# MPEG-4 Part 2 Decoder Demonstration

## *User Guide*

UG234 (v1.1) March 15, 2006

XILINX®

www.BDTIC.com/XILINX

Xilinx is disclosing this Document and Intellectual Property (hereinafter "the Design") to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED "AS IS" WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems ("High-Risk Applications"). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

© 2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

# Revision History

## MPEG-4 Part 2 Decoder Demonstration
## UG234 (v1.1) March 15, 2006

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 02/09/06 | 1.0 | Initial Xilinx release. |
| 03/15/06 | 1.1 | Minor edits on pp. 18, 26, and 27. |


www.BDTIC.com/XILINX

# Contents


## Preface: About This Guide

**Guide Contents** . . . . . 9
**Additional Resources** . . . . . 9
**Conventions** . . . . . 10
Typographical . . . . . 10
Online Document . . . . . 11

## Chapter 1: Introduction

**System Overview** . . . . . 13
**ML402 Board** . . . . . 15
**VIODC Board** . . . . . 15

## Chapter 2: Theory of Operation

**System Overview** . . . . . 17
**MicroBlaze Subsystem** . . . . . 18
Reading a Compressed Bitstream from Compact Flash . . . . . 18
Compressed Bitstream Input to MPEG-4 Decoder Core . . . . . 18
MicroBlaze Interface to DDR Memory . . . . . 18
Reading Status and Writing Control Registers . . . . . 19
**MPEG-4 Part 2 Decoder Core** . . . . . 19
Decoder External Memory Connection . . . . . 19
Decoder Output . . . . . 19
Decoder Output Storage in DDR Memory . . . . . 21
**Macroblock to Raster to Memory Controller** . . . . . 21
**DDR Memory Controller** . . . . . 22
**VGA Controller and Color Space Converter** . . . . . 23

## Chapter 3: Source Code, Project Files, and EDIF Netlist

**Overview** . . . . . 25
**MPEG-4 Part 2 Decoder Demonstration Implementation** . . . . . 27
Loading the EDK Project . . . . . 27
Recompiling One Step at a Time . . . . . 27
Hardware Component Processing . . . . . 28
Software Component Processing . . . . . 28
Synplicity Synthesis . . . . . 28
ModelSim Simulation . . . . . 28



www.BDTIC.com/XILINX

www.BDTIC.com/XILINX

# Schedule of Figures

## Chapter 1: Introduction

*Figure 1-1:* **System Overview** . . . . . 14
*Figure 1-2:* **ML402 Evaluation Platform Simplified Block Diagram** . . . . . 15
*Figure 1-3:* **VIODC Basic Block Diagram** . . . . . 16

## Chapter 2: Theory of Operation

*Figure 2-1:* **Block of Pixels** . . . . . 20
*Figure 2-2:* **Macroblock (Six Blocks of Pixels) 4:2:0 YUV** . . . . . 20
*Figure 2-3:* **Conversion from Macroblock YUV 4:2:0 to YUV 4:4:4** . . . . . 20

## Chapter 3: Source Code, Project Files, and EDIF Netlist

*Figure 3-1:* **Decoder Output-to-Video Driver Source Directory Structure** . . . . . 26


www.BDTIC.com/XILINX

www.BDTIC.com/XILINX

# Schedule of Tables

## Chapter 1: Introduction

## Chapter 2: Theory of Operation

*Table 2-1:* **DDR Memory Controller Command Format** . . . . . 19
*Table 2-2:* **Status and Control Register Definitions** . . . . . 19
*Table 2-3:* **Pixel Storage Format, YUV, 4:2:2** . . . . . 21
*Table 2-4:* **Memory Controller Ports** . . . . . 22

## Chapter 3: Source Code, Project Files, and EDIF Netlist

*Table 3-1:* **System Source Code** . . . . . 27


www.BDTIC.com/XILINX

www.BDTIC.com/XILINX

# About This Guide

This document details the implementation and use of the MPEG-4 Part2 Decoder Demonstration system that is included with the Video Starter Kit. The design can be used as a demonstration of an FPGA performing video decompression or as a development platform for those interested in building their own systems.

## Guide Contents

This manual contains the following chapters:

- Chapter 1, "Introduction," provides an overview of the MPEG-4 decoder demonstration system, the FPGA hardware evaluation platform, Xilinx IP cores and embedded software.
- Chapter 2, "Theory of Operation," describes the interconnection of the primary elements of the system and how these elements interact to perform as a system.
- Chapter 3, "Source Code, Project Files, and EDIF Netlist,"provides information on source code, project files, and EDIF netlist that are available for all elements of the system.

## Additional Resources

For additional information, go to http://support.xilinx.com. The following table lists some of the resources you can access from this website. You can also directly access these resources using the provided URLs.

| Resource | Description/URL |
|---|---|
| Tutorials | Tutorials covering Xilinx design flows, from design entry to verification and debugging<br>http://support.xilinx.com/support/techsup/tutorials/index.htm |
| Answer Browser | Database of Xilinx solution records<br>http://support.xilinx.com/xlnx/xil_ans_browser.jsp |
| Application Notes | Descriptions of device-specific design techniques and approaches<br>http://support.xilinx.com/apps/appsweb.htm |
| Data Sheets | Device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging<br>http://support.xilinx.com/xlnx/xweb/xil_publications_index.jsp |


www.BDTIC.com/XILINX

| Resource | Description/URL |
|---|---|
| Problem Solvers | Interactive tools that allow you to troubleshoot your design issues<br>http://support.xilinx.com/support/troubleshoot/psolvers.htm |
| Tech Tips | Latest news, design tips, and patch information for the Xilinx design environment<br>http://www.support.xilinx.com/xlnx/xil_tt_home.jsp |

