Open message to verificators

My esteemed verification professionals -
Especially those who verify my RTL.

I would like You to consider few suggestions of mine, that i believe will make my (and Yours) life easier and tape-outs closer.

The usual scenario is that my Verilog RTL modules are verified remotely. Mostly with SV+UVM. That by itself is not very productive, but i can't change that. When test fails, what i get back are waves file and log file. That is what i work with to nail the bug.

Following are several suggestions to ease the debug process and allow me to see the daylight even during short winter days (used to be our kids, but they are all grown up now).

#1 : print clock number instead of simulation time in log messages.

add "int cycles;" to test-bench verilog code.
Increment it on each posedge of the system clock. Specify this count as part of every UVM message info or error.

Timescales today tend to be in ridiculously fine time units. So the time points reported in messages are too long and have too many zeroes. With shorter integer clock count it should be much easier to navigate as the count appears in the waves too.

#2: add counter to UVM error messages and have this count in waves too.

When UVM outputs an error, please add the relevant clock count, but also add the running number of this error message. Add integer to the test-bench and deposit the error number into this bus.
UVM error messages are cryptic enough, without the need to hunt them down across the simulation.

Optionals, very nice to have:

#3: split ERRORs to WRONG, ERROR, CORRECT

In my verification environments, there is difference between comparison error and just an error.
Comparison error (usually printed as "**WRONG"**) - means reference model vs RTL produced different answers. When comparison is correct, the log message of "CORRECT" is produced. Not always printed on screen, but usually logged into the log file. Having WRONGS/CORRECTS helps to pinpoint the context of each wrong. On the other hand, ERRORs indicate something happened that should not happen (like fifo underrun, or environment fault).

#4 Panic wires (something You could take advantage of).

Panic wires are my way to imitate assertions. In RTL modules i tend to define wires named panic_xxxx. They get assigned expressions that should never be true.
e.g: **wire panic_paddr_bad = (^paddr) === 1'bx**;
wire panic_select = src0 && src1 ; // when they should be 1 hot.
wire panic_baud = uart_baudrate>1000;
It includes possible bugs, but also wrong setup configurations, like illegal setup values.

Assertions are not controlled by me. On the other hand panic wires are always silently there, can be viewed in waves and disappear in synthesis.

First thing when i get waves of failed test, is to check all panics. Sometimes violation of setup values is all there is for me to check, before shipping it back to verification. Smart verificators check those themselves after a while.

Opposite are “should_happen” wires. Which is my way (although seldom actually used) of making sure the verification covered my worries.

#5 talk VCD to me.

Waves of all simulators can be translated to VCD. VCD format of waves is text format, it can be gzipped and shared with me. Moreover when translated from Synopsys/Cadence/Mentor formats to VCD it can be trimmed to carry only interesting modules and times.
VCD file has couple of big advantages for me.

1: I can view it on my laptop without taking licenses.
2: use tool to scan VCD file to extract features without staring at green/red waves.
(try to analyze AXI bus transactions without it).