REPLACE in section 2.6

A string literal can be assigned to an integral type ~~character, or a packed array~~, as in Verilog-2001.

A string literal can be assigned to an unpacked array of bytes ~~characters, and a zero termination is added like in C~~.

REPLACE in section 3.7

SystemVerilog includes a string data type, which is a variable size, dynamically allocated array of **`byte`** ~~characters~~. SystemVerilog also includes a number of special methods to work with strings.

Verilog supports string literals, but only at the lexical level. In Verilog, string literals behave like ~~packed arrays~~ integral values of a width that is a multiple of 8 bits. A string literal assigned to an integral variable ~~packed array~~ ~~is~~ of a different size is either truncated to the size of the ~~array~~variable or padded with zeroes to the left as necessary.

....

When using the string data type instead of an integral variable ~~packed array~~,

....

A string literal can be assigned to a string~~, a character, or a packed array~~ or an integral type.

...

A string, string literal, or ~~packed array~~ integral value can be assigned to a string variable. The string variable shall grow to accommodate the ~~packed array~~ integral value. If the size (in bits) of the ~~packed array~~ integral value is not a multiple of 8, then the ~~packed array~~ integral value is ~~zero filled on the left~~ padded with zeroes to the left as necessary.

Add to the examples in section 3.7

```
a = {"Hi",b}; // OK
r = {"H",""}; // yields "H\0" "" is converted to 8'b0
b = {"H",""}; // yields "H" "" is the empty string
a[0] = "h";   // OK same as a[0] = "hi" ~~)~~
```

REPLACE in table 3-2

The comparison behaves like the ANSI C `strcmp` function (or the ~~compare~~ `compare` string method) except that embedded null bytes are included.

Comparison. Relational operators return 1 if the corresponding condition is true using the lexicographical ordering of the two strings `Str1` and `Str2`. The comparison behaves like the ANSI C `strcmp` function (or the ~~compare~~ `compare` string method) with regard to the lexical ordering. Both operands can be of type **`string`**, or one of them can be a string literal.

REPLACE in section 3.7.6

— `str.compare(s)` compares `str` and `s`, as in the ANSI C `strcmp` function (with regard to lexical ordering and return value)~~, with a compatible return value~~ and embedded null bytes are included.

REPLACE in section 3.7.7

— `str.icompare(s)` compares `str` and `s`, like the ANSI C `strcmp` function (with regard to lexical ordering and return value), ~~with a compatible return value,~~ but the comparison is case insensitive and embedded null bytes are included.

RELACE in section 23.4

The `$bits` system function returns ~~logic X'x~~0 when called with a dynamically sized type that is currently empty.