# 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`**`}` |


www.BDTIC.com/XILINX

| Convention | Meaning or Use | Example |
|---|---|---|
| Vertical ellipsis<br>.<br>.<br>. | Repetitive material that has been omitted | `IOB #1: Name = QOUT'`<br>`IOB #2: Name = CLKIN'`<br>.<br>.<br>. |
| 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.<br>Refer to "Title Formats" in Chapter 1 for details. |
| Blue, underlined text | Hyperlink to a website (URL) | Go to http://www.xilinx.com for the latest speed files. |


www.BDTIC.com/XILINX

www.BDTIC.com/XILINX

Chapter 1

# Introduction

This document details the implementation and use of the MPEG-4 Part2 Decoder Demonstration system that is included with the Video Starter Kit. The purpose of the designs is twofold:

- As a demonstration of an FPGA performing video decompression
- As a development platform for those interested in building their own systems.

As a demonstration system, the accompanying Compact Flash card is used to hold a single compressed video stream and FPGA configuration bitstreams that perform the decompression and drive the video display. An embedded processor within the FPGA reads the bitstream from the Compact Flash card and sends it to the MPEG-4 decoder. The output from the decoder is then reformatted for display on an external monitor. The designs are targeted for the Xilinx ML402 development platform board.

For users interested in building their own video decompression system, the included detailed implementation documentation and the accompanying source code can be used as a starting point. The HDL source code is included for all of the implementation modules except the MicroBlaze™ soft core processor and MPEG-4 decoder. The Xilinx MicroBlaze processor is a free core that is configurable through the EDK development platform. C source code for the software running on the MicroBlaze system is also included. The MPEG-4 decoder is a purchased core in which a netlist for the user's specific set of parameters is provided.

## System Overview

The MPEG-4 decoder demonstration system is composed of an FPGA hardware evaluation platform, Xilinx IP cores and embedded software operating together to perform video decompression on industry standard encoded video bitstreams.

Figure 1-1 details the primary hardware elements implemented to construct the MPEG-4 decoder demonstration system. Note that there are three hierarchical levels:

- Functions implemented in the Virtex™-4 XC4VSX35 device
- Support devices on the ML402 board
- Externally connected devices, a video display, and a PC for system monitor message display

The elements found within the heavy-line rectangle are implemented in the Virtex-4 XC4VSX35 device on the Xilinx ML402 development board. These include: MPEG-4 decoder core, DDR Memory Controller, Color Space Converter, VGA interface, Macroblock Format Converter, and the MicroBlaze soft core processor and its associated peripherals.


www.BDTIC.com/XILINX

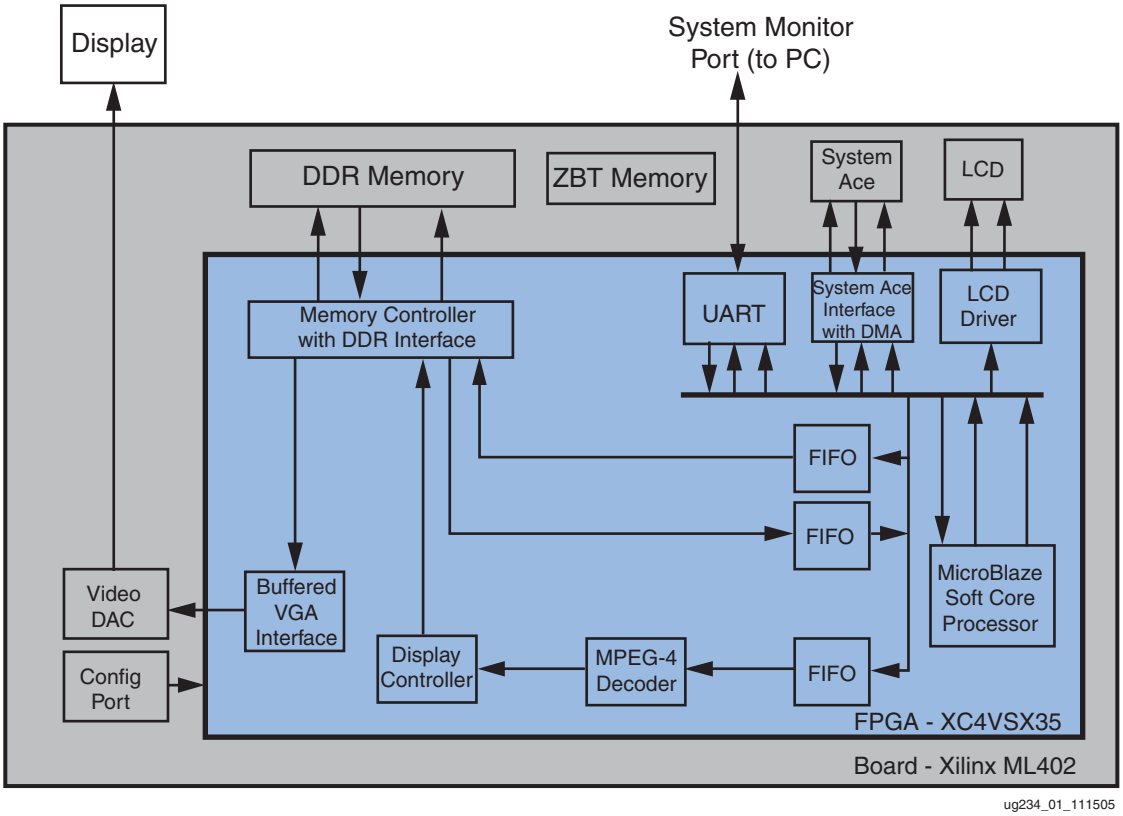


*Figure 1-1:* **System Overview**

Elements outside the heavy-line rectangle and within the grey rectangle are found on the ML402 as discrete devices. These include: ZBT memory, DDR memories, System ACE™ and Compact Flash connector, 2-line LCD display, and a Digital-to-Analog Converter (DAC).

