COE608 Computer Organization and Architectures
Winter 2017

Test Set-Up Documentation for CPU Simulation (Lab6)
CPU Implementation and Emulation on DE2 Board (Bonus)

This document is provided to assist with the creation of a final Quartus II project for the testing of the Semi-RISC CPU. The document describes the provided support files and how they are to be used in conjunction with the CPU VHDL files to create a working CPU on the DE2 board for bonus marks. A section will also discuss setting up the simulation-only project to test CPU operation for Lab6 before programming it onto the board.

1. Available Support and Files

To test the hardware implementation of the CPU certain utilities are needed. To observe the operation of a series of instructions, the Program Counter, as well as the contents of the two working registers, A and B has to be observed. In addition, an instruction memory module is required to feed instructions to the processor. The top-level entity file (CPU_TEST_Sim), observation utilities for the Program Counter and Registers (7-segment decoder and LCD decoder/controller) and a memory component generated using Altera's MegaCore library set are provided. All that remains for the creation of a complete system is the instantiation of the CPU design into the top-level entity file, and the configuration of the instruction memory unit with the appropriate instructions stored (the program to be executed).

The complete set of files required for the project are provided together. It includes a .**bdf** file, as well as VHDL files for the various components to be used in the design. All these files should be copied from the course directory to a local project directory, and all should be made part of the project.

2. Top-Level Simulation Entity File (Lab 6 Demo - Required)

The file *CPU_TEST_Sim.vhd* is provided for a simple simulation of the CPU. This file can be set as the top level entity file to allow you to perform simulations quickly. This file does not contain any display utilities, since information during simulation will be present in the generated waveform simulation reports. The file *CPU_TEST_Sim.vhd* consists of two components: your completely assembled CPU (cpu1) and the instruction memory unit (system_memory). This cpu1 place-holder CPU component is present inside the file, and indicates how the CPU should be connected to the system. Therefore you are to create a cpu1 file as specified by this component's entity and connect your datapath, control unit and reset circuit accordingly. Because this *CPU_TEST_Sim.vhd* file is used for simulation, all CPU outputs are connected to ports. In addition, to facilitate debugging, the instruction state indicator (T) is also present as an output. This allows you to determine what state the CPU is in during simulation. *DE2_pin_assignments.qsf* pin assignment file must be included in the design prior to compilation by using the **Assignments => Import Assignments...** option.

Please note that the simulation top-level file has two clock inputs, one for the memory and one for the CPU. The memory clock should be set to a maximum frequency of 20 MHz (50ns period), or lower. The CPU clock should be set to a frequency (**AT LEAST 2 TIMES SLOWER**) than the memory clock (100ns minimum). The data memory module implemented in lab4a must also be set to the clock frequency as the instruction memory (i.e. 50ns).

Figure 1 shows an example simulation output, as well as how the clocks have been set. This is what you should expect to see during simulation.

