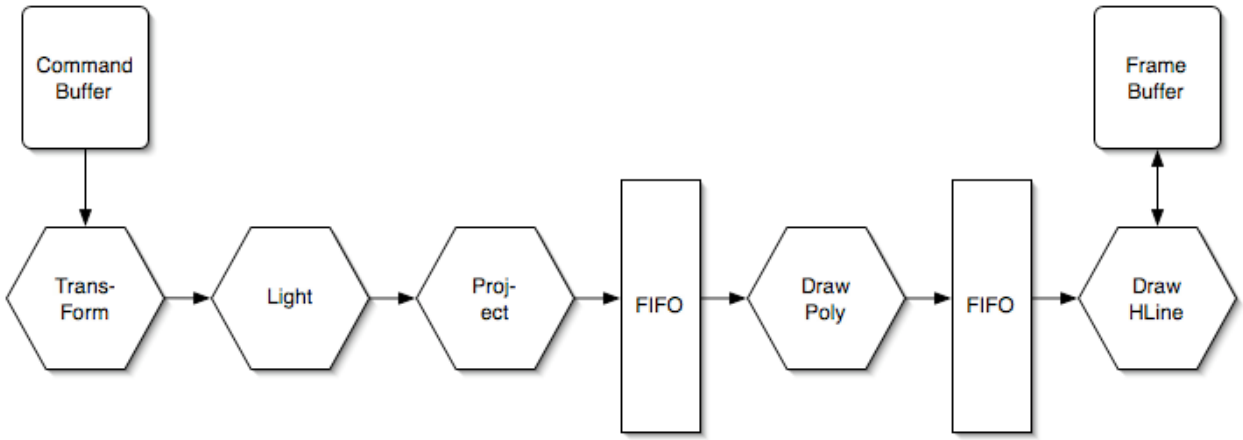
Project 3

Project 3 is the beginning of the 3D rendering pipeline. Prior projects (2-a, 2-b) where required accessory components (DAC, framebuffer, interface). Here we begin the main pipe. We are going to build the pipeline back to front. That is, we are going to start with the very *last* stage and work ourselves backward towards transformation. We are doing this because we can run the previous stages in software and transmit the results over the serial link to the card (hence, we can easily test our results). Here is an overall picture of what we are building:



This is a loosely coupled pipeline architecture (like what is used in the Pentium 2,3 & 4). The idea is you conventionally pipeline each of the fixed length stages (such as transform, light, drawhline, etc), but between variable length sections you place a FIFO. The goal of each of the early stages is to keep the FIFO full, but not needlessly so (for example, you could extensively pipeline the Transform stage, but you may not have to, instead you may want to extensively parallelize the drawhline stage).

Project 3 is to build the very final stage of this pipe (that is, the FIFO and DrawHLine sections). You should use your software simulator (or the provided solution) to duplicate all of the prior stages. Modify your serial interface to accept the commands to dump drawhline commands into the FIFO (as well as maintaining the framebuffer control – you also want to add a framebuffer clear command).

Notes:

- You should use 16 bit fixed point math. What's good enough for GeForce 3 is good enough for us ;-). Your pipeline is going to use two types of 16 bit numbers. One of these should be optimized for handling user coordinates and the other optimized for handling screen coordinates. The switch should occur at the Project stage (a later assignment). My suggestion – feel free to choose your own path – is to use X * 2^12 for user coordinates and X * 2^4 for screen coordinates. This should give you +/- 8 for user coordinates, and +/- 2048 for screen coordinates. You should TEST the usefulness of this fixed point system by modifying your software simulator. TRUST ME – TEST IT BEFORE IMPLEMENTING IT IN HARDWARE.

- The FPGA we are using has 40 hardware multiplies. Each of these is 18 bits (so they can comfortably produce a 16 bit result).
- Strive for working first, and efficiency second. If you where at a company producing this hardware you would probably spend a significant amount of time running simulations to determine exactly how much to parallelize and pipeline each stage in order to maximum your throughput and efficiency. Alas, we are going to take some short cuts and just implement it. Feel free to make appropriate decisions about pipelining.
- BUILD YOUR OWN FIFO. Please do not use one from opencores or elsewhere. Please make your FIFO parameterizable, since you will (hopefully) be reusing it in another part of your pipe.

Note, *there are a ton of design decisions you will have to make. Simply choose rational options. Feel free to alter details as you see fit. At a high level, we want to see a working component, but by all means pour your creativity into the design.*

Suggested coarse of action:

I suggest you begin by modifying your serial interface to accept the new commands (switch-framebuffer, clear-framebuffer, new-hline]. Then build the clear-framebuffer and switch-framebuffer logic (you should have the switch-framebuffer logic largely in place from project 2-b). The clear-framebuffer logic needs to zero out the framebuffer and reset the Z-buffer to the maximum (farthest away) distance.

You may want to have the graphics card ACK back to the PC when it is ready to receive another command. Although I believe you will still be processing commands far faster than you can send them, it isn't clear if this will always be the case. Hence, building that logic in now may save you time in the future.

Start by sending simple new-hline commands. For example, a dot. Work up to drawing a box at the same Z distance. Finally, work upwards to eventually sending the entire teapot.

At the end of this assignment you should have a teapot displayed on the screen. Of course, most of the computation will have been done in software.

Grading

As always we grade upon whether or not it works first. Do strive to make it work first and then worry about optimizations, tweaks, enhancements, etc.