Exercise 19: Inertial Integrator (HW5 Problem2)

- The Pitch readings we obtain from the gyro is not an angle, but rather an angular rate (we get °/sec). Therefore we have to integrate to get degrees of rotation.
- Don't worry, integration simply means summing over time. So we simply need an accumulator register that sums in the signed angular rate readings we are getting. (**ptch_int** = **ptch_int** – **ptch_rt**) (OK subtracting rate due to orientation of the sensor mount on the platform).
- Imagine the "Segway" device was just maintaining level, and not pitched forward or back. The pitch angular rate readings would be small negative and small positive numbers, and would on average sum to zero.
- As awesome as the inertial sensor is (and it is pretty freaking amazing) it is not that good. Just the act of soldering an inertial sensor to a board will affect the offsets of its gyro readings. So even if it was perfectly factory compensated, it will not be once it is in your application.
- The small offset will be integrated over time and the reading obtained purely by integrating the gyro pitch readings will drift. We need to "fuse" the accelerometer readings (AZ) with the gyro integration to correct for this drift.

Exercise 19: Inertial Integrator (HW5 Problem2)

inertial_integrator signal interface:

Signal:	Dir:	Description:
clk, rst_n	in	clock & reset
vld	in	High for a single clock cycle when new inertial readings are valid.
ptch_rt[15:0]	in	16-bit signed raw pitch rate from inertial sensor
AZ [15:0]	in	Will be used for sensor fusion (acceleration in Z direction)
ptch[15:0]	out	Fully compensated and "fused" 16-bit signed pitch.

inertial_integrator internal register:

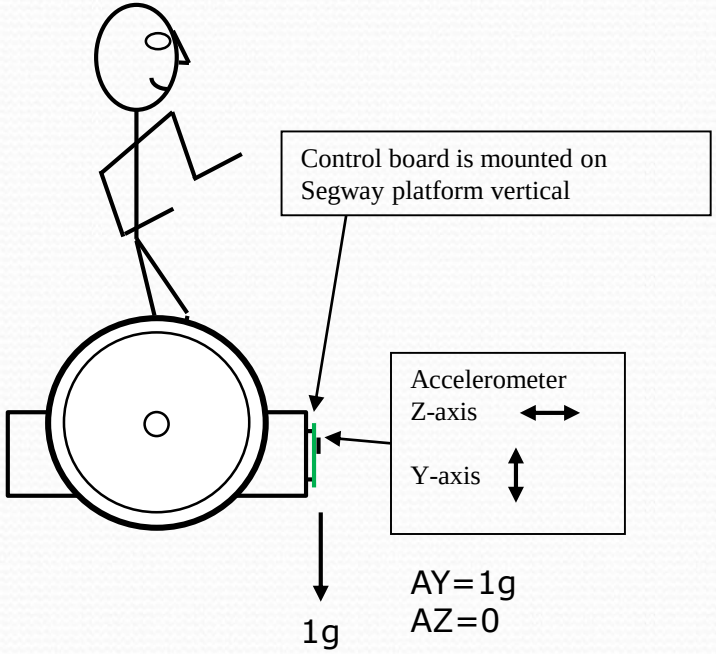
Register:	Purpose:
ptch_int[26:0]	Pitch integrating accumulator. This is the register ***ptch_rt*** is summed into

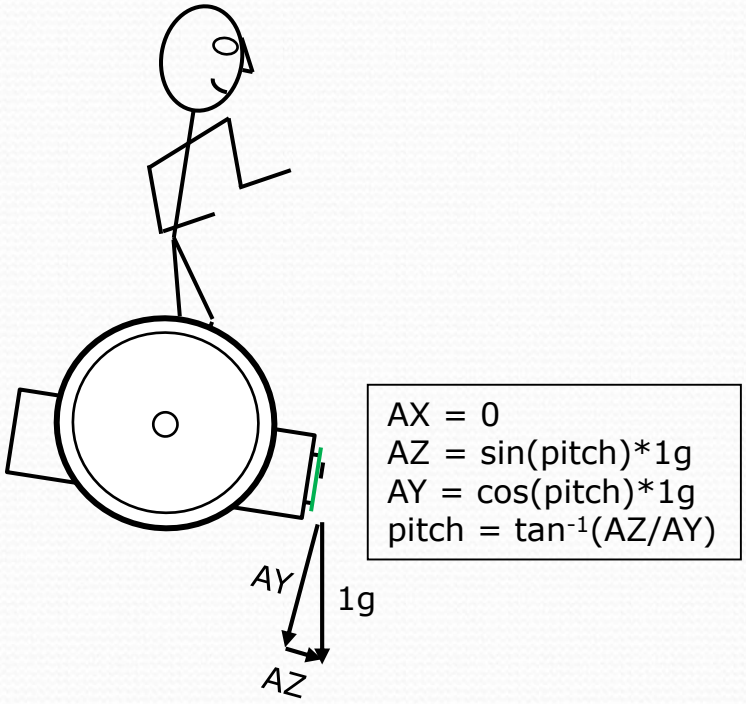
- On every **vld** pulse this unit will integrate **ptch_rt_comp** signal into **ptch_int[27:0]** (we integrate the **negative** of **ptch_rt_comp** due to orientation of sensor mount)
- Bits [26:11] of **ptch_int** form **ptch[15:0]** (i.e. divide by 2^11). This was somewhat of an arbitrary scaling factor, just go with it, don't question it.
- There is also a fusion correction term (**fusion_ptch_offset**) that is summed into **ptch_int**. This will be discussed more later.