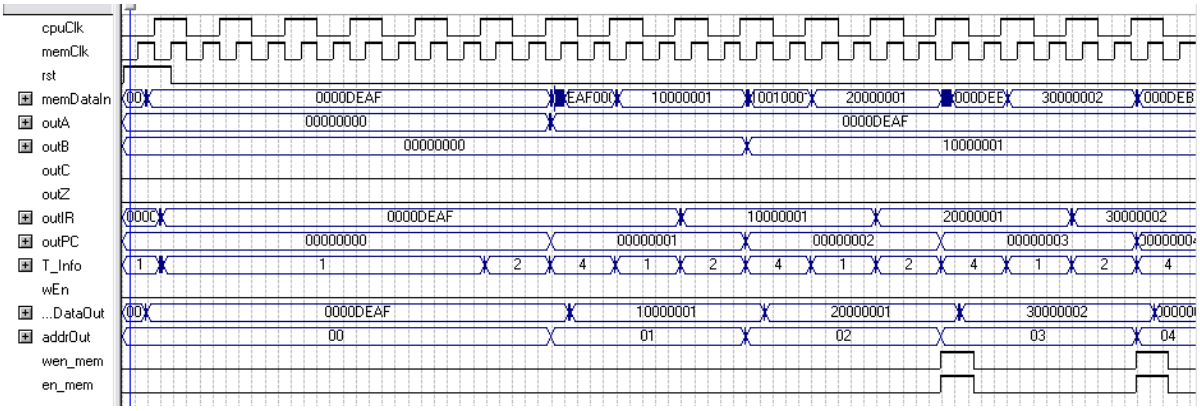

Figure 1: Sample Simulation Results

3. System Memory for CPU Simulation Setup

To permit simulation as well as emulation (hardware testing) of the CPU design, an instruction memory module is required. The students are provided with a memory specification file implementing a RAM (64 words of 32-bit each) MegaCore function. The files describing this device are *system_memory.bsf*, *system_memory.cmp* and *system_memory.vhd*. As Figure 2 shows, the memory is fed directly from the on-board 50 MHz clock, and uses the 6 least significant bits of the Address Register output to access 64 memory words, each 32-bits wide. No changes need to be made to the provided setup.

NOTE: since each memory word is 32-bits wide, it is not necessary for the Program Counter to increment by 4. The code for the program counter can be changed so that incrementing by 1 is achieved. Alternatively, all instructions must be placed in memory at intervals of 4 words.

To specify the data to be loaded into the memory unit a special file known as the **Memory Initialization File (MIF)** must be used. The student must **create a new file** of this type (File - New - Memory Initialization File), and give it an appropriate name as specified in CPU_TEST_Sim. During creation, the memory file size will have to be specified; a size of **64 words**, with a word size of **32 bits** should be specified to match the memory unit. Once open in the editor, the file permits the specification of data values at each of the 64 locations. It is up to the student to determine what values should be loaded in each location to implement instructions or a program. The CPU_specification and lab6 documents should be consulted for further information on simulation and emulation of instructions.

To specify the above MIF file as the input of the memory unit, the RAM unit itself will have to be "edited". This is achieved by using the MegaWizard Plug-In Manager. Then go to Project Navigator/Hierarchy and open project. Then right click on system memory:main and choose MegaWizard Pliug-in Manager. Under Tab 1: Parameter Settings, the Mem Init sub-tab should be selected, and the "Yes, use this file for the memory content data" option should be enabled. Your newly-created MIF file should be specified as the memory input. Once the MIF file is specified, finish the process and confirm all the changes to be made. Please note that once a file has been specified as input for a memory, this step need not be repeated when the MIF file is edited at a later date. All that is required will be the project to be **recompiled** whenever a change is made to the MIF file. Also, because the original system memory file was created using an older version of Quartus II, it is possible that the block-diagram file (.bsf) of the memory changes once you edit your RAM unit. The result of this change is that the new diagram will have the input and output pins listed in a different order (for example, the data input will be listed first, rather than the address input). If this is the case, you will have to reconnect these memory signals appropriately if you decide to implement the bonus, as connected in Figure 2.

