1 Introduction

In this assignment, we will modify the implementation of the prefix sum from PA3 using a two-address microcode design. You will design the microcode ROM, the sequencer design, and make some modifications to the main datapath to comply with the new design. Each section is described in detail below.

2 Design Overview

Figure 2.1: Overall architecture of the assignment.
Figure 2.2: State machine for prefix sum logic.

The following modules are included as a part of PA4:

1. **array.vhd**: This serves the same purpose as in PA3. This module is what allows the use of VHDL arrays as inputs/outputs, and as such should not be modified.

2. **pa4.vhd**: This is the new top module to be used. As in PA3, it contains the I/O ports to the test bench, the required port maps for each module, and the logic for ready. You are not required to modify this file, although you should look through it understand how the modules are stitched together.

3. **pa3_datapath_ext.vhd**: This module is similar to the **datapath.vhd** module in PA3, but contains a few changes:
   - The iteration / SDX / LDX logic that was previously implemented in the controller.vhd is moved to the internal datapath. When reset is asserted, iter should be set to 1 asynchronous to the clock. Otherwise on the rising edge of the clock, whenever the ROM table asserts the output shift1 (which corresponds to the SH2 state), iter should be incremented by 4. LDX and SDX, which are now internal signals, should be asynchronously assigned based on iter and the input signals that are being output from the ROM. Note the logic is the same as in PA3: when the state is DONE, LDX should be set to 1 (since there are no more reads being performed), and is otherwise set to iter. When the state is LD or DONE, the SDX should be set to iter-4 (to allow the last iteration to be saved to their correct destination), and is otherwise iter. Using the ROM outputs, you should determine what signals correspond to each state, and use those in the LDX and SDX logic.
   - There are new inputs / outputs. The input m (which contains the array size) is now an input. In addition, ready_flag and start_flag are inputs, which correspond to the DONE and ST state, respectively. A new output goto_done serves as the quantifier to the two-address microcode. Whenever iter is equal to m-3, goto_done should be set to ‘1’ to indicate a transition to DONE should take place. Otherwise, goto_done should remain ‘0.’
• The logic that you completed for the PA3 datapath (i.e. your logic for selecting the adder inputs, and your logic for synchronous writes to the registers / b array) will be the same. That portion can be simply copied over from PA3, granted it is implemented correctly.

4. microcode_rom.vhd: In this module, the state machine given by figure ?? is to be converted into an SM chart that is suitable for a microcode implementation. The module takes addr as an input (which is the current state), and produces the TEST, NSF, and NST signals. The TEST signal is a single bit, and encodes the two possible qualifiers. NSF and NST contain the next address of a given state, provided the TEST bit is true or false, respectively. Additionally, the output signals load, shift[1:0], ready_flag, and start_flag must be output correctly for the given states, also shown in the state machine.

5. sequencer.vhd: The sequencer is shown in the upper half of figure ???. This module contains the logic for sequencing the addr signal. The test MUX selects the quantifier based on the TEST bit being output by the ROM table. The second MUX then either selects NSF or NST as the next address, depending on if the quantifier was true (‘1’) or false (‘0’). Note that the register holding the addr signal should still support asynchronous resets, and in this case, should set addr to the starting state independent of the clock whenever reset is set to ‘1.’

3 Test Bench

There are three total test benches: ucode_test1.vhd, ucode_test2.vhd, and ucode_test3.vhd. These testbenches have the same functionality as those in PA3. Using each of these test benches, you should should simulate your design and ensure it works for each case.

4 Deliverables

Please submit a single PDF containing the following:

1. Your SM diagram that you converted from the state machine in figure ??.

2. Your two-address microcode table that you derived from the SM chart. You should have one entry for each address (state), and should list test, NSF, NST, and each output for each entry.

3. Your code of the following:
   • microcode_rom.vhd
   • sequencer.vhd
   • pa3_datapath_ext.vhd

4. The screenshots listed below. For the waveform screenshots, please clearly show a, b, reset, clk, ready, TEST, NSF, NST, load, shift[1:0], ready_flag, start_flag, goto_done, addr, and R1 - R4. Some of these will need to be added to the waveform from the datapath, sequencer, and microcode ROM modules.
   • The output waveform from ucode_test1.vhd from 0ns to 100ns.
   • The output waveform from ucode_test2.vhd from 0ns to 130ns
   • The output waveform from ucode_test3.vhd from 240ns to 320ns