This proposal updates section 22, System tasks and functions, to handle SV3.1 data types and include data types as arguments.

Replace all the text in section 22.2 $bits with

```
size_function ::= // not in Annex A
$bits ( expression )
$bits ( type_identifier )
variable_identifier::bits() // method form
type_identifier::bits // method form
```

Syntax 22-1—Size function syntax (not in Annex A)

The `bits` system function returns the number of bits required to hold an expression as a bit stream. See section 3.16 bit-stream cast for a definition of legal types. A 4 state value counts as one bit. Given the declaration:

```
logic [31:0] foo;
```

Then $bits(foo) shall return 32, even if the implementation uses more than 32-bits of storage to represent the 4-state values.

Given the declaration:

```
typedef struct {
        logic valid;
        bit [8:1] data;
} MyType;
```

The expression `MyType::bits` shall return 9, the number of data bits needed by a variable of type `MyType`.

The `bits` function can be used as an elaboration-time constant when used on fixed sized types; hence, it can be used in the declaration of other types or variables.

```
typedef bit [MyType::bits : 1] MyBits;// same as typedef bit [9:1] MyBits;
MyBits b;
```

Variable b can be used to hold the bit pattern of a variable of type MyType without loss of information.

The `bits` system function returns logic X when called on a dynamically sized type that is currently empty .It is an error to use the `bits` system function directly on a dynamically sized type identifier.

Replace section 22.4 array query functions with

```
array_query_functions ::= // not in Annex A
        array_dimension_function ( array_identifier [ , dimension_expression ] )
        | array_dimension_function ( type_identifier [ , dimension_expression ] )
        | $dimensions ( array_identifier )
        | $dimensions ( type_identifier )

array_dimension_function ::=
        $left
        | $right
        | $low
        | $high
        | $increment
|       $size
dimension_expression ::= expression
array_query_methods ::= // not in Annex A
        array_dimension_method ( [ dimension_expression ] )
        | dimensions

array_dimension_methods ::=
        left
        | right
        | low
        | high
        | first
        | last
        | increment
|       size
```

Syntax 22-2—Array querying function syntax (not in Annex A)

SystemVerilog provides system functions to return information about a particular dimention of an array variable or type. The default for the optional dimension expression is 1. The array dimension can specify any fixed sized index (packed or unpacked), or any dynamically sized index (dynamic, associative, or queue).

— `$left` shall return the left bound (msb) of the dimension
— `$right` shall return the right bound (lsb) of the dimension
— `$low` shall return the minimum of `$left` and `$right` of the dimension
— `$high` shall return the maximum of `$left` and `$right` of the dimension
— `$increment` shall return 1 if `$left` is greater than or equal to `$right`, and -1 if `$left` is less than `$right`
— `$size` shall return the number of elements in the dimension, which is equivalent to `$high - $low + 1`
— `$dimensions` shall return the number of dimensions in the array, or 0 for a singular object

The dimensions of an array shall be numbered as follows: The slowest varying dimension (packed or unpacked) is dimension 1. Successively faster varying dimensions have sequentially higher dimension numbers.
For instance:

```
// Dimension numbers
//    3    4       1    2
reg [3:0][2:1] n [1:5][2:8];
```

For an integer or bit type, only dimension 1 is defined. For an integer N declared without a range specifier, its bounds are assumed to be `[$bits(N)-1:0]`. If an out-of-range dimension is specified, these functions shall return a logic X.

When used on a dynamic array or queue dimension, these functions return information about the current state of the array. If the dimention is currently empty, these functions shall return a logic X. .It is an error to use these functions directly on a dynamically sized type identifier.

Use on associative array dimentions is restricted to index types with integral values. These functions will return:
— `$left` shall return 0
— `$right` shall return the highest possible index value
— `$low` shall return the lowest currently allocated index
— `$high` shall return the largest currently allocated index
— `$increment` shall return -1
— `$size` shall return the number of elements currently allocated

If the array identifier is a fixed sized array, these query functions may be used as a constant function and passed as a parameter before elaboration. These query functions may also be used on fixed sized type identifiers in which case it is always treated as a constant function.

Given the declaration below:

Typedef logic [16:1] Word;
Word Ram[0:9];

The following functions all return 16:

```
$size(Word)
$size(Ram,2)
Word::size
Ram::size(2)
```