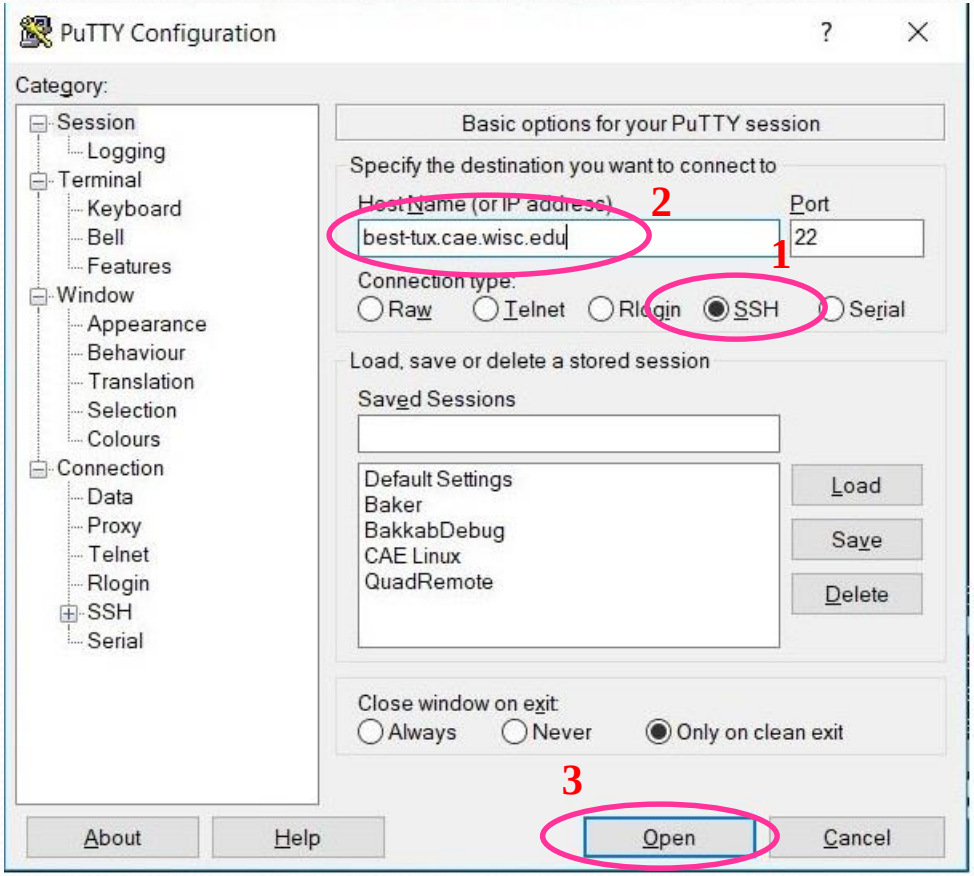
Exercise 13 Synthesizing your UART.v

- In HW3 you made **UART_tx.sv** and **UART_rcv.sv**. If you followed the signal naming outlined then you should be able to download **UART.v** from this exercise and use it directly. Download and inspect **UART.v**.

- Using PuTTy (installed on CAE machines) login to a Linux machine (easy as 1,2,3)

- Once logged in to Linux do an **ls**. You should see you have an **ece551** directory already (at least you should if you created on in your **I:drive** area.) Everything on your **I:drive** exists on Linux…magic!



Exercise 13 Synthesizing your UART.v

- Since your I:drive is mapped to your linux home and vice versa file transfer is not necessary. Create an exercise13 directory under your ece551 directory.
  - You can do this in linux by doing a: **cd ece551** and then doing a: **mkdir exercise13**
  - OR you can create it in Windows and it will appear in your Linux.

- Move your **UART_tx.sv** and **UART_rcv.sv** files that you should have completed for HW3 to your exercise13 directory. Also move the downloaded **UART.v** to your exercise13 area.

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

Exercise 13 Synthesizing your UART.v

- Write a synthesis script (**UART.dc**) to synthesize your UART. The script should perform the following:
  - Read in **UART_tx**, **UART_rcv.sv**, and **UART.v**
  - 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
  - Set a drive strength of 50 ohms on all inputs.
  - Define an output delay of 0.75ns on all outputs.
  - Defines a 0.10pf load on all outputs.
  - Sets a max transition time of 0.1ns on all nodes.
  - Employ the TSMC32K_Lowk_Conservative wire load model.
  - Produce an area report (**UART_area.txt**) (after synthesizing)
  - Writes out the gate level verilog netlist (**UART.vg**)

- Submit to the dropbox for Exercise13
  - Your synthesis script (**UART.dc**)
  - The output reports for area (**UART_area.txt**)
  - The gate level verilog netlist (**UART.vg**)

For reference…my UART was 470 square microns

(Some Hints Are Given On Next Page)

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_rcv.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
```

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

```
link
```