LSU EE 7700-2 **Homework 4** Solution **Due: 14 April 2008**

For the following assignment read the description of the GeForce 3 in Erik Lindholm, Mark J. Kilgard, Henry Moreton, "A User-Programmable Vertex Engine," SIGGRAPH 2001, p.149-158 and also read the description of the GeForce 6800 in John Montrym and Henry Moreton, "The GeForce 6800," IEEE Micro Magazine, vol. 25, no. 2, March 2005, pp. 41-51. Both papers are linked to the course references page, `http://www.ece.lsu.edu/gp/ref` *and require a password if accessed from outside* `lsu.edu`.

Problem 1: According Montrym 2005 the GeForce 6800 was designed specifically so that a particular resource would likely be the bottleneck. (They don't use the word bottleneck.)

Ideally there would be no bottlenecks, but there is no way a particular GPU design could have no bottlenecks for all of the different code that might run on it. This resource we are talking about is an expensive one so the designers don't want it idle unless there is nothing to do.

(*a*) What is the resource?

The resource is memory bandwidth.

(*b*) Suppose that for some candidate design and GPU code the resource is not the bottleneck. How might the design make the resource the bottleneck by *adding* to other parts of the design. (Be reasonably specific.)

Suppose the pipeline is accessing lots of memory, perhaps combining four different textures in some elaborate bump mapping scheme. Further suppose that the pipeline is running at full speed but memory bandwidth is not completely being used because something else is the bottleneck. Suppose that bottleneck is a piece of code in the fragment shader that's trying to do the equivalent of a shift-and-mask operation using only floating-point instructions. The bottleneck could be removed by adding integer fragment shader instructions. Another option would be to add additional fragment processors (without adding more memory ports).

Problem 2: Compare the vertex processor design in the GeForce 3 and the GeForce 6800.

(*a*) Describe two interesting similarities and two interesting differences between them.

Solution given in next part.

(*b*) For each difference explain why the design changed. Try to be reasonably specific.

Similarity: Both include a vector functional unit for most floating-point instructions and a special function unit for long-latency scalar operations such as reciprocal.

Similarity: Both use quad data types.

Similarity: Both have the same register organization (constants, temporaries, input registers, output registers).

Difference: The 6800 vertex processor has a texture unit.

Reason: to enable graphics techniques in which a texture is used to specify something about a vertex, for example, to displace it slightly.

Difference: In the 3-series all instructions take the same amount of time, in the 6800 series the special-unit instructions can take longer.

Reason: In the 3-series extra stages may have been added to the vector instructions so that they would take as long as the special instructions; with enough threads this does not hurt performance and it simplifies the control logic. Since the 6800 has texture access it has even greater variation in execution time, especially considering texture cache misses, and so the control logic would have to be able to deal with irregular completion times. If the control logic can deal with irregular completion times there is no need to pad the vector unit with extra stages.

Difference: The 6800 vertex processor can execute branches, the 3-series can't.

More elaborate vertex shaders require branches.

Problem 3: The 6800 has six vertex processors, each operating independently, as an MIMD (multiple instruction, multiple data) group. An alternative would be to operate the six vertex

processors as an SIMD (single instruction, multiple data) group, that is, a single PC would be used for all six VPs. (This has nothing to do with multithreading).

There are certain additions to the instruction set of the 6800 that would not be nearly as useful if the six VPs operated as an SIMD group.

(*a*) What are those extensions?

Branches.

(*b*) Why would SIMD make it difficult?

Because in an SIMD array all processors get the same instruction and so its difficult for some to have executed a taken branch and some to have executed a not-taken branch.