H.264 Motion Estimation Engine v1.0

User Guide

UG453 (v1.1) April 23, 2008

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/24/07	1.0	Initial Xilinx release.
04/23/08	1.1	Updated “Running the Test Bench” in Chapter 4.

Table of Contents

Appendix: Supporting Information

Schedule of Figures

Schedule of Tables

About This Guide

This document is intended to guide the user through all aspects of installation, demonstration, simulation, verification, and general usage of the Motion Estimation Engine core. It should be read in conjunction with the Xilinx *Motion Estimation Engine Product Specification* (DS648).

Guide Contents

This manual contains the following chapters:

- "Introduction," provides information about the core, recommended design experience, additional core resources, technical support, and core and document feedback.
- "Installing the H.264 Motion Estimation Engine Core," provides instructions for installing the core.
- "Designing with the H.264 Motion Estimation Engine Core," describes how to include the core into the designer's system, along with a list of the files provided.
- "Simulating the H. 264 Motion Estimation Engine Core," describes a test bench architecture that was created in the ModelSim® environment for simulation of the core.
- "Verifying the System,"describes the verification environment delivered with the core.
- Appendix , "Supporting Information,"provides the input sequences, directory tree structure, regression test summary, and references.

Additional Resources

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/literature.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*
Helvetica bold	Commands that you select from a menu	**File → Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *design_name*
	References to other manuals	See the *Development System Reference Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Square brackets `[ ]`	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[`*`option_name`*`]` *`design_name`*
Braces `{ }`	A list of items from which you must choose one or more	**`lowpwr =`**`{`**`on`**`|`**`off`**`}`
Vertical bar `|`	Separates items in a list of choices	**`lowpwr =`**`{`**`on`**`|`**`off`**`}`

Vertical ellipsis
.
.
.
	Repetitive material that has been omitted	`IOB #1: Name = QOUT'`
		`IOB #2: Name = CLKIN'`
		.
		.
		.

Horizontal ellipsis `...`	Repetitive material that has been omitted	**`allow block`** *block_name loc1 loc2 ... locn;*

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.

Convention	Meaning or Use	Example
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-II Platform FPGA User Guide.*
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Chapter 1

Introduction

This user guide is required reading for the engineer using or considering using the Xilinx® Motion Estimation Engine core. The release of this product is given in ZIP file form.

About the Core

The Xilinx Motion Estimation Engine core accepts input parameters on a frame and macroblock basis and a stream of pixels in macroblock format and generates output motion vectors, Sum-of-Absolute Difference (SAD) values, and coded block pattern with best motion vector for each block.

Recommended Design Experience

Although the Motion Estimation Engine core is a fully-verified solution, the challenge associated with implementing a complete design varies depending on the configuration and functionality of the application. For best results, previous experience building high-performance pipelined FPGA designs using Xilinx implementation software and user constraints files (UCF) is recommended.

Contact your local Xilinx representative for a closer review and estimation for your specific site requirements.

Additional Core Resources

For detailed information and updates about the Motion Estimation Engine core, see the documents located on the Motion Estimation Engine product page. In general, this document should always be used in conjunction with the following:

- The Xilinx Motion Estimation Engine Product Specification (DS648)
- The MPEG4 Part 10 specification ([Ref 1])
- JM10.2 H.264 Codec Reference C Code

Technical Support

For technical support, go to www.xilinx.com/support. Questions are routed to a team of engineers with expertise using the Motion Estimation Engine core.

Xilinx provides technical support for use of this product as described in this guide.

Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

Feedback

Xilinx welcomes comments and suggestions about the Motion Estimation Engine core and the accompanying documentation. For comments or suggestions about the Motion Estimation Engine core, submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

- Product name
- Core version number
- Explanation of your comments

For comments or suggestions about this document, submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments

Chapter 2

Installing the H.264 Motion Estimation Engine Core

This chapter provides instructions for installing the H. 264 Motion Estimation Engine core.

The Motion Estimation Engine fixed netlist is provided under the Xilinx LogiCORE™ Site License Agreement, which conforms to the terms of the SignOnce IP Site License standard defined by the Common License Consortium.

