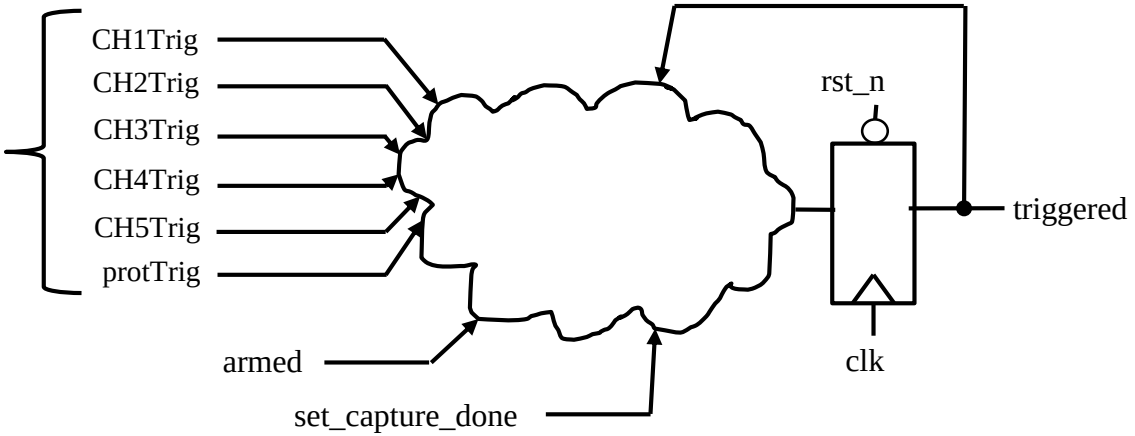
ECE 551
HW3

- Due Thurs March 4th in class
- Work Individually
- Remember What You Learned From the Cummings SNUG paper
- Use descriptive signal names and comment your code

HW3 Problem 1 (**10pts)** Trigger Logic

Done as Exercise08 on Fri Feb 21st

trigger sources:
All trigger sources must be high for a trigger to occur (if, for instance, CH4Trig was not supposed to influence the trigger it would be set high external to this block)



All trigger sources have to be high in order for **triggered** to get set.

armed is an external signal and indicates we have already sampled enough samples such that we know we can accept a trigger immediately. **triggered** cannot go high unless **armed** is asserted.

set_capture_done comes from an external control unit and is asserted once the logic analyzer has finished capturing channels into the RAMqueue. It should clear **triggered**. It should have priority over all the other signals (**armed** & the trigger sources).

Once **triggered** goes high it should stay high until **set_capture_done** knocks it down.

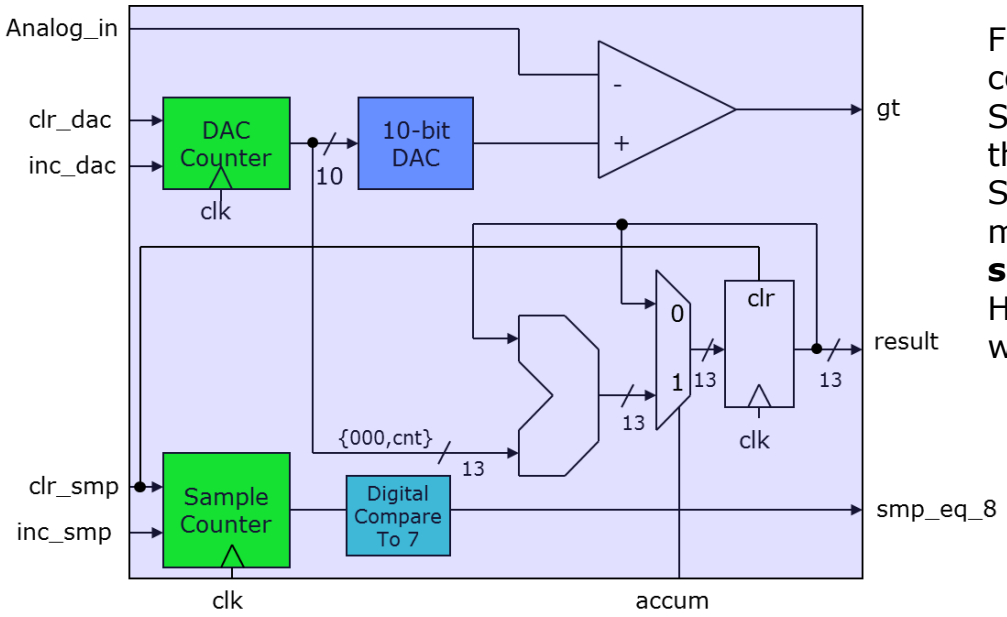
Reference Ex08 for what should be tested.

