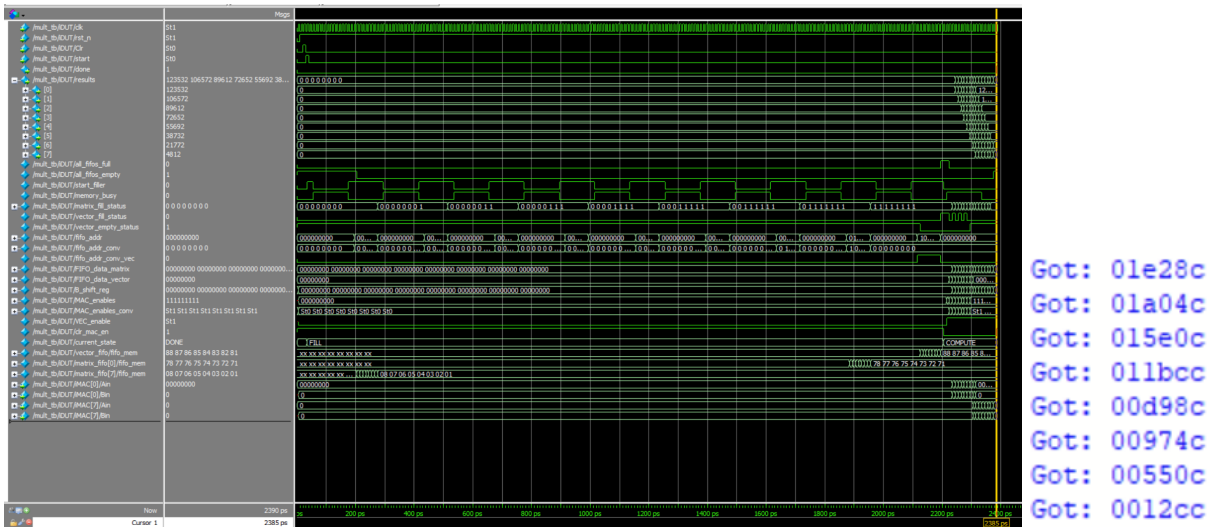
Henry Wysong-Grass
Minilab1A
Minilab1B
2026-02-05

Minilab1A and Minilab1B Report

Minilab1B

TESTBENCHING

We used a testbench to “kick off” the simulation. Since our vec_mult unit was self-contained, we only supply an address to it and set “start” high which begins the system. We then both automatically check the design against a gold standard as defined in our test bench, and also manually by looking at the waveform and by using a calculator to find the expected output. We only check the cumulative MAC outputs when the “done” signal is raised. Below is the sample waveform from the functioning module.



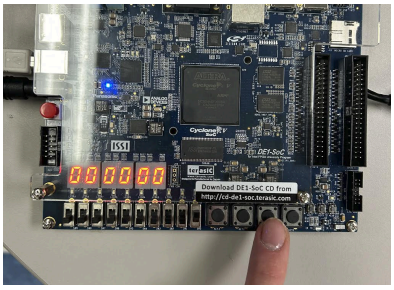
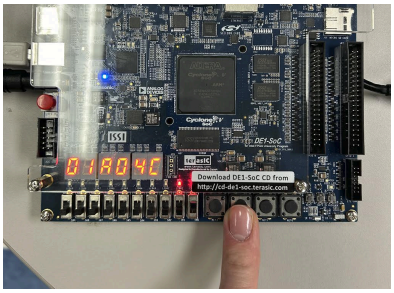
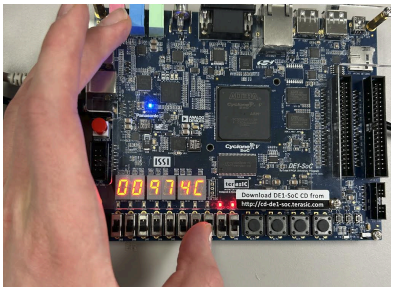
Above, the left image shows the waveforms from the finalized design and the right shows the expected/actual outputs of the unit.

FIXING TIMING

Our timing was initially not met due to our MAC unit. By adding in one pipeline stage and changing how a temp register was assigned, we were able to shave off all timing errors and meet timing. Adding a pipelined stage improved our negative slack by 1ns, leaving us with 4ps of time to take off. By creatively adding extra “temp” registers, we were able to add a further ~300ps of slack allowing us to comfortably meet timing.

