ECE 551

HW4 *(100 pts)*

- Due Thurs Nov 4th@ 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, Ayan Deep Hazra 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. **(10pts)** Project Team Formation

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

Team Name:	KASA
Person1:	Kab Kalaichelvan
Person2:	Anirudh Jayendra
Person3:	Suvya Santhan
Person4:	Ayan Deep Hazra

HW4 Problem 3 (**20pts)** Synthesize your UART

Started as Exercise17 on Mon Nov 1st

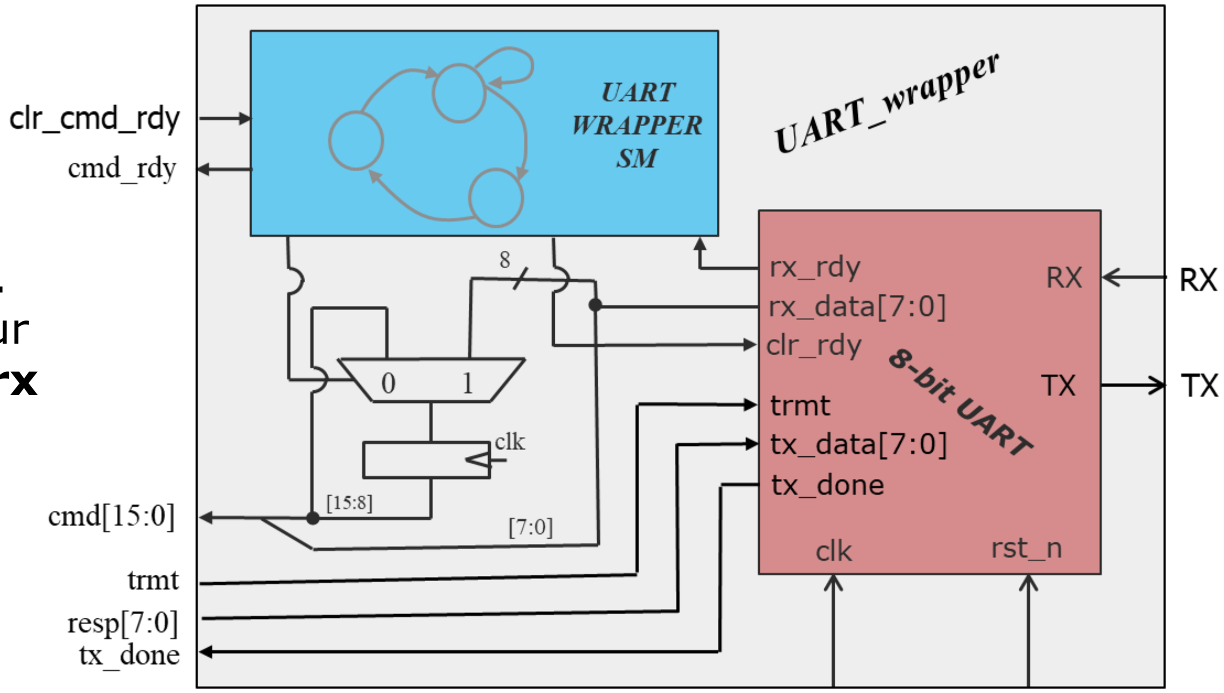
- In Ex14 & 15 your **UART_tx** and **UART_rx** were combined to produced a UART transceiver (**UART.v**). You will now synthesize UART.v and its childred (UART_tx & UART_rx).
- You will synthesize these modules using Synopsys on the CAE linux machines.
- (**NOTE:** if your UART modules are a big ball of stink, and one of your project partners has a better versions you can use their code. This exercise can be done as a team of 2)
- Write a synthesis script to synthesize your **UART.v**. The script should perform the following:
  - Defines a clock of 500MHz frequency and sources it to clock
  - Performs a set don't touch on the clock network
  - Defines input delays of 0.5 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_LVT) for all inputs except clock and rst_n
  - Defines an output delay of 0.75ns on all outputs.
  - Defines a 0.15pf load on all outputs.
  - Sets a max transition time of 0.15ns on all nodes.
  - Employ the TSMC32K_Lowk_Conservative wire load model.
  - Produces a min_delay report
  - Produces a max_delay report
  - Produces an area report
  - Writes out the gate level verilog netlist (UART.vg)
- Submit to the dropbox.
  - Your synthesis scripts (UART.dc)
  - The output reports for area (UART_area.txt)
  - The gate level verilog netlist (UART.vg)

Started as Exercise14 on Mon Oct 18th

HW4 Problem 4 (**25pts)** UART_wrapper/RemoteComm

- "The Knight" receives a 16-bit command that tells it the navigation moves it will make via Bluetooth. The Bluetooth module sends this command via UART *(a byte based protocol).* You need to make a wrapper to package two bytes into a single 16-bit command.

