Lab 5: Introduction to FSM + Datapaths

Objective

The objective of this lab is to introduce the student to a combined FSM and Datapath design problem. This lab is worth 150 pts and is a 1-week lab.

All files referred to in this lab are available in this ZIP archive listed on the WWW page for this lab. To unzip on Unix machines, do '*unzip zipfile.zip*'.

Map all designs produced in this lab to the Flex 10K family, EPF10K20RC240-4 (for the Professional edition of the software, you will need to click on the 'show all speed grades' button on the device menu in order to get the "-4" speed grade). Use the FAST synthesis option.

To Do (part 1, FSM Implementation):

You are to complete the design for a Synchronous SRAM that has a block transfer capability. The interface to the 64x8 XFER SRAM is given below:

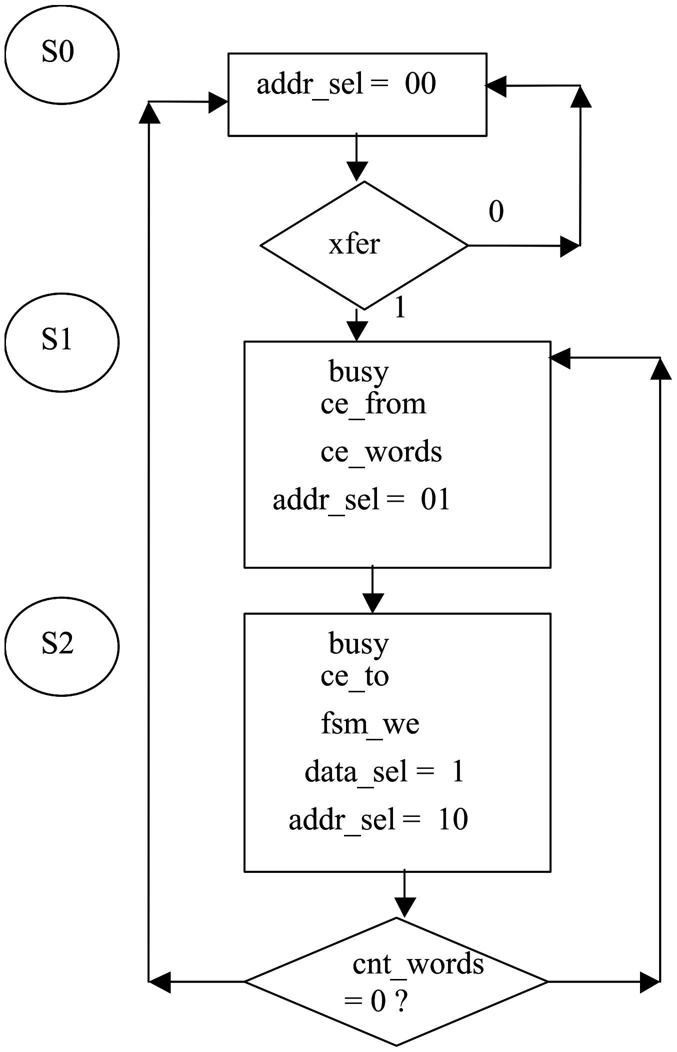
INPUTS:

- *clk, reset* Clock and high true asynchronous reset
- *din[7:0]* Data input bus for SRAM
- *we* Write Enable signal
- *addr[5:0]* Addr Bus
- *cmd_we* When this line is high, the value on the address bus is loaded into one the WCNT counter if din[1:0]=”00”, into the FROM counter if din[1:0]=”01”, or into the TO counter if din[1:0] = “10”.
- *xfer* control line that starts block transfer

OUTPUTS:

- *busy* status output that indicates a TRANSFER operation is in progress
- *dout[7..0]* Data output bus

The *xferram.gdf* schematic contains the datapath for this design. Your job is to complete the finite state machine implementation by modifying the file *fsm.vhd* to implement the ASM specification shown below:



State S0 simply waits for *xfer* assertion. *Addr_sel=00* is a mux select that selects the normal *addr* inputs to drive the RAM address lines.

State S1 reads a word from the RAM using the *from* counter address. *ce_from* increments the *from* counter, *ce_words* decrements the *wcnt* counter. *Addr_sel=01* is a mux select that selects the *from* counter to drive the RAM address lines.

State S2 writes the word read in the previous state to the RAM using the *to* counter address. *Fsm_we* asserts the RAM write line. *Data_sel=1* is a mux select that selects the word read from the previous cycle to be steered to the RAM data inputs. *Addr_sel=10* is a mux select that selects the *to* counter to drive the RAM address lines. The transfer operation is finished if the wcnt register = 0 (*cnt_words = 0*). The *cel_to* output increments the *to* counter address.

You should examine the *xferram.gdf* schematic and trace the connections of the FSM symbol to all of the datapath components to understand the relationship between the above signal names and the datapath components that they control. Use the above ASM to help you understand the transfer operation itself. The transfer operation is accomplished in states S1 and S2 where S1 reads a word from memory and S2 writes a word. The number of words copied is specified in the *wcnt* counter. The beginning source location is specified in the *from* counter, and the beginning destination location is specified in the *to* counter. The values of the *wcn*t, *to*, and *from* counters must be loaded via the *cmd_we*, *addr*, and *din* inputs before the transfer operation is started.

Testing your Design, part 1:

After you have completed your modifications to *fsm.vhd,* you can test your design via the *xfergold.scf* waveform file. This waveform first writes values into the RAM starting at location 1, then copies 10 words starting at location 5 to location 30H. After the copy is complete, locations are read starting from 30H. Another copy operation is also performed later in the test waveform.

To Do (part 2, adding a *zero*'ing functionality):

Modify both the datapath and finite state machine to support the ZERO operation in addition to the transfer operation. Add an input pin called 'zero' that will zero the number of words specified by the WCNT register starting at the location specified by the TO register. There are posted class notes on a RAM with the zero'ing function –you only have to figure out how to incorporate this same functionality into this design. Add whatever states and datapath components necessary to the finite state machine and datapath to support the required functionality.

The *zero* operation should take one clock cycle per write operation.

Call your new design *xferram2.gdf*, and your new VHDL file *fsm2.vhd* so that you do not overwrite the work you have done in Part 1.

The waveform *xfergold2.scf* is a golden waveform that tests both the transfer and zeroing operations.

Assume that the *xfer* and *zero* inputs will not be asserted at the same time.

Check Off:

You must get the functionality of both Part #1 and Part #2 verified by the TA.