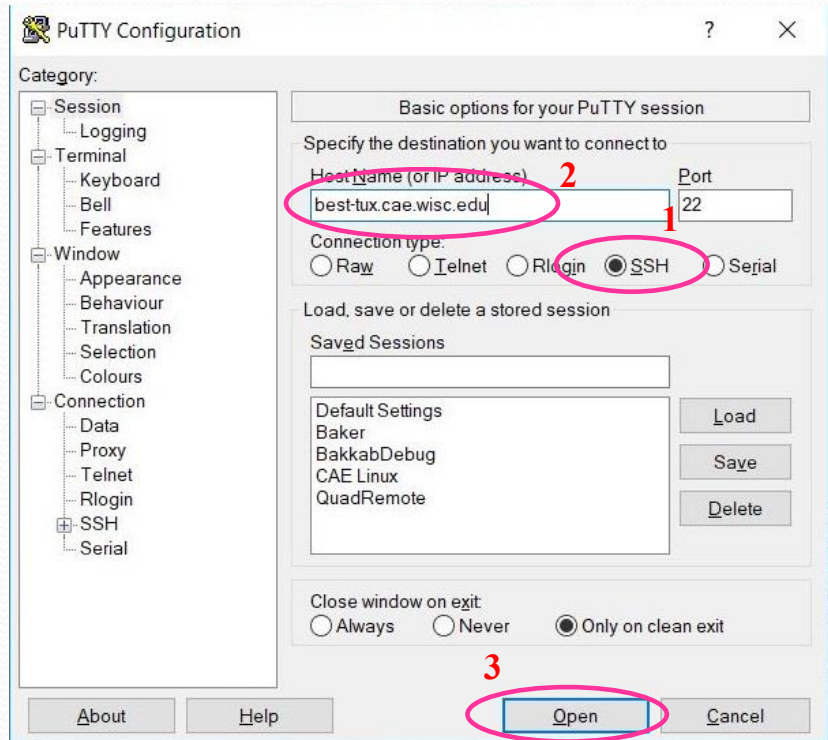


Exercise 14 Synthesizing your *telem* (HW4 Problem 3)

Do this with a partner, but
both people turn in results
(we don't have enough licences)

- In HW3 you made **telemetry.sv** (which in turn instantiated a provided **UART_tx.sv** block). Today's exercise is to synthesize **telemetry**.
- 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 14 Synthesizing telemetry.sv

- Since your I:drive is mapped to your linux home and vice versa file transfer is not necessary. Create an **exercise14** directory under your **ece551** directory.
 - You can do this in linux by doing a: **cd ece551** and then doing a: **mkdir exercise14**
 - OR you can create it in Windows and it will appear in your Linux.
- Copy your **telemetry.sv** Verilog that you should have completed for HW3 to your exercise14 directory. Also copy over **UART_tx.sv**
- 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 14 Synthesizing telemetry.sv

- Write a synthesis script (**telemetry.dc**) to synthesize your SPI. 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 1 from the Synopsys 32nm library (NAND2X1_RVT) 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.
 - 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 report
 - Produces a max_delay report
 - Produces an area report
 - Flattens the design so it has no hierarchy
 - Writes out the gate level verilog netlist (**telemetry.vg**)
 - Writes out a Synopsys Design Constraints file (**telemetry.sdc**)
- Submit to the dropbox for Exercise14
 - Your synthesis script (**telemetry.dc**)
 - The output reports for area (**telemetry_area.txt**)
 - The gate level verilog netlist (**telemetry.vg**)

Some Hints are Given on Next Page:

For reference...my cell area was 699 and my total area was 853. I would expect yours to be "in the same ballpark"

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
```