Hardware Accelerated Shaders Using FPGA's

Luke Goddard *
National Centre for Computer Animation
Bournemouth University

Ian Stephenson †
National Centre for Computer Animation
Bournemouth University

Attempts to hardware accelerate production renderers have been restricted by the hardware available. Standard interactive graphics hardare is generally ill-suited to production style rendering, while the cost of custom hardware prevents its widespread adoption. Field Programable Gate Arrays (FPGA's) offer a solution to this problem by allowing custom circuits to be dynamically created using standard commercial hardware logic boards.

By retargeting a RenderMan SL compiler to produce FPGA circuits, we create a hardware accelerated shading engine, which dynamically rebuilds its hardware at render time, to create acceleration circuits for each surface type required, allowing complex programmable shaders to executed at performance levels approaching hard-wired shaders.

1 Field Programmable Gate Arrays

FPGA's are "soft" hardware — a required circuit is described in the programming language HDL (Hardware Description Language), and compiled. This compiled circuit description can then be downloaded into an FPGA board installed in a standard computer. Once downloaded, the circuit operates as if it had been fabricated by traditional means. This allows us to design the hardware required for production rendering and fabricate it on an FPGA board costing no more than a standard graphics accelerator.

As FPGA's are also re-programmable, new circuit descriptions can be download into them at run time. This allows the hardware to be tuned, not just for the generic task of rendering, but on a per scene, or per surface basis. By compiling RenderMan shaders from SL into HDL, we can create a hardware implementation of a specific shader. These can be loaded as required into the FPGA at rendertime, allowing fully programmable shaders to be genuinely hardware accelerated rather than simply run on an array of graphics processors.

2 Implementation

Generation of hardware descriptions is handled in multiple passes. Shaders are first compiled from SL into a custom assembly language, which describes the shader as a set of pipelined modules. An assembler then converts this description into HDL, combining standard modules with customized control hardware. The HDL compiler then turns this description into a final circuit layout which can be loaded onto the FPGA board. Figure 1 shows an overview of the shading architecture.

Shaders are constructed as a series of computation nodes connected together by a backbone of RAM. The parameters and constants of the shader are written into the RAM through memory mapped registers on the PCI bus. Each column of RAM holds one of the shaders parameters and acts as a circular buffer to provide this data to nodes that require it.

Once a point enters into the shading pipeline it is assigned a label that represents its current level of completion. As the point progresses through the pipeline, the nodes evaluate this label and only manipulate its parameters if it is at a corresponding stage. This provides a method of flow control and allows loops and conditional statements to be constructed by disabling unwanted nodes.

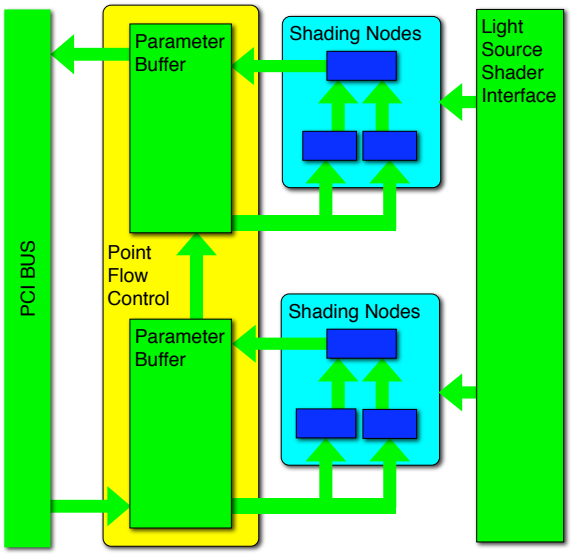

Figure 1: The FPGA Shader Architecture

Depending on its size and complexity, a shader my be divided into sequential regions, each with its own buffers. Points which have completed executing on all the nodes in one region are shifted to the next. This expands the capacity of the pipeline and improves performance by reducing the number of nodes accessing each RAM.

After finishing all of the required operations points remains circulating within the final buffer until requested. Removing a point clears space for another to enter that region of the pipeline. The shaded points are then passed back to the software system for hiding, though implementation of a simple hardware hider within the FPGA would be possible.

3 Results

The resulting shading engine is capable of executing a subset of the RenderMan Shading Language and producing images comparable to software shaded implementations. The hardware accelerated shaders can process points as fast as data can be streamed to them over the PCI bus. As the system is pipelined, more complex shaders increase latency, but not necessarily throughout.

Current generation FPGA's are still limited in the size of circuit they can synthesize, which restricts the complexity of shader which can be used. In addition reprogramming of the FPGA is slow, so performance is impaired when many different shaders are required. Both of these restrictions are improving with newer FPGA chips, but could also be alleviated by the use of mutiple FPGA boards, either splitting shaders over several chips, or using one board while the other is being reprogrammed.

*e-mail: goddardl@ntlworld.com
†e-mail: ian@dctsystems.co.uk