Exercise 16: Auth_blk (work in teams of 2)

- 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 0x67 ('g') over its UART TX line. This will intern cause the **pwr_up** signal to be asserted

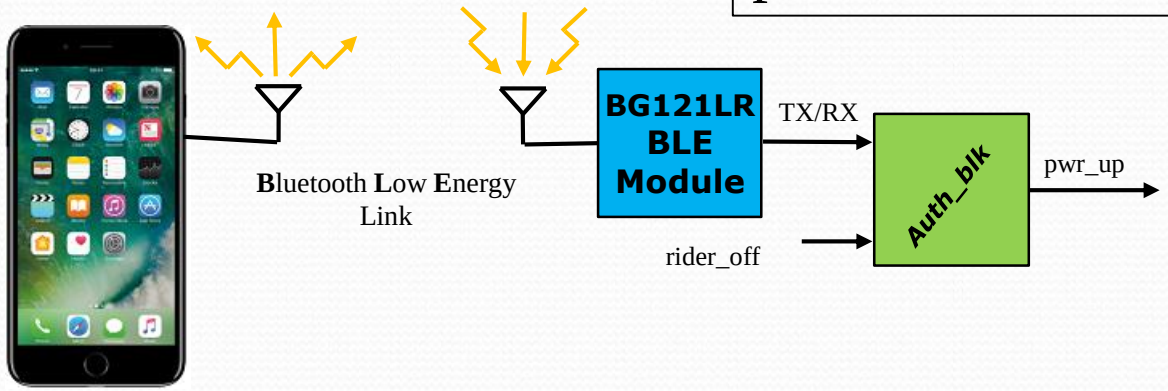
- When the phone app deliberately disconnects, or is disconnected due to range the BLE121LR module sends out 0x73 ('s') over its UART TX line . The "Segway then shuts down (if the weight on the platform no longer exceeds MIN_RIDER_WEIGHT (**rider_off** signal from en_steer block).

- The UART will send at 19200 using 8N1 variant.

- Of course once the "Segway" is enabled 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.

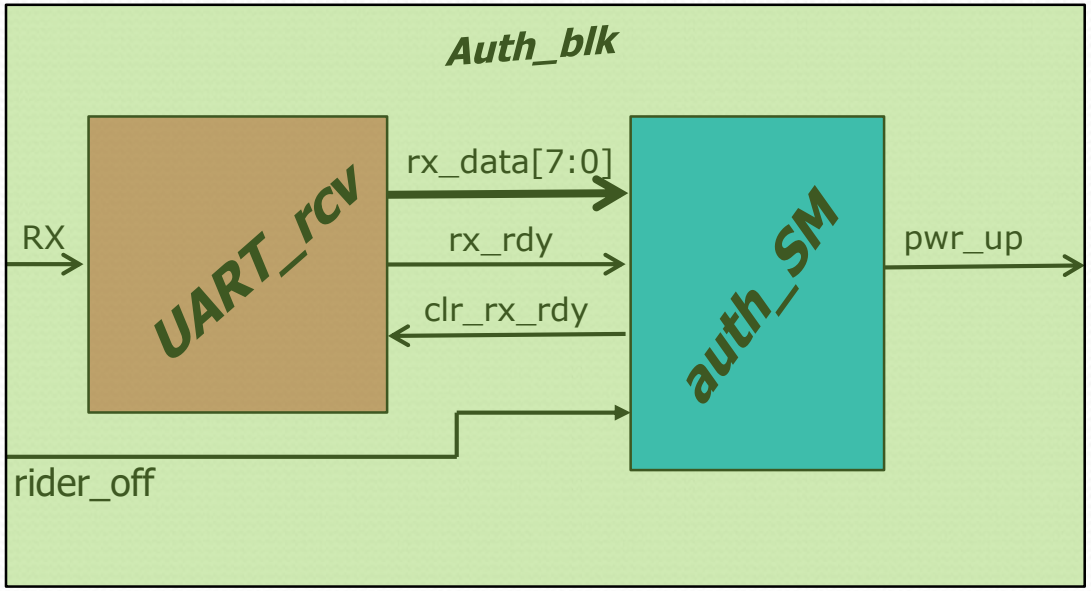
- **pwr_up** goes to the ***balance_cntrl*** unit to enable it.

