Atlys HDMI Demonstration Project

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Revision: November 2, 2011

Overview

This project demonstrates the following video processing capabilities of the Atlys:

1) Reading in video data from an HDMI port and storing it in a designated region of the onboard DDR2.
2) Modifying video data stored in DDR2 using an embedded MicroBlaze processor.
3) Outputting video data stored in DDR2 to a display device attached to an HDMI port.
4) Implementing an EDID transmitter so that attached devices will recognize the Atlys as a display device

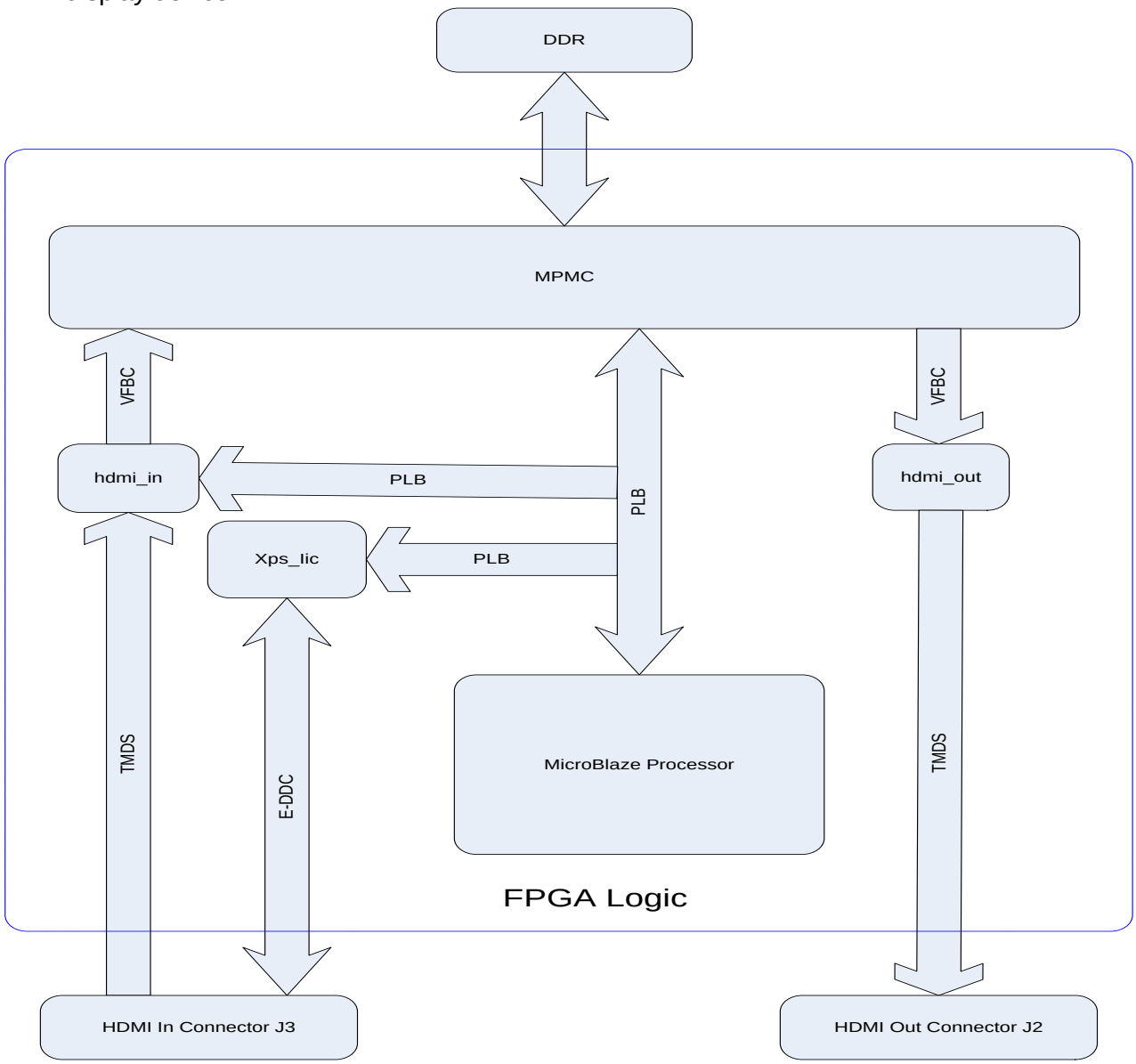

Figure 1. Block Diagram

The project was designed using Xilinx EDK 13.2. It incorporates a custom HDMI input core and a custom HDMI output core that attach directly to an MPMC core with VFBC interfaces. The HDMI output core is configurable within Xilinx Platform Studio (XPS) and the HDMI input core is software configurable. Both of these cores were designed to be reusable within other projects. The project also includes an xps_iic core for controlling the E-DDC lines described in the DVI 1.0 standard. This allows the Atlys to be detected by computers as a display device.

See Figure 1 above for a block diagram of the implemented hardware (interrupt and push button GPIO cores not shown).

