In my previous instructable, we saw how to design a simple direct mapped cache controller. This time, we move a step ahead. We will be designing a simple four-way set associative cache controller. Advantage? Less miss rate, but at the cost of performance. Just like my previous blog, we would be designing and emulating an entire processor, main memory and cache environment to test our cache controller. I hope you guys find this as a useful reference to understand the concepts and design your own cache controllers in future. Since the model for processor (test bench) and the main memory system are exactly the same as my previous blog, I won’t be explaining them again. Please refer to the previous instructable for details regarding it.

**Step 1: Specifications**

Quick look thru the specifications of the **Cache Controller** presented here:
Step 2: RTL View of the Entire System

Complete RTL representation of the Top Module is shown in the Figure (excluding the processor). Default specs for the buses are:

- All Data Buses are 32-bit Buses.
- Address Bus = 32-bit Bus (But only 10 bits are addressable here by the Memory).
- Data Block = 128 bits (Wide Bandwidth Bus for Read).
- All components are driven by the same clock.
Step 3: Test Results

The Top Module was tested using a Test Bench, that simply models a non-pipelined Processor, just like we did in the last instructable. The Test Bench generates Read/Write Data requests to the Memory frequently. This mocks typical "Load" and "Store" instructions, common in all programs executed by a processor.

The test results successfully verified the functionality of the Cache Controller. Following are the test stats observed:

- All Read/Write Miss and Hit signals were generated correctly.
- All Read/Write Data operations were successful in all four ways.
- pLRU algorithm is successfully verified for the replacement of cache lines.
- No data incoherence/inconsistency problems detected.
- The Design was successfully timing verified for a Maxm. Clock Frequency of operation = 100 MHz in Xilinx Virtex-4 ML-403 Board (whole system), 110 MHz for Cache Controller alone.
- Block RAMs were inferred for the Main Memory. All other arrays were implemented on LUTs.

Step 4: Attached Files

Following files are attached here with this blog:

- .VHD files of Cache Controller, Cache Data Array, Main Memory System.
- Test Bench.
Notes:

- Go through the documentation for full understanding of the specifications of the Cache Controller presented here.
- Any changes in the code have dependency on other modules. So, the changes should be done judiciously.
- Pay attention to all the comments and headers that I have given.
- If for any reason, Block RAMs are not inferred for the Main Memory, REDUCE the size of the memory, followed by changes in address bus widths across the files and so on. So that the same memory can be implemented either on LUTs or Distributed RAM. This will save the routing time and resources. Or, Go to the specific FPGA documentation and find the compatible code for Block RAM and edit code accordingly, and use the same address bus width specifications. Same technique for Altera FPGAs.

For queries and feedback,
mail me: iammituraj@gmail.com

Mitu Raj

https://www.instructables.com/FC13QH7J/VUTDZET/FC13QH7J/VUTDZET.rar
Would you please give me the password?

Password????Thanks

Hello, what's the password for this file?

what is the password for this file?

hope you got it

password?

What's the password? Thanks

Like @shrigizmo said, need the password to access the files.

hope u got it