ECE 551

HW4 *(100 pts)*

- Due Fri Oct 31st @ 11:55PM
- Work Individually
- Use descriptive signal names
- Comment & indent your code
- Code will be judged on coding style

HW4 Problems 1&2 (**10pts) + (5pts)**

1. **(10pts)** Complete the Synopsys Design Vision tutorial. Sign below, (preferably in blood).

I, _____ completed the Synopsys Design Vision tutorial. If I had any problems with it, I discussed them with the TA or Instructor, either in person, or through email.

2. **(5pts)** Project Team Formation

Form a 3 or 4 person project team, **Come up with a team name**, and fill in the table below:

Team Name:	
Person1:	
Person2:	
Person3:	
Person4:	

HW4 Problem 3 (**15pts)** Auth_blk.sv

You are on your own for this one…it is not covered by an exercise.

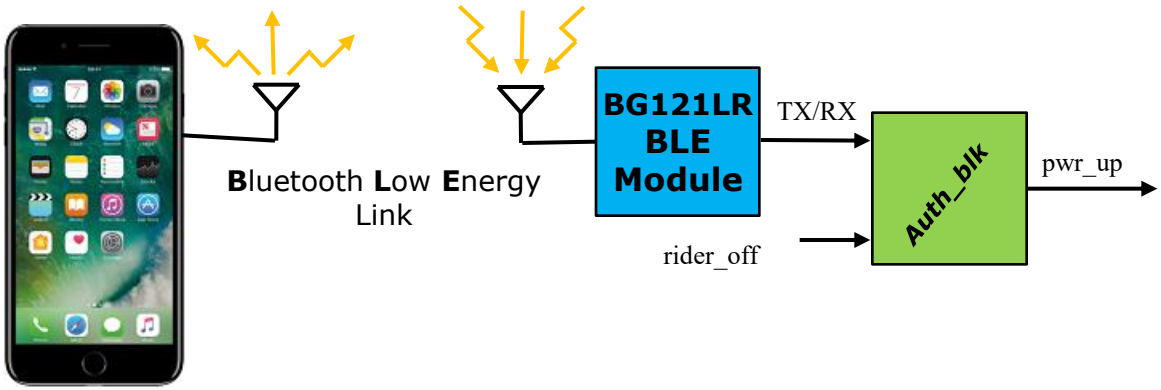
- An authorized rider will carry a phone with an app that sends the proper authorization code to a BLE module on the "Segway" device. The Segway control board has a BLE121LR module on that will be advertising a "Segway" service. When the users phone scans the service, connects, and sends the appropriate authorization code it will cause the BLE121LR module to send out 0x47 ('G') over its UART TX line. This will in turn cause the **pwr_up** signal to be asserted

- When the phone app deliberately disconnects, or is disconnected due to range the BLE121LR module sends out 0x53 ('S') over its UART TX line . The "Segway then shuts down (if the weight on the platform no longer exceeds MIN_RIDER_WEIGHT (**rider_off** signal from en_steer block).

- The UART will send at 9600 using 8N1 variant.

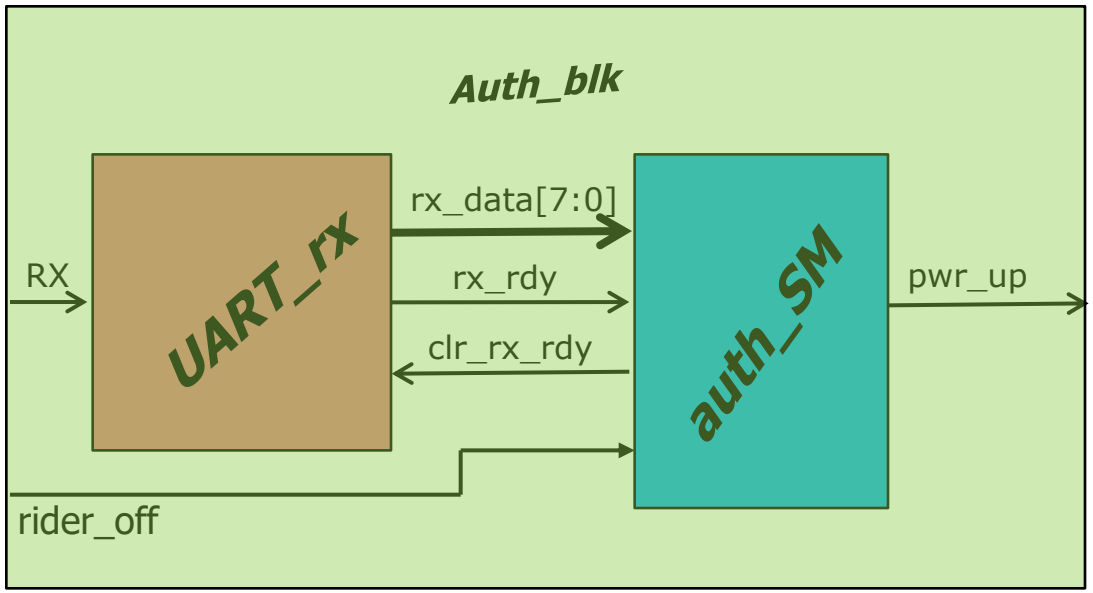
- Of course once the "Segway" is enabled (**pwr_up** asserted) it will stay enabled as long as there is a rider on the device. We wouldn't want it to power down and throw the rider just because the phone went dead or the BLE link was interrupted. We have a signal (**rider_off**) that indicates the weight on the platform does not exceed the minimum allowed rider weight.

- **pwr_up** goes to the ***balance_cntrl*** unit to enable it.



HW4 Problem 3 (**15pts)** Auth_blk.sv

- **Auth_blk** will be constructed from a UART receiver (***UART_rx***) and a simple state machine comparing the receptions to 'G', 'S', and monitoring the **rider_off** signal.

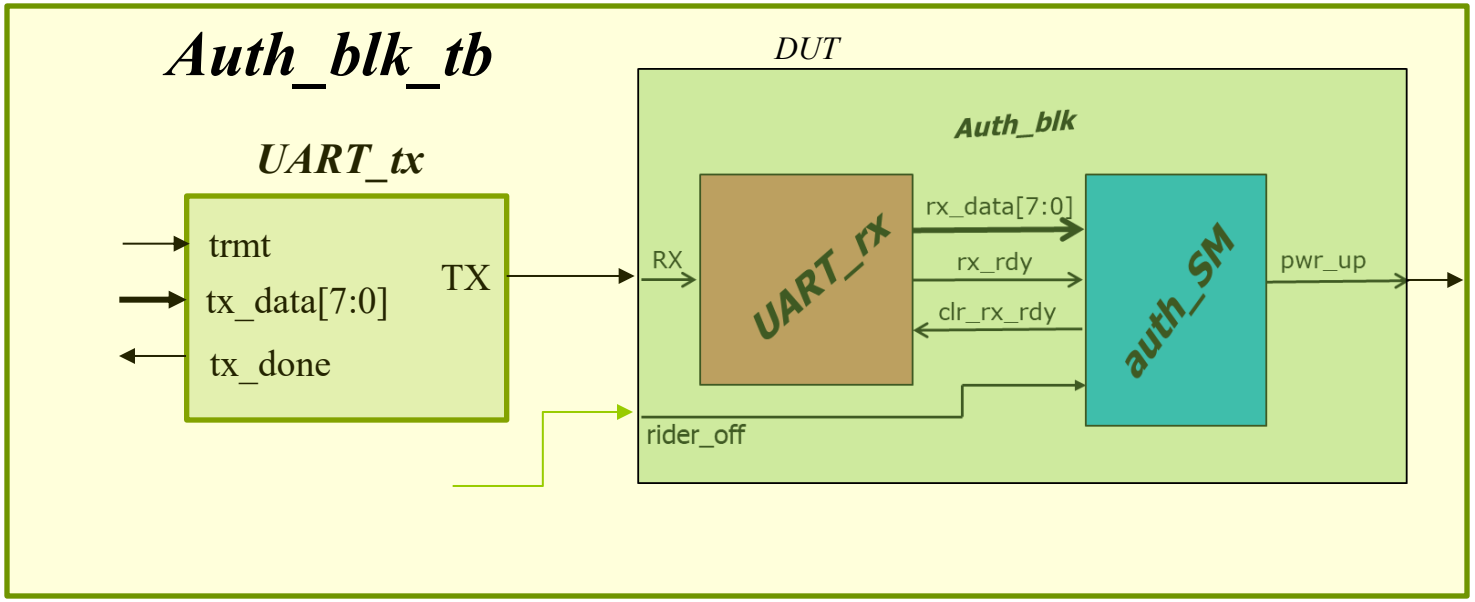


Draw a bubble diagram. Simulate its operation (in your head) under different scenarios. If it helps…my solution had 3 states.

- **pwr_up** is asserted upon reception of 'G' (0x47). It is deasserted after the last reception was 'S' and the **rider_off** signal is high.

- **UART_rx** was developed as part of HW3

HW4 Problem 3 (**15pts)** Auth_blk.sv (Testing It)



You need to develop both **Auth_blk.sv** and a test bench for it (**Auth_blk_tb.sv**). Remember you have a **UART_tx** block that was part of HW3.

Test more than the simple case of **pwr_up** and then down. Try to traverse all states and arcs of your SM.

Submit: **Auth_blk.sv**, **Auth_blk_tb.sv**, & proof it ran/passed in your testbench.

HW4 Problem 4 (**15pts)** Synthesize your **balance_cntrl.sv**

- Write a synthesis script (**balance_cntrl.dc**). The script should perform the following:
  - Defines a clock of 125MHz frequency and sources it to clock
  - Performs a set don't touch on the clock network
  - Defines input delays of 0.3 ns on all inputs other than clock
  - Defines a drive strength equivalent to a 2-input nand of size 2 from the Synopsys 32nm library (NAND2X2_RVT) for all inputs except clk and rst_n
  - Defines an output delay of 0.75ns on all outputs.
  - Defines a 50fF load on all outputs.
  - Sets a max transition time of 0.15ns on all nodes.
  - Employs the Synopsys 32nm wire load model for a block of size 16000 sq microns
  - Compiles, then flattens the design so it has no hierarchy, and compiles again.
  - Produces a min_delay report
  - Produces a max_delay report
  - Produces an area report
  - Flattens the design so it has no hierarchy
  - Writes out the gate level verilog netlist (**balance_cntrl.vg**)

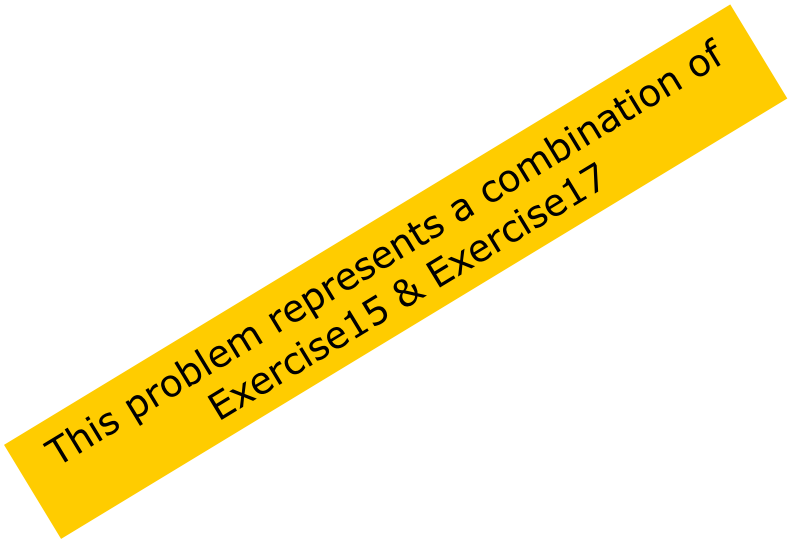
Started as Exercise18 on Weds Oct 29th

For reference my area was 4000 square microns

- Submit:
  - Your synthesis script (**balance_cntrl.dc**)
  - The output reports for area (**area.txt**)
  - The gate level verilog netlist (**balance_cntrl.vg**)

HW4 Problem 5 (**20pts)** Balance Control & Testing

- See Exercise15 (adding ***fast_sim*** to PID and creating **balance_cntrl.sv**)
- See Exercise17 (random testing of ***balance_cntrl***)
- Submit:
  - balance_cntrl.sv
  - PID.sv
  - balance_cntrl_chk_tb.sv (self-checking random vector testbench)
  - Proof balance_cntrl.sv passed the above testbench.



HW4 Problem 6 (**35pts)** SPI Tranceiver

- See Exercise16 (SPI Tranceiver)
- Did you do a good job on ***SPI_mnrch***? If not polish it up now.
- Submit:
  - SPI_mnrch.sv
  - SPI_mnrch_tb.sv
  - Proof your DUT passed the self-checking testbench

We start this problem as Exercise16 on Mon Oct 20th