DAY-55
#100DAYSRTL

“System Verilog: Structure & Union”

Introduction:-

Structures and unions are special data types which are used to group variables having different data types. We have seen that arrays also provide the grouping of various variables or elements. But in arrays, the data type of each element is the same, whereas structure and unions can have elements having different data types

Structure:-

A structure is a collection of elements having different data types. These elements can be accessed as a whole or individually. Like Enums these can be used to make user-defined data types. When a structure is declared, the memory is allocated for all the elements.

Syntax:-

```
<typedef> struct { <element1>; <element2>; ... } <name>;
```

Code practising :-

```
module structure1();
  typedef struct { int roll; string name; int mark; } student;
    student s1, s2;
    initial begin
      s1 = '{ 23, "LEO", 78 };
        $display("s1 = %p", s1);
        $display("s2 = %p", s2);
        // chaning one of the element
        s1.name = "Vikram";
        $display("s1 = %p", s1);
        // copyind one struct to another
        s2 = s1;
        $display("s2 = %p", s2);
    end
endmodule
```

Result:-

```
s1 = '{roll:23, name:"LEO", mark:78}
s2 = '{roll:0, name:"", mark:0}
s1 = '{roll:23, name:"Vikram", mark:78}
s2 = '{roll:23, name:"Vikram", mark:78}
```

Like the arrays, a structure can also be packed or unpacked. This concept of a packed and unpacked structure is exclusive to System Verilog. All structures are by default unpacked in nature.

Code practising :-

```
module structure2();
    typedef struct packed{
        int roll;
        bit [10:0] mark;
    } student;
    student s1, s2;
    initial begin
        s1 = { 10'd23, 10'd78 };
        $display("s1 = %p", s1);
        // In this initialization the data will be
        // stored correctly
        s2 = { 32'd23, 11'd98};
        $display("s2 = %p", s2);
    end
endmodule
```

Result:-

```
s1 = '{roll:11, mark:1102}
s2 = '{roll:23, mark:98}
```

Packed Vs Unpacked Stracture:-

Packed Structure	Unpacked Structure
members can be of packed data type only	Members can be of any data type
Initialization can be done for several members at once using the concatenation operator as well. This is because the packed structure is treated as a vector.	Initialization can be done for several members at once but only with the assignment operator, i.e., '{}. The concatenation operator cannot be used.

In a packed structure, we can only use a packed data type. For example, a string cannot be used inside a packed structure. Also, all array types except a packed array cannot be used inside the packed address

Union:-

Unions are like structures, but in a union, we can only access one element at a time. When unions are declared the memory is allocated only for the largest data type.

Syntax:-

```
<typedef> union { <element1>; <element2>; ... } <name>;
```

Code practising :-

```
module union1 ();
  union { int n;byte d;} test;
    initial begin
        test.n = 'hFFFF;
        $display("n = %0h", test.n );
        $display("d = %0h", test.d );
        test.d = 'h48;
        $display("n = %0h", test.n );
        $display("d = %0h", test.d );
        $displayh("test = %p", test);
    end
endmodule
```

Result:-

```
n = ffff
d = ff
n = ff48
d = 48
test = '{n:0000ff48, d:48}
```

Structure vs Union:-

Structure	Union
Struct keyword is used for declaration	Union keyword is used for declaration
The size of a structure is the sum of all the members of the structure	The size of a union is the size of the largest member of the union
All members are assigned different memory location	The same memory location is shared by all the members
Individual members can be accessed at a time	Only one member can be accessed at a time
Several members can be initialized at once	Only the first member can be initialized

Unions can also be declared as packed, but in the packed union, all the elements should be of the same size. I do not find packed union useful as all members have the same size

and also store the same value at a given time. Thus, a packed union can be replaced with a simple data type variable**.**

Tagged Unions:-

We have seen that the unions in System Verilog are not very useful as the value changed for one variable is reflected in other variables defined inside the union. System Verilog introduced a new concept of tagged unions. In a tagged union, we can assign a tag to a particular variable of any data type. Using this tag we can know that union is currently used to store which variable. This is useful when we want to have a packet that can store different values at different times

Syntax:-

```
<typedef>union tagged{<element1>;<element2>;.. }<name>;
```