External connections to a monitor capable of displaying VGA video is required. Also present is a serial configuration port for downloading the FPGA configuration bitstream from a PC to the ML402 board.

The embedded MicroBlaze processor operates as the overall system-level controller, handling such functions as the user interface, reading compressed bitstreams from Compact Flash, transmitting the bitstream into the MPEG-4 decoder core, and monitoring all system status flags.

The goal of the MPEG-4 decoder demonstration system is twofold. First, is to have a standalone system showing a Xilinx FPGA-based system performing a complex video application, which includes not only high performance video processing engines, but also an embedded processor to handle common system-level tasks. Viewing the resulting output video is an important quality measurement metric for all video systems.

Secondly, users interested in building their own video system which includes an MPEG-4 decoder can use the demonstration platform as a starting point. Many of the commonly required elements of a system are integrated and tested. The addition of the Video Input and Output Daughter Card (VIODC) allows a multitude of different video input and output formats to be processed.


www.BDTIC.com/XILINX

The MicroBlaze subsystem hardware and software were developed using the Xilinx EDK tools. All files are included in the source repository.

## ML402 Board

The Video Starter Kit is composed of a hardware platform, software tools, support libraries, and demonstration applications. The hardware platform is composed of two boards: an ML402 evaluation platform and a VIODC. The ML402 evaluation platform is powered by the Virtex-4 XC4VSX35 device and supported by industry-standard peripherals, connectors, and interfaces. Designed for use with the Xilinx System Generator for a DSP development platform, the Virtex-4 ML402 SX Evaluation Platform provides an entry-level development environment for creating video signal processing designs. Figure 1-2 is a simplified block diagram of the ML402 board.

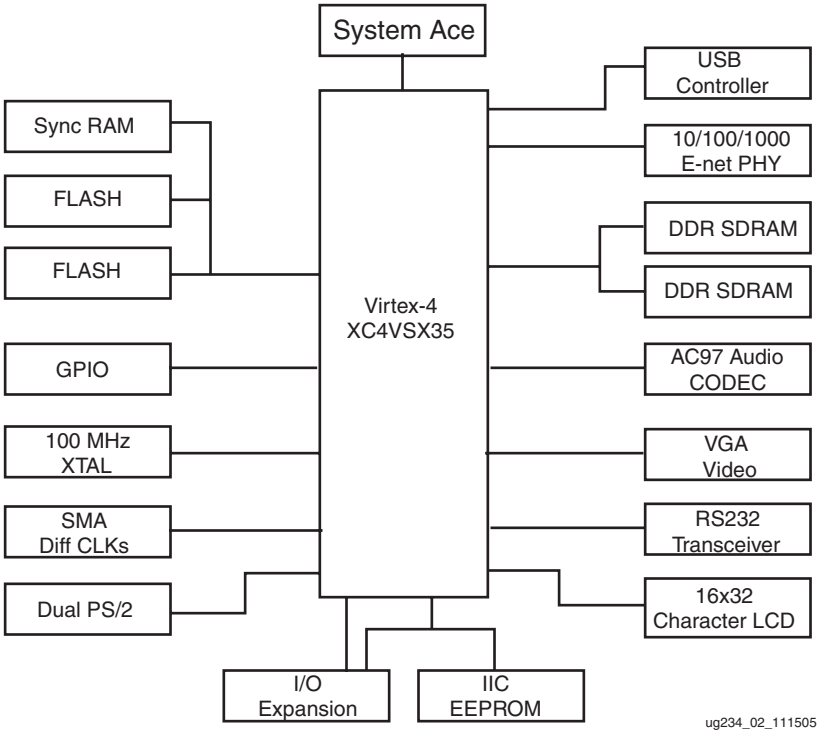


*Figure 1-2:* **ML402 Evaluation Platform Simplified Block Diagram**

The VIODC board adds support for industry standard video input and output sources, including: component and composite video, S-Video, HD-DVI, DVB-ASI, and SDI. An additional input for a stereo camera is available. The VIODC daughter card interfaces to the ML402 board through the I/O Expansion headers. For further information, refer to the ML402 documentation for a complete description.

## VIODC Board

The VIODC basic block diagram is shown in Figure 1-3. At the heart of the board is a Virtex-II Pro XC2VP7 device, which provides all timing signals and physical interfaces to the video input and output devices. For further details, refer to the VIODC documentation. Note that for the first phase of the MPEG-4 Part 2 Decoder, the VIODC board is not required. Instead, the video is output through the VGA port on the ML402 board.