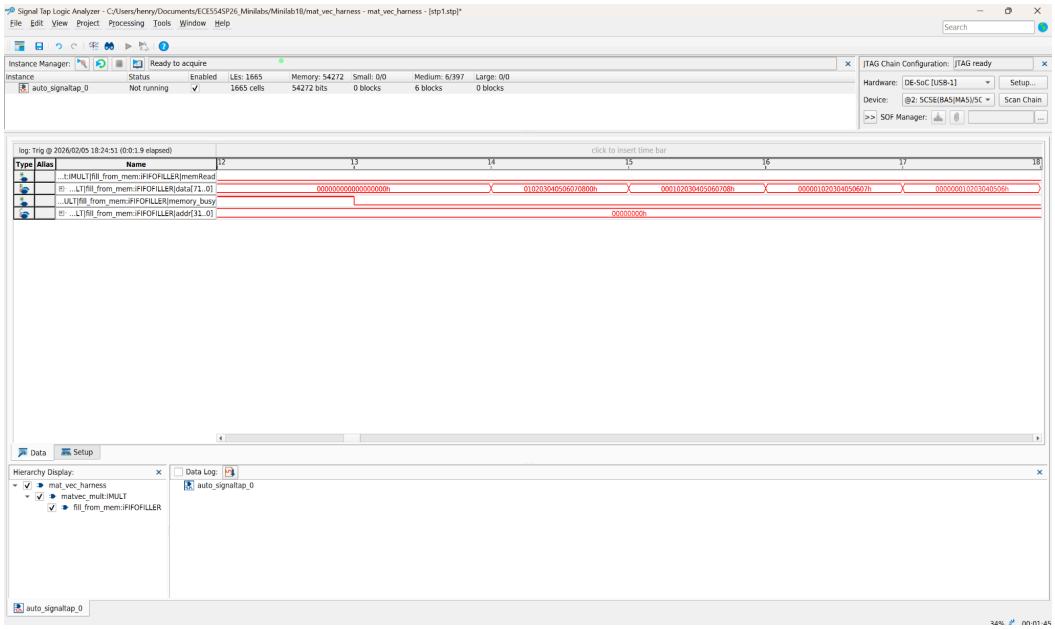
TESTING THE SYNTHESIZED DESIGN

We were able to test the synthesized design by mapping it correctly to the FPGA and adding support for button visibility. We had an externally accessible clr, rst_n and start button along with being able to select which vector element we want to display with the switches. Below you can see this in action and working.

		
A reset/cleared board before starting	Start has been asserted (key[2]) and the calculation is now done.	Another value is selected showing. The state is shown (Done)

SIGNALTAP

To use signaltap on the Avalon MM interface, we added the pre-synthesis system clock along using the memRead signal as our trigger. We additionally monitor data and, memBusy and the address that we are reading from. After reuploading and triggering on the execution, we see our interface is correctly implemented as we pull the correct data (as seen in the image below), and that all signals rise and fall correctly.



COMMENTARY ON PROBLEMS

The largest problem we had was with interpreting how the FIFO's should be laid out. We quickly got our interface and matrix_vector multiplier working, but struggled to figure out if our project was actually

performing up to the specifications. We eventually brought out the calculator and realized we were, but this took a while to double check. Additionally, meeting timing was difficult as, even after pipelining, we still had 4ps of time to cut out which proved more difficult in the MAC. Only by being clever with our internal register assignments were we able to prevent having to pipeline the whole multiplier.

Minilab1A (pretty much the same as minilab0)
Git Component:
To create my repository, I used GitHub to create a new repository with the specified name and publicity. I then cloned the repository via https over command line onto my local machine. (Tokens and authentication already configured from past) After creating a new directory, I manually copied the requested files into the minilab0 directory and then committed and pushed using “git add .” and “git commit -m “minilab0 report commit” along with “git push” respectively.

Resource Utilization Discussion: (this was originally written for the Xilinx boards, but still remains generally true. The exact reports are still available in this repository)

In the utilization reports, we can see that the number of LUTs in use drops from synthesis to implementation. The synthesis report actually notes most of why this happens. During implementation, more aggressive optimization techniques are used to both drive down excessive LUT use and keep our design efficient and lean.

Modelsim waveform sample: