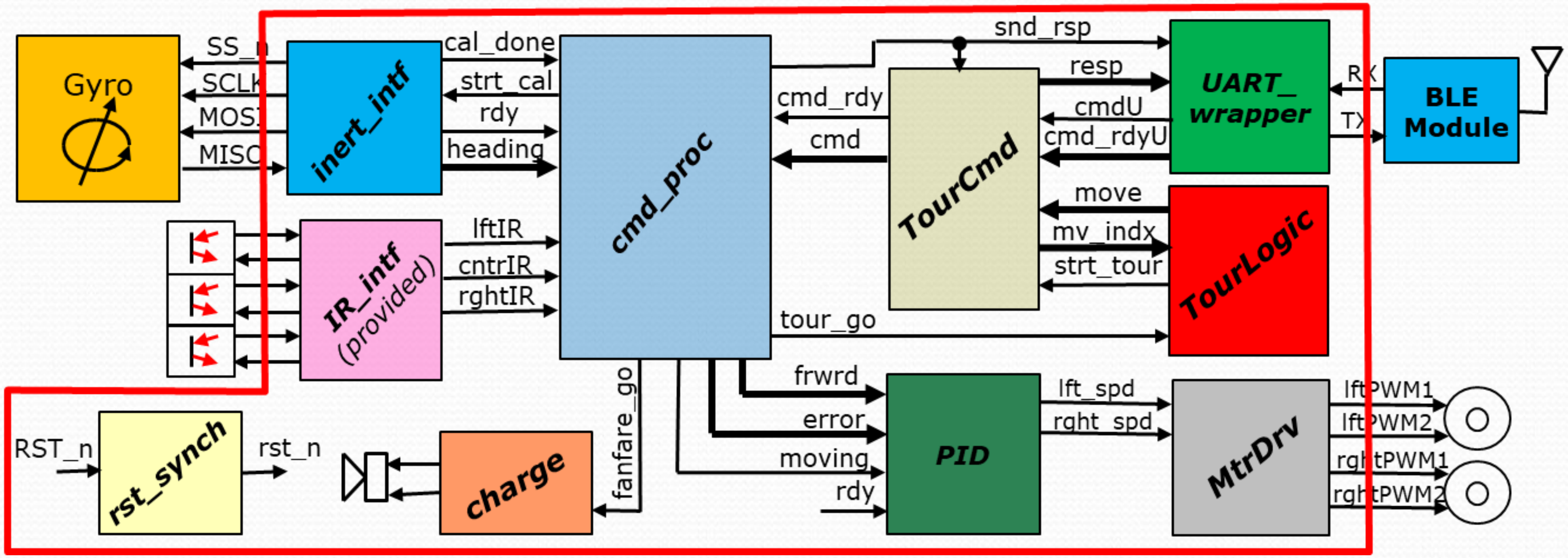
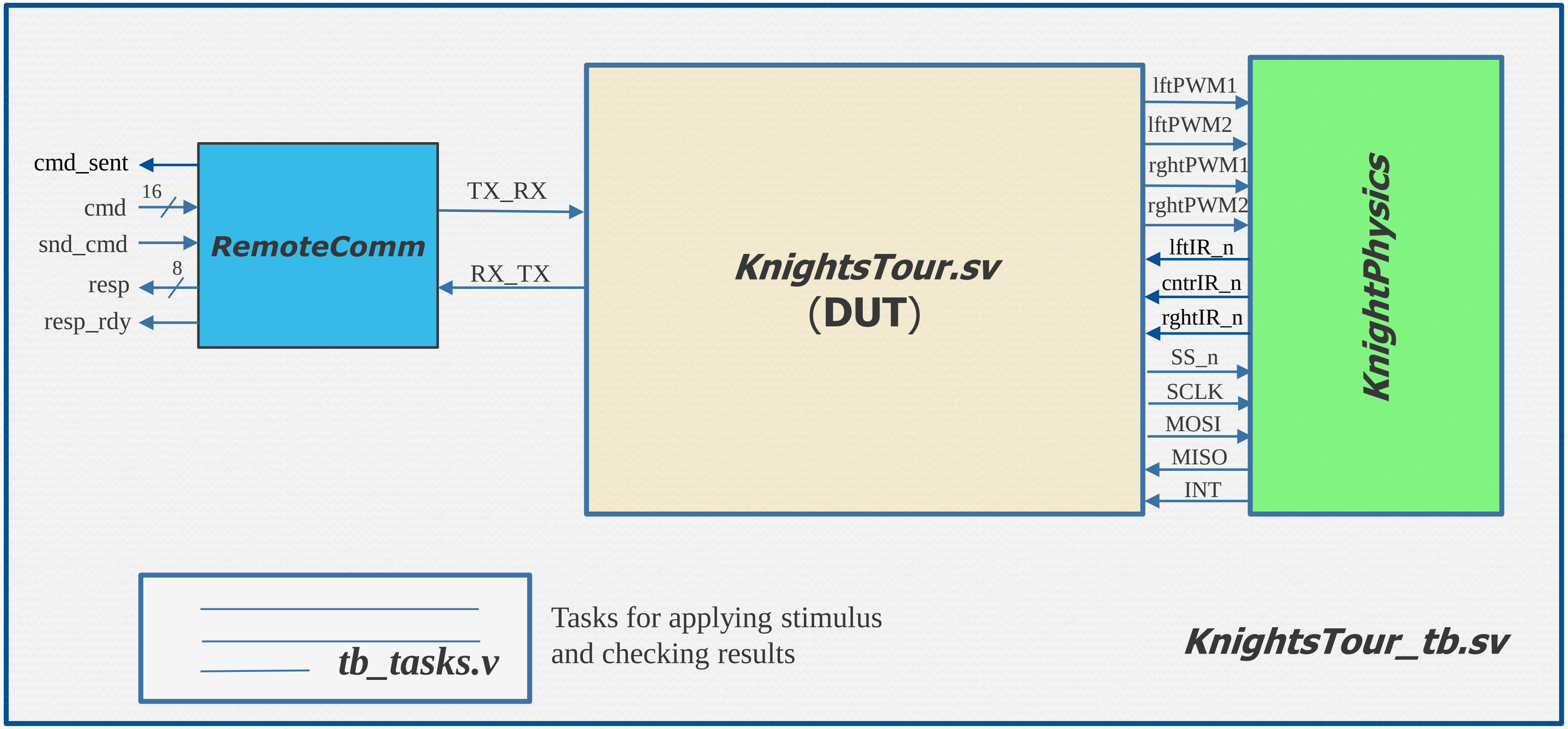
Tips for Testing top Level of the KnightsTours