Install the core by performing a manual installation after downloading the core from the web.

System Requirements

Windows

- Windows® 2000 Professional with Service Pack 2 or greater
- Windows XP Professional Service Pack 2 or greater

Software

- Xilinx ISE™ 9.1i with Service Pack 3
- ModelSim® 6.1c SE
- MicroSoft Visual C++ 6.0
- ActivePerl 5.8.3

Manual Installation

1. Download the IP Update ZIP file from the following location and save it to a temporary directory:

   https://secure.xilinx.com/webreg/register.do?group=h264_me

   Prompted to enter a login name and password, enter your Xilinx login name and password.
2. If you are new to Xilinx, click Create an Account and follow the instructions.
3. Three ZIP files are given. Unzip them all **into the same directory.** They are as follows (x_y is the release version number, e.g., 1_0).
   a. `H264_MotionEstimation_verx_y_utils.zip` – unzip this first.
   b. `H.264_MotionEstimation_verx_y_ReleaseNetlists.zip`
   c. `H264_MotionEstimation_verx_y_InputSequences.zip`

4. Allow the extractor utility you use to overwrite all existing files and maintain the directory structure defined in the archive. See "Appendix , "Supporting Information"" for an illustration of the correct directory tree structure.

Netlists

The netlists given in this release of the H. 264 Motion Estimation Engine are all in the `\ReleasedNetlists` directory. They have been synthesized using Synplify_Pro 8.8.0.4 for Spartan™-3A (S3A), Virtex™-4 (V4), and Virtex-5 (V5) FPGAs:

- `MotionEstimation_S3A.edf`
- `MotionEstimation_V4.edf`
- `MotionEstimation_V5.edf`

To learn how to instantiate the cores in your system, refer to Chapter 3, "Designing with the H.264 Motion Estimation Engine Core."

Chapter 3

Designing with the H.264 Motion Estimation Engine Core

This chapter describes how to include a Motion Estimation Engine core into the next hierarchy of system architecture.

VHDL Template Files

To help the user design the core into his system, the release provides the following template files. Neither the core source code nor the HDL libraries have been provided in source form but in object code format for simulation purposes.

Note: **These VHDL template files are for reference only.**

1. Instantiation template code:

`/HDL/LowCost_MotionEstimation/MotionEstimation/src/MotionEstimation.vho`

2. Model Tech VHDL simulator test bench code:

`/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/MotionEstimation_TB.vho`

Motion Estimation in the H.264 Encoder

Figure 3-1 shows the Motion Estimator in a typical implementation of an encoder. For more information on the system-level integration, refer to the H.264 specification [Ref 1].

