FPGA-BASED 3D GRAPHICS PIPELINE

Technical Manual

Gabriel Cojocaru

1. Theory of Operation

For this design I will be using the DE1-SoC FPGA development board from Altera. The Board has a Cyclone V FPGA with a Hard Processor System (HPS) on the same die. The HPS is connected to the FPGA fabric with a high-performance bus that can be configured to support 32, 64, 128, or 256 bits data bus. The HPS is an ARM Cortex A9 Dual-Core processor capable to run a Linux distribution. The design will use the HPS system to run a minimal distribution of Linux from Altera that has the necessary kernel drivers to talk to the peripherals attached to the HPS system. The FPGA fabric is also memory-mapped into the HPS side so that the circuits implemented in the FPGA can be accessed by programs running in user space by calling mmap().
The first structure visible to the HPS is a FIFO used to receive commands and data from the program. To send vertices to the FIFO, the program writes to the virtual address returned by mmap() + the offset at which the FIFO is located. On the FPGA, a state-machine continuously pulls data (if available) and interprets it and decides how to act depending whether it is a command or just data. The remaining of the system is shown in the figure below.

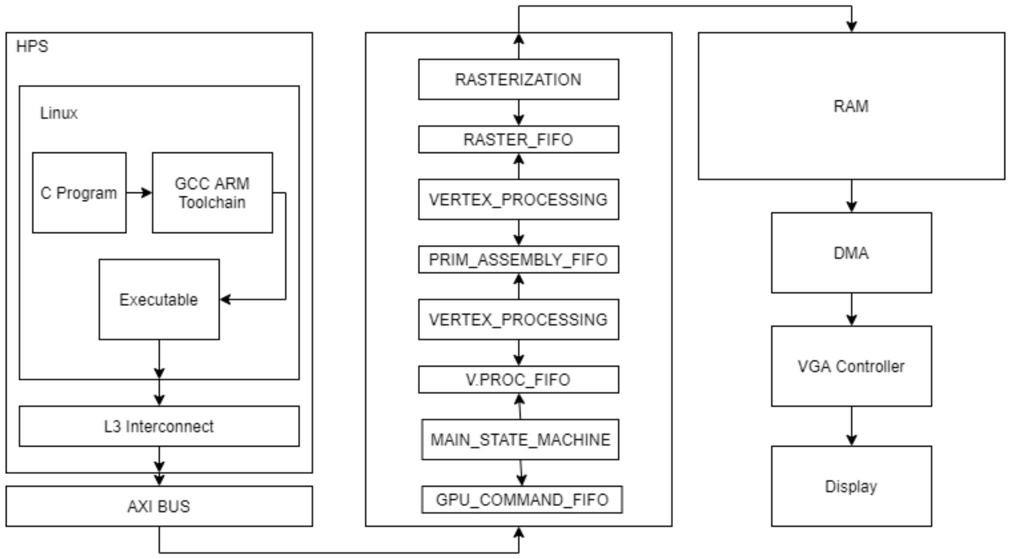

Figure 1. Software and Hardware system overview

From the figure, once the data is in the FIFO between the HPS and the FPGA fabric, the first state-machine reads the data and interprets it. If the program sent a list of vertices that define a primitive, the module relays the data to the Vertex Processing stage. If the data sent is configuration for the pipeline, then the Primitive Reader stage executes that and repeats the process again. Once the vertices are in the Vertex Processing stage, they are transformed according to the Transformation Matrix that the program sent to the GPU. Multiplying a vertex with the transformation matrix results in scaling, rotation, and translation of each vertex. Once the vertex is processed, it is sent to the next stage. Once the a triangle’s vertices passed through all but last stages of the pipeline, the Rasterizer module takes the 3 vertices of the triangle generates all the pixels within the triangle, computes the color using linear interpolation, then writes the RGB color to the corresponding address in RAM. A DMA module is configured to read the framebuffer from RAM and pass it to the VGA module to be displayed. To allow for writing and reading from the framebuffer by the 2 different modules interacting with it, the design will implement double buffering to allow the Rasterizer to write freely to one of the buffers while the DMA reads from another. Once an entire framebuffer is read from memory, the DMA module switches and reads from the other buffer, and the rasterizer does the same.

