ECE 4170: Introduction to HDL with Applications to Digital System Design

School of Electrical and Computer Engineering
Georgia Institute of Technology
Spring 2007 Semester
Instructor: Prof. Sudhakar Yalamanchili

Template for Project Functional Design

Submission Date: April 11th 2007
Have Questions?: Please e-mail or meet TA
Your functional design report needs to have the following sections.

1. Functional Design of Computation Core

Describe the internals of the Computation Core in the form of a datapath. Identify the components in the datapath that will be active depending on the current or past values of the Computation Core's input signals. For some projects it may be useful to describe the behavior of the Computation Core as a state machine. You should include all major computational components, as well as the types of data these blocks will use as input and output. Include a short description of what functions each component will be responsible for, and address relevant timing issues (i.e., how the component will stay synchronized with other components).

1) *Datapath Components:* Describe the major pieces of the datapath, including general functionality, and I/O for each piece (essentially an entity description). Provide a register transfer level (RTL) schematic with all signals identified.
2) *Inputs & Outputs*: Describe in detail the **contents** and the **format** of the input-data and the output-data of the application running on the SoC.
3) *Timing Issues:* Describe how different components will stay synchronized with respect to their data, and whether state machines will be used.

2. Partitioning and Soft Processor Integration (Optional)

As part of the previous section you would have described the application functionality that would be implemented on the custom-hardware. In this section describe the details of the hardware necessary to implement that functionality.

Describe the roles of the Wishbone-Interface-Controller (WIC) and the Computation Core *as they apply to your application*. Also describe the purpose of all the interface signals between the WIC and your computation and how each of the blocks (the Wishbone-Interface-Controller and the Computation Core) react to changes in values over the interface signals. For example, if an input signal to a block is asserted describe the output signals that are asserted by the block in response, and why they are asserted. The purpose of having a separate WIC component is to separate the details of generating memory-addresses and the details of the Wishbone protocol from the design of the datapath that does the actual computation. The datapath is implemented in the Computation Core. Technically it should be feasible to replace the WIC with a different

interface-controller while keeping the Computation Core unchanged, and be able to connect the resulting custom-core to a completely different kind of bus.

1) *Partitioning of Functionality*: Describe what parts of the application will be mapped to the CPU (S/W partition) and the Custom-hardware (H/W partition).
2) *Intermediate sharing and transfer of data*: Describe specifically what data will be shared or transferred between the S/W and H/W partitions.
For each type of data-transfer:
a. Show when the transfer will happen.
b. Describe the mechanism for transfer of data (CPU-initiated reads/writes to memory or custom-hardware, Custom-hardware initiated reads/writes).
c. The size of the transferred/shared data (1-2 words or blocks of a larger size).

3. Schedules and Milestones

Identify the timing of the following milestones that you need to meet in order to complete this project. The milestones should be associated with the following tasks:

1) Custom-Hardware Implementation and Testing (ModelSim)
   a. VHDL implementation of the *major individual components* of the custom-core (have a milestone for each important component)
   b. *Integration testing* of multiple components. These include:
      i. Some sub-set of the datapath (*describe which*)
      ii. The computation-core as a whole
      iii. The WIC (if applicable)
      iv. The complete custom-hardware
      v. The custom-hardware attached to SoC reading and writing from memory
2) S/W implementation and Testing (for OpenRisc only)
   a. Running application on a host PC: Testing the part of the application that will be mapped to the SoC.
   b. Design of APIs (function-calls or pointer-based read-write sequences) that would be used by the S/W partition (i.e. 'C' code running on the CPU) to communicate with the custom-hardware.
   c. Running the S/W partition on the SoC CPU (simulating under ModelSim).
3) Completion of full integration testing (ModelSim) : You would need to verify correct execution by using the reference input data-sets and comparing the produced outputs against the reference output data-sets. The output reference data sets are produced by a software reference implementation.

4) Synthesis and test (Xilinx ISE): Synthesize the design and test using the Xilinx ISE environment. You are free to pick any part and speed grade, although faster speed grades will lead to better clock cycle times at the expense of high cost.

Schedule: For each milestone assign a date targeted for completion. Plan your work so that the Full Integration Testing milestone has a target date no later than **April 26th**.

Prioritization: Give a higher priority to tasks/milestones relating to the completion of your custom-hardware and testing with ModelSim. **As a minimum, you need to have the custom-hardware implementation completed and tested in ModelSim by the end of the semester.**