This was the design problem on the Midterm



Auth_blk

- **Auth_blk** will be constructed from a UART receiver (UART_rcv) and a simple state machine comparing the receptions to 0x67, 0x73, and monitoring the **rider_off** signal.

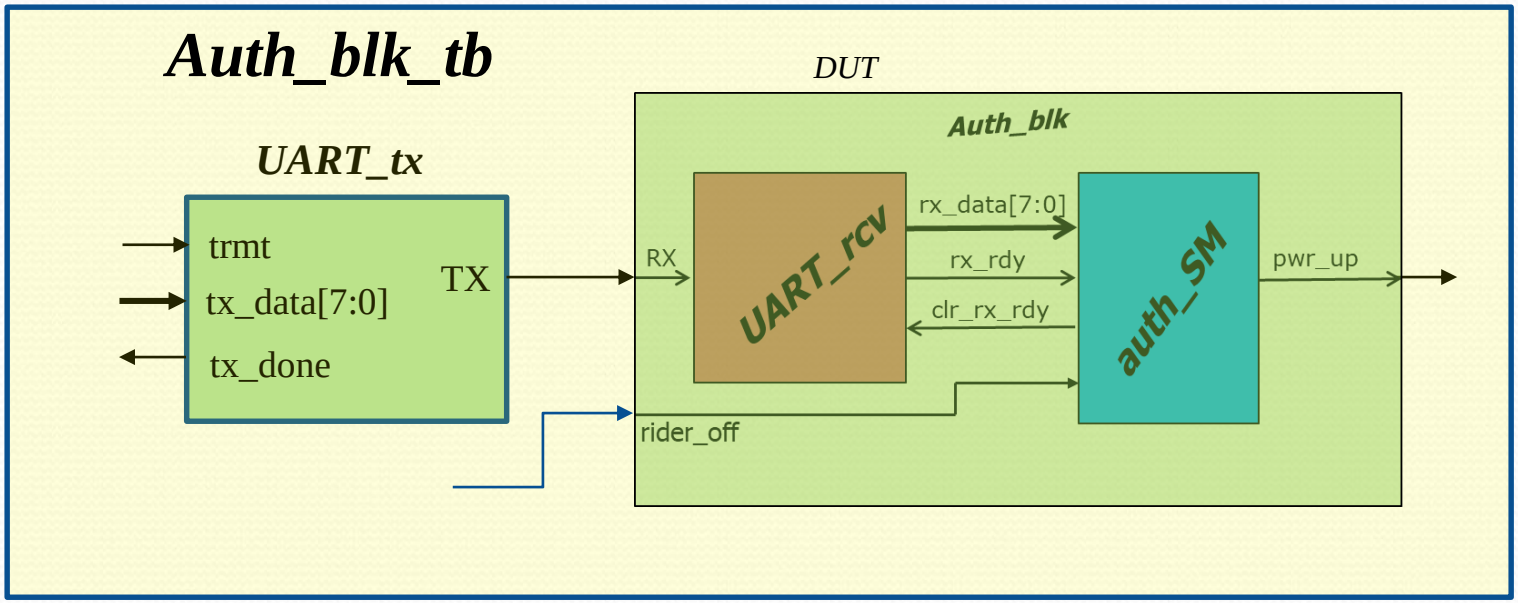


Work with your partner to draw a bubble diagram. Discuss its operation under different scenarios. If it helps…my solution had 3 states.

- **pwr_up** is asserted upon reception of 'g' (0x67). It is deasserted after the last reception was 's' and the **rider_off** signal is high.

- **UART_rcv** will be developed as part of HW3

Exercise 16: Auth_blk (Testing It)

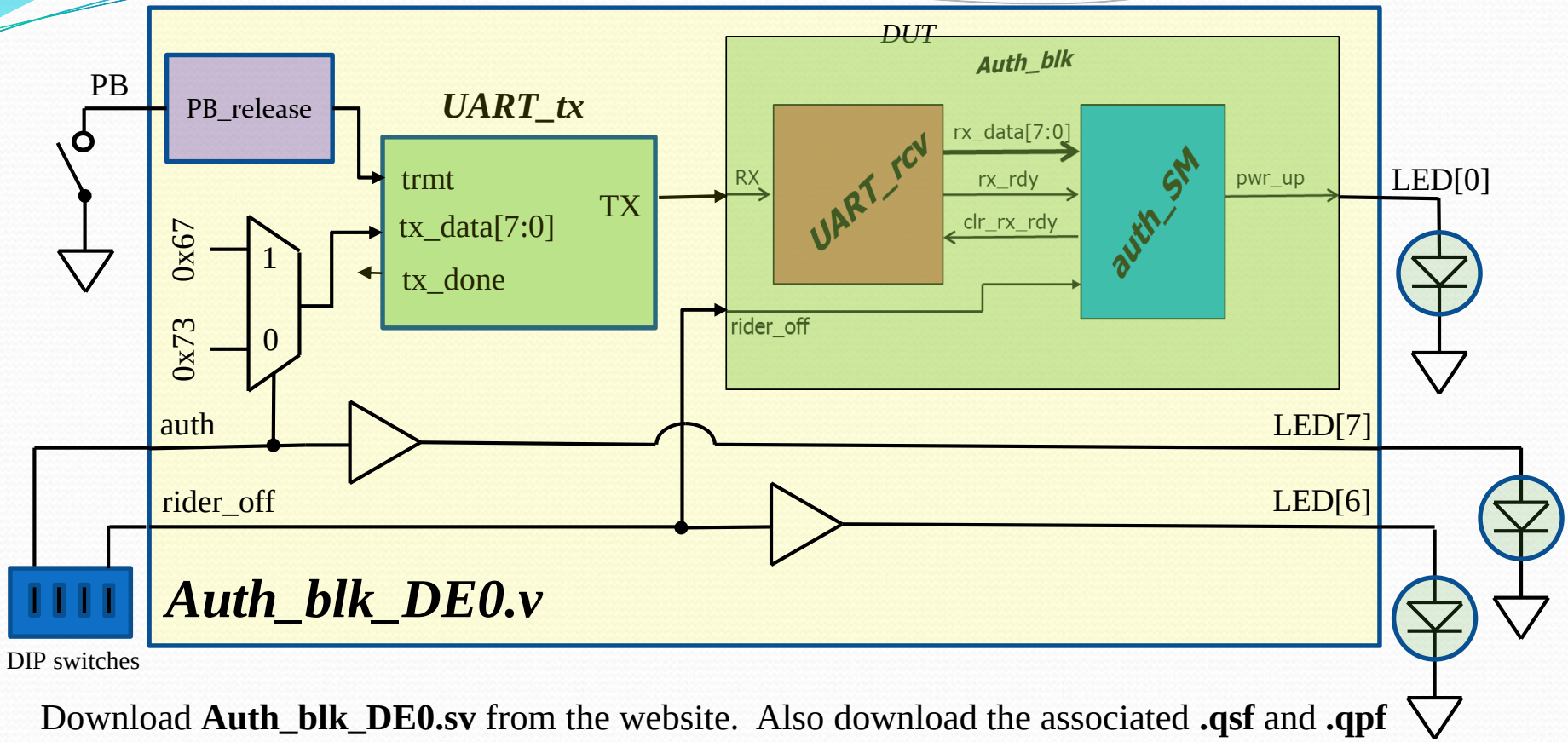


One of you create **Auth_blk.sv** while the other works on creating a test bench for it (**Auth_blk_tb.sv**). Remember you have a **UART_tx** block that was part of HW3. You can use my posted solutions for both **UART_rcv** and **UART_tx** if yours is not good.

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

Ensure it is good. We will test it on DE0-Nano next class

Exercise 16: Part II (Testing on DE0)



Download **Auth_blk_DE0.sv** from the website. Also download the associated **.qsf** and **.qpf** files.

You should have a **rst_synch.sv** and a **PB_release.sv** from exercise10.

You should be able to figure (by inspection) out how this should behave when mapped to the DE0

When you have it working call either Fego or Eric over to get checked off.