Robot Arm Control Final Report

Nathan Ruetten

Eva Denille

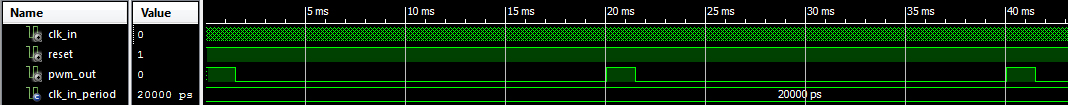
5/4/13

*Overview*

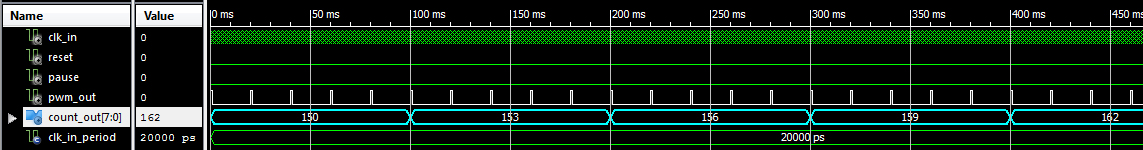
In the Robot Arm Control lab, we used VHDL to be able to control a robotic arm. This lab refined and taught use more about VHDL such as hex numbers. As always to run any this through this language, there was a clock divider. It made all of the “checks” work together. The mode generator is how we control the arm. Depending on the combination of the switches, we decide the direction of the arm whether it pans or tilts. The counters loads and passes the correct vector to the PWM. Its primary purpose is to split the pan and tilt data, whether to pass the inputted data, and know if the arm is already at the requested position. The PWMs arm to output the data to the arm and feed back to the counter the arm has been moved. There is one more block that is needed for when both the pan and tilt are requested at the same time. We called it D\_swep. Its job is to count till the pan finishes its sweep, then allow tilt to sweep. We also have a debouncer to cleanly reset all the modules.

*Major Simulation Results*

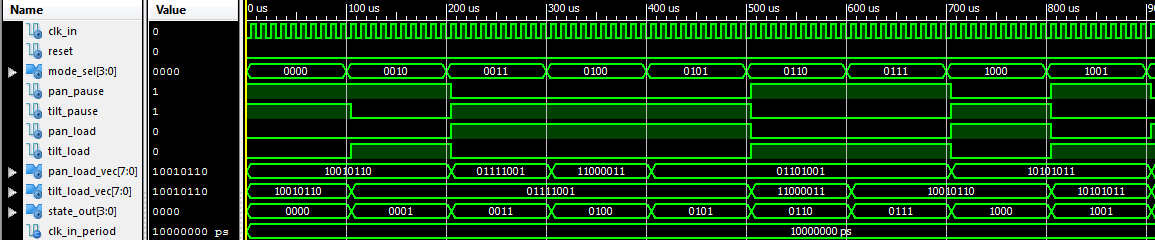
Combination for week 1(clk\_div & pwm)



Combination for week 2(clk\_div, counter, & pwm)



Mode Generator



*Problems & Conclusion*

We had issues from the start but most of these for week one was refreshing VHDL. We had some major problems with the counter. The issues involved understanding hex numbers. We added an extra library and had some of the cases sent from the mode generator throw off the counter. We also had a little confusion how the mode generator was supposed to be done. The lecture in class for the rotary encoder threw us behind a little. We also had a few speed bumps syncing the program to the board. We had to check that everything was located in the correct places.

From this lab we know how to control a robotic arm using VHDL. The knowledge from this lab can be carried on to other systems that require multiple inputs and outputs. Looking at modern control

*Block Diagram*