Projects

ECE 464 / ECE 520 / DS 510P

Spring 2003
Dr. Paul D. Franzon

My intent is the ECE 464 students work in project 1 in pairs, and ECE 520 students work on project 2 in pairs. VBEE and NTU students can either work on project 1 by themselves or on project 2 in pairs.

ECE 464 (Pairs) and Individual Off Campus student project (VBEE/NTU)

You are going to design a parameterized multiport FIFO and an LRU arbiter. Both of these are commonly used hardware modules.

MultiPort FIFO

Design a parameterized multiport First In First Out (FIFO) buffer. It is parameterized as follows:

- Width of data (N)
- Number of words in FIFO (M)

It has two input ports and two output ports. The I/O of the FIFO are as follows:

clock  // clock
reset   // clear internal address registers
        // (active low)
In1 [N-1:0]  // input port 1
In2 [N-1:0]  // input port 2
New1  // new input on port 1
New2  // new input on port 2
Out1 [N-1:0] // output port 1
Out2 [N-1:0] // output port 2
Request1 // request new data on port 1
Request2 // request new data on port 2
Full    // goes high when FIFO full

It behaves like a normal FIFO, except that if both New1 and New2 go high, In1 and IN2 are both written to the FIFO (on that clock cycle) and in that order; and if both Request1 and Request2 go high, then the next two items on the FIFO are written out to Out1 and Out2 in that order. If no request is made on a particular clock cycle Out stays the same as it was before. For example, consider the following timing diagram, for a 4-bit wide FIFO:

Clock 00110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011
Design implement and test a module with this functionality.

**LRU Arbiter**

Multiple units share a bus but only one unit can use it on any clock cycle. An arbiter is the unit that decides which unit gets to use the bus in any clock cycle. The units make their requests and then receive their grants at the start of the next clock cycle. Some algorithm must be used to decide which unit gets the bus in case of multiple requests. You are going to implement an arbiter that uses a “Least Recently Used” (LRU) algorithm. I.e. In the case of multiple requests, the requester that has not used the bus for the longest period of time gets to use it. In the case that no unit satisfies this criteria, then it does not matter which unit gets it.

You are going to implement this for a bus with 8 units connected to it.

The module I/O are as follows:

- **clock**  // clock
- **reset**  // reset the LRU tracking registers
- **Request[7:0]**  // 8 request lines
- **Grant[7:0]**  // 8 grant lines

An example timing diagram is as follows:

- **clock** 0011001100110011001100110011
- **reset** 1000011111111111111111111111
- **Request** 00 01 02 03 03
- **Grant** 00 00 01 02 01 02

**ECE 492B students are to do this project in self-selected pairs.**
**VBEE and NTU students are to turn in individual projects.**

**ECE 520 and Pair Off-Campus Projects**

*(You need to find a partner to do these projects)*

**Graphics Rendering Engine**

You will be designing part of a simplified Black and White 3-D Graphics pipeline. Attached is a description of the basic operation of a 3-D rendering engine, and a description of the Bresenham line drawing algorithm. You are responsible for building a rasterizer that implements flat shading. You will form 2-person teams for this project. Only triangular polygons will be used. Any memories you need are not to be
synthesizable or synthesized. Use a simple behavioral memory description, e.g. as shown in class.

The input memory will be organized as follows with up to 128 Polygons, each with the following entries:
- Vertices. 3 vertices, each with (x,y) coordinates, each being 7 bits long.
- Normal vector. Normal vector (x,y,z), each component being 7 bits long.
- Diffuse intensity, $k_d$, expressed as an 8-bit greyscale.

In addition, you will assume the following:
- The light source vector is (0,0,1).
- The constant ambient light $K_a$ is 16.
- The intensity of the point light source is 1.
- The specular light component, $k_s$ is 0.

The output memory will be organized as a 128x128x8-bit pixel map.

You will be expected to meet a minimum performance requirement of being able to sustain a rendering rate of 250,000 triangles per second. I recommend you use the 0.35 μm technology. To minimize “cheating” please notice the following limitations:
- The ONLY permitted memories are the ones to store the items above.
- To determine your performance, run a test case where the “display” is 100% filled with right angle triangles where the two short sides are the same length. You should have 128 equally sized triangles.
- When the edges of two triangles overlap, use the color from the lower and leftmost triangle.

i.e. step & repeat a pattern like this, over the “display” so that you have 128 triangles:

![Diagram showing triangle shading and shared boundaries](image)

The easiest way to do this is when you are “filling” the triangle, don’t fill the bottom and right sides. E.g. At “this side is yellow above” (the line drawn here is the shared edge), when you are doing the green fill, stop one pixel from the actual edge pixel, and when doing the yellow fill, fill in the actual edge pixel.

General Instructions
Preliminary Project Report

Your preliminary report is due in class on Monday, March 18 (off campus students, see the syllabus for due date) and is mainly intended as a progress report to demonstrate you are on top of important project issues. Items to be included in this report include the following:

- Block diagrams of your design, clearly identifying any design hierarchy, all registers, all module I/O. Neat block diagrams are expected, not hand-drawn sketches.
- For the ECE 520 project, any appropriate high-level (e.g. C) code showing the algorithm has been correctly captured.

Grading for this report will be as follows (out of 10):
- 9-10 : All major design elements correctly identified; behavioral code complete with test cases. Neat and clear documentation.
- 7-8 : At least one major design element missing, or C code incomplete, or confusing and poor documentation.
- 5-6 : Scrappy, but honest, capturing some elements but conveying no real understanding of the design.
- 0-4 : Extremely poor attempt.

Main Project Report and Demo

Your main project report is due on Monday April 21 (VBEE and NTU students see syllabus). There are penalties for late turn-in.

On-campus students: Turn the report in at your demonstration. Project demos will take place in the week of and after the due data.

Your project report is to include the following:
- Written description of your approach, including block diagram of your design, description of your verification strategy and a discussion of any part of the design, synthesis or verification that you consider “tricky”, novel or noteworthy.
- Specifically note and document your area and your throughput
- Full listings of the following:
  - Verilog files, including test fixture
  - Synthesis scripts
  - Extracts from View_command.log generated by design_analyzer, including the results from the read command and timing verification. (If you use dc_shell, you will have to rerun the script in design_analyzer in order to obtain this).
  - Plots from the final design.
  - Simulation run results (waveforms or equivalent
  - High level model of the design, if appropriate

You will be graded on the following factors (roughly in order of importance):
- Design correctness in terms of correct functionality
- Design correctness in terms of using good Verilog and Synthesis strategies, structuring, coding approach, etc.
- Correctness of synthesis approach
- Design elegance in terms of Verilog readability, and “coolness” factors
- Design efficiency – performance and area
- Completeness of Verification – do you just simulate a handful of cases, or has some thought and effort gone into it?
- Quality of reporting

The grading scale will be roughly as follows:

100 Excellent job. Several special features.
95 Very good job. At least one special feature.
90 Works properly. Good basic design.
80 - 89 Some significant problems that have a good probability of meaning this design would not be acceptable to an industrial team. e.g. Synthesis problems, features that did not work or poor documentation.
70 - 79 Major problems but made a good attempt, or understood very little of a working design. This includes a terrible report but design that would otherwise get an 80.
0-60 Some fraction of the above. Give something if they made an attempt.