The red box surrounds all the blocks that are your DUT (**KnightsTour.sv** *(provided)*). To test it you will need blocks to represent the **b**luetooth **l**ow **e**nergy module that sends commands (your **RemoteComm** block does this); and something that models the inertial sensor & IR sensor readings (**KnightPhysics.sv** *(provided)*).

Tips for Testing top Level of KnightsTour:



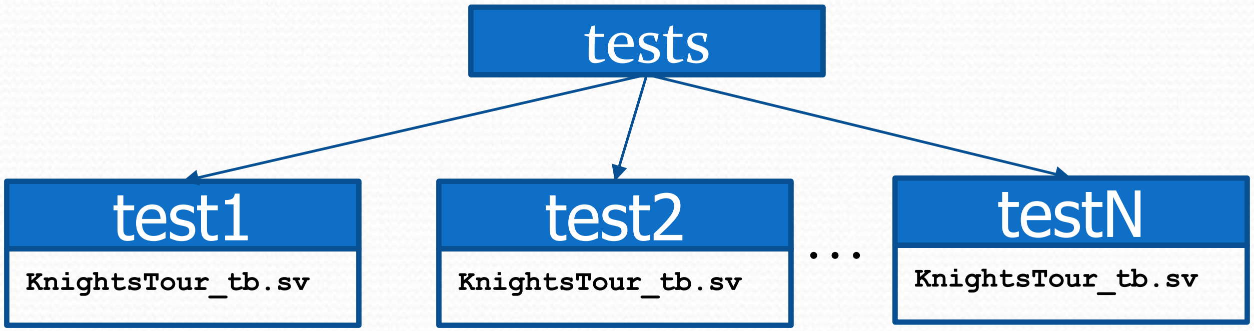
A testbench shell (**KnightsTour_tb.sv**) is provided. Its use is optional, but recommended. **KnightPhysics.sv** is a model of the inertial sensor, the physics of the robot, and the chess board itself. It monitors the PWM signals and does a crude modeling of the physics, deriving quantities such as wheel velocities (**omega_lft/omega_right**) and yaw rate of the robot platform (**heading_v**), which in turn feeds the embedded iNEMO sensor model.

Tips for Testing top Level of "The Knight":

- Use of Tasks can be helpful to make your test bench more readable & less repetitive
  - Initialize
  - SendCmd
  - ChkPosAck
  - …
- What to test? *(start simple and work up to more complex)*
  - Do things initialize properly
    - Are PWM's running and midrail values just after reset?
    - Does **NEMO_setup** eventually assert? *(look inside KnightPhysics)*
  - Send command to calibrate
    - Does **cal_done** assert
    - Do you get a positive acknowledge
  - Send command to move west 1 square.
    - Does **error** signal change, does right wheel duty cycle increase relative to left?
    - Does heading start to change and eventually converge to desired heading?
    - Does omega_sum (inside KnightPhysics) ramp up. Does cntrIR_n fire?
- Ensure tests are "abbreviated" by having **FAST_SIM** set to 1.