ptch_rt_comp = ptch_rt – PTCH_RT_OFFSET **(PTCH_RT_OFFSET=16'h03C2)**

What is Sensor Fusion?

- One can get pitch and roll from an accelerometer. The earth's gravity always provides one **g** of acceleration straight down.





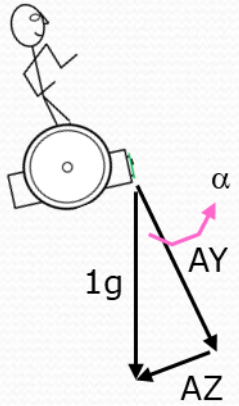
So why get pitch and roll from the gyro?
We can just use AY and AZ

Accelerometers give noisy readings. Vibrations plus acceleration in linear direction come into play.

What is Sensor Fusion?

- Accelerometers are noisy, and Gyros are subject to long term drift.
  - Accelerometer readings are noisy (vibrations and linear acceleration)
  - Gyro readings are amazingly quiet (even in presence of vibration) but since we are integrating, any remaining offset error will result in a long term drift.
- Sensor fusion is an attempt to take the best from each to create a low noise, no drift version of pitch.
  - The fusion data is dominated by the integrated gyro readings, thus it stays low noise.
  - However, pitch is also calculated via accel readings, and the long term average of the integrated gyro readings is "leaked" toward the accelerometer results. In electrical engineering terms the accelerometer gets to establish the DC operating point, but the gyro readings determine the transient response.
  - **AZ[15:0]** will be used to calculate the pitch as seen by the accelerometer (**ptch_acc**). If **ptch_acc>ptch** then **ptch_int** will have a constant added to it. If **ptch_acc<ptch** then **ptch_int** will have a constant subtracted from it.

Trigonomic Simplifications



- Remember this one? *(you should have learned it in a math class somewhere along the line)*
  - For small angles (α) the $\tan^{-1}$(opposite/adjacent) ≈ opposite/adjacent
  - Taking it one step further for small angles: adjacent ≈ hypotenuse = unity
  - So… for us pitch is simply proportional to AZ.
  - This is only true for small angles, but the "Segway" platform better be at a small angle from horizontal or the rider is in serious trouble

Perform this math inside ***inertial_integrator.sv***

```
ptch_acc_product = AZ_comp * $signed(327);                    // 327 is fudge factor
ptch_acc = {{3{ptch_acc_product[25]}},ptch_acc_product[25:13]};   // pitch angle calculated from accel only

if (ptch_acc>ptch)          // if pitch calculated from accel > pitch calculated from gyro
    fusion_ptch_offset = +512;
  else
    fusion_ptch_offset = -512;
```

fusion_ptch_offset is added into next integration of **ptch_int** on every valid reading from the inertial sensor. **fusion_ptch_offset** is a 27-bit wide constant (same width as **ptch_int**)

AZ_comp = **AZ** - **AZ_OFFSET** where **AZ_OFFSET** is a localparam we will assign to 16'hFE80