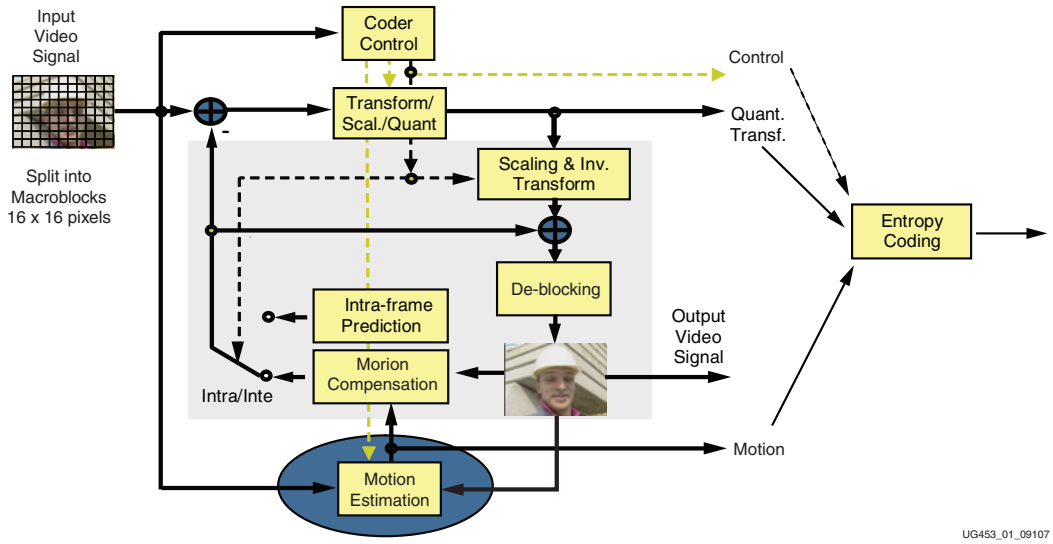

Figure 3-1: **H.264 Encoder**

Motion estimation requires a definition of a search region and a search scheme to find the best match (motion vector) within a search region. Rate-Distortion theory also takes into consideration vectors that may be close together, but not necessarily the best SAD value in finding the optimal motion vector to choose for entropy coding. Searching within the search region can be done exhaustively, but the computational overhead is prohibitive. Thus, a search scheme with seed motion vectors and regions to search are desired. In this implementation of the motion estimation, 10 seed vectors are provided for each 8x4 block with eight 8x4 blocks in a 16x16 macroblock. A 4x3 region is searched to the right and down from the seed vector.

Another difficult part of motion estimation is the localization of memory to allow for easy access to the search region. In this implementation, a sliding window of eight macroblocks high and seven macroblocks wide is utilized to keep the external memory transfers efficient and the bandwidth manageable.

Processing begins with frame parameters being updated each frame and the first macroblock being sent to the core. Macroblock parameters in the form of seed vectors (80) are send to the core and the H.264 Motion Estimation Engine processes and produces SADs and the list of Motion Vectors searched corresponding to the SADs produced. See Figure 3-2.

H.264 allows for variable block sizes within a macroblock. If through the search process the best motion vector (minimum SAD) for two blocks is identical, then the block can be merged to be a larger block, that is, two 8x4 blocks to an 8x8 block. The Variable Block Size Decision block performs this processing and provides as an output the coded block pattern for a macroblock and corresponding best motion vectors and SADs.