Tips for Testing top Level of "The Knight":

- Seems like a lot to test…How to attack it.
  - Don't try to put it all in one giant test!
  - Have a test suite
  - Perhaps a directory of tests
    - Store off a different version of **KnightsTour_tb.sv** for each test
  - Could even have a python script to run the tests in batch

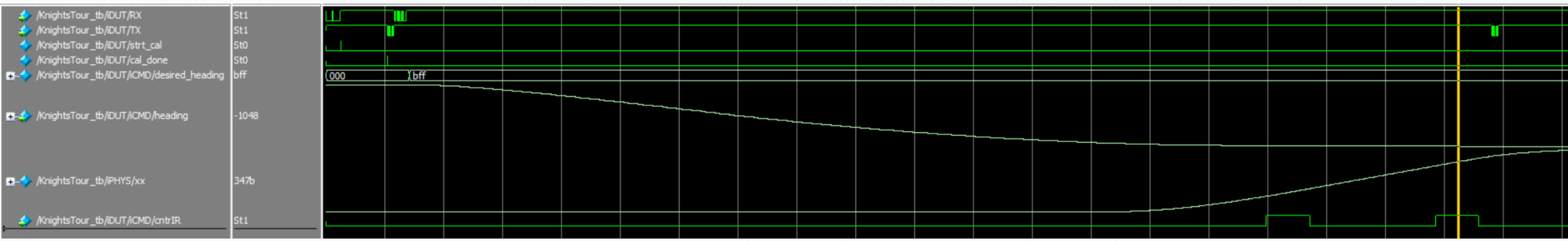


- When using **KnightPhysics.sv** to model the physics of the robot and layout of the board the runs are long. A run of entire tour command is impractical.

- Remember vsim on Linux is much faster than ModelSim student edition.

Tips for Testing top Level of "The Knight":

- Running Closed Loop with KnightPhysics model



Here we see a run where first the ***cal*** command was sent. Then after the positive acknowledge for **cal** was received *(0xA5),* a command to move east 1 square was issued.

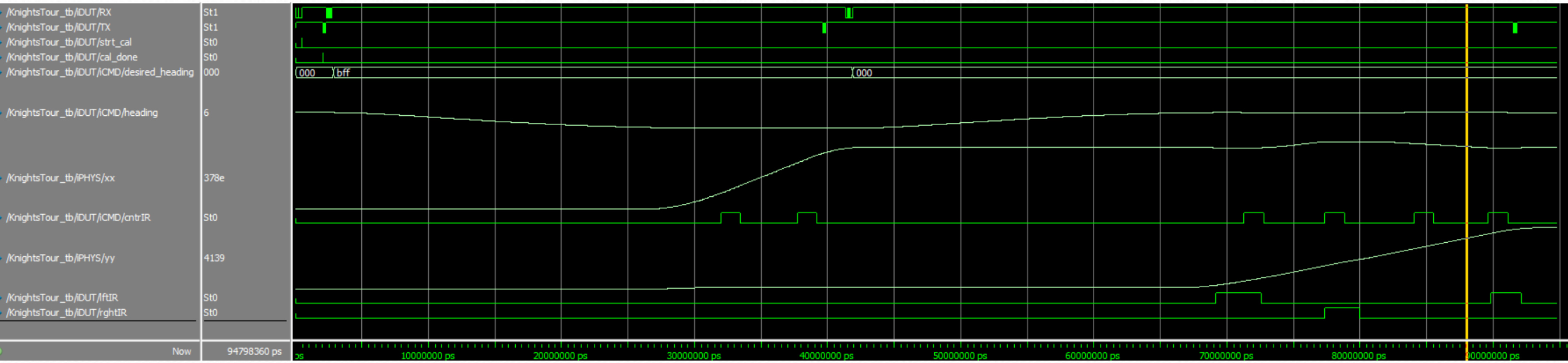
First we see the heading *(plotted as analog)* become close to -1024 *(dead east).* When the heading drops below the threshold specified in ***cmd_proc*** then the **frwrd** register would start ramping up.

We also see the **xx** signal from ***KnightPhysics*** is plotted *(as analog hex value).* **xx** represents the robots x position on the board multiplied by 4096. **xx** and **yy** start at 0x2800 *(if you divide 0x2800 by 4096 you get 2.5)* which represents the middle of the middle square.

