Boolean equations describing combinational logic are specified in Verilog with a continuous assignment statement consisting of the keyword assign followed by a Boolean expression.

To distinguish arithmetic operators from logical operators, Verilog uses the symbols (&), (/), and ( & ) for AND, OR, and NOT (complement), respectively.

\[
\text{assign } D = (A \&\& B) \mid\mid (!C);
\]
The values of $E$ and $F$ during simulation are determined dynamically by the values of $A$, $B$, $C$, and $D$. The simulator detects when the test bench changes a value of one or more of the inputs. When this happens, the simulator updates the values of $E$ and $F$.

The continuous assignment mechanism is so named because the relationship between the assigned value and the variables is permanent.

The mechanism acts just like combinational logic, has a gate-level equivalent circuit, and is referred to as implicit combinational logic.
User-Defined Primitives

The user can create additional primitives by defining them in tabular form.

These types of circuits are referred to as user-defined primitives (UDPs).

One way of specifying a digital circuit in tabular form is by means of a truth table.

UDP descriptions do not use the keyword pair module . . . endmodule.

Instead, they are declared with the keyword pair primitive . . . endprimitive.
• It is declared with the keyword primitive, followed by a name and port list.

• There can be only one output, and it must be listed first in the port list and declared with keyword output.

• There can be any number of inputs. The order in which they are listed in the input declaration must conform to the order in which they are given values in the table that follows.

• The truth table is enclosed within the keywords table and endtable.

• The values of the inputs are listed in order, ending with a colon (:). The output is always the last entry in a row and is followed by a semicolon (;).

• The declaration of a UDP ends with the keyword endprimitive.