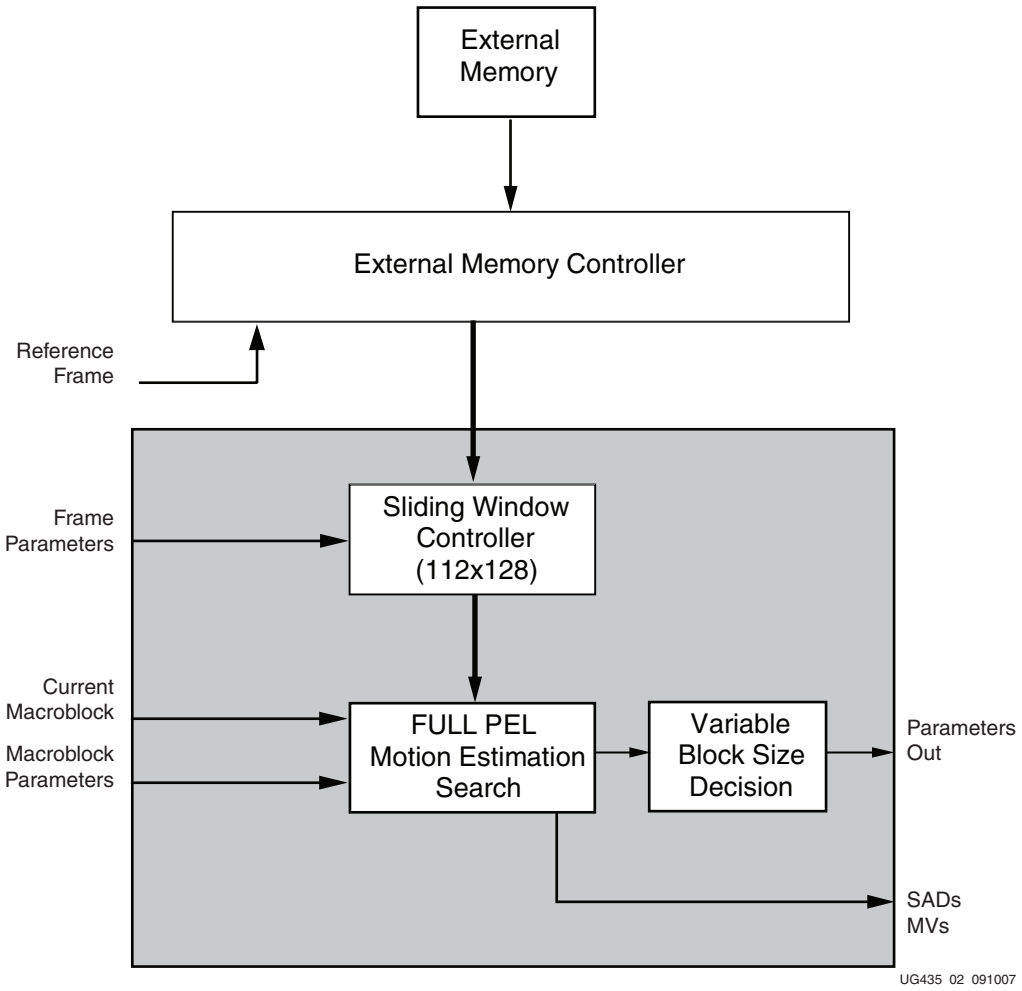

Figure 3-2: **Motion Estimation Engine Block Diagram**

XILINX®

Chapter 4

Simulating the H. 264 Motion Estimation Engine Core

This chapter describes a test bench architecture that was created in the ModelSim environment for simulation of the Motion Estimation Engine core. It does **not** describe the verification platform. The simulation described may be used for visualization of the I/O signals at the periphery of the cores. Although the same precompiled libraries are used in the verification of the Motion Estimation Engine core, this process is described in detail in Chapter 5, "Verifying the System."

Test Bench Release

For the purposes of running, viewing, and understanding the test bench and the architectures used, the release provides the following files under `\HDL\LowCost_MotionEstimation\MotionEstimation`:

1. Libraries of object (precompiled) source:

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/work/`

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/memxlib/`

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/Sim_tools/`

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/unisim/`

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/mig_ddr2_s3a_v1_06_a/`

2. Test bench stimulus files:

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/stimuli/ (input stimulus files)`

3. ModelSim-specific script files:

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/MotionEstimation_prim.do`

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/MotionEstimation_core.do`

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/MotionEstimation_user.do`

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/vsim_gui.bat`

4. Testbench source file:

   `/HDL/LowCost_MotionEstimation/MotionEstimation/Testbench/MotionEstimation_TB.vhd`

Running the Test Bench

To run the test bench:

1. Double-click on the `vsim_gui.bat` file. This spawns the ModelSim GUI. Two wave windows are given. One (`MotionEstimation_user`) is meant to contain user-defined signals. The user may use this window to view any internal signals of his choice. The other (`MotionEstimation_core`) shows the signals at the periphery of the core to give the user a feel for the typical interfacing activity required.

   If the the following error is received, the user needs to run the refresh command, **vcom -refresh**:

   ```
   #**Error: (vsim-13) Recompile work.vfbc_infrastructure because
   C:\opt\Modeltech_6.1f\xilinx_libs_9_2_03i_ip2\.\unisims.vcomponents
   has changed.
   ```

2. In the ModelSim environment, enter **run -all**. This runs for about 15 minutes.

   The stimulus data for this simulation is held in the `\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\stimuli` directory, consisting of several input text files and some expected output text files. The stimulus data files and expected output files provided contain data extracted from the reference C model (see [Ref 1]).

The simulation generates these output files:

`\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\MotionEstimation_MVs.out.txt`

`\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\MotionEstimation_ParamsMB.out.txt`

`\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\MotionEstimation_SADs.out.txt`

that are directly comparable to the reference expected output.