As the robot moves along the x we see two pulses on **cntrIR**. One as it leaves the center square and one as it enters the square to the east. Upon the 2nd rise of **cntrIR** the **frwrd** register is ramped down. Once it hits zero the Knight sends a positive acknowledge *(0xA5)* and the command is over. When finished the new **xx** position is in the range of 0x3680 *(near the center of the square to the east of center).*

Tips for Testing top Level of “The Knight”:

- Running Closed Loop with KnightPhysics model



This is a simulation that starts the same as the previous slide (cal followed by move east 1 square). Then it waits 150000 clocks and issues a move north 2 squares. During this 2nd move we some instances of **lftIR** and **rghtIR** firing as the robot “hits the guardrails” in the x direction.

This is a test that shows the **lftIR/rghtIR** are having the desired effect and keeping the x bounded at the robot moves in the y.

Synthesis of KnightsTour:

Contraint:	Value:
Clock frequency	333MHz for clk (3.0ns period)
Input delay	0.4ns after clk for all inputs
Output delay	0.4ns prior to next clk rise for all outputs
Drive strength of inputs	Equivalent to a NAND2X2_LVT gate from our library
Output load	0.1pF on all outputs
Wireload model	16000 area model
Max transition time	0.15ns
Clock uncertainty	0.15ns

Use of **compile_ultra** is not permitted
(has been seen to cause random functional bugs in the past)

1) Create synthesis script
2) Meet both max and min delay timings
3) Report Area
4) Simulate post synthesis netlist

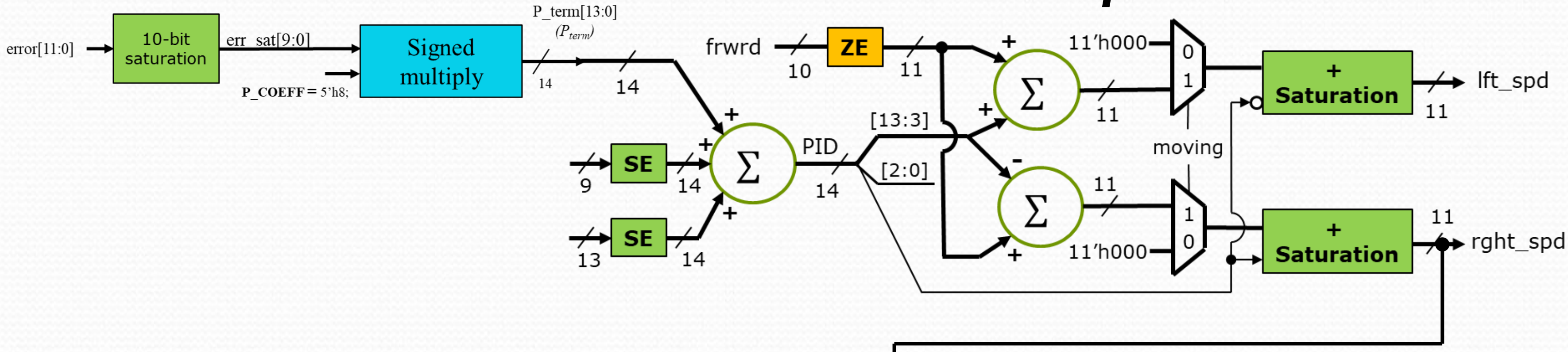
My total area was: 16242

Synthesis Hints:

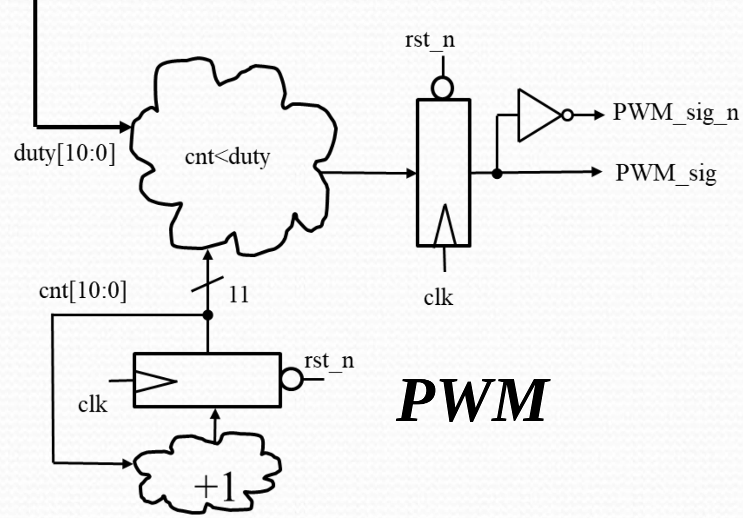
Ensure you are using newest provided **inertial_integrator**

Pterm

Rest of PID



- Add pipelining flops
- Do it in separate copy
- Introduce to "known good" code one change at a time
- **NOTE:** if you pipeline a signal you may have to delay any qualifying enable signals. *(i.e. if you pipeline **err_sat** you should create a piped version of **err_vld** for **integrator** and **prev_err** flops in I & D logic)*



PWM