Project Guidelines

Project Schedule

10/03/2012	**Submit team member form**
11/15/2012	**Proposal**
01/22/2013	**Project Demo**
01/24/2013	**Final presentation**

Project Description

In this term project, you will work as a team (with 4 ~ 7 members) to design a general or application-specific embedded processor that can work with a compiler and communicate with external modules (either memory or accelerators for a specific application.) via on-chip buses. Requirements are described as bellowed:

A 0.18um technology is provided as the basis of this project. Lessons learned here can be applied to other technologies.

Basic requirements (70%)

- **Must be a pipelined structure and compatible to the target processor. This course supports Andes N9. If chosen otherwise, the selected processor shall be at least equivalent to the same class of Andes32 and with equivalent instructions.**
- **Must be compatible with a compiler and a debugger of the target processor.**
- **Data width shall fit the specification of the target processor and shall be at least, i.e., 32 bit.**
- **Operating speed is at least 40 MHz for post-synthesized version. (The best ever in this class run 250 MHz.)**
- **Its instruction set shall have at least 35 instructions, including branch, I/O instructions.**

- **There shall be at least 8 addressing modes.**
- **The address space for I/O controller registers is part of the same address space as data memory address, i.e., I/O registers are memory mapped.**
- **External main memory shall use non-ideal latency model, including data memory and instruction memory. The delay shall be set to either at least 4 times of your processor clock, or at least 80ns. It shall be noted that memory does not need to be synthesizable, however, they shall have the specified delay in behavior.**
- **Include interrupt mechanism and interrupt service routine for handling requests from other devices, such as sending data and control signals to DMA controller, or receiving data and control signals from DMA controller.**
- **Must provide performance counters, as those specified in homeworks, inside the processor.**
- **Gate count shall be larger than 20,000.**
- **Must work with the target compiler without problem for straight code.**
- **Must implement and verify L1 Cache and Forwarding.**
- **Must implement and verify at least 1 hardware interrupt and interrupt service routine by receiving data from external devices.**
- **Must complete code analysis by nLint and reach at least 95% error free.**
- **Must verify every instruction individually with testbenches learned in homeworks, specifically capability of self-verification with error locations and error messages.**
- **Must verify by sorting 100 numbers stored in external memory or any equivalent test bench.**
- **Must verify by calculating Fibonacci series from F_0 to F_{40} stored in external memory.**
- **Must verify by a to-be-announced program specified by TA later.**
- **Must verify by running a sequence of instructions (at least 1000) that completed a meaningful function based on your desired target application. If your design is for a general processor, a meaningful function is like add up numbers from 1 to 20 and**

store back to external memory, or perform NxN matrix multiplication, etc. If you target 3D-graph application, a meaningful function would be to compute pixel position for a cube when rotating 60 degree horizontally.

Advanced features (30%)

- **Complete and verify at least 20 more instructions other than those in basic requirements and include instructions facilitating 64-bit addition/subtraction & store/load.**
- **Must verify by calculating Fibonacci series from F_0 to F_{90} stored in external memory.**
- **Complete code analysis by nLint and reach at least 98% error free.**
- **Implement stack or other mechanisms to facilitate function calls or recursive function.**
- **Add L2 Cache.**
- **Add a DMA.**
- **Develop AHB bus accessible from the processor with a random access time (1 or 2 cycles). It shall be note that AHB could be just a behavioral model, i.e., not need to synthesize.**
- **Design dynamic branch prediction**
- **Boot from an external ROM with a simplified O.S.**

Bonus (15%)

- **Complete layout using IC compiler****
- **Perform post-layout simulation** and show all benchmarks work correctly**

NOTE:

- **The course will provide a technology file for simulation and synthesis later.**

****For those want to do chip layout, your team shall need to access**

TSMC 0.18um technology in your own lab since we are not allowed to provide this technology for this course.