Fusion Calcs…litte more explicit

- During normal operation the pitch integrator (**ptch_int**) is summing the compensated rate signal (**ptch_rt_comp**) (OK actually the negative of it), but we are also going to sum in this **fusion_ptch_offset** term to "leak" the gyro angular measurement to agree with the accelerometer gyro measurement.

  - ```
    ptch_int <= ptch_int - {{11{ptch_rt_comp[15]}},ptch_rt_comp} +
                fusion_ptch_offset;
    ```

- During normal operation **fusion_ptch_offset** will be +512 if **ptch_acc > ptch** and will be -512 if **ptch_acc < ptch.**

Break your team into two groups. Each group code a version of **inertial_integrator.sv**. Compare your code with each other once you have finished and discuss the differences.

Get as much completed as possible today.

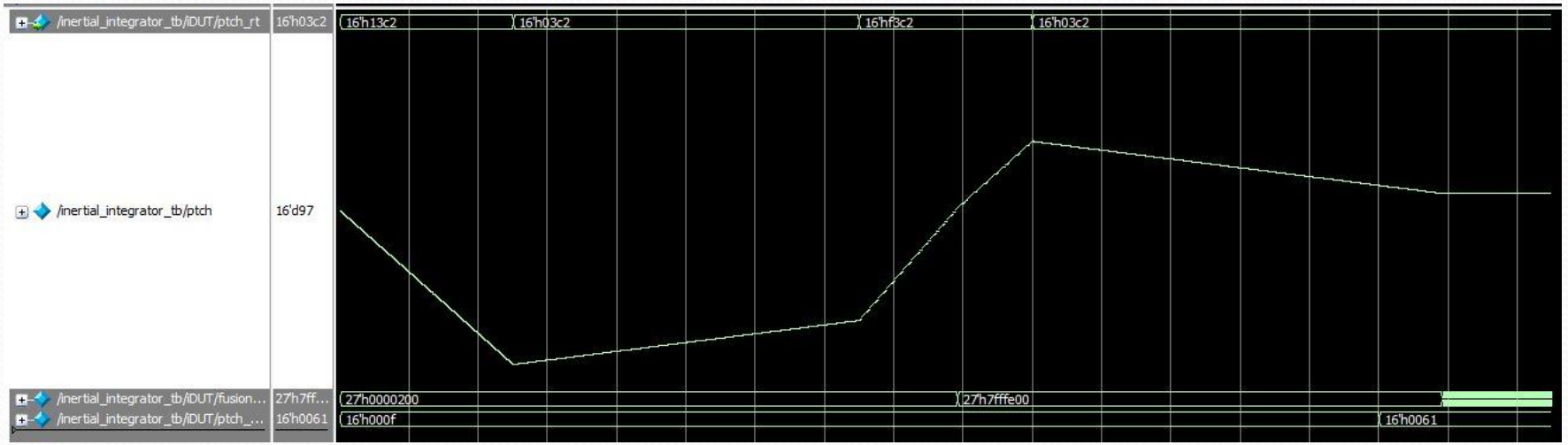
You will create a testbench in the next class period.
    ```

Testing Inertial Integrator:

- Create a simple test bench that instantiates ***inertial_integrator*** and can apply stimulus.
- After reset has deasserted we will apply a **ptch_rt** of 16'h1000 + PTCH_RT_OFFSET. Apply an **AZ** stimulus of 16'h0000. Ensure **vld** is held high.
- Maintain this stimulus for 500 clocks. The **ptch** output should be trending more and more negative since we integrate the negative of **ptch_rt**.
- Now zero out the pitch rate by applying a value of PTCH_RT_OFFSET to **ptch_rt**.
- Hold this stimulus for 1000 clocks. Due to the fusion with the **AZ** reading of zero you should see the **ptch** reading trending slowly back toward zero.
- Now apply a stimulus of PTCH_RT_OFFSET – 16'h1000 to **ptch_rt** for 500 clocks. The **ptch** output should trend steeply into positive territory.
- Again zero out the pitch rate by applying PTCH_RT_OFFSET to **ptch_rt**.

Testing Inertial Integrator (continued):

- Hold this stimulus for 1000 clocks. The **ptch** output should be trending slowly back toward zero.
- Finally adjust AZ to 16'h0800. When ptch gets down to about 100 it should level off as the ptch calculated from AZ should match that from fusion and the offset should start toggling back and forth from -512 to +512
- It is possible to display a waveform as an analog value using "format". The waveform as shown uses 300px vertical to display a range of -1000 to +1000 for ptch. Yours should look similar if your DUT is working.



- You may have to use radix (demical) and format (custom analog) to get this to display

Inertial Integrator: (what to turn in)

As a team you should have two groups of two that each created and tested an inertial integrator .

One person in each group submit:

inertial_integrator.sv
inertial_integrator_tb.sv
waveform picture of analog waveform

Ensure that the comments near the top of **inertial_integrator.sv** have the name of the other person who contributed to that group of 2.