Submit to the dropbox: **trigger_logic.sv**, **trigger_logic_tb.sv**, and proof that you ran it.

HW3 Problem 2 (**20pts**) SM Design

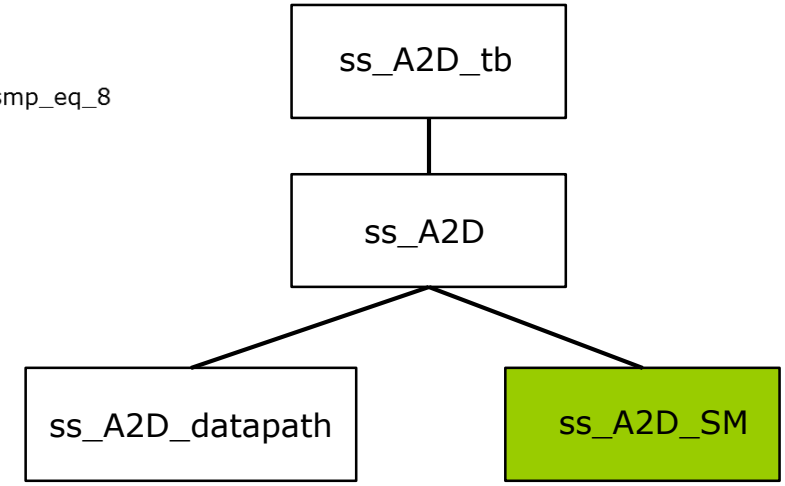
You are on your own for this one. It is your SM warm up

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 3 (**20pts**) Dual PWM

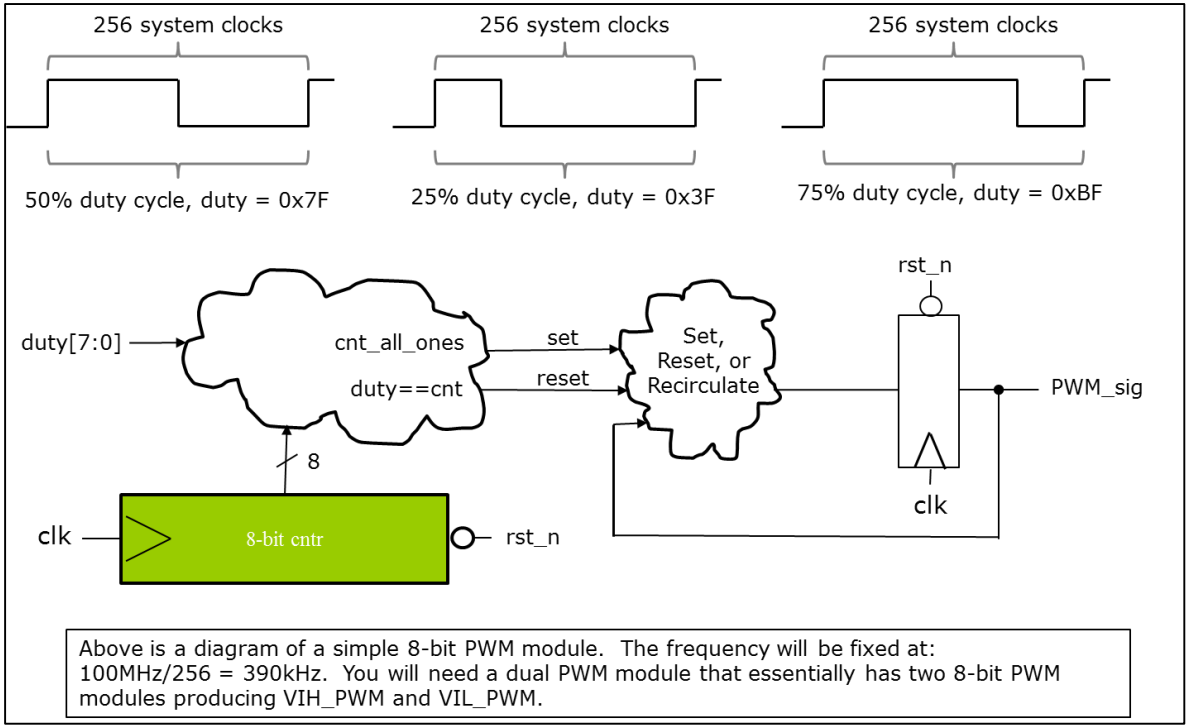
Done as Exercise09 on Mon & Weds Feb 24th & Feb 26th

