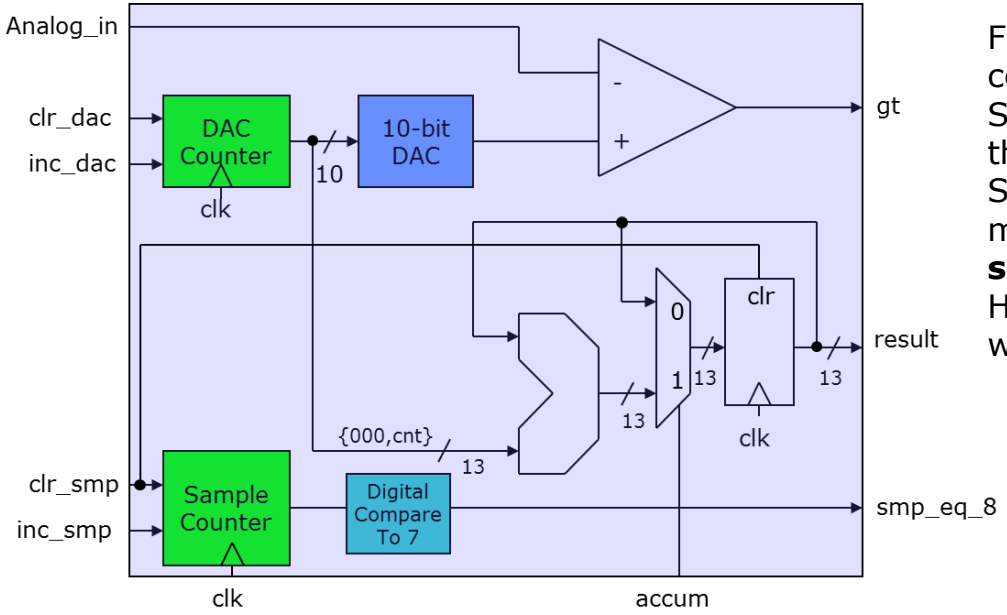
ECE 551
HW3

- Due Wed Oct 18th @ class (9:55AM)
- Work Individually
- Remember What You Learned From the Cummings SNUG paper
- Use descriptive signal names and comment your code

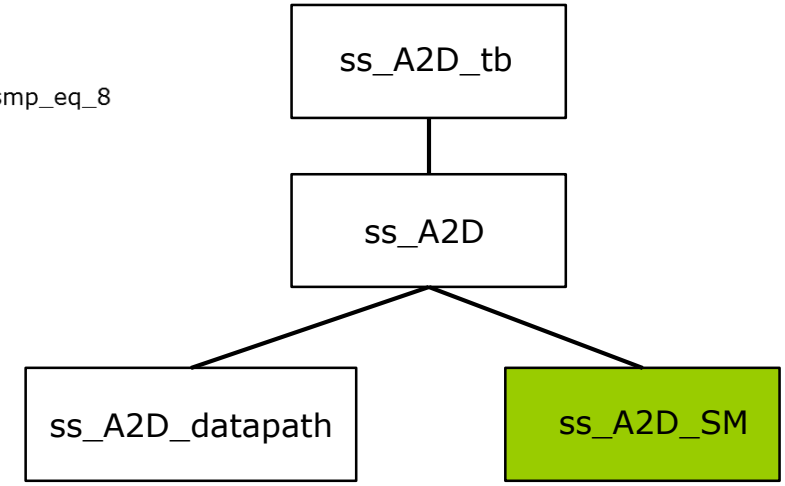
HW3 Problem 1 (**20pts)** SM Design

A Single Slope A2D converter (capable of averaging 8 samples) was discussed in Lecture1. The block diagram is shown below. On the course webpage under HW3 you will find files: **ss_A2D_tb.v**, **ss_A2D.v**, **ss_A2D_datapath.v**, **ss_A2D_SM_shell.sv**.



Flush out **ss_A2D_SM_shell.sv** to complete the control (state machine). Simulate your resulting design using the provided self checking test bench. Submit your verilog for the state machine (file should be called **ss_A2D_SM.sv** to the dropbox for HW3). Also submit proof that it worked.

Hierarchy of design is as shown. Testbench instantiates DUT (ss_A2D). Which in turn instantiates datapath and statemachine. You are simply flushing out the SM code and renaming the file from ss_A2D_SM_shell.sv to ss_A2D_SM.sv. You are also simulating to prove it works.



HW3 Problem 2 (**25pts**) ESC Interface

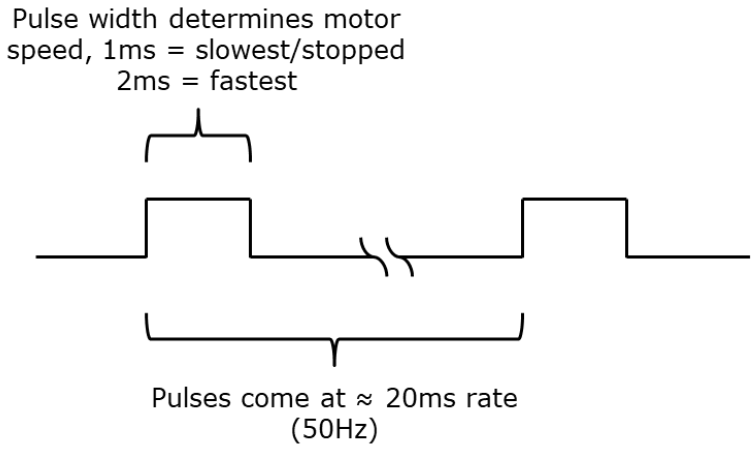
In the vicinity of slide 38 of the Project Spec you will find a section on ESC (Electronic Speed Control). Read these slides carefully.

You will be implementing the ESC controller, with the following interface:

Signal:	Dir:	Description:
clk	in	50MHz clock
rst_n	in	Active low asynch reset
SPEED[10:0]	in	Result from flight controller telling how fast to run each motor.
OFF[9:0]	in	**Unsigned** offset added to SPEED to compensate for variation in ESC/motor pairs. Give all 4 motors the same pulse width and they don't run close to the same speed. This offset is used to compensate that to first order.
PWM	Out	Output to the ESC to control motor speed. It is effectively a PWM signal.

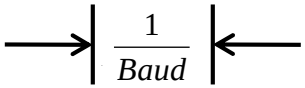
- If both SPEED and OFF were zero the PWM width would be 1ms.
- For every count of SPEED or OFF the speed would increase by 0.32usec

This problem will have been completed as part of in class exercises. Just submit your ESC_interface.sv that was checked off in class.

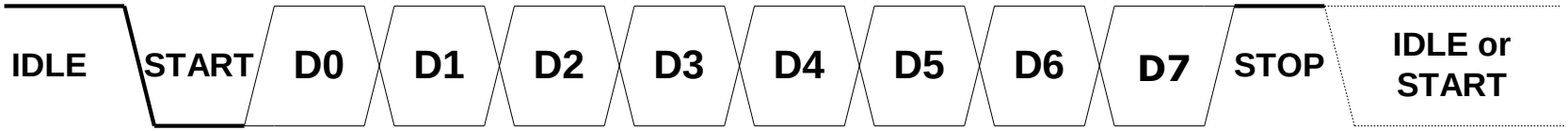


What is UART (RS-232)

- RS-232 signal phases
  - Idle
  - Start bit
  - Data (8-data for our project)
  - Parity (no parity for our project)
  - Stop bit – channel returns to idle condition
  - Idle or Start next frame

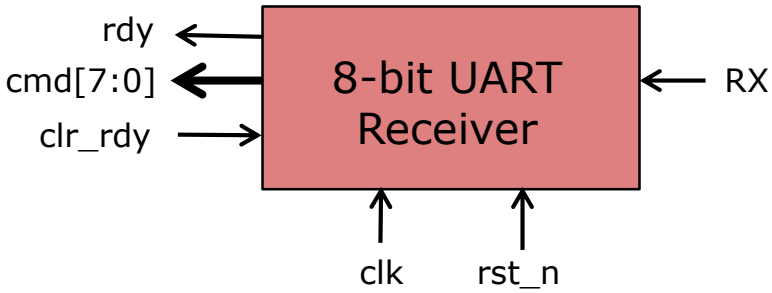


Baud rate will be 19,200 with 50MHz clock ➔ 2604 divider ➔ 12-bit



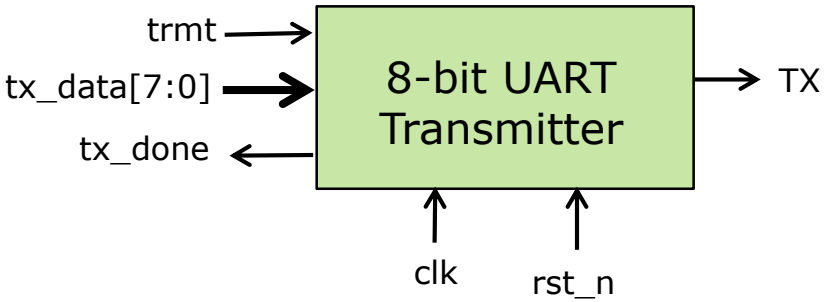
- Receiver monitors for falling edge of Start bit. Counts off 1.5 bit times and starts shifting (right shifting since LSB is first) data into a register.
- Transmitter sits idle till told to transmit. Then will shift out a 9-bit (start bit appended) register at the baud rate interval.

