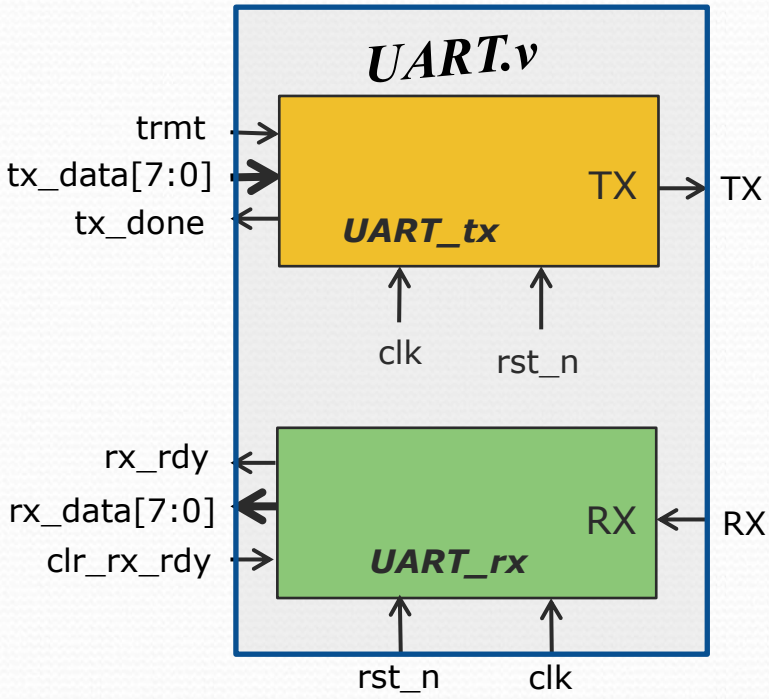
Exercise 18 Synthesizing your *UART.sv* *(HW4 Problem 3)*

- You should have a UART transceiver *(**UART.sv** see Ex15)* that makes use of your ***UART_tx*** and ***UART_rx*** blocks.



Signal:	Dir:	Description:
clk,rst_n	in	50MHz clk, active low asynch reset
trmt	in	Initiates UART transmisison
tx_data[7:0]	in	Data to transmit
tx_done	out	Indicates transmission complete
rx_rdy	out	New 8-bit data ready
clr_rx_rdy	in	Knock down rdy flag
rx_data[7:0]	out	8-bit data received
TX	out	Serial data out line
RX	in	Serial data in line

Exercise 18 Synthesizing your *UART.sv* *(HW4 Problem 3)*

- Under the "Tutorials" section of the class webpage there is a video about accessing CAE linux machines remotely. There is another one about FTPing files.