The project spec has a couple of slides (around slide 10) about what PWM is, and how we are using it in this project.

Create a simple 8-bit PWM block (**PWM8.sv**). Create a test bench for it and simulate/test/debug it.

Now create **dual_PWM.v** by instantiating two copies of your PWM8 block.

<u>Submit to the dropbox:</u>
PWM8.sv
PWM8_tb.v
dual_PWM.v
Proof that you simulated.



Above is a diagram of a simple 8-bit PWM module. The frequency will be fixed at: 100MHz/256 = 390kHz. You will need a dual PWM module that essentially has two 8-bit PWM modules producing VIH_PWM and VIL_PWM.

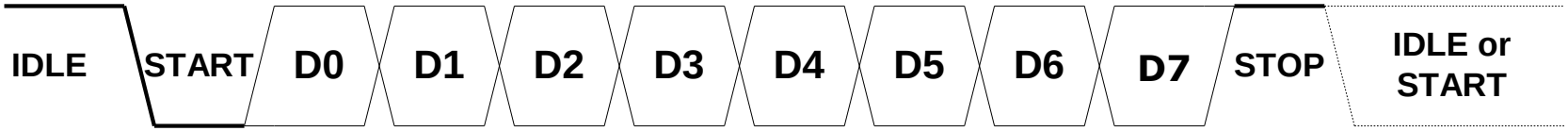
Signal:	Dir:	Description:
clk	In	100MHz main system clock
rst_n	In	Asynch active low reset
VIH[7:0],VIL[7:0]	In	8-bit vectors specifying the duty cycle (thresholds) for VIH and VIL respectively.
VIH_PWM,VIL_PWM	Out	The PWM outputs that will be low passed to create the VIL and VIH thresholds.

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

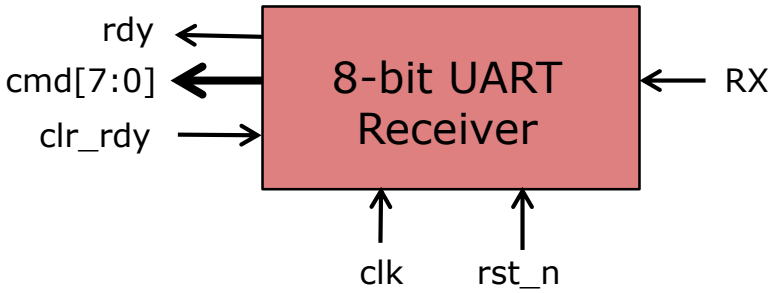
$\frac{1}{Baud}$

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



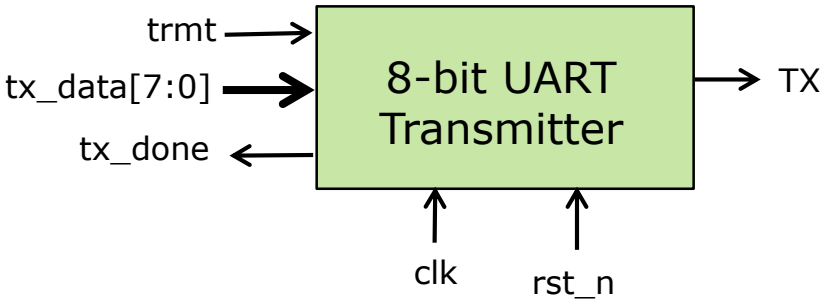
- Receiver monitors for falling edge of Start bit. Counts off 1.5 bit times and starts shifting (right shifting since LSB is first) data into a register.
- Transmitter sits idle till told to transmit. Then will shift out a 9-bit (start bit appended) register at the baud rate interval.

