Procedural Texture Mapping in FPGAs

Andy Ye and David Lewis
Outline

1. Introduction to Texture Mapping and Procedural Textures.

2. Outlining the Proposed Hardware Accelerator for Procedural Textures.

3. Some Results.
Introduction to Texture Mapping

Texture
• Two dimensional or three dimensional images specified by a coordinate system.
• Texture information contained in a texture image is typically color, but can also be other physical properties like density, weight, refraction and reflection coefficients, etc.

Texture Mapping
• The process of determining the color of screen pixels using 2-D or 3-D textures and a database of 3-D surface descriptions.
Texture Mapping

Why is Texture Mapping Important in Computer Graphics?
- Greatly increase the realism of computer generated images.
- Simple to implement and consumes little CPU time.

What makes Texture Mapping Difficult to Implement?
- Textures requires a large amount of memory to store (a few megabytes to a few hundred megabytes of memory is required).
- Large memory required for texture mapping can make the graphic system expensive and significantly reduce the performance of the graphic system.

What are the possible solutions?
- Using special compression algorithms for texture mapping.
- Using special hardware accelerators to generate Procedural Textures.
Procedural Textures

What is Procedural Texture?
• Textures synthesized using computer programs.

Advantages
• Flexibility
  Can be used to model things that do not exist or no longer exist in nature (e.g. Dinosaur Skins).
• Efficiency and Realism
  Can realistically and efficiently model many fractal images like marble, wood grain, cloud and terrain.
• Compactness
  Procedural textures programs are typically a few hundred lines long and can create textures that need a few megabytes of memory to store.

Disadvantages
• Not all textures can be realistically modeled as procedural textures.
• Require much more CPU time to produce than digitized photos.
G. Y. Gardner, *Simulation of Natural Scenes Using Textured Quadric Surfaces*
Basic Structure of a Hardware Accelerator for Procedural Texture Programs

Motivation
- Using a special set of hardware to calculate procedural textures on the fly and to save valuable CPU time.

Structure
- Inputs to the hardware are a set of texture coordinates (u, v, w).
- Output of the hardware is a value that represents color, density, refraction or reflection coefficients, etc.

![Procedural Texture Algorithm Diagram]
Marble Texture Algorithm

Improvements on the Marble Texture Algorithm

- Loop unrolling.
- Table lookups for sin and cosine functions.
- Generating random number in XOR arrays.
- Substituting floating point arithmetic by low precision fixed point arithmetic.
Marble Texture Algorithm (Hardware):

Computation Complexity per Pixel:
- addition/subtraction: 73.
- multiplication/division: 24.
- Other Elementary functions: XOR arrays 48.
- Memory accesses for tables stored in memory: 0.
- 55\% of an Altera 10k50 (1597 LUTs).
1. 4-bit wide data path
2. 5-bit wide data path
3. 6-bit wide data path
4. 7-bit wide data path
5. 8-bit wide data path
6. 25-bit wide data path
Three Dimensional Wood Texture
Conclusion

Proposed a Hardware Accelerator for Procedural Textures

- Replaces the texture memory.
- Creates procedural textures on the fly.

Presented a Circuit Design for the Marble Texture