- A file **UART.v** is available for download. It simply combines your **UART_tx** and **UART_rx** together in a single module.



- Create **UART_wrapper.sv** (according to the diagram above). Instantiate the downloaded **UART.sv** and then add the simple datapath and control SM around it. We will work on testing it Friday during Exercise15.

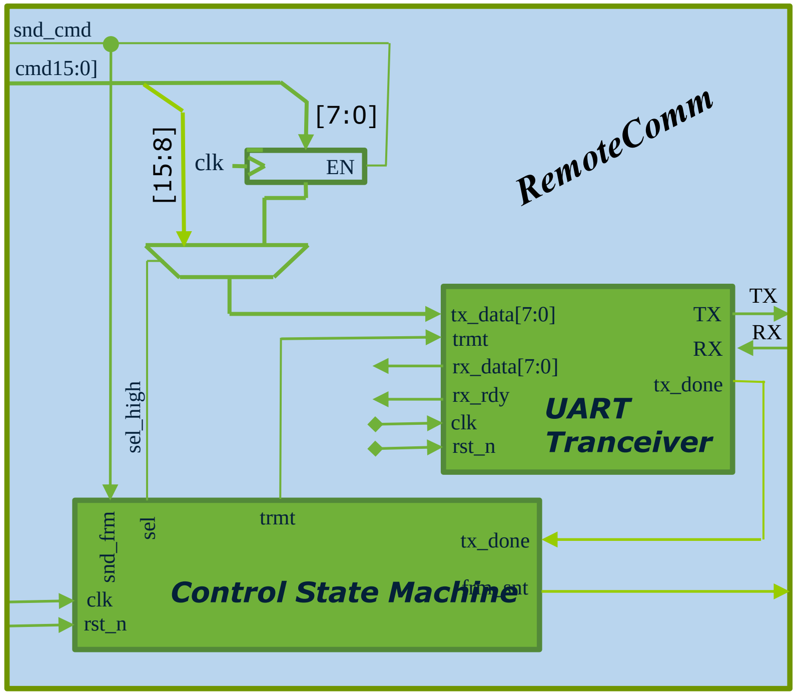
HW4 Problem 4 (25pts) UART_wrapper/RemoteComm

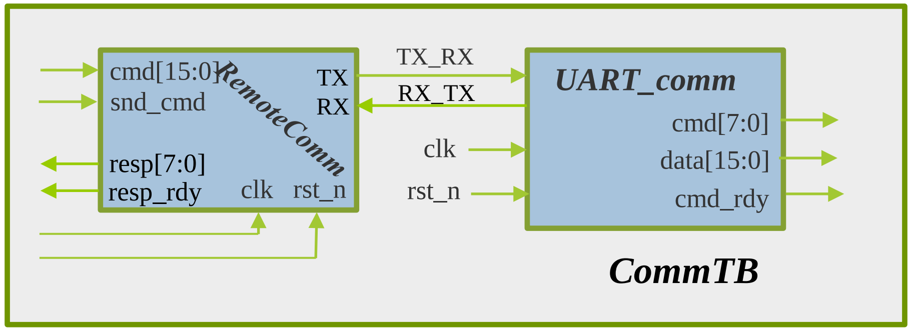
- How are you going to test **UART_Wrapper**? Wouldn't it be nice to have a block that accepted a 16-bit command and sent it as two 8-bit UART transmissions?

- **RemoteComm** performs the opposite function as **UART_wrapper**. It takes a 16-bit command and sends it as two 8-bit bytes over UART.

- Create **RemoteComm.sv**

- Use **RemoteComm.sv** to create a self-checking test bench for **UART_wrapper**. Call it **CommTB.v**.





Submit: **UART_wrapper.sv**, **RemoteComm.sv**, & **commTB.sv** as well as proof your self-checking ***commTB*** passed.

HW4 Problem 5 **(35pts)** SPI Tranceiver

Started as Exercise16 on Wed Oct 27th

- Reference Exercise16 for detailed description
  - Submit:
    - ✓ SPI_mnrch.sv
    - ✓ SPI_mnrch_tb.sv
    - ✓ Proof of testbench run (transcript window output).