Digital VLSI Circuits and HDL

Mazad S. Zaveri, PhD

Generic structure of a module

module modulename (list of input and output variables/ports);
    input list of input variables;
    output **reg/wire** list of output variables;

    instances of sub-module

    instances of local **wire** or **reg** variables

    your code will go here

endmodule

Variables in a module

- input (input port of the module)
  - Do not have to be additionally declared as reg or wire
    - Will be used on the RHS of assign statements
    - Will be used on the RHS of expressions, inside an always block
    - Will be passed (to an input port) of sub-modules
  - Any module will never provide/drive a value to it (never on the LHS, or never passed to output of a sub-module instance)
- output (output port of the module)
  - Reg
    - Value provided using always block (LHS of an expression)
  - Wire
    - Value provided using assign statement (LHS)
    - Value provided using gate-level modeling (has to be passed to output of the gate instance)
    - Value provided using a sub-module instance (has to be passed to output of the sub-module instance)
- Variables that are not input/output ports (their use is only temporary/local or to act as interconnection)
  - Used for interconnecting sub-modules
    - Wire
  - Used for virtual interconnection between assign statements
    - Wire
  - Used inside always or initial block (given value as LHS of an expression)
    - Reg

Modeling digital circuits using Verilog

- Gate-level modeling
  - Using instances of gate primitives
    - AND, OR, NAND, NOR, Ex-OR, Ex-NOR
  - This method is cumbersome for large designs
    - Not suitable for modeling higher-level algorithm/behavior
- Data-flow modeling (using assign statements)
  - Based on continuous assignment
    - Suitable for ONLY combinational logic
  - Not suitable for modeling sequencing elements (such as latch or FF)
- Behavioral modeling (using always and initial blocks)
  - initial blocks are used only in a test-bench
  - always blocks generally have a *sensitivity list*
    - Block is executed/evaluated when one/more of the variables in the *sensitivity list* changes its values
  - Suitable for modeling higher-level algorithm/behavior (combinational logic and sequencing elements)
    - Can use popular programming-constructs, such as if-else, case