ECE 551
HW5

- Due Wednesday Nov 21st by class time
- Work Individually on Problem 1
  - This is submitted to the dropbox
- Rest of HW can be done as a team
  - These problems are submitted via dropbox
  - **Please**…only **one** person submit all problems for the team!

HW5 Problem 1

Does your synthesis script do an: **ungroup –all –flatten** Command? It should to produce a **.vg** netlist that has no hierarchy and is just gates.

1. (25pts) Post Synthesis Simulation of your **SPI_mstr16.sv**

Posted on the class webpage (under the tutorials section) is a file about Post Synthesis Validation (simulation). Read it carefully.

As part HW4 you created **SPI_mstr16.sv** and tested in in a self checking test bench (**SPI_mstr16_tb.sv**). As part of HW4 you also wrote a synthesis script (for you UART). Adapt the synthesis script to apply to synthesizing your **SPI_mstr16** module and use Synopsys to produce **SPI_mstr16.vg**. Then simulate the gate level netlist.

Submit to the dropbox:

a) Your **SPI_mstr16.vg** gate level netlist that you simulated
b) Simulation results proving success. Yes…you could turn in something here that fooled me into thinking you did this when you actually didn't. However, you would be hurting yourself, because you will need post synthesis simulation to work for the class project, and you can't fool me for that.

(**Note:** you may want to try post synthesis simulation of a few of your other blocks to ensure they are ready for the project)

HW5 Problem 2

2. (40pts) Inertial Integrator

We will work on this problem in Exercise19

Around slides 24-28 of the project spec you will find a section on the "inertial_integrator" unit. Carefully study these slides.

For this HW problem you will be coding that block (**inertial_integrator.sv**).

You will then be creating a testbench for inertial_integrator called **inertial_integrator_tb.sv.**

The details of this problem are specified in detail in Exercise19.

One person from the project team submit a final version of:

inertial_integrator.sv
inertial_integrator_tb.sv
"analog" waveform showing proper behavior

HW5 Problem 3 (A2D Interface)

3. (25pts) A2D Interface (In class exercise 17 & 18)

If your project team got checked off on exercise 18 then you are done with this problem.

4. (10pts) balance_cntrl cleanup.

Select whoever's implementation of **balance_cntrl.sv** (see ex14) that you wish to use. That version needs a couple of signals added to it, and one **parameter** added to it.

Signal:	Dir:	Description:
pwr_up	in	This signal would come from the Auth_blk.sv and is used to enable the device. The outputs lft_spd/rght_spd of balance_cntrl.sv should be zero if pwr_up is zero.
too_fast	out	This signal is asserted if either lft_spd or rght_spd is greater than 1536. It goes to piezo interface to warn the rider they are approaching control hazard conditions.

See next slide for parameter to be added.

HW5 Problem 4 (clean up of balance_cntrl)

4. (10pts) balance_cntrl cleanup (continued)

The integrator (of I term of PID) was designed to integrate on a timescale appropriate for the angular momentum of the Segway. Meaning it is deliberately slow because the Segway has a hefty thing with a large moment of inertia.

Fullchip simulations would take too long to converge with the I term used as is. You will introduce a parameter called **fast_sim**. It should be defaulted to zero, but we can pass a value in to balance_cntrl.sv when it is instantiated.

This parameter (f**ast_sim**). Should speed up the integral term by 16X. **Do not actually change your implementation of the integrator**. Instead where **integrator** is used to make **PID_cntrl** we will use bits [17:2] of **integrator** instead of a sign extended bits [17:6] as we currently do.

This is a switch based on parameter. So you keep the code for [17:6] too, you just make a selectable option for [17:2].

Fix up a version of **balance_cntrl.sv** and submit it.