`\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\stimuli\MotionEstimation_MVs.out.txt`

`\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\stimuli\MotionEstimation_ParamsMB.out.txt`

`\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\stimuli\MotionEstimation_SADs.out.txt`

Note: Xilinx has provided a `.vho` source file as an example of how to instantiate the Motion Estimation Engine core. It is **not** provided as compilable source code for ModelSim. All simulations are to be run using the precompiled libraries provided.

In Chapter 5, "Verifying the System," of this document, the process by which stimulus and expected results are generated from the reference code is described. The generated files may be used in place of the default simulation files provided with the release, but must reside in the locations mentioned above during simulation.

XILINX®

Chapter 5

Verifying the System

This chapter describes the verification environment delivered with the Motion Estimation Engine core. Ultimately, the system is verified by using long regression tests with several different resolution input sequences. The output of the hardware from these tests must exactly match the output given by the reference software that runs with the same stimulus.

Verification Platform Release

For the purposes of running, viewing, and understanding the verification process, the release provides the following files:

1. Libraries of object (precompiled) source (same as Simulation):

   `\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\work\`

   `\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\memxlib\`

   `\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\Sim_tools\`

   `\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\unisim\`

   `\HDL\LowCost_MotionEstimation\MotionEstimation\Testbench\mig_ddr2_s3a_v1_06_a\`

2. Source video sequences:

   `\InputSequences\`

See the "Appendix, "Supporting Information" for the full file list.

3. Verification scripts:

   `\Verification\MotionEstimation\MotionEstimation_Verification.pl`

   `\Verification\MotionEstimation\MotionEstimation_Verification_Level1.bat`

4. Testbench support module:

   `\TestBenchSupport\TestBenchSupport.pm`

5. Software reference code executables:

   `\Software\ArchC_Rev3\bin\lencod.exe`

Running the Verification Tests

There are five tests that can be run by the user. Each test has varying characteristics, including varying video formats, parameter settings, etc. A description of the tests is given in the Appendix, "Supporting Information" and is also summarized in the batch scripts listed under item 3 (Verification scripts) above.

Verification is automated down to running the simple editable `Level1` batch script. It contains the command line:

```
perl -I"..\..\TestBenchSupport" MotionEstimation_Verification.pl
[TestLevel][No. Frames][Test#]
```

`TestLevel:` `1` (only 1 is available)

`No. Frames:` `between 1 and 10 inclusive`

`Test#:` `between 0 and 5 inclusive` (0 runs all tests in order from 1 to 5)

Edit the file as desired. Double clicking on the batch file invokes the appropriate test to be executed. Level 1 simulation is summarized in Table 5-1.

Table 5-1: **Verifcation Level Summary**

Verification Level	Reference Executable	HW Representation	Notes
Level 1	`ArchC_rev3\bin\lencod.exe`	MTI Simulation (precompiled libraries)	• **Reference** – uses structurally modified reference code to generate stimulus and expected Motion Estimation Engine outputs for verification at Motion Estimation level. • **Unit under test** – uses same precompiled object code libraries as simulation. Uses stimulus files generated above as simulation input stimulus. • Detailed – used for debugging. • RTL simulation, thus, only a couple of frames per hour, depending on the frame size.

Verification Process (Level 1)

Table 5-1 shows the verification process (Level 1).

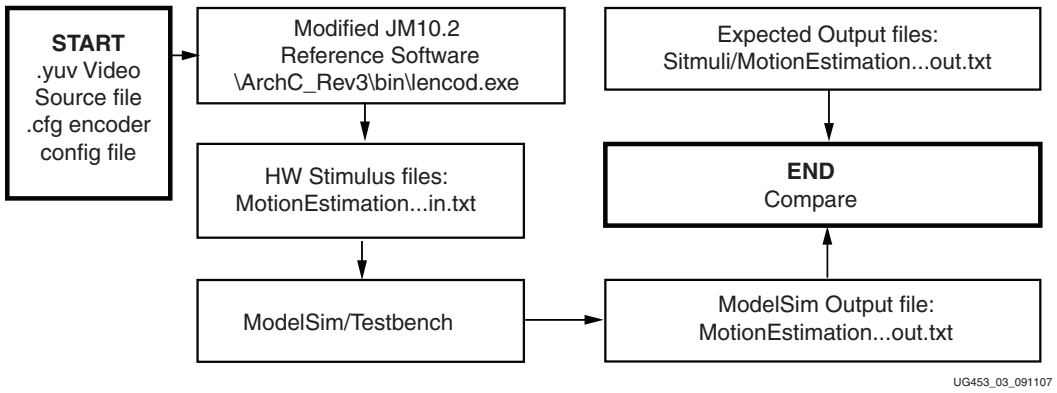

Figure 5-1: **Verification Process (Level 1)**

The command line in the batch script invokes the following processes in this order:

1. A DOS window is spawned.
2. A Perl script (`MotionEstimation_Verification.pl`) is called, which sets up the required test(s) in order with the appropriate parameters.

   This Perl script calls functions in the general-purpose test bench support module that runs the reference code:

   `\Software\ArchC_Rev3\bin\lencod.exe`
3. The executable uses the video sequences in `\InputSequences` as input.
4. This code generates the stimulus files:

   `\Verification\MotionEstimation\Level1\Testxx\stimuli\MotionEstimation_...in.txt`

   …and the expected output reference file:

   `\Verification\MotionEstimation\Level1\Testxx\MotionEstimation_...out.x`
5. ModelSim is then called in batch mode (no GUI is invoked with the release) which generates the HW simulation output file:

   `\Verification\MotionEstimation\Level1\Testxx\MotionEstimation_...out.txt`
6. This file is then compared to the reference output file generated previously and a DOS report given. The results are also summarized in:

   `\Verification\MotionEstimation\EncoderVerification_Summary_Level1.txt`

   Note: The whole process takes some time. For QCIF (176 x 144), it should take at least a 15 minutes to run two frames on a modern laptop. This time increases for tests that bring extra complexity into the stimulus generation, for example, formats with increased frame size, main and high profile settings, etc.

Verification Notes

When test number is set to 0 in the batch script file, all tests are run in sequence from 1 to 10. When doing this, by default, all tests are run regardless of whether they have been run in the past. If some tests have already been run, but some others have not been run or their results deleted, then the user may wish to run only the remaining tests. The user can do this by editing the `MotionEstimation_Verification.pl` script, commenting out the line:

```
$ForceRegenerateAll = "1";
```

Use '#' to comment).

Also, if you want to take this approach, but rerun one or some tests selectively, delete the test directory of the test(s) you want to rerun:

MotionEstimation\Level1\testxx

and rerun the script.

XILINX®

Appendix

Supporting Information

Input Sequences

The files provided as video input source files are:

```
\InputSequences\city_4cif_30\city_4cif_30.hdr
\InputSequences\city_4cif_30\city_4cif_30.yuv
\InputSequences\football_cif_30\football_cif_30.hdr
\InputSequences\football_cif_30\football_cif_30.yuv
\InputSequences\foreman_qcif_30\foreman_qcif_30.hdr
\InputSequences\foreman_qcif_30\foreman_qcif_30.yuv
\InputSequences\shields_720p_60\shields_720p_60.hdr
\InputSequences\shields_720p_60\shields_720p_60.yuv
\InputSequences\tractor_1088p_30\tractor_1088p_30.hdr
\InputSequences\tractor_1088p_30\tractor_1088p_30.yuv
```

All sequences are 10 frames long.

Directory Tree Structure

Figure A-1 shows the structure of the Motion Estimation directories after unzipping the three ZIP files.

Regression Test Descriptions

Table A-1 describes the regression tests used by showing the main digressions from the default settings. See the generated JM code configuration file for more information.

Table A-1: **Regression Test Summary**

Test Number	Stream	Image size	Test Specifics	Details
1	foreman_qcif_30	176x144	Baseline	1 Reference Frame
2	football_cif_30	352x288	Baseline	1 Reference Frame
3	city_4cif_30	704x576	Baseline	1 Reference Frame
4	shields_720p_60	1280x720	Baseline	1 Reference Frame
5	tractor_1088p_30	1920x1080	Baseline	1 Reference Frame

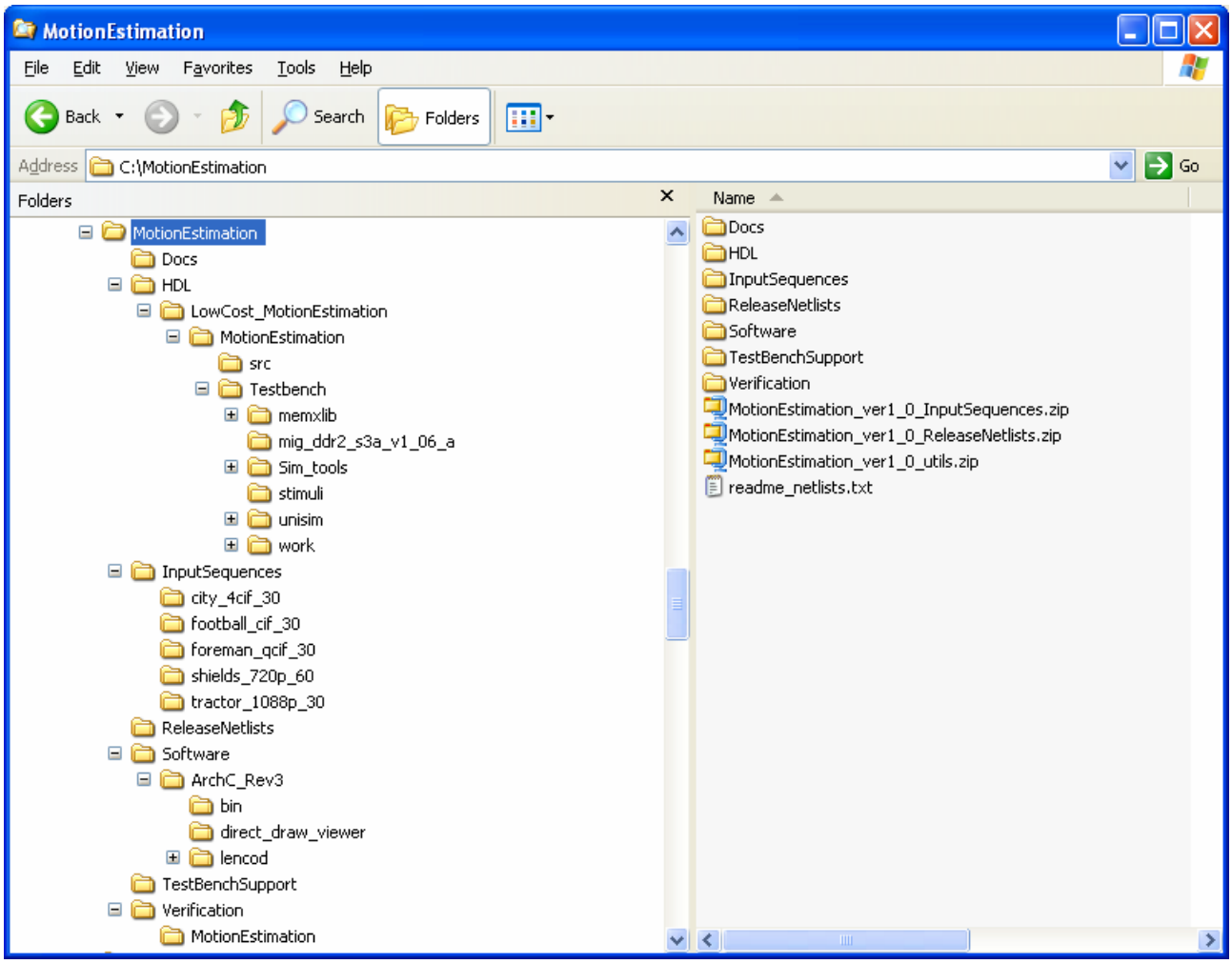

Figure A-1: **Directory Tree Structure**

References

1. ITU-T/ISO/IEC, *Advanced Video Coding for Generic Audio Visual Services*, H.264 03/2005.