LC3 Final Project

# Overview

This project was designed to bring together all of the things we learned in this class. The most difficult part of the project was the sheer size of it, with many different aspects to the Testbench and a relatively complex Design Under Test.

# Test Plan

* All opcodes are supported
* Instructions can be executed in any order
* Instructions can be repeated
* A load after a store returns the value previously written
* A store command modifies memory correctly
* Branching can jump forward or backward
* Branching depends on n, z, and p
* Any register can be a source
* Any register can be a destination
* Reset can happen during any clock cycle of LD
* Reset can happen during any instruction

# Test Strategy

Cover Groups

* All opcodes used
* All opcodes are preceded and followed by all others
* All opcodes are followed by the same opcode
* Reset is tested during all cycles of Load
* Reset is tested with all opcodes
* Branch occurs with N, Z, and P
* All opcodes are tested with all SRC1, SRC2, and DST registers

Assertions

* Flags aren’t high at the same time
* Bus isn’t driven by more that one thing
* Registers are 0 after a reset
* MemWe is only ever high for one clock cycle
* Only store instructions write to memory

Tests

* Coverage
* Large number of instructions
* Different tests for different DUTs

# Test Approach Tradeoffs

Memory Model

* DUT memory module and golden memory array
* Memory locations are randomized at startup
* Instructions written to the DUT memory at Fetch

Checking Memory after each instruction

* Checking the entire memory took a long time so we only did it when necessary
* We check the current memory location only after each instruction

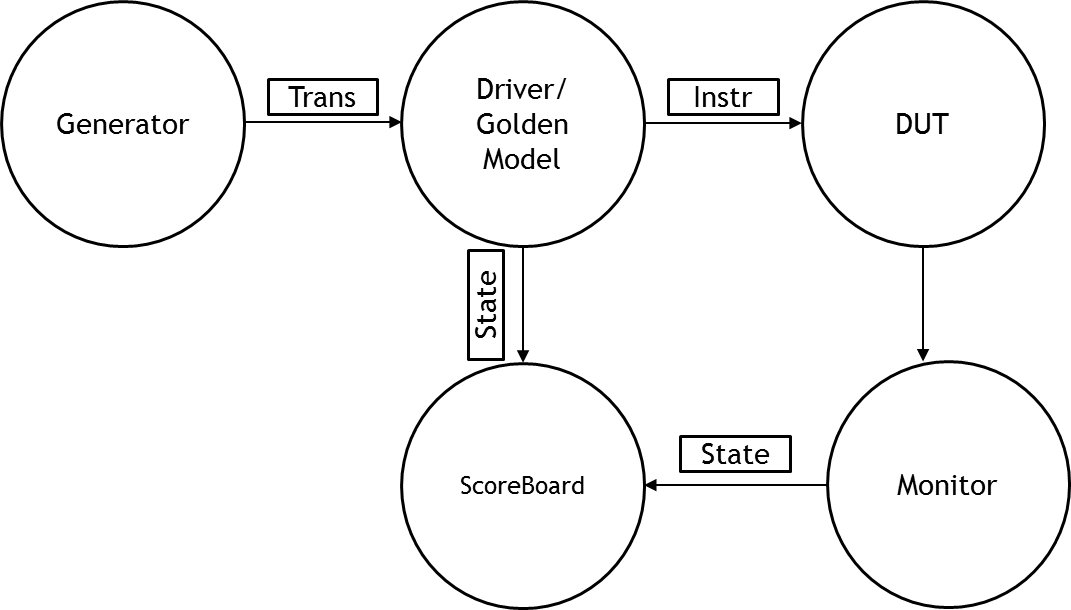
Handling two different DUTs

* We created different tests for each DUT which contains a Config flag
* Each time a cross module reference is needed we check the Config flag

