IP Project 3 – Video Output

What we want:

A Xilinx EDK **pcore** peripheral that uses the **HDMI output** port of the Atlys board to send images to a TFT monitor. For that it needs to connect as a **Burst Master** to the **AXI Spartan-6 FPGA DDRx Memory Controller**. The base address for the image should be set by an AXI slave register.

What you get from us:

- Documentation for the VmodCAM module
- Two VmodCAM demo projects that streams images to the video output, one in VGA, on in HD resolution. It is a Xilinx ISE project, so it doesn't make any use of EDK or processor systems.
- A project ` Atlys_HDMI_PLB_demo` that streams HDMI image from input to output through memory. It uses PLB-based HDMI components. The use of PLB is now deprecated.
- Documents about the Spartan 6 AXI memory controller and slave and master burst peripherals

What to demonstrate:

Video output on a TFT display that includes some smoothly animated component. You can communicate with the groups that do the video input from the camera module. Build an EDK system that has both the video input and video output components, and show that it displays the camera image on a monitor. (If you don't want to be dependent on other groups, you have to write animation on Microblaze. Be aware that you cannot write a complete image of pixels from a Microblaze in time to keep the frame rate).

A few hints:

- Understand how HDMI and TFT transmit image information.
- Try to get the VmodCAM project working first, and try to understand how the data flows inside
- Get the Atlys_HDMI_PLB_demo working with the test image mode. Don`t bother with the HDMI input. Try to understand how the PLB video output component works.
- Do Tutorial m05 (upcoming) on how to build AXI peripherals with the Create Peripheral Wizard
- Write test images from the Microblaze into memory, so that you are not dependent on video input. Move a small object around on the images so you can see if the frames update correctly and fast enough.
- The master burst component will not be coherent with the Microblaze's cache. To make sure that you read and write the same data, circumvent your cache by declaring all your memory variables, arrays and pointers as *volatile*.
- Make the core flexible on the line stride used (How many bytes are reserved for one line of the image: 1KByte, 2KByte, 4Kbyte)