2. Schematics and Drawings

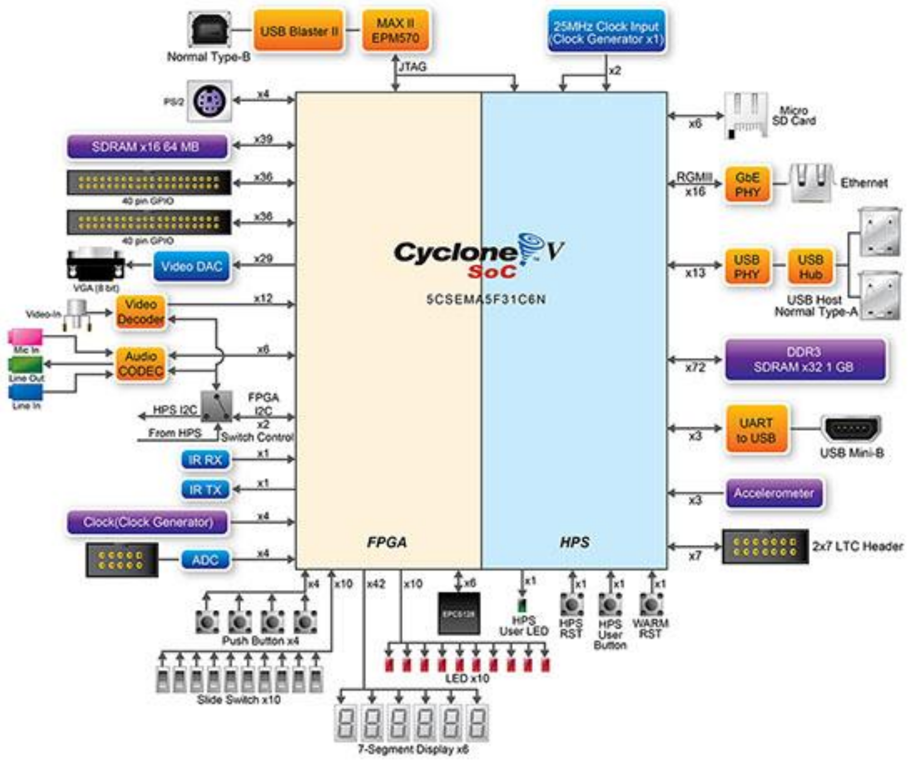

Figure 2. DE1-SoC System Overview

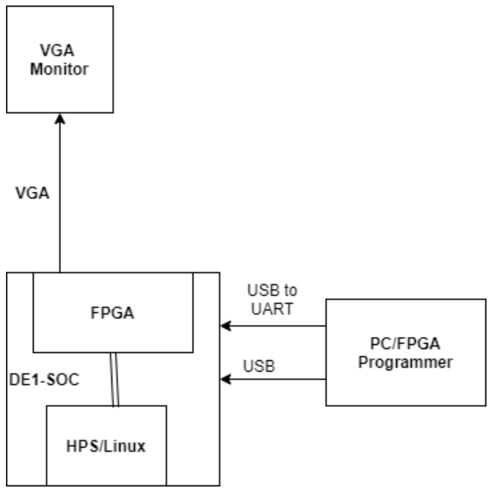

Figure 3. DE1-SoC board being used in this design

3. Bill of Materials

RefID	Part Name	Part/Catalog No	Total Cost [$]
p1	The DE1-SoC Development Kit	P0159	175
p2	Universal AC Adapter	FCB-3017-ULX	2
p3	SanDisk microSD 32GB	3695074	8.99
p4	Dell 19 Monitor	B013VEPG3W	51
p5	GCC ARM Embedded Toolchain	N/A	
p5	Linux Kernel	N/A	
p6	HPS header file with addresses for Memory-Mapped hardware	N/A	
p7	Quartus Software	N/A	

4. Manufacturing/Replacement Details

If the DE1-SoC becomes faulty, you an order a replacement board from the Altera website here.