www.BDTIC.com/XILINX

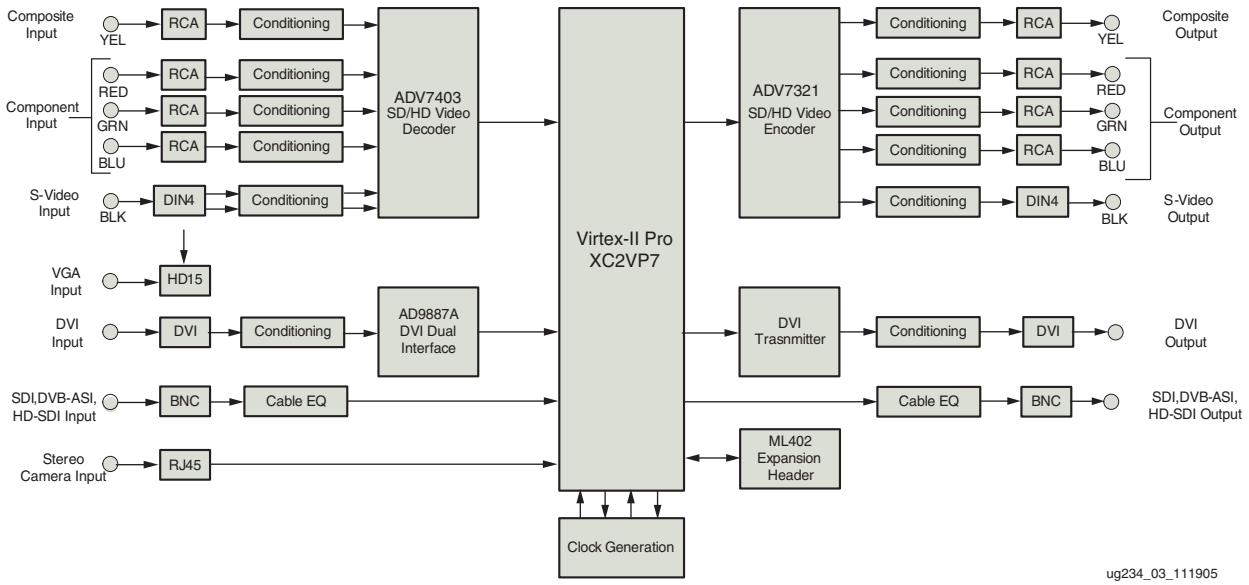


*Figure 1-3:* **VIODC Basic Block Diagram**


www.BDTIC.com/XILINX

Chapter 2

# Theory of Operation

This chapter depicts the interconnection of the primary elements of the system and how these elements interact to perform as a system.

## System Overview

The primary elements within the system are:

- MicroBlaze soft processor core
- DDR Memory Controller core
- MPEG-4 Part 2 Decoder core
- Macroblock Format Converter
- Video Output Driver and Color Space Converter

The processor performs three main functions within the system:

- Management of compressed bitstreams
- Graphics overlay
- Video clip loop control

Managing the compressed bitstreams entails "opening" the compressed bitstream file found on the Compact Flash (CF). The System ACE device provides the physical-level interface to the CF and provides an abstraction layer that makes the memory appear to be a disk. The compressed video is read from CF, stored in the DDR memory, and written into the input FIFO of the decoder. The video is stored in DDR memory to meet the input bandwidth of the decoder. The MPEG-4 decoder input FIFO is configured to generate an interrupt whenever the FIFO goes "almost empty." The interrupt service routine, running on the MicroBlaze, determines whether it needs to transfer more of the compressed bitstream to the input FIFO. Or, if the end of the video clip has been reached, the interrupt service routine terminates the video processing.

The MPEG-4 decoder core processes the compressed bitstream and produces a raw video output. During processing, intermediate video frames are stored in external memory for reference. The final reconstructed frame is output in Macroblock format, converted to raster scan format, and stored in DDR memory in a 4:2:2 YUV format. The MPEG-4 Part 2 Decoder core is available from Xilinx by contacting a Xilinx salesperson or the website for more information.

External DDR memory is accessed by multiple elements: the MicroBlaze processor, Macroblock Format Converter, and the Video Output Driver. A DDR Memory Controller is used to abstract away the physical-level interface and to arbitrate between the multiple elements requiring access. There are two read and two write ports on the memory controller. The MicroBlaze port can be used to write graphics onto the display, while the


www.BDTIC.com/XILINX

Macroblock Format Converter and Video Output Driver ports are used to process the compressed video bitstream.

# MicroBlaze Subsystem

The MicroBlaze embedded soft processor core is responsible for the overall system control including:

- Reading the compressed bitstreams from Compact Flash
- Managing the writing of the compressed bitstream to the MPEG-4 decoder
- Monitoring status signals and normal operational conditions
- Initializing the video output frame buffers.

Development of the MicroBlaze subsystem was done completely within the Xilinx Embedded System Development tools called EDK v8.1. This includes the configuration of MicroBlaze processor with attached peripherals, including: System ACE interface, MEPG-4 decoder, serial communications port, VGA controller, and interfaces to enable reading and writing registers. Transferring command and data information to other devices, such as the memory controller and MPEG-4 decoder, are handled by register reads and writes.

## Reading a Compressed Bitstream from Compact Flash

A compressed bitstream is stored on a CF card, which is read using the System ACE device on the ML402 board. The MicroBlaze subsystem was configured with the System ACE interface attached as an On-chip Peripheral Bus (OPB) to the processor. Refer to the MicroBlaze documentation for a complete description of the OPB peripherals.

Access to the CF device is performed through memory read and writes of data and commands. The System ACE device requires a series of commands, similar to a disk drive, to perform reads and writes. The EDK development system provides C-callable functions that are used by the MicroBlaze processor to open, close, read and write memory to the System ACE device. Source code is included that demonstrates opening, closing, and reading of the compressed bitstreams from the CF.

## Compressed Bitstream Input to MPEG-4 Decoder Core

A compressed bitstream is read from the CF and written into the DDR memory. After the whole bitstream has been transferred interrupts get enabled. The input FIFO for the decoder has an almost-empty flag that generates an interrupt when active. An interrupt handler gets called, which reads bitstream data from the DDR memory and writes it to the input FIFO of the decoder. Source code to manage reading from the CF and servicing the interrupts is included.