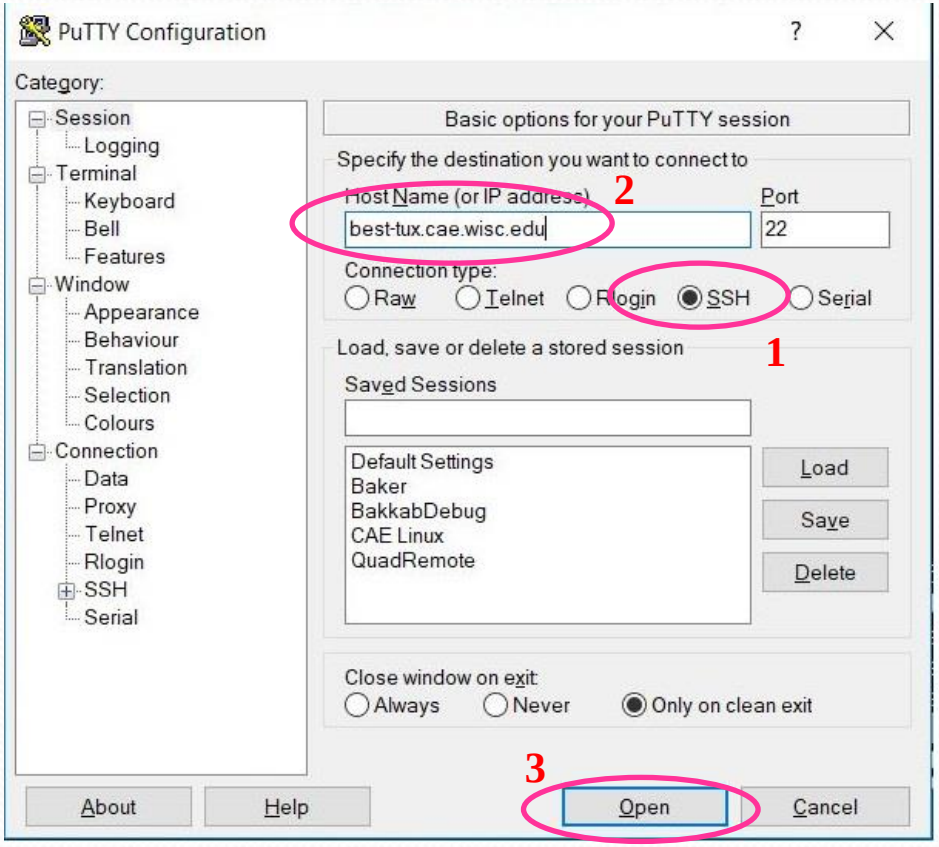
- Using either PuTTy or MobaXterm, or whatever SSH terminal client you want to use login to:
  best-tux.cae.wisc.edu

- Once logged in to Linux make an ece551 directory:
  `mkdir ece551`

- Change directory into your newly created **ece551** directory.
  `cd ece551`

- Now make an **Exercise18** directory inside **ece551**.
  `mkdir Exercise18`

Here showing how to login using PuTTy



Exercise 18 Synthesizing UART.sv

- Now you have to FTP (File Transfer Protocol) your needed verilog files over to your **Exercise18** directory in Linux. Please refer to the video on the Canvas page….Mac users…you have to figure this out for yourselves.

- Now we will kick off Synopsys in command shell mode (which is how "real" engineers use it)

```
dc_shell
```

- When writing a synthesis script you need to know each line (Synopsys command) of the script works stand alone. If it does you copy it into the script. You **do not** just write a script and run it all at once in Synopsys. There will be errors you will miss if you do this.

- Synthesis scripts usually end with a **.dc** extension (**.dc** for design compiler)

Exercise 18 Synthesizing UART.sv

- Write a synthesis script (**UART.dc**) to synthesize your UART. 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 0.4 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 0.4ns on all outputs.
  - Defines a 0.10pf 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 & 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**)

For reference…my cell area was 720 and my total area was 893. I would expect yours to be "in the same ballpark" +/- 150

Some Hints are Given on Next Page:

cp ~ejhoffman/ece551/public/designvision/.synopsys_dc.setup .

vi .synopsys_dc.setup

cd ece551
cd Exercise18

dc_shell

Hints for Synthesis Scripting.

- About 1/3 of the way through Lecture06 there is an example synthesis script.
- A synthesis script is just a series of commands that you can directly type into ***dc_shell***. Always test each command in ***dc_shell*** first to make sure you are using it right, then copy that line into your synthesis script.
- When reading in System Verilog you have to use: **read_file –format sverilog {file_names}**. Note sverilog not verilog.
- When reading in several files of a design that have hierarchy it is best to do as follows:

read_file -format sverilog {./UART_tx.sv ./UART_rx.sv ./UART.sv}

```
read_file -format sverilog {UART_tx.sv UART_rx.sv UART.v}
```

  - NOTE: reading the children first, and parents later

- Then it is important to set the level you wish to synthesize next

```
set current_design UART
```

set current_design UART

- Also can be necessary to get it to traverse the design hierarchy so it knows who the children are:

```
link
```

最後兩句話:

remove_design -all

source commMod.dc