Design Advantage™
Comit Systems, Inc. Newsletter
We build them for YOU.℠
Volume 2 Number 4 September 1999

Automated Incremental Save in Verilog-XL* Narrows Down Error Tracking

By Chirinjeev Singh

Saving frequent value-change dumps during simulation is invaluable in tracking down errors, if it can be done without slowing down the simulation too much, running out of disk space, or excessive monitoring.

Combining '$incsave' (which can be put in an 'always' loop to keep saving the incremental database during simulation after a specified interval of time), with the Verilog '$system' function call, provides an efficient, "automated" way of
maintaining an ongoing dump by removing previously saved incremental files if no error is encountered for a long time.

Since '$incsave' normally takes the name of the file as a fixed string, any algorithm we propose will require string
manipulation for changing the name of the incremental files, each time with a unique name. A string is used as a variable in the '$incsave' command to generate a new file each time.

The algorithm looks straightforward, but since string
manipulation is not so obvious in Verilog, this method of automatically assigning unique file names can be easily overlooked. The Verilog code is shown below:

```
/************************************************
Author : Chirinjeev Singh
Date : August 10 1999
Functional Description:
. Every N ns
  . If Error
      . Print error message
      . Print Iteration_Num
      . $finish
    else
      if (first iteration)
        do a complete save
      else
        do an incremental save
        if Iteration_Num > M
          delete saved file for Iteration_Num-M
        end if
      end if

***********************/
module saveRestore;

/*——FOLLOWING LINES CAN BE EDITED BY USER——*/

parameter SAVE_TIME = 500000; // Time after which
you want to save the database
parameter SAVED_ITRNS = 12; // Number of previous
saved iterations

// Specify the dir , else save files will be dumped
in the cwd.
//Example : replace inc_ by /users/csingh/dat/inc_
reg [31:0] baseStr; initial baseStr = "inc_";

/*———— End of the user edit area ————*/
// errFlag can be made an input to this module.
//Currently  assumed as a global variable accessed
by all the modules.

reg errFlag; initial errFlag = 0;
reg [11:0] itrnNo; initial itrnNo = 'h0;

always begin
 #(SAVE_TIME);
 if (errFlag) begin
   $display("Error at Time: %0d\n",$time);
   $display("Iteration Number is %s",getAscii
(itrnNo));
   errFlag = 0;
   $finish;
 end // if (errFlag)
 // Save the database for the first time
 else if (itrnNo == 'h0) begin
   $save("save.dat");
   itrnNo = itrnNo + 1;
 end
 else begin
   $display("%s",{baseStr,getAscii(itrnNo),".
dat"});
   $incsave({baseStr,getAscii(itrnNo),".dat"});
   itrnNo = itrnNo + 1;
   if(itrnNo > SAVED_ITRNS) begin
     $system({"\rm -rf inc_",getAscii(itrnNo -
SAVED_ITRNS),".dat"});
   end
 end
end // always begin

function [23:0] getAscii;
input [11:0] itrnNo;
begin
 getAscii[23:0] = {convNibToAscii(itrnNo[11:8]),
convNibToAscii(itrnNo[7:4]),
                   convNibToAscii(itrnNo[3:0])};
end
endfunction

function [7:0] convNibToAscii;
input [3:0] nib;
begin
  case(nib)

4'h0,4'h1,4'h2,4'h3,4'h4,4'h5,4'h6,4'h7,4'h8,4'h9:
    convNibToAscii = nib + 'h30;
  4'ha,4'hb,4'hc,4'hd,4'he,4'hf: convNibToAscii =
nib + 'h57;
  default: $display("ILLEGAL VALUE FOR NIBBLE");
  endcase
end
endfunction
endmodule
```