## MicroBlaze Interface to DDR Memory

The MicroBlaze subsystem includes an interface capable of reading and writing to the external 64 MB DDR memory through the DDR memory controller. The ML402 board DDR memory is organized as 32-bit words and is designed to be accessed in bursts of various lengths. The MicroBlaze processor accesses a minimum of four and a maximum of 60 32-bit words per command. Writes into memory are accomplished by first transferring all of the data into a write data FIFO and then issuing a write command to a separate write command FIFO. For a read operation, the user writes a read command into the memory


www.BDTIC.com/XILINX

controller's read command FIFO. Table 2-1 describes the format of the read and write commands.

*Table 2-1:* **DDR Memory Controller Command Format**

| Bits [31 down to 26] | Bits [25 down to 0] |
|---|---|
| Transfer Length | Starting Address |

## Reading Status and Writing Control Registers

The MicroBlaze processor has access to a status register that can be read to determine the state of various elements. See Table 2-2 for status and control register definitions.

*Table 2-2:* **Status and Control Register Definitions**

| Register | Description |
|---|---|
| Decoder Status | Provides state information for the decoder input FIFO. |
| Memory Controller Status | Provides state information for the memory controller FIFOs. |
| Decoder Reset | Software reset of the decoder peripheral. |
| Memory Controller Reset | Software reset of the memory controller peripheral. |

# MPEG-4 Part 2 Decoder Core

The MPEG-4 Part 2 Decoder core is capable of accepting a compatible compressed bitstream and producing a 4:1:1 YUV video output. This core is included in this demonstration system to illustrate the capabilities of Xilinx FPGAs in the video CODEC area. Note that although the core has the ability to support video formats up to 4CIF resolution, the implementation in this demonstration has been limited to CIF resolution. For further information about the full featured MPEG-4 Part 2 Decoder core, contact the local Xilinx sales representative.

## Decoder External Memory Connection

Performing the decompression operation requires storing a number of video frames for reference. The FPGA does not have enough internal memory. Therefore, an external memory device is required. An external Zero Bus Turn-around (ZBT) SRAM interface is integrated into the MPEG-4 Part 2 decoder.

## Decoder Output

The output of the MPEG-4 Part 2 Decoder core is an uncompressed video data stream in 4:2:0 YUV format, where the data is output one Macroblock at a time. A Macroblock is composed of six blocks, each of which is an 8 pixel by 8 pixel portion of the video frame as illustrated in Figure 2-1.


www.BDTIC.com/XILINX

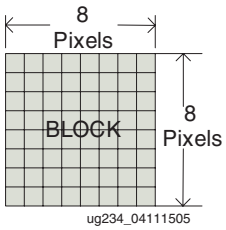


*Figure 2-1:* **Block of Pixels**

Figure 2-2 illustrates the forming of six blocks into the larger structure, a Macroblock. As can be seen the Macroblock has four Y blocks, representing 256 total pixels in a 16x16 pixel arrangement. A single U and single V Block, each containing 64 pixels of data, is replicated 3 times each to create the full YUV values for each pixel.

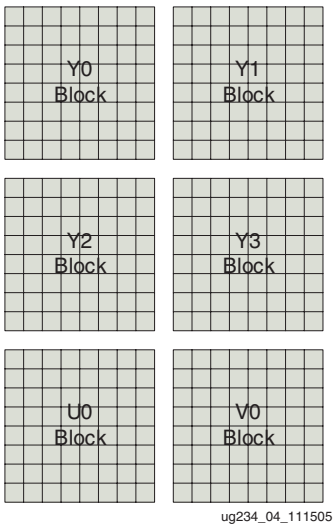


*Figure 2-2:* **Macroblock (Six Blocks of Pixels) 4:2:0 YUV**

Figure 2-3 illustrates the conversion from a YUV 4:2:0 format to YUV 4:4:4, by replicating the U and V values so that they are processed with the associated Y pixel locations. Note that for each U and V value, four values are created through replication.

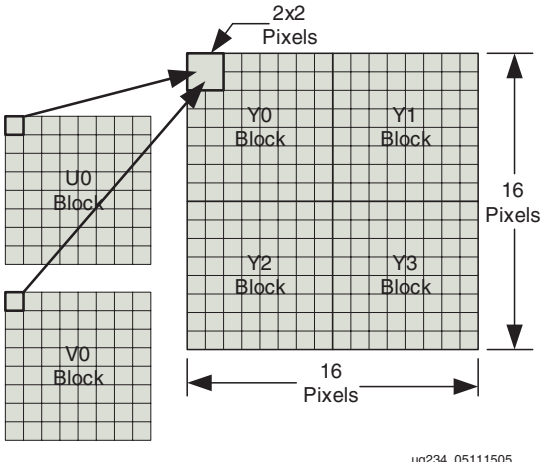


*Figure 2-3:* **Conversion from Macroblock YUV 4:2:0 to YUV 4:4:4**


www.BDTIC.com/XILINX

### Decoder Output Storage in DDR Memory

Output data from the decoder is stored into one of three video frame buffers in the DDR memory. Before storing the decompressed data, the format is converted from 4:2:0 YUV data organized as Macroblocks to 4:2:2 YUV data, where the data is organized in a raster format. Raster format is equivalent to rows of pixels, with N rows of pixels creating a full video frame. Conversion is performed by the Macroblock Format Converter block, which is detailed in the next sectionis detailed in the next section. The 4:2:2 format refers to the fact that for every 2 pixels in a row there is one U and one V pixel.

Table 2-3 details the actual bits of the DDR 32-bit data words. Note that the MicroBlaze processor has the ability to read and write the DDR memory. Therefore, if one chooses to create images for display, one would need to follow this format when writing the individual pixels. All three video frames must be written by the MicroBlaze processor with the same data to create stable graphics or background.

