Projects

ECE 464/ECE 520/DS 510P

Spring 2002
Dr. Paul D. Franzon

My intent is that all NCSU students work in pairs, and off-campus students work by themselves. If you wish to work by yourself, the standards and requirements will not change. I will match-make individuals in class.

Please read my comments at the end about getting started early.

1. ECE 464 students, DS510P and NTU students

A. DDR SDRAM Memory Interface

You are to design a DRAM interface unit for a single DDR SDRAM chip. The part you are to use is the Micron MT46V2M32 32-bit wide 64Mb DDR SDRAM. Specification data sheets and a Verilog model are available from www.micron.com

A simple description of the interfaces to your module set can be found below:

Briefly, the operation is intended as follows: Whenever, the address input changes, the interface is to perform a series of reads and writes to the memory, or do a refresh as appropriate. The number of reads or writes are determined by the burst length. The operation of the pins on the ASIC side are as follows:

Address[?:0] // start address of access
// YOU WORK OUT THE ADDRESS RANGE
// change indicates new access being requested
Burst[2:0] // length of burst 2, 4, 8 or full page (000)
Data[31:0] // data – bidirectional
Read/Write# // 1= read, 0=write (to the DRAM)
NextRefresh // The next address change is to initiate a
// refresh on the line containing that address
Clock // input clock
ClockDiv2 // above clock at half rate

Address // Memory Interface
Burst
Data (32-bits)
Read/Write#　　CAS#
NextRefresh　　RAS#
Clock　　Etc.
ClockDiv2　　ClockDiv4
reset
On the DRAM side, you are to provide signal interfaces for all the necessary pins specified in the datasheet, except for its clock (CK) – that can come from your test fixture but it needs to share a common origin with Clock above. Note the following:

- All outputs from this block, to either side, are to be registered.
- The data buses will need tri-state drivers outside the register.
- You can use the Auto precharge option.
- You are to use Auto Refresh.
- Use only sequential burst mode.
- You never need to change DM (but still provide it).
- Don’t take advantage of the memories ability to pipeline accesses between different banks.

For most of you, your library will not permit operation as fast as the fastest version of this memory. That’s OK. What you need to do is to work out the fastest you can pump data into and out of the DRAM successfully and then structure the rest of your design to feed that pipeline. I strongly recommend that you test this data interface on the Micron Verilog file before doing the rest of the design.

I also strongly recommend that you structure your design so that only one clock and one edge is used in each module. (i.e. You will need multiple modules for the design.)

Points will be awarded for the data bandwidth you achieve to and from the SDRAM. Make sure that this interface is as fast as you can make it.

The hardest part of this project is understanding the memory interface requirements rather than the design itself. At first the data sheet is daunting. However, it is penetrable and you working out what the interface is forms part of the project. Do not expect us to generate a simplified version of the interface requirements for you, nor explain in detail what the different signals mean and do. You are graduating soon. Being able to read and understand a data sheet is an important and expected skill.

ECE 464 students are to do this project in self-selected pairs.
VBEE and NTU students are to turn in individual projects.

2. ECE 520 and Pair Off-Campus Project
(ECE 520, NTU and VBEE should find a self-selected partner to do these projects)

Note: These descriptions are intentionally structured with high-level detail only. It is up to you to learn what you need to know to get the design done and to work out how to approach it.

A. IP Forwarding Engine

A new algorithm for IP forwarding has been developed and is available in this locker (forward_paper.pdf and forward_patent.pdf). The paper describes the hardware algorithm while the patent describes both the algorithm and how to build the table used. You are to implement this algorithm in hardware as follows:

- Use a commercial DRAM (e.g. a micron part as above) to store the forwarding table, and use an on-chip SRAM to store the compacted table. Do not synthesize the SRAM. Implement as a 2-D array as part of the test fixture.
- Implement the table forwarding algorithm in hardware. Do not implement the table generation algorithm in hardware; generate it with software and read it into the SRAM.
- Test cases will be supplied later in the semester.

The hardest part about this project is forcing yourself into the discipline of determing how to approach the design, how to structure the hardware, design the controller etc. Many of you will be tempted to simply
translate C into Verilog. If this works at all, it will produce a crappy design. DESIGN THEN CODE. Note that the 464 project is one (minor) part of this project.

**General Instructions**

**Main Project Report and Demo**

Your main project report is due on **Friday April 14** (April 22 for VBEE and NTU students). On campus students will be penalized for late turn-in (10% per week or part thereof, up to 30%). I’ll separately inform off-campus students about my late policy for them.