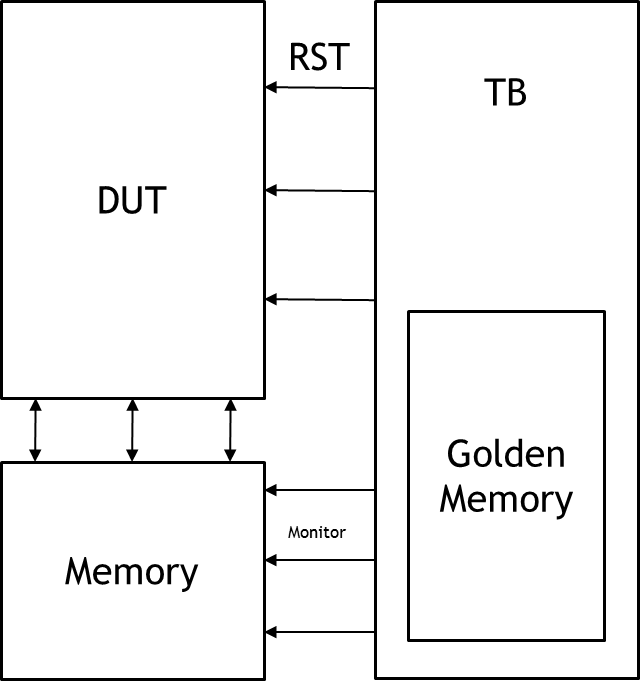
# Test Bench Block Diagram

# 

# Program Flow Diagram

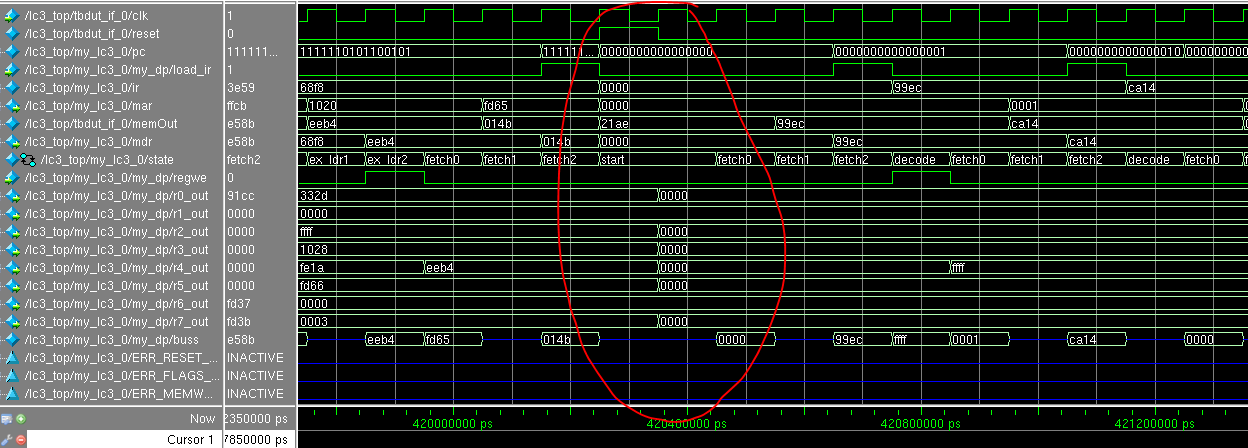


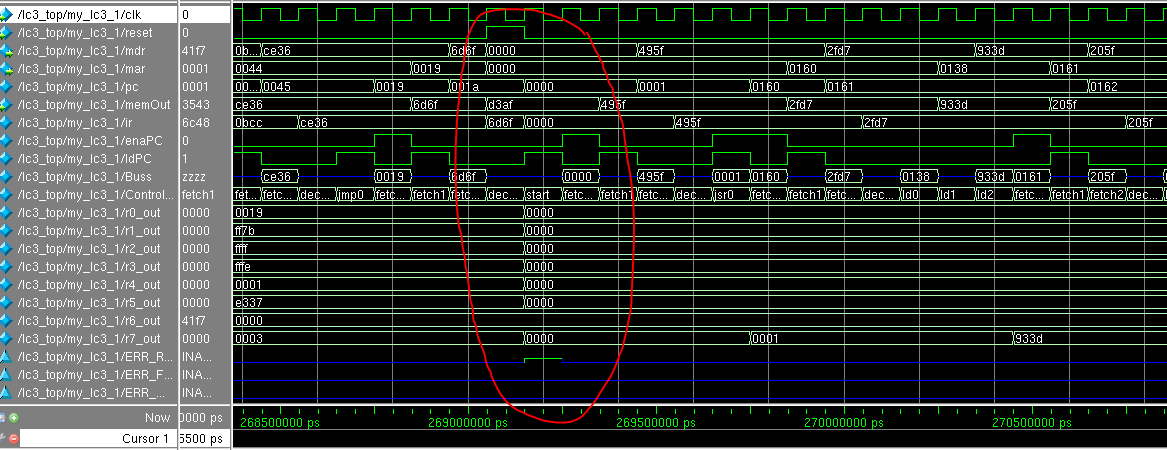
# Test Approach



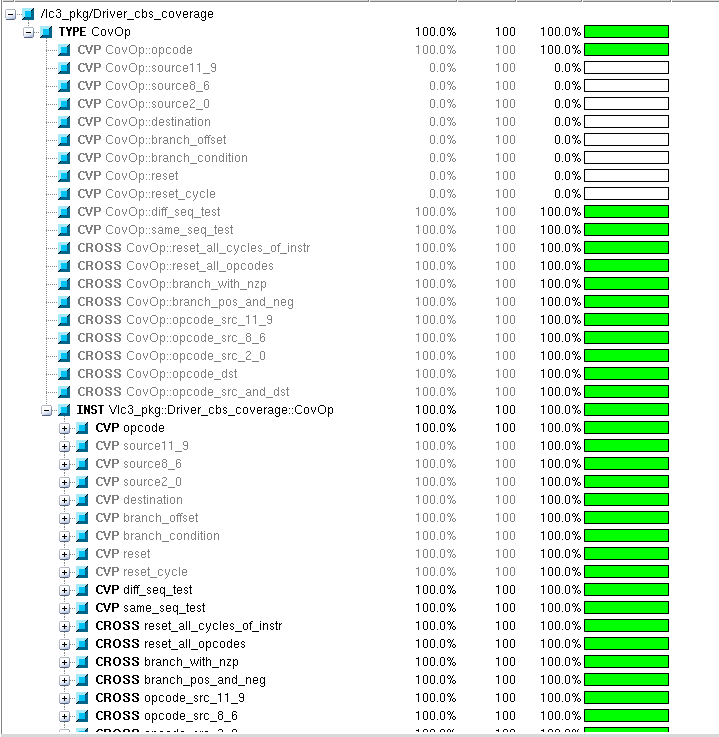
# Waveforms

Ammon’s DUT



Khalil’s DUT

# Functional Coverage



# Bug Report