*Table 2-3:* **Pixel Storage Format, YUV, 4:2:2**

| Bits [31 to 24] | Bits [23 to 16] | Bits [15 to 8] | Bits [7 to 0] |
|---|---|---|---|
| V data | U data | Y1 data | Y0 data |

## Macroblock to Raster to Memory Controller

The Macroblock to Raster to Memory Controller (MB_2_Raster_2_MC) performs three primary functions:

- Conversion of Macroblocks to raster format
- Storage in one of three logical frame buffers in DDR memory
- Frame synchronization with the video driver

Macroblocks being produced by the MPEG-4 Part 2 decoder are converted into a raster scan format more conducive to the display process and then stored into one logical video frame in DDR memory. Three frame buffers are used on a fixed rotating schedule and each frame is capable of handling images with up to 1024x1024 resolution.

The MB_2_Raster_2_MC has data written into internal Macroblock buffers by the MPEG-4 Part 2 Decoder, converts the data into a raster format, and transfers the data to the memory controller data FIFO. A write command and start address instructs the memory controller where to store the data.

Write operations to DDR by the MB_2_Raster_2_MC must be to a video frame that is not being used to display video to avoid image corruption. A synchronization operation is used between the MB_2_Raster_2_MC and the video output driver; this insures that the frame of video being displayed is not written at the same time.


www.BDTIC.com/XILINX

## DDR Memory Controller

Access to the external DDR memory devices is controlled by the DDR Memory Controller. The main goal of the memory controller is to make it appear that each of the multiple devices connected always appear to have the memory available and to abstract away the complexities of controlling the attached DDR memories. The memory controller must perform a number of different functions:

- Execute read and write commands
- Arbitrate access to the memory
- Perform refresh and DDR initialization

Access to the DDR memory must be done on a single-user basis, but in this system there are four competing functions. There are two read and two write ports, see Table 2-4 for a description of each port and its purpose.

Arbitration allows for the memory controller to identify which request is next and to service those requests in a reasonable time frame. In this implementation, a simple "round-robin" arbitration scheme is used. After a port is granted access, it will maintain control until all commands are processed. One reason that a very simple arbitration scheme can be employed is because of the relatively high memory bandwidth available. In this version of the design, the memory controller is operating with a 100 MHz clock. It performs two 32-bit data word transfers per clock cycle, resulting in a theoretical maximum bandwidth of almost 800 MB/s. This is an order of magnitude higher bandwidth than is required for this application. A more complex scheme might be required when looking to support higher frame rates and higher resolutions.

*Table 2-4:* **Memory Controller Ports**

| Direction | Port Function |
|---|---|
| Write | MicroBlaze to DDR memory. Provides ability for the processor to initialize the frame buffer memory, perform on screen display functions, and use the DDR memory for data storage. |
| Read | MicroBlaze from DDR memory. Provides ability for the processor to read the contents of each frame buffer and to read stored data. |
| Write | Macroblock to Raster to Memory Controller (MB_2_Raster_2_MC). Provides ability for the decoder to write decompressed video into the frame buffers. |
| Read | Video display driver. Provides ability for video display driver to read frames from the DDR memory for display. |

After access to a port is granted by the arbiter, the simple command is decoded. The direction of the operation is inherent in the port type, read or write. The transfer length and start address found in the command are used to initialize internal memory controller state machine variables. For writes to DDR memory, data is read from the write port input FIFO and written to the appropriate DDR memory location. Read commands read the DDR memory and write the data to the read ports output data FIFO. Input and output FIFO status flags are used to indicate the state of the operation.

DDR memory requires periodic *refresh cycles* to be performed to ensure that the memory retains stored data. The memory controller has built-in refresh timers and state machines that automatically interject refresh operations as needed. Note that the refresh cycles have higher priority than either read or write operation.


www.BDTIC.com/XILINX

# VGA Controller and Color Space Converter

Final output video is displayed on a VGA compatible monitor, such as found connected to a typical PC. All required signals are generated by the VGA controller module. This module performs several functions:

- Generates read commands to memory controller to fetch video data one row at a time.
- Generates all VGA control signals: VSYNC, HSYNC, Pixel Clock, etc.
- Converts the frame data from YUV 4:2:2 to YUV 4:4:4.
- Converts from YUV to RGB color space.

The display output driver generates the horizontal (HSYNC) and vertical timing (VSYNC) required to drive a monitor. Currently, the timing is generated to support a 4CIF image (720 pixels by 480 rows) at 30 frames per second. At the start of each line of video (when a new horizontal sync is generated), a request to the memory controller is made to read an entire line of video, which stores the entire line in its read data output FIFO. Data is read out of the memory controllers data output FIFO during the time the pixels are displayed on the screen at the precise pixel rate that is required. Note that each 32-bit word read from the frame buffer represents 2 Y values and a single U and single V value (see Table 2-3).

In addition, the VGA controller must generate a frame address that cycles through the three different frame buffers. Three buffers are required because the MPEG-4 decoder is not synchronized to the output display timing. Therefore, a simple ping-pong buffer arrangement does not work. To prevent the decoder from writing to an area of memory that is currently being displayed on the screen, an extra frame buffer must be used. That way, the decoder and the VGA controller can both be assured to always have access to the portion of the memory that they need access to.

Before the data can be displayed, a color space conversion process must first convert the 4:2:2 YUV data to RGB. First, the 4:2:2 YUV data must have the U/V data replicated to produce 4:4:4 YUV data. Next, a straightforward mathematical set of equations (shown below) is implemented to convert the YUV data to RGB. Saturation logic is also present to ensure that the RGB values do not exceed their maximum or minimum allowed values.

$$\text{Red} = Y + 1.371(V - 128)$$

$$\text{Green} = Y - 0.336(U - 128) - 0.698(V - 128)$$

$$\text{Blue} = Y + 1.732(U - 128)$$


www.BDTIC.com/XILINX

www.BDTIC.com/XILINX

Chapter 3

# Source Code, Project Files, and EDIF Netlist

## Overview

Source code is available for many of the components in the demonstration. The MPEG-4 decoder core and MicroBlaze processor source are not included. Instead, an MPEG-4 pre-generated netlist for up to CIF resolution is provided in the reference design. License for the MPEG-4 Simple Profile Decoder and Encoder can be purchased from Xilinx. Source code is included with the Video Starter Kit for the following modules:

- Macroblock to Raster Converter
- Video Output Driver
- Color Space Converter
- DDR Memory Controller
- Wrapper File (integrates all these elements into a single design)

Each of the modules is stored in a separate directory structure, illustrated in Figure 3-1. Each directory area has a similar structure, containing: source, and simulation and synthesis subdirectories. The *source* directory contains the VHDL or Verilog files. In the case of the *MB2Raster_Converter,* there is also an EDIF file that was created using CoreGen for a block RAM, which is instantiated in the source code. The *simulation* directory contains the testbench used to verify that particular module. Support for simulation in ModelSim is included containing *.do* files and ModelSim project files that can be used for simulating that portion of the design. The *synthesis* directory contains the Synplicity project file used for synthesizing the module.

A directory structure is also included for the entire MicroBlaze system, built by using the Embedded Development Kit (EDK) tools from Xilinx. Using the EDK development tools, a user can open the complete EDK project and modify any portion of the processor subsystem required.

***Note:*** The entire system is built in two primary sections: one for the microprocessor and one for the MPEG-4 decoder, memory controller, and all support modules. This structure enables easy access to the working decoder subsystem if a designer wishes to use a different processor, for instance a DSP or PPC.


www.BDTIC.com/XILINX

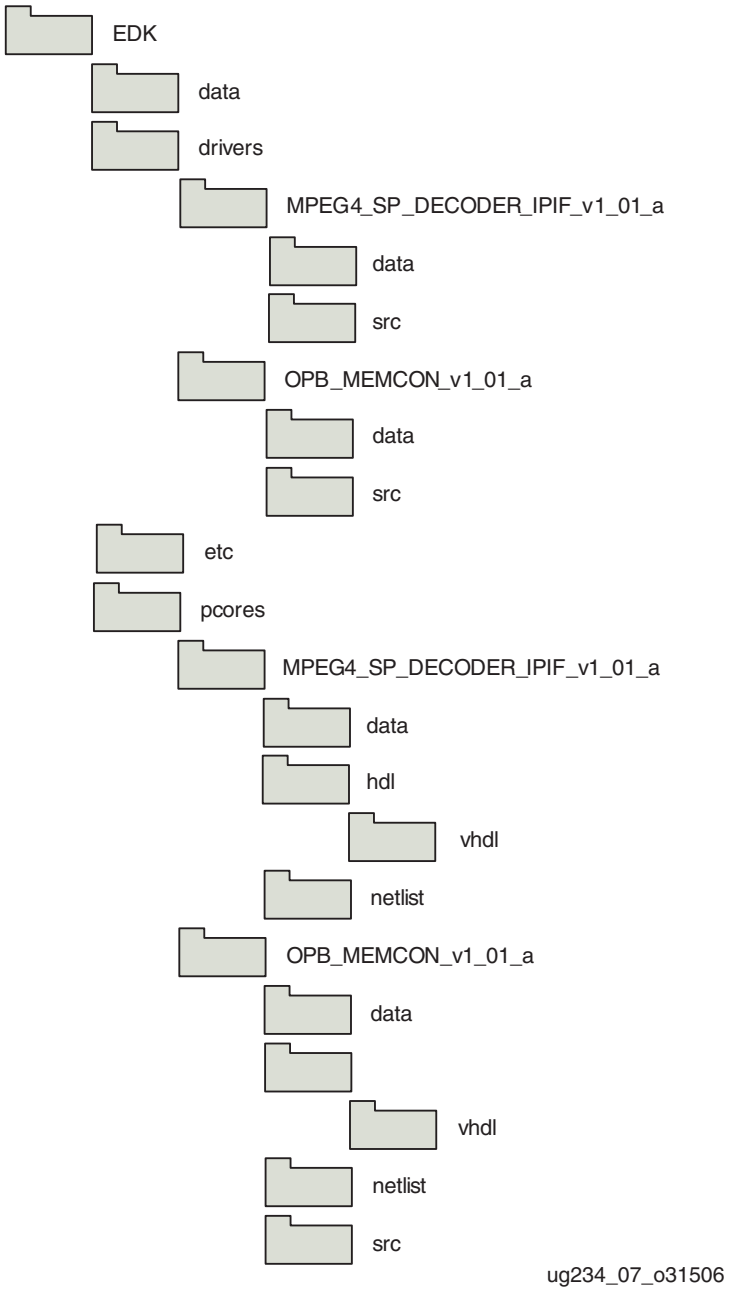


*Figure 3-1:* **Decoder Output-to-Video Driver Source Directory Structure**


www.BDTIC.com/XILINX

Source code, project files, and EDIF netlist are available for all elements of the system. See Table 3-1.

*Table 3-1:* **System Source Code**

| Component | Source File |
|---|---|
| MicroBlaze Processor | Project directory capable of rebuilding entire subsystem. Included is C source code for all control functions. |
| MPEG-4 Part 2 Decoder | EDIF netlist for up to CIF resolution images |
| Memory Controller | Verilog source files compatible with Synplicity synthesis tools |
| VGA Controller | VHDL compatible with Synplicity and XST synthesis tools |
| Color Space Converter | VHDL compatible with Synplicity and XST synthesis tools |
| Macroblock to Raster to Memory Controller | VHDL compatible with Synplicity and XST synthesis tools |

# MPEG-4 Part 2 Decoder Demonstration Implementation

The MPEG-4 decoder demo design has already been compiled for the ML402 board and resides on the Compact Flash card included with the Video Starter Kit. The demo can be run directly from the Compact Flash card. All source files for the demo design have been included so that the user can run the XPS software to re-implement the design and make modifications if so desired.

## Loading the EDK Project

The demonstration design was developed using the Embedded Development Kit (EDK). A user can open the EDK project in XPS choose **File->Open Project**. Then browse to:

```
<project installation directory>\Examples\hdl_demos\ml402_mpeg4_dec\EDK
```

And select:

```
system.xmp
```

The `system.xmp` file completely configures the EDK tools and loads the associated project software and hardware components. After the project has been loaded into the EDK system and with the ML402 board powered on and a Xilinx download cable attached, a user can simply click the EDK download button and all necessary steps are executed to re-compile the design and download to the ML402 board.

## Recompiling One Step at a Time

Alternatively, a user can choose to recompile the system one step at a time, finally downloading the implemented design to the board. The EDK system relies on a number of text-based files to specify the configuration of the system.

Within the project management window, two tabs are available: Applications and System. The System tab contains all the hardware components of the EDK system. The


www.BDTIC.com/XILINX

Applications tab contains all the software. By selecting either tab, the user can review the underlying source code for both hardware and software portions of the system.

## Hardware Component Processing

The system.mhs file is a textual description of the components included in the MPEG-4 Part 2 Decoder Demonstration system and includes the configuration of each component and connectivity. When the Generate Netlist button is clicked, the EDK system determines which hardware files need to be processed and automatically runs all steps required to produce a netlist for each of the system components. For those components that are synthesized, the appropriate synthesis tools are automatically invoked. Clicking the Generate Bitstream button processes the synthesized netlists through translate, map, PAR, and Bitgen phases.

## Software Component Processing

Within the source tab for the demonstration system, the user can browse and modify the available C source code for the demonstration system. Double clicking on the `stream_video_ddr.c` opens the file in an editor for the user to browse or modify. The Generate Libraries button compiles the necessary software libraries for MicroBlaze and the peripherals included in the system into the project directory. User source code is compiled by clicking the Compile All User Applications button, which for the demonstration system compiles the `stream_video_ddr.c` source code.

Clicking the Update Bitstream button to create the file `download.bit` used to configure the FPGA device. The final step is to click on the Download button which configures the FPGA on the ML402 board and automatically begins execution of the software code.

## Synplicity Synthesis

For synthesis, a Synplicity project file (`.PRJ`) has been supplied that allows a customer to synthesize the entire design or the specific component that is defined in that particular directory (such as the YUV2RGB Converter).

## ModelSim Simulation

For simulation, two Modelsim scripts (`.DO` files) have been provided for each design component. The first is a `testbench.do` file which is used to initialize a simulation. This script should be run from the ModelSim console window to startup a simulation. It references all the necessary libraries and then calls a second DO file (`waveFormat.do`) which is used to open up a wave diagram and to populate the waveform window with a list of signals that might be of interest.

However, there is one file that is not provided—a ModelSim project file (`.MPF`) which is needed to load a project before a simulation can be run. ModelSim projects cannot be provided because there are absolute paths in the file for the source files and simulation libraries. Instead, a user must create a new project and ModelSim will use the customer's own `modelsim.ini` file to setup the project parameters. All that the customer needs to do is add the appropriate source files and testbench file to use the supplied DO files to perform a simulation.

The necessary files for each design are as follows (in the order shown). All the files will be located in either the source or simulation directory for the component being tested unless otherwise noted. (Note that there is no simulation for the Memory_Controller only. This was only tested as part of the overall design).


www.BDTIC.com/XILINX

1. MB2Raster_Converter ->
   a. `mb_buffer.vhd`
   b. `dc_mb_2_raster_mc_if.vhd`
   c. `tb_dc_mb_2_raster_mc_if.vhd`
2. YUV2RGB_Converter
   a. `YUV2RGB_Converter.vhd`
   b. `YUV2RGB_Converter_tb.vhd`
3. VGA_Controller
   a. `VGA_Controller.vhd`
   b. `VGA_Controller_tb.vhd`
4. uBlaze_2_MemCntlr_MB2Raster_YUV2RGB_VGA (this is the top-level design)
   a. `glbl.v`
   b. `clk_module.v` (located in Memory_Controller/source)
   c. `delay.v` (located in Memory_Controller/source)
   d. `tristate_iobuf.v` (located in Memory_Controller/source)
   e. `ddr_iobs.v` (located in Memory_Controller/source)
   f. `blockram.v` (located in Memory_Controller/source)
   g. `UltraFIFO.v` (located in Memory_Controller/source)
   h. `user_interface.v` (located in Memory_Controller/source)
   i. `controller.v` (located in Memory_Controller/source)
   j. `ddr.v` (located in Memory_Controller/source)
   k. `ddr_top.v` (located in Memory_Controller/source)
   l. `mb_buffer.vhd` (located in MB2Raster_Converter/source)
   m. `dc_mb_2_raster_mc_if.vhd` (located in MB2Raster_Converter/source)
   n. `YUV2RGB_Converter.vhd` (located in YUV2RGB_Converter/source)
   o. `VGA_Controller.vhd` (located in VGA_Controller/source)
   p. `uBlaze_2_MemCntlr_MB2Raster_YUV2RGB_VGA.vhd`
   q. `uBlaze_2_MemCntlr_MB2Raster_YUV2RGB_VGA_tb.vhd`


www.BDTIC.com/XILINX

www.BDTIC.com/XILINX