UART Receiver/Transmitter



A host computer will send commands to the Logic Analyzer via a UART serial peripheral

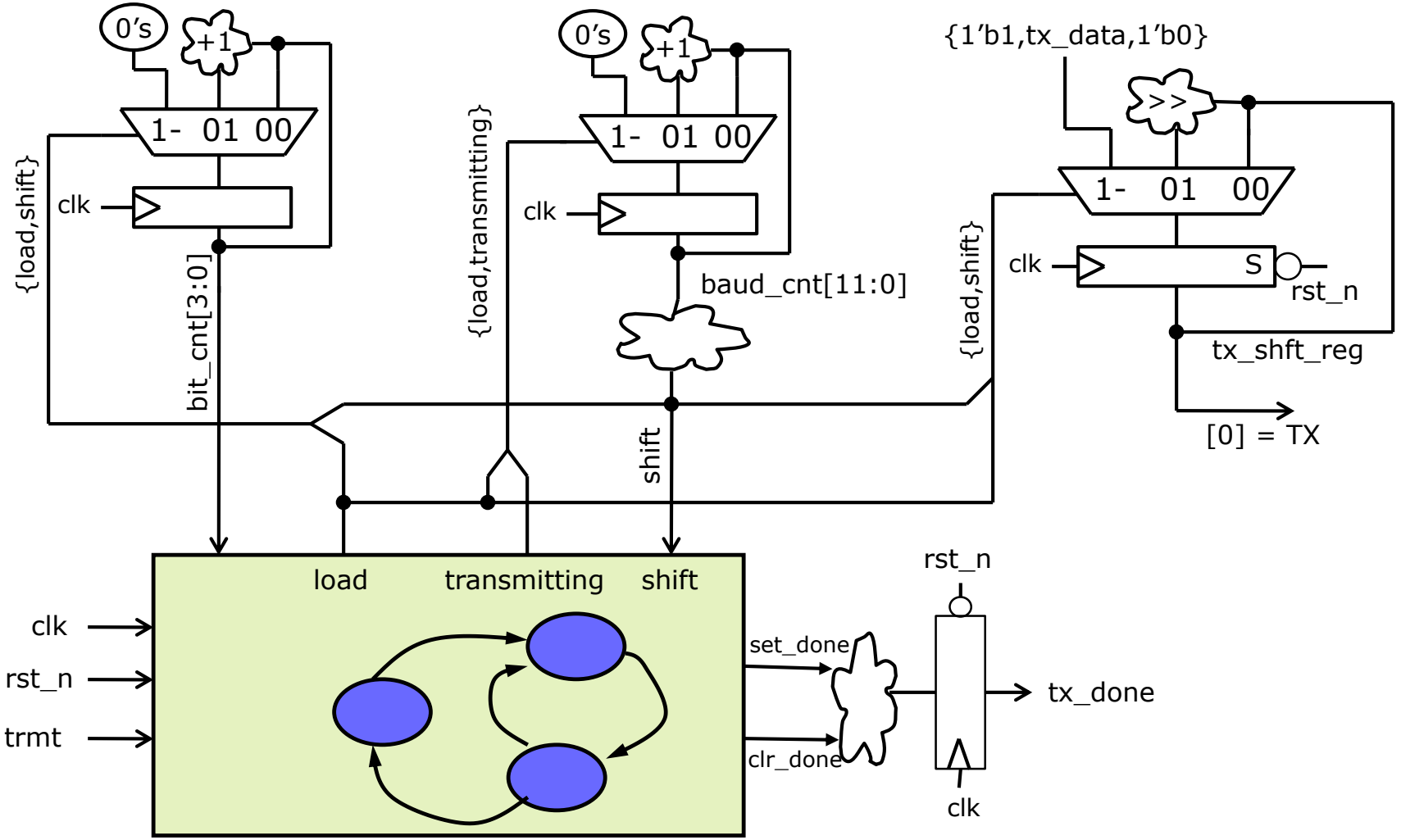
Signal:	Dir:	Description
clk,rst_n	in	100MHz system clock & active low reset
RX	in	Serial data carrying command from host computer
rdy	out	Asserted when a byte has been received. Falls when new start bit comes, or when ***clr_rdy*** knocks it down.
cmd[7:0]	out	Byte received (serves as command to LA)
clr_rdy	in	Asserted to knock down the **rdy** signal.



The follower sends responses back to the host computer. These reponses are sent via a UART serial peripheral.

Signal:	Dir:	Description
clk,rst_n	in	100MHz system clock & active low reset
TX	out	Serial data output back to host
trmt	in	Asserted for 1 clock to initiate transmission
tx_data[7:0]	in	Byte to transmit (response from LA)
tx_done	out	Asserted when byte is done transmitting. Stays high till next byte transmitted.

Possible Topology of UART_tx



HW3 Problem 3 (**20pts)** UART Transmitter

Implement a the UART Transmitter (**UART_tx.sv**).

Make a simple test bench for it. This is one instance in which I would not spend too much time on the test bench. You can just instantiate your transmitter and send a few bytes. Verify the correct functionality (including baud rate) by staring at the green waveforms. You will make a more comprehensive test bench in the next problem.

Submit **UART_tx.sv** to the dropbox for HW3.

HW3 Problem 4 (**25pts + 10pts)** UART Receiver

Implement a the UART Receiver (**UART_rcv.sv**).

Since you have a transmitter too, it is now easy to make a self checking test bench. Architect the test bench as shown. Does the 8-bit value you transmit match the value you receive when the transmission completes?

Submit **UART_rcv.sv** and your test bench to the dropbox for HW3.