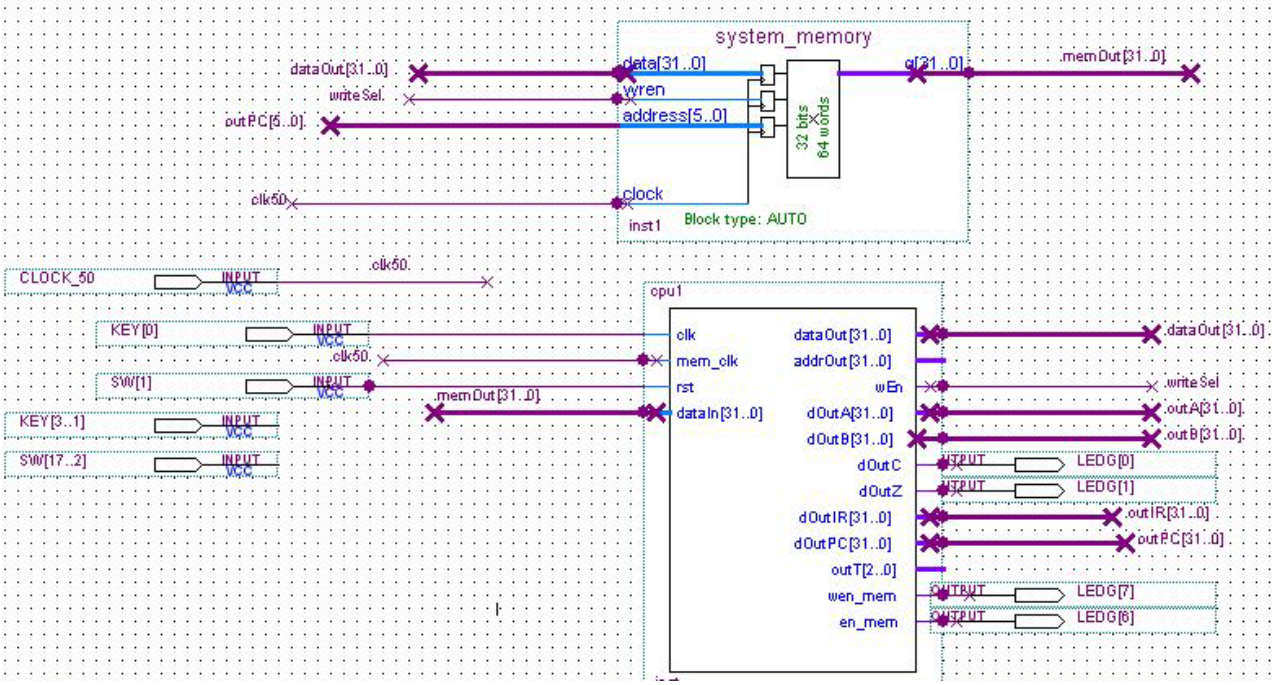

Figure 2: Memory and a Sample Processor Module

4. Top-Level CPU Implementation and Emulation Entity File (Bonus only)

A top-level block diagram .bdf file is provided in the /courses/coe608/bonus directory. This folder contains instances of the LCD control unit, the 7-segment decoder, and the required output pins to generate a complete hardware implementation of the CPU_TEST_Sim. It also contains the block diagrams and pins necessary to emulate the CPU.

NOTE: the provided document DOES NOT contain output pins for simulation purposes. A separate VHDL file (CPU_TEST_Sim.bdf) is provided and should be used as the top-level entity for emulation purposes.

Ensure that CPU_TEST_Sim.bdf is set to top-level entity for this bonus section, and that the vhdl file used in lab6 is not included - these two files will conflict. The top-level entity can be divided into two portions: the display section, shown in Figure 3 and the actual CPU instantiation section shown in Figure 2. The display section needs no additional editing on the part of the student. However, it is important that the names of the wires provided in the block diagram file are not to be changed; if the names are changed, the corresponding changes will have to be carried out throughout the design.

Figure 2 shows how a CPU design would be instantiated into the design. It is expected that the CPU unit will incorporate at least the following ports:

- outputs for the two working registers A and B
- Outputs for the status registers C and Z
- output for IR
- the output of the Program Counter
- the Write-Enable control signal for instruction memory
- the wen_mem and en_mem signal for reading and writing to data memory
- the data output and input from memory
- Clock and Reset inputs.

It will be up to the student to correctly connect these wires to the ports of their specific design.

