Verilog Real Constants

- **Real Constant**
  - **real constant numbers is represented as described by IEEE Std 754-1985, an IEEE standard for double-precision floating-point numbers.**
    - **Example**

      1.2
      1.2E12
      0.1e-2
      .12 // is an invalid form of real number

- **Real numbers shall be converted to integers by rounding the real number to the nearest integer, rather than by truncating it.**
  - **The real numbers 35.7 and 35.5 both become 36 when converted to an integer and 35.2 becomes 35.**
  - **Converting –1.5 to integer yields –2, converting 1.5 to integer yields 2.**

Verilog Constant

- **Unsized constant numbers**

  659 // is a decimal number
  'h 837FF // is a hexadecimal number
  'o7460 // is an octal number
  4af // is illegal(hexadecimal format requires 'h)
- **Sized constant numbers**

  4'b1001 // is a 4-bit binary number
  5 'D 3 // is a 5-bit decimal number
  3'b01x // is a 3-bit number with the least significant bit unknown
  12'hx // is a 12-bit unknown number
  16'hz // is a 16-bit high-impedance number
- **Using sign with constant numbers**

  8 'd -6 // this is illegal syntax
  -8 'd 6 // this defines the two's complement of 6, held in 8 bits—equivalent to -(8'd 6)
  4 'shf // this denotes the 4-bit number '1111', to be interpreted as a 2's complement number,or '-1'.
  // This is equivalent to -4'h 1
  -4 'sd15 // this is equivalent to -(-4'd 1), or '0001'
  16'sd? // the same as 16'sbz
- **Using underscore character in numbers**
  - **27_195_000 is the same as 27195000**

- **A string is a sequence of characters enclosed by double quotes ("") and contained on a single line.**
  - **In an expression is treated as unsigned integer constants represented by a sequence of 8-bit ASCII values**
  - **Example**

```
reg [8*12:1] stringvar;
initial begin
        stringvar = "Hello world!";
end
```

  - special characters in string
    - \n Newline; \t Tab ; \\ \ character; \”

Verilog Constant

- Automatic left padding

```
reg [11:0] a, b, c, d;
initial begin
   a = 'h x; // yields xxx
   b = 'h 3x; // yields 03x
   c = 'h z3; // yields zz3
   d = 'h 0z3; // yields 0z3
end
reg [84:0] e, f, g;
e = 'h5; // yields {82{1'b0},3'b101}
f = 'hx; // yields {85{1'hx}}
g = 'hz; // yields {85{1'hz}}
```

- Vectors
  - A net or reg declaration without a range specification shall be considered 1 bit wide and is known as a scalar. Multibit net and reg data types shall be declared by specifying a range, which is known as a vector.
  - Declaration examples:
    - wand w; // a scalar net of type "wand"
    - tri [15:0] busa; // three-state 16-bit bus
    - reg a; // a scalar reg
    - reg [3:0] v; // a 4-bit vector msb –v[3]; lsb v[0]
    - reg signed [3:0] signed_reg; // a 4-bit vector in range -8 to 7
    - reg [-1:4] b; // a 6-bit vector reg
    - wire w1, w2; // declares two wires
    - Reg [4:0] x, y, z; // declares three 5-bit regs

Verilog Identifiers

- **The first character of a simple identifier**
  - shall not be a digit or $
  - it can be a letter or an underscore
- **Identifiers shall be case sensitive**
  - **Example:**
    shiftreg_a
    busa_index
    error_condition
    merge_ab
    _bus3
    n$657
- Escaped identifiers start with backslash (\) and provide means of including any printable ASCII in an identifier
  - **Example:**
    \busa+index
    \-clock
    error-condition
    \net1/\net2
    \{a,b}
    \a*(b+c)

Strengths

- Two types of strengths can be specified in net declaration as follows:
  - *Charge strength* shall only be used when declaring a net of type trireg.
  - *Drive strength* shall only be used when placing a continuous assignment on a net in the same tatement that declares the net.

Verilog Data Type

- Nets:
  - The net data types can represent physical connections between structural entities, such as gates. A net shall not store a value (except for the trireg net). Instead, its value shall be determined by the values of its drivers.
- Varilables
  - A variable is an abstraction of a data storage element. A variable shall store a value from one assignment to the next. An assignment statement in a procedure acts as a trigger that changes the value in the data storage element. The initialization value for reg, time, and integer data types shall be the unknown value, x.

Charge Strength

- **It can be one of three strenght:**
  small< medium < large
- The default charge strength of a **trireg** net shall be **medium**.
  – Example

```
trireg a;        // trireg net of charge strength medium
trireg (large) #(0,0,50) cap1;
trireg (small)signed [3:0] cap2;
```

Drive strength

- supply1< strong1< pull1< weak1
- Supply0< strong0< pull0< weak0
  – Example:
    - nor (weak1,strong0) n1(out1,in1,in2);