Lab 3
MD5

Prof. Kredo

Due: By 23:59 Friday, March 27

Introduction

In this lab you will accomplish several goals:

- Extend Lab 2 to compute an MD5 digest
- Implement a basic state machine
- Verify module operation by writing testbenches
- Test your design on the DE1-SoC board

Lab 3 must be done individually.

Requirements

Your objective is to extend Lab 2 so that your system computes the MD5 digest of the 128-bit value entered by the user. You will accomplish this by performing the operations outlined in Lab 2 when the `SW9` is low and then computing the MD5 digest after the user toggles `SW9` high and presses the *Action* button. Thus, your Lab 3 implementation will have two modes of operation: (1) identical to Lab 2 when `SW9` is low and (2) compute MD5 digests when `SW9` is high.

The normal sequence of events is provided below. You will not be tested beyond this process.

1. Ensure `SW9` is in the low position
2. Reset the board with the *Reset* button
3. Enter or modify a 128-bit value as described in Lab 2
4. Raise `SW9`
5. Press the *Action* button (which causes your system to compute the MD5 digest)
6. Press the *Left* and *Right* buttons to display the MD5 digest
7. Lower `SW9` and continue at Step 3

Since your system must be able to repeatedly compute MD5 digests, you should not modify the 128-bit value input by the user when you compute the digest. When `SW9` is high, the *Left* and *Right* buttons should scroll through the MD5 digest similar to the operation of Lab 2.

I/O

Use the following buttons for the system operations. The *Capture* button from Lab 2 now becomes the *Action* button.

Function	Physical Button
Action	KEY3
Left	KEY2
Right	KEY1
Reset	KEY0

The DE1-SoC board contains a 50 MHz clock for use in your system. **Do not divide or modify the clock unless necessary to meet timing requirements.** Your system should run with a single clock.

Testbench

For each module in this lab, you must write a testbench that verifies the functionality of the module. You do not need to test every possible input combination (for example, testing all 256 combinations of data from the switches), but your testbenches should thoroughly test your modules. When writing your testbenches, follow the guidelines below.

- Include an error signal that your testbench raises whenever an error occurs.
- Clearly comment your testbench code to document the exact behavior you are testing.
- Ensure your tests cover all paths through your module description.

Your top-level module should contain very little or no logic. For this lab, you do not need a testbench for your top-level module.

Evaluation

Submit your assignment by creating a project archive in Quartus II by selecting 'Archive Project...' from the Project Menu. Submit only your project archive (qar) file. Before archiving your project, ensure the 'File set' is 'Source Control' by clicking the 'Advanced...' button in the Archive Project window.

Your lab will be evaluated based on:

- 10 points: Performs Lab 2 functionality
- 20 points: Separate system mode when `SW9` high
- 40 points: Computes correct MD5 digest
- 10 points: System repeats without reset
- 20 points: Testbenches

Hints

- Test and debug in steps. Start with a subset of the lab requirements, implement it, test it, and then add other requirements. Performing the testing and debugging in steps will ease your efforts. In general, the Evaluation requirements are ordered in a way to ease this implementation process.
- Write your testbenches early and verify each module works before integrating it into the system and before programming your board.
- Altera has many free online video tutorials to help learn the Quartus II software suite.

Synopsis Constraints File

Put the lines below in a text file with the extension sdc, such as lab3.sdc, to inform Quartus about the clock in your system. Replace `YOUR_CLOCK_NAME` with the name of your clock signal in your top-level module.

```
create_clock -period 20 [get_ports YOUR_CLOCK_NAME]
derive_pll_clocks
derive_clock_uncertainty
```

If you use a PLL, then put the undivided clock name as `YOUR_CLOCK_NAME` and not the slower, divided output clock.