USER GUIDE

RACE GUIDELINES CHECK

For the checking of race guidelines and rules we have incorporated a new attribute with read Verilog.

Syntax - read_verilog -rules -help <filename> <filename.v>

-rules

- This attribute is used for the check of race guidelines.
- The first input after the -rules attribute, user need to enter the file name with .txt or .rpt with any location where the results of race guidelines will be stored. If the user do not enter the filename, by default race_guidelines.rpt file with the required result will be created in the yosys folder itself.
- For having the information about the warnings details, just simply use -help

  Syntax - read_verilog -rules -help <filename.v>

- After entering the filename with .rpt or .txt, enter the file in which user want to check the race guidelines.

If user wants to check the race guidelines we have assumed that user doesn't wants to synthesize that code, or will synthesize that code after checking for race guidelines. **So once the race guidelines checked, user need to restart yosys and continue with normal yosys functioning with read_verilog without -rules**.

For example, if user wants to check race guidelines for the file fsm.v, the command looks like

read_verilog -rules report.rpt fsm.v

The resultant race guidelines will be stored in report.rpt in the yosys folder itself.

Apart from this a summary report will be generated in the yosys folder itself with the name summary.rpt. This file will contain the module name and associated number of warnings generated by violating the race guidelines.

Following are the race guidelines cover in this code section. User can see this details about guidelines by using **-help attribute along with -rules** as shown in syntax above.

Guideline 1: NB_SEQ

When modelling sequential IC, use non-blocking assignments.

Consider following example in which shift register is designed using blocking statement.

```
always @(posedge clk) begin
        q1 = d;
        q2 = q1;
        q3 = q2;
end
```

The resultant synthesized model consist of only one flip flop due to the stratified event queue concept.

Thus it is advisable and also safest to use non-blocking assignments in such sequential blocks.

Guideline 2: NB_LATCH

When modelling latches, use non-blocking assignments.

A similar analysis as of guideline 1 would show that it is also safest to use non-blocking assignments to model latches.

Guideline 3: B_COMB

When modelling combinational IC with an always block, use blocking assignments.

Consider following example in which shift register is designed using blocking statement.

```
always @(a or b or c or d) begin
   tmp1 <= a & b;
   tmp2 <= c & d;
   y <= tmp1 | tmp2;
end
```

The resultant program will not work because of the functioning of non-blocking assignments. To generate proper output we need to add tmp1 and tmp2 to sensitivity list which can have different complexity.

Thus it is advisable and also safest to use blocking assignments in such combinational blocks.

Guideline 4: NB_SEQ_COMB

When modelling both sequential and combinational logic within the same always block, use non-blocking assignments.

This guideline is very useful when designing FSM. There are some forms of fsm in which designer needs to use both combinational as well as sequential block in single always block.

Thus it is advisable and also safest to use non-blocking assignments in such blocks. Designer can always use two different always block for the same.

Guideline 5: NO_B_NB

Do not mix blocking and non-blocking assignments in the same always block.

This guideline generally do not cause any errors or discrepancy in pre and post synthesis results provided both assignments are done to different variables. Synopsys tool will report error for the same.

Thus it is advisable and also safest to use not use non-blocking & blocking assignments in such blocks.

Guideline 6: NO_SAME_VAR

Do not make assignments to the same variable from more than one always block.

The synopsys tool will give following warning for this practice:

Warning: there is 1 multiple-driver net with unknown wired-logic type

Thus advisable to not assign values to same variable in different always block.

Guideline 7: NO_POS_NEG_EDGE

Do not provide posedge and negedge together in the sensitivity list of a single always block.

The reason behind this is, in every library FF's and Latches with such sensitivity list is not always available.So avoid using both edges simultaneously.

CDC (Clock Domain Crossing) Check

For checking CDC in sequential blocks, use the 'synth' yosys command. No separate attribute is needed for CDC check.

The synth command on execution will check for the CDC between FF's and will also verify if synchronisers have been used for the resolution.

The synth command in yosys is used for technological independent mapping.

Syntax - synth -top top_module_name -flatten