

**DUE:** October 1 (Friday)

The purpose of this project is to be familiar with the Altera DE2 board and Quartus II/ModelSim-Altera software tool. The chapter numbers refer to the book “Rapid Prototyping of Digital Systems,” 2<sup>nd</sup> edition (i.e., the “Altera book”). In every step of all projects, you need to turn in a brief statement about the project final status. For example, “The project works.” or “The project did not work because ....”.

1. (15%) Create a project about a Boolean function  $F(A,B,C) = A \text{ AND } (B \text{ OR } C)$  in VHDL, simulate the design, and load it into the Cyclone II chip on the DE2 board. The inputs are connected to three push buttons and the output is connected to an LED. Refer to pages 28 and 29 of DE2 User Manual (DE2 user manual/DE2 UserManual.pdf on the DE2 System CD) for the pin assignment. You need to turn in the following: (1) the VHDL file and (2) a simulation waveform printout.
2. (10%) Create a project for the simple computer design in Chap. 8. (The VHDL source file, SCOMP.VHD, can be found in the Altera book CD.) Compile and simulate the design. You need to do the following prior to compilation:
  - Replace `lpm_ram_dq` with a dual-port RAM generated by using “MegaWizard Plug-In Manager” (under “Tools” of Quartus II menu items). The input data and address to the RAM should be registered (synchronized with clock) while the output data should not be. The signal `memory_address_register` should be connected to both address ports.
  - Add the VHDL file for the RAM to the project.
  - Instantiate a memory component. Note that you should use an inverted clock to access the memory. (That is, use “not clock” in the port map.)

For the simulation, use a large enough “End Time” so as to verify the execution of the “loop forever” instruction. Turn in the following: (1) a simulation waveform printout and (2) the amount and percentage of hardware resources used (in terms of Logic Elements, I/O Pins, and Memory Bits). The hardware resource usage information is contained in the compilation report.

**DUE:** October 13 (Wednesday)

The purpose of this project is to modify the simple computer design so that it supports a single step operation. Again the chapter numbers refer to the Altera book.

1. (10%) Use the DEC\_7SEC component from Chap. 5 to create a design project. The design should be loaded into the Cyclone II chip so that, whenever a push button is pressed down, a hexadecimal digit using a seven-segment display increases its value by one (starting from an initial value of zero). Refer to pages 28 to 33 of DE2 User Manual (DE2\_user\_manual/DE2\_UserManual.pdf on the DE2\_System CD) for the pin assignment. (The four push buttons on DE2 are already debounced.) Note that, for the DEC\_7SEC component (a 7-segment display decoder) in Chap. 5, segments a to g in the VHDL code correspond to LED segments 0 to 7, as shown on p.30 of the DE2\_UserManual. Turn in the following: (1) the VHDL file(s), excluding source files from Chap. 5, and (2) the amount and percentage of hardware resources used (in terms of Logic Elements, I/O Pins, and Memory Bits).
2. (15%) Create a project for a modified simple computer design based on the processor in your first project derived from Chap. 8. The processor does not advance to the next instruction until a push button is pressed down. The 8-bit OP code of the current instruction should be displayed using two digits of seven-segment display. Use Hex5 and Hex4 on the DE2 board for those two digit display. Turn in the following: (1) the VHDL file(s), excluding source files from Chap. 5, and (2) the amount and percentage of hardware resources used (in terms of Logic Elements, I/O Pins, and Memory Bits).

**DUE: October 27 (Wednesday)**

(25%) The purpose of this project is to modify the TOP\_SPIM processor design so that it supports pipelined operations. Again the chapter numbers refer to the Altera book.

Use components from Chapter 13 to create a design project and then modify the processor design to be pipelined. Neither data forwarding nor hazard detection circuit needs to be implemented. You need to use MegaWizard Plug-In Manager to generate *synchronous* memory components to replace the instruction and data memories.

Turn in the following: (1) VHDL file(s), excluding unmodified source files from Chapter 13, (2) the test program in the instruction memory (i.e., “program.mif”), (3) a simulation waveform printout, and (4) comments on why the simulation printout demonstrates the correctness of the design.

**HINTS:** The following quotes from p. 23 of the Altera book are relevant to the design problem.

- Synthesize and check the control module first, since it is simple to see if it works correctly when you add the pipeline registers.
- To minimize changes, pipeline registers must be placed in the VHDL module that generates the input to those registers. As an example, all the pipeline registers that store control signals must be placed in the control module.
- Use the following notation to add new latched signals. Add a “D\_” in front of a signal name to indicate it is the *input* to a D flip-flop used in a pipelined register. Signals that are *input* to two successive flip-flops would be “DD\_” and three would be “DDD\_”. As an example, *instruction* would be the registered version of the signal *D\_instruction*.
- Add signal and process statements to model the pipeline modules – see the PC in the ifetch.vhd module for an example of how this can work. If necessary, you may move MUXes to different modules.
- Some control signals must be reset to zero, so use *synchronous reset* for these pipeline registers. Critical pipeline registers with control signals such as regwrite or memwrite should be cleared at reset so that the pipeline starts up correctly. The MIPS instruction ADD \$0, \$0, \$0 is all zeros and does not modify any values in registers or memory. It is used to initialize the IF/ID pipeline at reset.
- Since only eight registers are implemented in the MIPS VHDL model, use the following modified test program. You need to modify “program.mif” which is to be used inside the instruction memory (in “ifetch.vhd”). Refer to “control.vhd”, and “execute.vhd” for relevant machine codes.

LW	\$1, 1(\$2)
SUB	\$3, \$4, \$5
AND	\$2, \$7, \$6
OR	\$4, \$4, \$5
ADD	\$5, \$6, \$7

**DUE:** November 12 (Friday)

The purpose of this project is to modify the pipelined TOP\_SPIM processor design from Project#3 so that it can handle data hazard with data forwarding. Again the chapter numbers refer to the Altera book.

(25%) Modify the processor design to add data forwarding and data hazard detection circuits. There is no need to handle control hazards. You need to design a test program that allows you to tell if the design works on the board. Turn in the following: (1) VHDL file(s), excluding unmodified source files from Chapters 5 and 13, (2) the test program (i.e., “program.mif” that contains appropriate comments about the instructions), (3) a description about why the test program can verify the newly added circuits, (3) a simulation waveform printout, and (5) comments on how the simulation results demonstrate the circuits work.

(10% Extra Credit) You need to fully complete it to get any credit.

Devise a way to verify that the processor works on the board. Once you verify it, turn in (1) a description about how you verify that the processor works on the board, and (2) VHDL files that are added or modified for the extra credit.