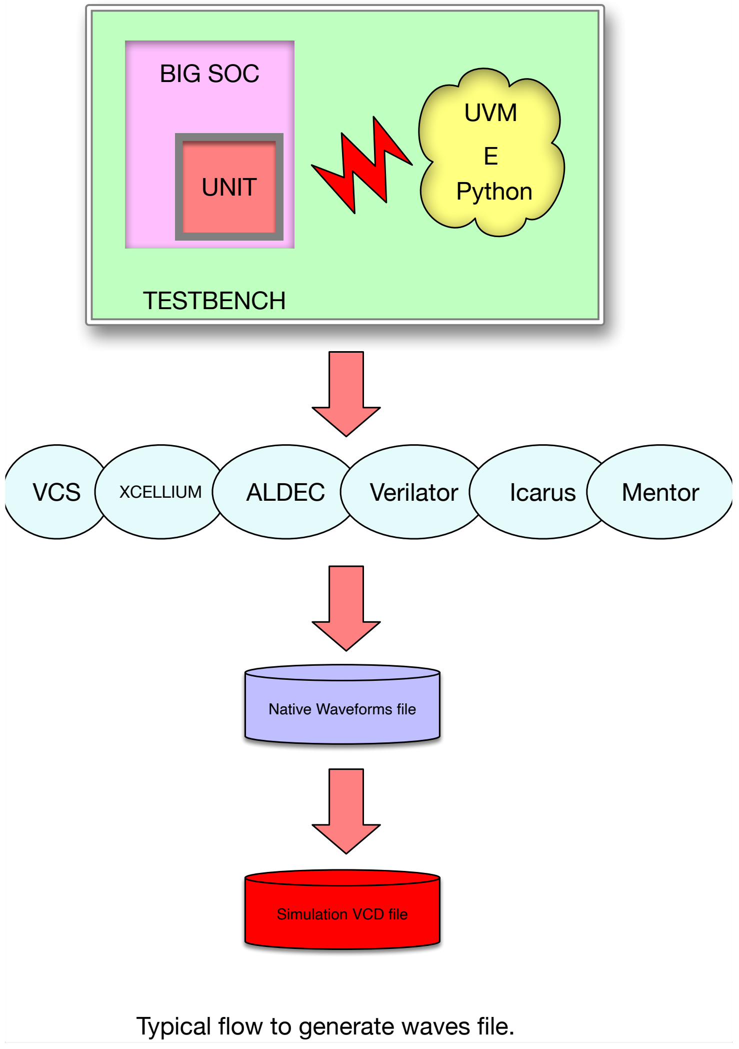
Extracting testbench from simulation waves

Just something i cooked in an hour. Another day, Elihai asked me whether i have a solution to some problem. I tend to answer "Yes" even before hearing the rest of the sentence, For the simple reason - chances are i already encountered it in the past. This time the task is simple to explain - **Given full chip simulation waves, we want to run unit level test with exactly the same unit inputs.**

Unit test environment probably has much more monitors and checkers. So when there is a problem on a chip level, better debug it "privately".

Interestingly enough, couple of weeks before, somebody else asked about similar problem.This gave me an idea to implement it. Using my tools. Not even asking "Gemini" about it.
I am sure it was done before, But what would be the fun in that?



Typical flow to generate waves file.

Using vcd_python3 app

This app is "C" code compiled with Python shared library. "C" code scans VCD file. Based on defined triggers, it passes control to python functions. These functions can peek into waves and get current values of any signal. The API is simple and looks like:
A = very.peek('tb.dut.cpu.opcode') (returns string of 0,1,x,z)
or **A = logs.peek('tb.dut.cpu.addr')**   (returns positive integer (or -1 for unknowns)

The triggers usually are posedge or negedge of clocks. Asking values on negedge triggered function can emit messages about transactions, or count number of events and so on.
invocation: **vcd_python3 dump.vcd module.py**
module.py is the code that includes all python functions.
"import veri" - imports the "C" code that allows to enquire the VCD values.
"**import logs**" - imports more helper functions.
And as python goes - You can import anything that helps.