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

**Off-campus students:** The above date is the postmark date.

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. Include a table listing the area and performance (clock cycle and throughput) of your design.

- Full listings of the following:
  - C models of the hardware (not appropriate for the DRAM model)
  - Verilog files, including test fixture
  - Synthesis scripts
  - View_command.log generated by design_analyzer. (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) pre- and post- synthesis. I realize that our poor SDF support in the library means that you might need to do the post-synthesis simulation at a slower clock speed, e.g. half of the synthesized speed.
  - High level model of the design, if appropriate

You will be graded on the following factors (point scale indicated):

**Grading Scheme**

**High level C model.** Is this model complete and accurate? Did they use it to generate test cases? [10 points]

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No model.</td>
</tr>
<tr>
<td>3</td>
<td>Incomplete attempt</td>
</tr>
<tr>
<td>5</td>
<td>Partial model.</td>
</tr>
<tr>
<td>7</td>
<td>Modeled but not used</td>
</tr>
<tr>
<td>10</td>
<td>Meets above standards.</td>
</tr>
</tbody>
</table>

**Block Diagram.** Do they have a block diagram? Is it complete, useful and accurate? Does it comply with the code? Is it neat and readable? [10 points]

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No diagram.</td>
</tr>
<tr>
<td>3</td>
<td>Incomplete attempt</td>
</tr>
<tr>
<td>5</td>
<td>Looks “OK” but scrappy.</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Meets above standards.</td>
</tr>
</tbody>
</table>

**Design.** Is the underlying design a good one? Is reasonable use made of pipelining? Is the design efficient in use of resources? Is the partitioning leading to fast synthesis? Are there no “dumb” things such as chains of adders, poorly shared resources, etc. Is it clear that they designed before coding? [20 points]
**Verilog Code.** Does the code conform with practices described in class? Is the in-code documentation useful and complete? Is the code readable? [10 points]

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Clearly thought through.</td>
</tr>
<tr>
<td>7</td>
<td>Design OK.</td>
</tr>
<tr>
<td>5</td>
<td>Design marginal</td>
</tr>
<tr>
<td>3</td>
<td>No real design. They just “coded.”</td>
</tr>
<tr>
<td>0</td>
<td>Does not work.</td>
</tr>
</tbody>
</table>

**Verification.** They get 5 points if it complies with the pre-supplied test cases, 0 if it does not. They get 10 points if it also complies with the hidden test case. This is for the 520 project. For the 464 project give 10 points if the verification looks reasonable, 5 if it is barely OK, 0 if they did not bother. [10 points]

**Synthesis Correctness.** Is the code synthesizable? Does the code pass all the LINT checks after the read command? Does the final design meet timing? Does the synthesis look “normal”? Again this is very binary (well ternary). 0 points if the code if there is no netlist produced. 5 points if there are ANY problems likely to produce a non-functioning design (LINT, HDL errors or does not meet timing). 10 points if everything is OK.

**Report.** Is the written report clean, concise but complete and accurate? [10 points]

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>High quality in this range.</td>
</tr>
<tr>
<td>7</td>
<td>Scaps of paper get 3-5 points.</td>
</tr>
<tr>
<td>3</td>
<td>No report.</td>
</tr>
</tbody>
</table>

**Optimization.** Is there anything elegant or noteworthy about the design? An extra efficient implementation? Extra performance or very low area? A very complete verification? Note the area and performance and compare it with the class average. [10 points]

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Efficient, “cool” or special</td>
</tr>
<tr>
<td>7</td>
<td>Average Area and Timing</td>
</tr>
<tr>
<td>3</td>
<td>Not very efficient.</td>
</tr>
</tbody>
</table>

**Individual Contribution.** Each individual’s score can vary by up to 10 points each way. Give 10 points if it is clear that each individual contributed fairly equally. Give 0-5 points if they contributed but it was substandard. Note why. Take up to 10 points off if it was clear the individual was free-loading and knows nothing about the design. [-10 to +10 points]

[100 points total]

**Some Generalities**

**Get started early.** The ECE 464 project should take about 20-40 hours per person while the ECE 520 project is around four times larger. You can not hope to finish this in one night, let alone one week. Most really good projects are done twice, the first time working out what not to do. In addition, we only have finite tool licenses and compute cycles. If everyone waits until the last minute you might be spending hours waiting for a Synopsys license or watching a server thrashing.
Also, I do expect you to partition tasks between team members and coordinate back together at the end.

Other Information

Additional information to help you do the project will be posted periodically in the notices section of the class website.