Verilog Coding Guidelines

Alax Yang

1. Ensure that the coding style does not contains combinational feedback loops.

2. Never assign "x" value to the signals.

3. Maximize the usage of parameters instead of text-macros.

4. Use parameters for state-encoders in FSM

5. Ensure that the design coding does not generate tri-state logic.

6. Use the concept of base+ offset for coding address burses.

7. Tie the un-used bits to a known value.

8. Connect based on port-names while instantiating.

9. Minimize top-level glue logic coding style.

10. Ensure only one module is defined in a file.

11. Prefer using case statements for truth-table like structure, instead of using long if-else statements

12. Ensure that default statement is used in the case statements.

13. While designing FSM, prefer mealy machine with two always block.

14. Ensure that the unused module inputs are driven and not floating.

15. Ensure that the design does not contain initial blocks and delay elements.

16. Ensure that the whole design is resetable to a known state. No internal logic generated asynchronous resets.

17. Ensure that the resets are not used as data or clock signals in the design.

18. Ensure that the reset is priority over any other signal in the case of an if-else construct.

19. Ensure that the signals which cross the clock-domains are synchronized.

20. Ensure that the design does not contain while statements.

21. Avoid using verilog UDP in the design.

22. Coding style which will not have simulation and synthesis mis-match results.

23. Coding Style which are DFT(Design for Test) friendly.

24. Use blocking statements for coding combinational logic

25. Use non-blocking statements for coding sequential logic

26. Never mix blocking and non-blocking in a same procedural block

27. Never assign a value to the same variable multiple times in different always blocks. Even though it is a non-blocking statements the design is prone to race-condition

28. Ensure that a complete list of signals present in the sensitivity list in an always block, thereby the simulation results for a pre-synthesis versus the post-synthesis match.

29. Ensure only one clock per sensitivity list.

30. Never use both “posedge clock” and “negedge clock” to trigger in one system.

31. Be careful using Synopsys Full-case and parallel case directives.

32. Prefer using “case” instead of “casex” and “casez”.

33. During the process of logic synthesis, avoid the usage of `include in the Verilog files that describe the design. By doing this, each of the modules can be synthesized by itself. In case of any instantiated modules, you need to provide those modules to the synthesis tool separately.