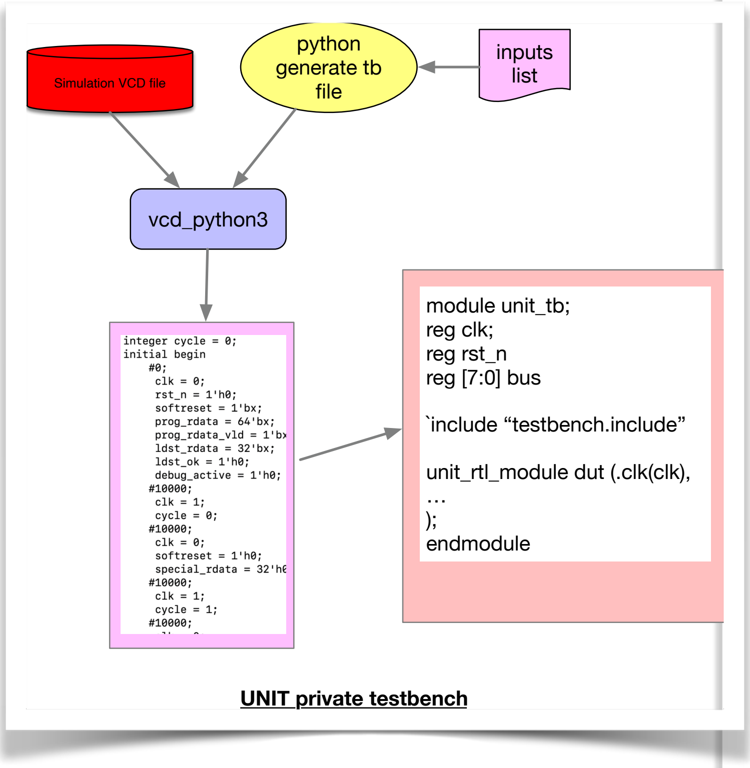
Testbench production

Let set trigger on posedge of some clock and negedge of the same clock. On posedge we just force the clock to "1". on Negedge we query all inputs of the unit and write their value to the resulting file. But first, on both functions, we write out the delay since last time (#1000).

Here is an example of how the resulting file looks like. It can optionally include the whole module - registers of inputs, wires of outputs, this initial sequence and instance of the DUT.

The python module that generates the testbench needs:

1. the VCD file. (VCD = value change dump. is readable waveforms format).
2. list of inputs (just names) - You can “grep input” from the file. Or give the module filename. the software will extract the inputs by itself.
3. compiled vcd_python3 (not that trivial).



