertical vector, plus cost and SAD (Sum of absolut differences). The resulting vector is the one optimizing the SAD cost. It defines a movement between the source and he reference for this Mb.

SourceInfo structure description

```
typedef struct {
Uint*   Qp;                          // Macroblock Qp
Uint8   uCoeffL[256];                // Luma coefficients
} SourceInfo;
```

* Qp fits in a byte. To simplify the access, Qp is extended to the width of the read data bus. The position of the byte in the word can be defined in the compilation parameters.

ReferenceInfo structure description

```
typedef struct {
Uint8   uCoeffL[256];                // Luma coefficients
} ReferenceInfo;
```

PreviousInfo structure description

```
typedef struct {
int8      sHvect;                    // Horizontal vector of a macroblock in the previous image
int8      sVvect;                    // Vertical vector of a macroblock in the previous image
} PrevData;

typedef struct {
PrevData  sPreviousData[ImageWidth]
} PreviousInfo;
```

ResultInfo structure description

```
typedef struct {
int8      sHvect;                    // Horizontal resulting vector
int8      sVvect;                    // Vertical resulting vector
Uint16 uSadCost;                     // Sum of SAD and cost clipped to 0xFFFF
} ResData;

typedef struct {
ResData  sResultData[ImageWidth]
} ResultInfo;
```

5.4 System integration

5.4.1 Integration with CPU bus

A typical multi-master system integration of the IP is represented on the figure below.