UART Receiver/Transmitter



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

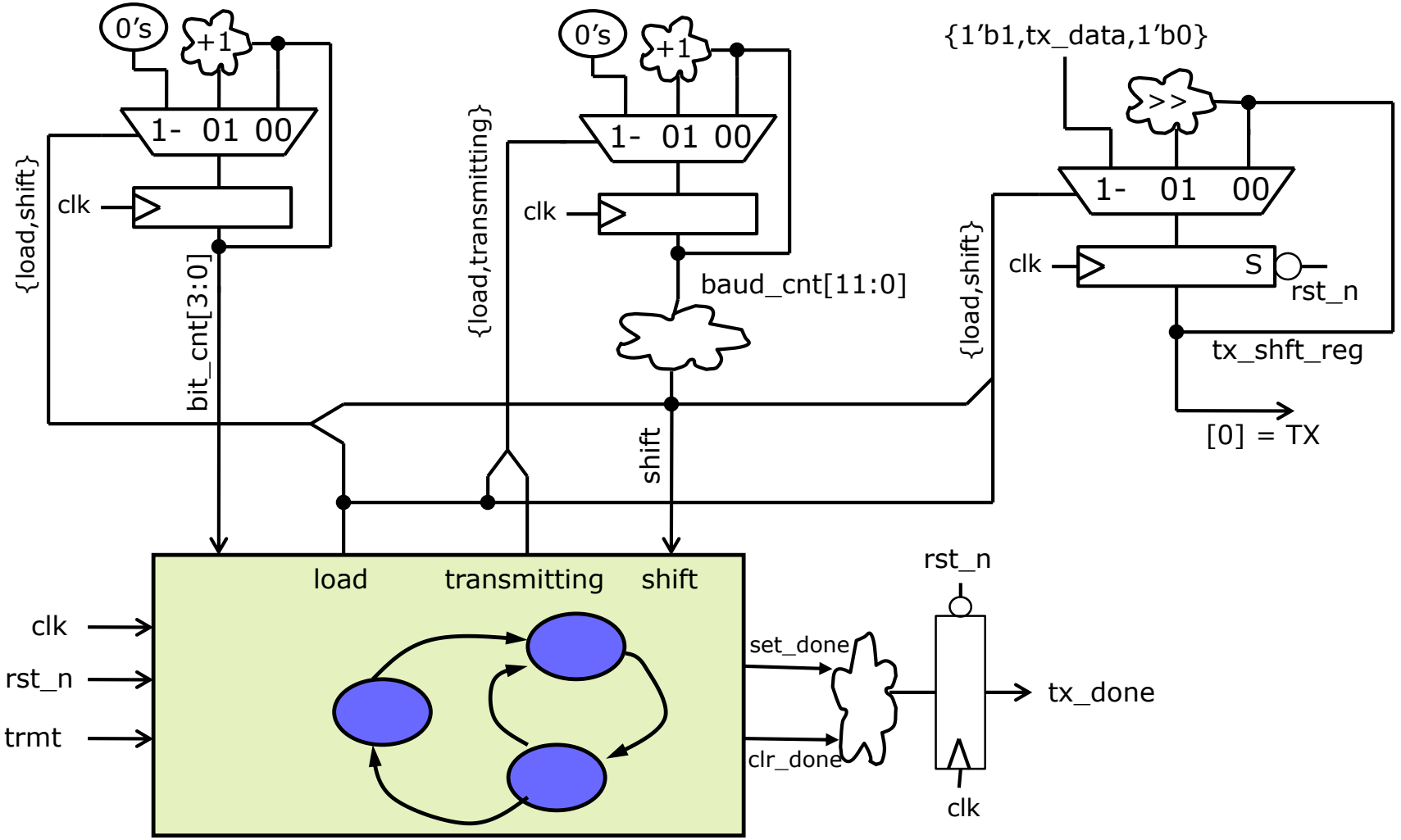
Signal:	Dir:	Description
clk,rst_n	in	100MHz system clock & active low reset
RX	in	Serial data carrying command from host computer
rdy	out	Asserted when a byte has been received
cmd[7:0]	out	Byte received (serves as command to LA)
clr_rdy	in	Asserted to knock down the **rdy** signal.



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 4 (**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.

Started as Exercise10 on Fri Feb 28th

HW3 Problem 5 (**20pts + 10pts)** UART Receiver

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

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

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