ECE 2031 Project Summary

Yuqi He

Georgia Institute of Technology
School of Electrical and Computer Engineering

Submitted
April 21, 2020

Introduction

In this semester's project, we create an I/O peripheral in VHDL between SCOMP and SRAM chip on the DE2 board which gives users 256k words of data memory to read and write. This document examines our approach to implement this I/O device with state machine and at the same time satisfy the timing requirement. We choose to use state machine because it will be easily implemented in VHDL and control signals properly. State machine is also a better way to be presented and understood. Our project accomplishes all the original proposal goals including 16-bit read and write access between SCOMP and SRAM with any address and signal timing requirement of about one single SCOMP state, but sometimes has minor bugs.

Device Functionality

Read SRAM Data

User should first set higher 2 bits of address to SRAM_ADHI and lower 16 bits to SRAM_ADLOW to form the desired SRAM address to be read. User should then set SRAM_CTRL to 10 to indicate read and wait for data to be latched. Finally, user should reset SRAM_CTRL to be 00 to its default value.

Write SRAM Data

User should first set higher 2 bits of address to SRAM_ADHI and lower 16 bits to SRAM_ADLOW to form the desired SRAM address to be written. After setting the address, user should set the desired write value to SRAM_DATA. User should then set SRAM_CTRL to 01 to indicate write and wait for data to be latched and write. Finally, user should reset SRAM_CTRL to be 00 to its default value.

Special Feature

In our design, user will only need two bits to indicate read or write. The normally third bit which is the drive line is compromised in our state machine. Thus, user will only need to indicate read or write and do not need to worry about drive line.

Special Situation

User needs to reset SRAM_CTRL to its default value after each read or write. If user forgets to reset, the state machine will produce undefined states and later read and write will produce incorrect results and will possibly cause data corruption.

Design Decisions

After we have finished our initial state machine, we have two implementation choices to go with. The first choice is to include in and out signals and all data inside our state machine. The second choice is to use tri-state buffer and include only necessary control signals inside our state machine. We went for the second choice because with only control signals, our code will look clean, readable, and can be easily modified based on other needs. For data, tri-state buffer will decide whether input should be passed to output based on control signal.

Conclusions

Despite short on time and special situation during COVID-19 period, we did finish our project and our solution satisfies every aspects of project requirements. For future improvements and optimizations, SRAM_CTRL reset process can be incorporated as part of state machine and user do not need to reset it manually. The weakness of our design is it do not have a good protection against incorrect instructions from users. If I revisit this project in the future, I will definitely spend more time on improving the protections against incorrect instructions since no one wants their data to be corrupted.