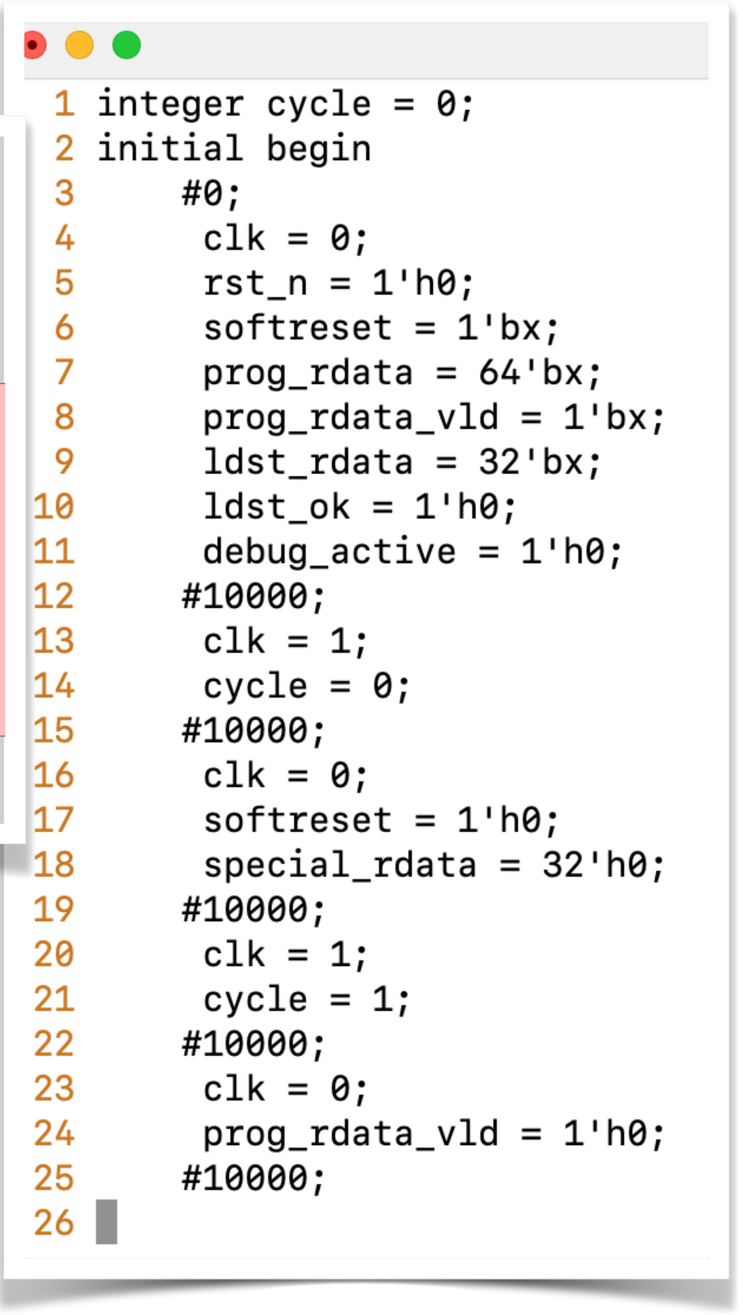
```
integer cycle = 0;
initial begin
    #0;
     clk = 0;
     rst_n = 1'h0;
     softreset = 1'bx;
     prog_rdata = 64'bx;
     prog_rdata_vld = 1'bx
     ldst_rdata = 32'bx;
     ldst_ok = 1'h0;
     debug_active = 1'h0;
    #10000;
     clk = 1;
     cycle = 0;
    #10000;
     clk = 0;
     softreset = 1'h0;
     special_rdata = 32'h0
    #10000;
     clk = 1;
     cycle = 1;
    #10000;
```

```
module unit_tb;
reg clk;
reg rst_n
reg [7:0] bus

`include “testbench.include”

unit_rtl_module dut (.clk(clk),
…
);
endmodule
```

UNIT private testbench



```
integer cycle = 0;
initial begin
    #0;
     clk = 0;
     rst_n = 1'h0;
     softreset = 1'bx;
     prog_rdata = 64'bx;
     prog_rdata_vld = 1'bx;
     ldst_rdata = 32'bx;
     ldst_ok = 1'h0;
     debug_active = 1'h0;
    #10000;
     clk = 1;
     cycle = 0;
    #10000;
     clk = 0;
     softreset = 1'h0;
     special_rdata = 32'h0;
    #10000;
     clk = 1;
     cycle = 1;
    #10000;
     clk = 0;
     prog_rdata_vld = 1'h0;
    #10000;

```

The conclusion

It works, and it took me an hour to write and try. This doc took me 3 times as much.

Soon i will add to **vlsimentor** (GitHub repository) a complete demo.
vlsistuff is another repository, where this software and more can be found.
Both repositories are open under MIT license.

*** git clone https://github.com/greenblat/vlsistuff.git
*** git clone https://github.com/greenblat/vlsimentor.git

Compiling the C code is not trivial, Any help to make it trivial is welcome.

thanks, Ilia