SystemVerilog Typedef Class

Sometimes the compiler errors out because of a class variable being used before the declaration of the class itself. For example, if two classes need a handle to each other, the classic puzzle of whether chicken or egg came first pops up. This is because the compiler processes the first class where it finds a reference to the second class being that which hasn't been declared yet.

```
class ABC;
    DEF     def;    // Error: DEF has not been d
endclass

class DEF;
    ABC     abc;
endclass
```

Compilation Error

```
file: typdef-class.sv
   DEF def;
     |
ncvlog: *E,NOIPRT (typedef-class.sv,2|5): Unrecogr
```

In such cases you have to provide a *forward declaration* for the second class using `typedef` keyword. When the compiler see a `typedef` class, it will know that a definition for the class will be found later in the same file.

Usage

```
1  typedef class DEF;  // Inform compiler that DEF
2                      // used before actual class
3
4  class ABC;
5      DEF     def;      // Okay: Compiler knows th
6                        // declaration will come lat
7  endclass
8
9  class DEF;
10     ABC     abc;
11 endclass
```

By using a `typedef` **DEF** is declared to be of type `class` which is later proved to be the same. It is not necessary to specify that DEF is of type `class` in the **typedef** statement.

```
1  typedef DEF;        // Legal
2
3  class ABC;
4      DEF def;
5  endclass
```

Using typedef with parameterized classes

A `typedef` can also be used on classes with a parameterized port list as shown below.

```
typedef XYZ;

module top;
    XYZ #(8'h3f, real)              xyz0;   // posi
    XYZ #(.ADDR(8'h60), .T(real))   xyz1;   // name
endmodule

class XYZ #(parameter ADDR = 8'h00, type T = int);
endclass
```

INTERVIEW QUESTIONS

- SystemVerilog Interview Set 1 (/systemverilog/systemverilog-interview-questions-set-1)
- SystemVerilog Interview Set 2 (/systemverilog/systemverilog-interview-questions-set-2)
- SystemVerilog Interview Set 3 (/systemverilog/systemverilog-interview-questions-set-3)
- SystemVerilog Interview Set 4 (/systemverilog/systemverilog-interview-questions-set-4)
- SystemVerilog Interview Set 5 (/systemverilog/systemverilog-interview-questions-set-5)
- SystemVerilog Interview Set 6 (/systemverilog/systemverilog-interview-questions-set-6)

Verification Guide

SystemVerilog typedef class

typedef class

Table of Contents

*A typedef*is used to provide a forward declaration of the class.
In some cases, the class needs to be instantiated before the class declaration. In these kinds of situations, the *typedef* is used to provide a forward declaration of the class.

typedef syntax

```
typedef class class_name;
```

typedef examples

Without typedef

In the below example,
There are two classes c1 and c2.

c2 is instantiated inside c1 and c1 inside c2. Both classes need the handle of each other. As execution will happen in sequential order.
Dependency between both the classes leads to a compilation error.

```
//class-1
class c1;
  c2 c;    //using class c2 handle before declaring it.
endclass

//class-2
class c2;
  c1 c;
endclass

module typedef_class;
  initial begin
    c1 class1;
    c2 class2;
    $display("Inside typedef_class");
  end
endmodule
```

Simulator Output

```
Error-[SE] Syntax error
Following verilog source has syntax error :
token 'c2' should be a valid type. Please declare it virtual if it
is an Interface.
"testbench.sv", 6: token is ';'
c2 c;
```

Click to execute on EDA playground

With typedef

The compilation error of the above example can be avoided by using a typedef.

```
typedef class c2;
//class-1
class c1;
  c2 c;    //using class c2 handle before declaring it.
endclass

//class-2
class c2;
  c1 c;
endclass

module typedef_class;
  initial begin
    c1 class1;
    c2 class2;
    $display("Inside typedef_class");
  end
endmodule
```

Simulator Output

```
Inside typedef_class
```

Click to execute on EDA playground

‹ Previous

Next ›