Insert as Section 22.2.1

22.2.1 Type size property

```
size_property ::= // not in Annex A
        type_identifier :: bits
```

Syntax 22-2-1—Type size property syntax (not in Annex A)

The `bits` built-in property returns the number of bits required to represent a variable of the given type. The `bits` property is available on any fixed-size data type, except chandle and classes. Given the declaration below:

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

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

The `bits` property can be used as an elaboration-time constant, 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.

Modify Section 3.14 (1st example of page 23) as shown

Structures can be converted to bits preserving the bit pattern, which means they can be converted back to the same value without any loss of information. The following example demonstrates this conversion. In the example, the `bits` property gives the size of a structure in bits (the `$bits` system function and the `bits` type property are discussed in Section 22.2):

```
typedef struct {
            bit isfloat;
            union { int i; shortreal f; } n; // anonymous type
} tagged; // named structure

typedef bit [tagged::bits - 1 : 0] tagbits; // tagged defined above

tagged a [7:0]; // unpacked array of structures

tagbits t = tagbits'(a[3]); // convert structure to array of bits
a[4] = tagged'(t); // convert array of bits back to structure
```