**Custom IP Core Generation**

1. Started with the xillydemo project. It contains the minimal processor-logic-peripheral system capable of running Xillinux.
2. Video DMA example – RGB2YCbCr
   1. Add VDMA IP core. Frame Buffer = 1. In advanced tab, turn off Fsync option (Free running).
   2. Connect Clk and Rst signals to their respective sources from “Processor System Reset” block.
   3. AXI Interconnect has only 3 Master interfaces. Change it to 4 and then connect M03\_AXI to S\_AXI\_Lite port the VDMA IP. Auto-generate address for this connection using Address Editor. This address will be used to configure VDMA IP core by writing to its registers.
   4. Enable “High performance AXI interface” HP1 with 64-bit in Zynq7 Processing System.
   5. Cannot connect the PS7's HP1 directly to VDMA's M\_AXI\_S2MM port. Use an AXI Interconnect to translate between AXI4 and AXI3. Configure this AXI Interconnect to have two slaves and one master. The two slaves are connected to MM2S and S2MM ports of VDMA. The Master is connected to HP1 of PS7. Use auto-address assignment again. This gives the full **512 MB** address range of the external **DDR-RAM** at its disposal for both reading and writing.
   6. Don’t connect the interrupt wire for now.
   7. The streaming interfaces of VDMA, AXIS\_MM2S and AXIS\_S2MM, source and sink the video streams respectively. Connect them together to make a video loopback. Else, connect a video filter e.g. RGB2YCbCr. RGB2YCbCr IP core requires clk, reset and clk\_en. Connect a constant ‘1’ to clk\_en.
   8. There are three reset signals. One for ZynQ7, one for AXI interconnects and one for peripherals. Connect appropriately or else hw won’t work!
   9. Modified “vdma\_ex.c” seems to work with the IP. Write a more workable code with OpenCV interface.
3. Implement the same IP in Matlab and import to Vivado.
   1. Simulink supports generating the AXI4-streaming protocol based IP core. But there are various settings that need to be carefully set.
   2. Set the toolpath for HDL Coder using the following command

hdlsetuptoolpath('ToolName', 'Xilinx Vivado', 'ToolPath', 'C:\Xilinx\Vivado\2016.2\bin')

* 1. The first option needs to be looked at is the AXI4-Stream Data Width. In VDMA IP core, we can set it to any multiple of 8 less than AXI4 Data Width, which in turn can be set in multiples of 32. AXI4 Data Width = 32, however, seems a reasonable choice since DDR-RAM bit-width is 32. For RGB-24 bit data, AXI4-Stream Data Width = 24 worked fine with RGB2YCrCb IP core in Vivado. However, Matlab seems to only generate the IP with 32-bit width (to support RGBA format). So, either find a way to change this in Matlab or use RGBA format in OpenCV. Using 32-bits per pixel, btw, seems a good choice because it will lead to aligned memory access. Unaligned access are supported in VDMA but will definitely create some timing delays.
  2. Second option is the valid data line.
  3. GenLock needs to be understood. It has something to do with the “shear” i.e. master should skip a buffer if slave is working on it or something like that. But this configuration is probably only useful when reading and writing is being done to the same buffers i.e. buffers are acting like a circular FIFO to act as a buffer between two different time domains. This is probably why the downloaded VDMA example software had same addresses for read and write buffers!



In this figure, VDMA 1 and VDMA 2 are probably acting as the circular FIFOs as described above.

In my application, VDMA role is different i.e. provide an interface between CPU and FPGA. In this case, the problem of shear can occur when CPU is writing and FPGA VDMA is reading but solving this problem would require stopping and restarting VDMA on per frame basis. So that VDMA starts the next cycle when CPU has finished writing the frame to the buffer. This could become the topic of a paper.

* 1. For simple IP cores, following information is relevant.



* 1. Finally implemented IP core for Sobel filter in Matlab. Using EOL, SOF, Valid and Ready signals. Absence of one of these signals (most likely EOL), causes frame synchronization issues. Also the valid signal must be used to enable the module for only the relevant time intervals. Matlab example uses 32-bit (RGBA) interface instead of 24-bit (RGB). Appropriate changes in OpenCV code were made i.e. conversion from 3-channel to 4-channel.
  2. Implemented Sobel filter IP using Matlab/Simulink environment. Matlab has two relevant toolboxes i.e. HDL Coder and Vision HDL Coder. It seems that Vision HDL toolbox has many useful built-in blocks like filters etc. but uses a different streaming protocol with blanking etc. Had to install Matlab r2017b to get this working because in r2015, this protocol was not being assigned to the Xilinx AXI4 Streaming protocol when generating the IP. AXI interconnect had to be inserted as well because Vision HDL toolbox streaming protocol uses a default video resolution of FHD and some default values for front and back porch. These need to be changed via software for proper synching. Enable AXI4 register read out option in HDL Workflow advisor in the last step. Otherwise, these register values can be set but not read.
  3. Select “Generic Xilinx Platform” as “Target Platform” because AXIS-Video is not available for Zedboard-specific platforms.
  4. Developed HSG IP Core using Vision HDL toolbox. Also implemented a SW model for SVM training and validation purposes. This task was not too simple to do. Both HW and SW models have different caveats. Histogram population using filters is a neat idea but led to initial problems in both models. In SW, Boolean signals input in imfilter lead to Boolean signal output while we want to sum the votes and the output should be 4 bits with saturation. Similarly, in HW the filters had to be adjusted to give saturated 4-bit outputs while being driven by 1-bit inputs. Earlier, bin number calculation was problematic in SW because of coding mistakes. Now, SW model uses the exact bin calculation using atan2 while HW uses the approximate method. The error is only in few pixels and mostly for bins 0 and 7 which is understandable due to limited precision.
  5. Including ‘sqrt’ operation was causing timing closure problem in fpga implementation. Removing it makes the IP work at 100 MHz which is the system wide clock. The timing closure problem could be “seen” in the observed video in the form of a flicker i.e. pixel values kept on changing even for static input images.
  6. Developed an SVM training setup to go with the HSG SW model. Works: detects faces. HW IP core is, however, not 100% identical to SW model as expected; some faces are skipped in SW, others in HW. Also, there is some randomness even for static images. Sometimes only few faces are detected. Entirely different set of faces are detected every time program is restarted. In a single session, however, the same set is detected without flicker. Some initialization problem?

1. Implement the same IP in bare Verilog using IP skeleton code generation in Vivado.

**Troubleshooting Notes**

1. “Bus error” in software on ubuntu. Check if the IP core AXI interface has been assigned address or not. In software this address is used to configure the IP. If this has not been assigned, then software cannot register this address as virtual memory address.