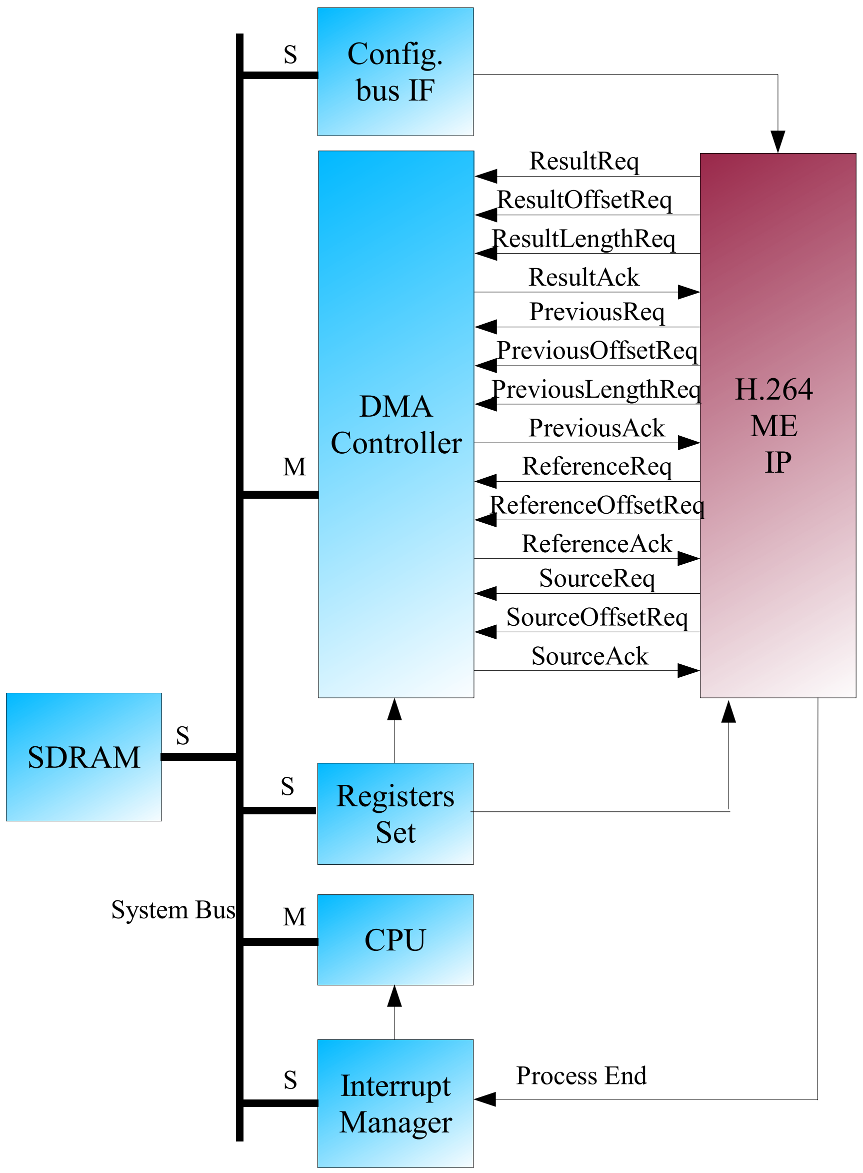


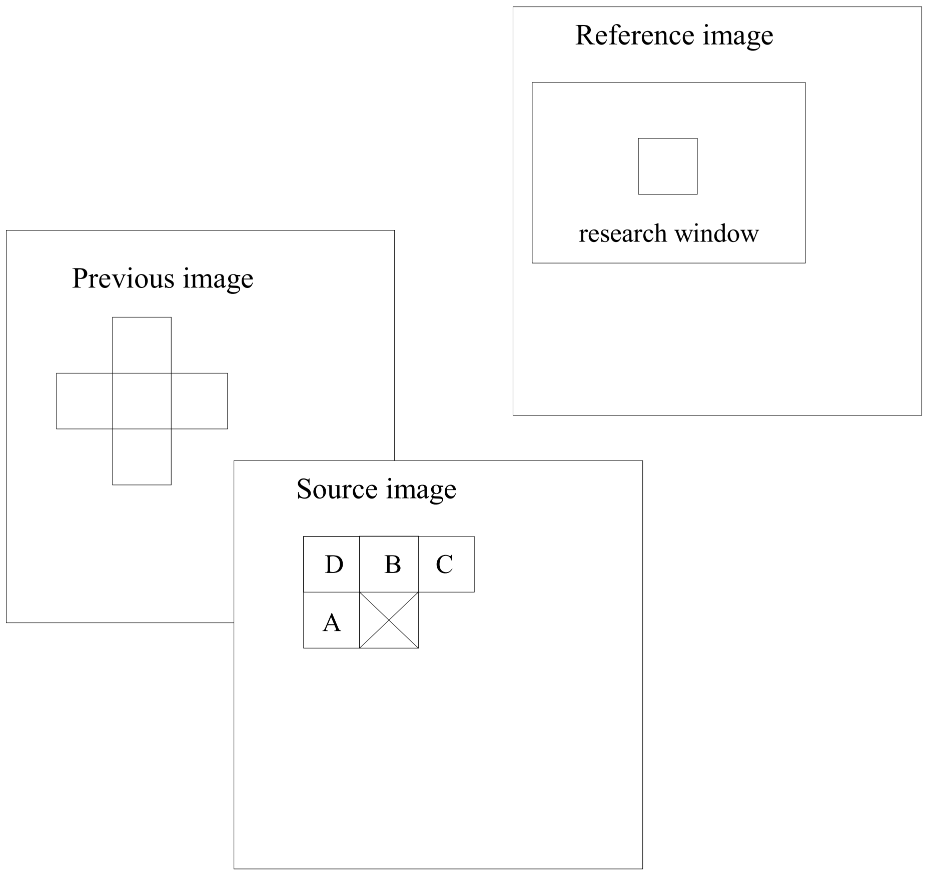
Figure 5- CPU bus system integration

A CPU manages the system. It is in charge of memory allocation, DMA controllers programming, and computation enabling. The DMA does arbitration between the different channels.

5.4.2 integration with AHB bus

The IP is simply configured by registers, then it works in an autonomous way, with Master data acces in external memory, for input and output streams.

5.5 Operating mode



For a given macroblock in the source, the IP optimizes the R-D cost:

$$J = Cost_{vector} + \sum \left| SrcPix - RefPix \right|$$

to select the best candidate position in the research window. The candidates are :

- The four neighbors A, B, C, D vectors
- The median vector of A, B & C, or the median of A,B & D if C does not exist.
- The null vector
- The vector of the corresponding macroblock in the previous image and those of its four neighbors

The vector cost is evaluated in CAVLC.
The Lagrangian multiplier takes automatically into account the quantizer of the macroblock.

The best position is then used as a start point for diamonds iterations. This consists in testing the R-D cost of the four moves by one (left top, right & bottom). If one of them gives a smaller cost, it is chosen as a the new start point. Else, the process stops. The MaxDiamondIterations parameter limits this number of iterations.

This process is a first rough Motion estimation, or pre-estimation. A fine motion estimation is done in the decision module of the complete encoder.

6 Deliverables

The deliverables includes:

- Bitstream IP VHDL source code
- User's manual
- Simulation files
- Reference software on PC (windows)

The reference software encodes a video file, and delivers output files, corresponding to the input and output of the IP. This enables to insert known data into the IP, and to compare the IP output with the software output.

An evaluation version of the IP is available.

As option, a PCI board with Altera Stratix II FPGA will soon be available, to ease real-time benches.