en_steer block (finish it up)

- In HW3 problem 1 you implemented a statemachine that was the “heart” of the enable steer block. However, it is not the entire **en_steer** block.

- There where signals **diff_gt_15_16** and **diff_gt_1_4** and **diff_gt_min**. These where given as inputs. However you have to now make these signals. They are based on the difference and sum of the load cell inputs (coming from **A2D_intf**).

- **en_steer** waited for the rider to be on the Segway and have their weight evenly distributed for 1.34 seconds before enabling steering. This required the use of a 26-bit timer. Waiting for 2^{26} clock cycles to be simulated in ModelSim will take far too long. We need to add the **fast_sim** parameter toff **en_steer** as well. When **fast_sim** is asserted we will only wait for bits [14:0] of this timer to become full. **fast_sim** should be a parameter passed down from the **Segway.v** level. You did something similar in HW5 to accelerate the integral term of the PID.

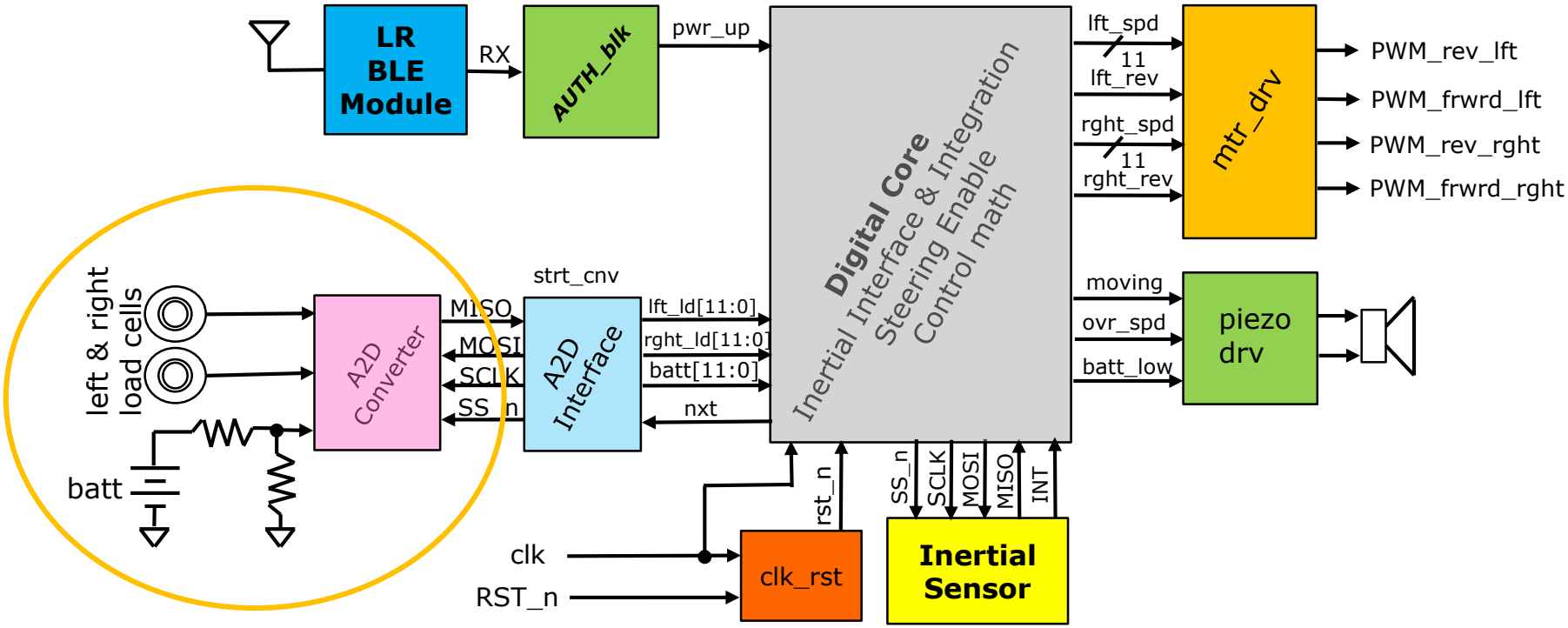
- The project spec did not originally specify **MIN_RIDER_WEIGHT**. It should be a **localparam** and should be set to 12’h200.

Tweak the Integrators

- **Inertial_Integrator:** From when you first implemented this there was a spec change. For the fusion correction there was a term **fusion_ptch_offset** that was set to either +512 or -512. This is being changed to +1024 or -1024.

- Integrator in P**I**D. While the device is powered, but not authorized yet (**pwr_up** not asserted) the motors are disabled but the PID math is still occurring. Meaning the integral term could be winding up (see video BadBehaviorSegway.mp4). We need to ensure integral term of PID is reset when **~pwr_up**.

Modeling ld_cell_lft/ld_cell_rght for fullchip testbench



I provided a cheesy model of the A2D converter (ADC128S). I also gave you the SPI block within it (SPI_ADC128S). However, for full chip simulation you will need a way to specify left and right load cell readings (and perhaps battery too) and have those translate into something that is accessed by your A2D_Intf block (SPI bus).

This means someone on your team has to modify ADC128S or augment SPI_ADC128S to create such a block that can be used as part of fullchip testing.

What needs to drive **nxt** into A2D_Intf to force it to perform round robin readings? Really just any periodic signal that happens often enough but not too often. Hey, don't we get **vld** readings from the inertial sensor 200+ times a second.

What Else

This document was just intended to cover some random miscellaneous things you need to do. Refer back to the project spec as that is the master document.

However, some of the main things that remain for you project team to do are:

- Finish any needed design blocks (**piezo** is only one we have not touched).
- Download **Segway.v** and flush out the fullchip DUT.
- Create your fullchip testbench (**Segway_tb.v** is an optional shell you can use).
- Create a suite of tests to thoroughly wring out the design.
- Run the test suite and debug
- Create a synthesis script and synthesize the design to create a .vg netlist
- Be ready to demonstrate you can run at least one of your tests post synthesis.

- Do all this work on the Linux side. Project demos are done in B555 on linux. Fullchip simulations are long and **vsim** on the linux machines runs faster than ModelSim on the Windows machines. Synthesis and post synthesis simulation only runs on the linux machines anyway.

- Want to run from home? Discover **mobaxterm** and remote login to:
best-tux.cae.wisc.edu