Running the Demo Application

The following equipment is required:

- Digilent Atlys Development Board
- Computer running Xilinx EDK 13.2
- 2 USB micro cables for programming and UART connection
- 2 HDMI cables
- 1 HDMI display or DVI display with HDMI/DVI adapter
- 1 HDMI video source or DVI video source with HDMI/DVI adapter

The following steps outline how to build the project in XPS and then execute the demo application on an Atlys. A basic understanding of Xilinx EDK is assumed. All file paths are given from the folder containing this document:

1) Open up project\system.xmp in XPS. Allow the tools to perform any necessary updates.
2) Export the Hardware design to SDK. This will take some time to complete.
3) Open up a new workspace in Xilinx SDK. If not done automatically, import the generated hardware profile.
4) Create an empty Xilinx C project and a standalone BSP. Copy all files from source\ to the src\ folder in the empty C project. Ensure to overwrite any existing files, and allow the project to build.
5) Attach the Atlys UART and PROG ports to your computer and switch the Atlys on. Connect to the UART port in a Terminal program (this can be done is SDK).
6) Under “Xilinx Tools”, select Program FPGA. In the dialog box that appears, ensure that “ELF file to Initialize Block Ram” is set as bootloop and click Program.
7) Once the FPGA is programmed, right click on your project’s built binary (.elf file) and select Run As-> Launch on Hardware.
8) Once the program has downloaded, attach a display device to HDMI OUT port J2 and a video source to HDMI IN port J3. Ensure that jumpers 6,7 and 8 are not shorted.
9) The onboard push buttons perform the following operations:

    - BTNC – Start/Stop recording data into the frame buffer from the input video source. A newly attached device will always be in the Stop state. When in the Start state, the data present in the frame buffer is constantly being overwritten by new data from the video source.
    - BTNR – Print a test pattern to the frame buffer. The program should be in the stop state in order to view the pattern before it is overwritten

- BTND – Invert the data in the frame buffer. The program should be in the stop state in order to view the inverted frame before it is overwritten
- BTNL – Print the width and height of the input source frame over UART. A width of 0 and height of 1 means that no video data is being received.

Additional Notes on Using the Demo Application

- Input sources that require EDID packets to recognize a device must be attached after the software has been loaded to the FPGA. This is because the IIC controller must be configured properly in order to respond to the EDID requests which occur just as the device is plugged in.
- The output frame cannot be configured from software, and will constantly output the data stored at the designated frame base address (set from XPS). If any of the values are changed in XPS then the corresponding values must be changed in the hdmi_demo.h header file.
- The transmitted EDID packet may be altered in the hdmi_demo.h header file.

hdmi_in Register Descriptions

Register descriptions are given along with their memory offsets from the hdmi_in base address. Bit numbers are assigned such that Bit(0) = MSB and Bit(31) = LSB. Note that all registers are only accessible when a video source is attached and providing a valid clock signal. Registers are reset whenever a video source is disconnected.

Control Register: 0x00 : R/W

Bits (0 : 30) – Reserved
Bit(31) – Write Enable – Enables the core to begin writing video data to the frame buffer. Default is '0'.

Status Register: 0x04 : R

Bits (0 : 29) – Reserved
Bit(30) – Frame Locked – Reading a value of '1' means that the frame dimensions of the incoming video signal have been determined.
Bit(31) – PLL Locked – Reading a value of '1' means that a valid clock signal is detected on the TMDS lines.

Frame Width Register: 0x08 : R

Bits (0 : 15) – Reserved
Bits (16 : 31) – Unsigned value that represents the width of the input frame in pixels

Frame Height Register: 0x0C : R

Bits (0 : 15) – Reserved
Bits (16 : 31) – Unsigned value that represents the height of the input frame in lines, minus 1

Line Stride Register: 0x10 : R/W

Bits (0 : 15) – Reserved
Bits (16 : 31) – Unsigned value that defines the line stride of the frame in pixels (described below). This value must have bits 26 to 31 equal to zero in order to be 128 byte aligned. Default value is 0 and must be set before enabling the core.

Frame Base Address Register: 0x14 : R/W

Bits (0 : 31) – Unsigned value that defines the physical address of the frame buffer. This address must fall somewhere within the DDR2 memory space and have enough trailing memory to fit the entire frame. This value must have bits 25 to 31 equal to zero in order to be 128 byte aligned. Default value is 0 and must be set before enabling the core.

Memory Format of Frame Buffer

Video data is stored linearly in memory as 16 bit pixels with the upper left hand pixel of the frame being located at the defined frame base address. The remaining pixels in the line are stored sequentially from this address. The beginning of the next line is determined by the value of the Line Stride. The Line Stride is the number of pixels (each being 2 bytes) in memory between the beginnings of two adjacent lines. By setting the Line Stride to a value larger than the Frame Width you may pad the end of each line with unused memory. This prevents problems that may arise if the input frame is wider than the frame being output.

The data at pixel address (x,y) with the upper left hand pixel being (0,0) may be accessed using the following function in an SDK application:

```
Xil_In16( FRAME_BASE_ADDR + x*2 + y*LINE_STRIDE*2 );
```

Note that it is currently a requirement of the hdmi_in and hdmi_out cores that any values provided for frame base address, line stride, or frame width be 128-byte aligned. This means that any line stride and frame width values must have bits 5 down to 0 equal to 0, and that the frame base address must have bits 6 down to 0 equal to 0. Not following this guideline may result in undefined behavior.

The arrangement of color data within a 16-bit pixel is as follows:

RED MSB	RED	RED	RED	RED LSB	GRN MSB	GRN	GRN	GRN	GRN	GRN LSB	BLU MSB	BLU	BLU	BLU	BLU LSB

Figure 2. Pixel Data Bit Ordering