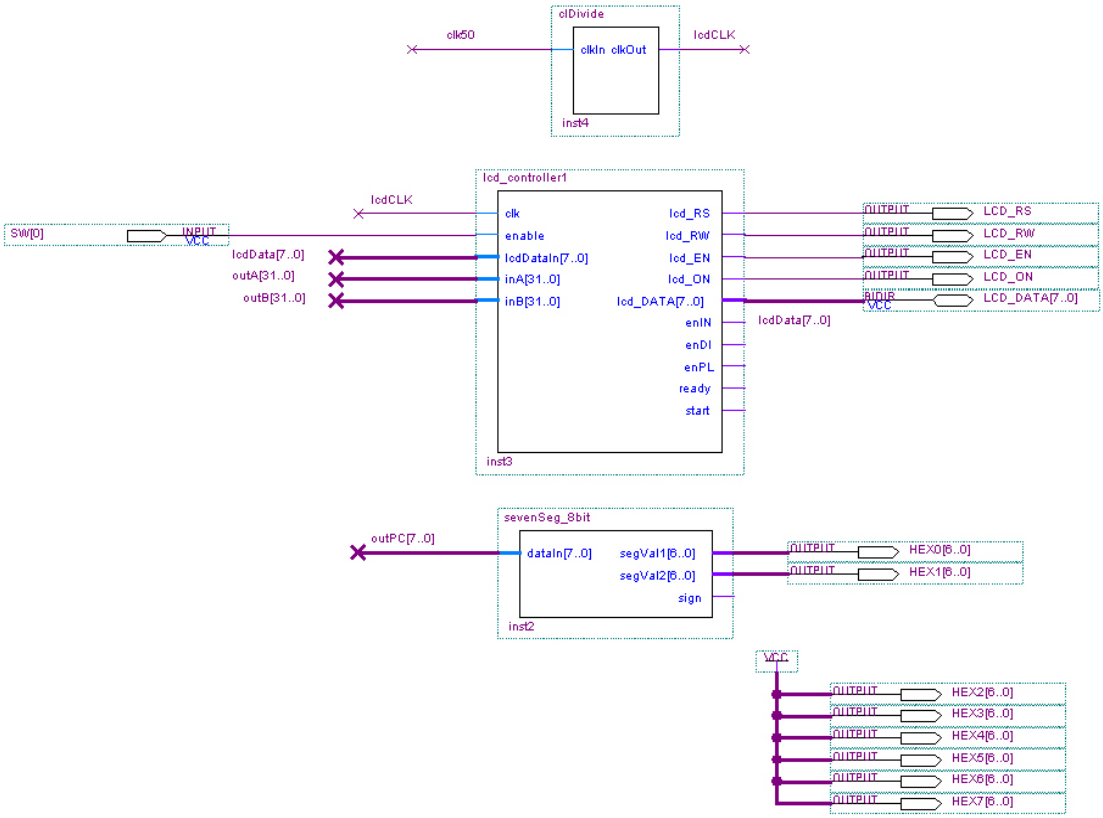

Figure 3: Display Section of .bdf

Please note that the *DE2_pin_assignments.qsf* pin assignment file must be included in the design prior to compilation by using the **Assignments => Import Assignments... option.**

The actual CPU clock will be generated by the students via the push-button KEY[0], and will be separate from the memory clock. As well, the reset signal will be implemented using the regular switch SW[1]. SW[0] enables and disables the LCD. LEDG[0] will light if a carry out flag is produced by the instruction, whereas LEDG[1] will output when a zero flag is encountered. Similarly, LEDG[7] and LEDG[6] will quickly flash when data has been successfully stored to the data memory (wen_mem and en_mem). The LCD will display register A and B's current data, and HEX[1..0] with the PC value of the CPU. Ensure that these inputs, outputs, and signals in the CPU_TEST_Sim.bdf correspond to this emulation description if pursuing this bonus emulation section. Note that the cpu1 block symbol will have to be generated based on your design and incorporated into the CPU_TEST_Sim.bdf provided. The system_memory.bsf may also need to be included in your project directory for incorporation into the top level bdf.