SystemVerilog Missing Conveniences

Matt Maidment

Overview

- Unpacked Array Type “Mismatch”
- Packed Array of Packed Structs
- Unpacked Variables In Ternary Operators

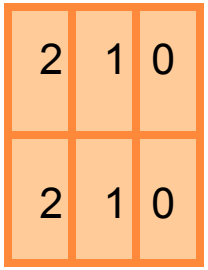
Unpacked Array Type "Mismatch"

- Unpacked Arrays of the same "shape" but different types are not assignment compatible
- Consider

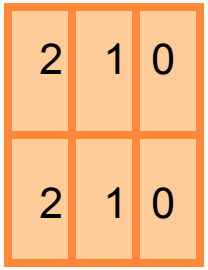
```
wire [2:0] wiremda [1:0];
logic [2:0] logicmda [1:0];
```

- They have the same shape

wiremda



logicmda



- But cannot be assigned to each other
  - `assign wiremda = logicmda;`
  - `logicmda = wiremda;`

Unpacked Array Type “Mismatch” (cont.)

- A nettype is not a type?
  - If so then it’s interesting properties are
    - Resolution Mechanism,
    - Direction of Assignment
  - If so then
    - Variable(s) to which it points
    - Shape of pointers

  Are interesting
- Why not same shape unpacked arrays to be assignment compatible?

Packed Array of Packed Structs

- If you want a hierarchy of packed structs then you cannot array any sub-struct
- Because of need for array of ps_t
- root_t cannot be packed
- Why not allow packed array of packed structs?

```
typedef struct packed {
 logic a1;
 logic a2;
} ps_t;
typedef struct packed {
  ps_t a3[1:0];
  logic [1:0] a4;
} root_t;
root_t pr;

typedef struct packed {
  ps_t [1:0] a3;
  logic [1:0] a4;
} root_t;
```

Unpacked Variables in Ternary Operators

- Given Type Compatible Unpacked Variables
- Cannot Do This:

```
ut = en ? us : ut;
```

- Why Not Allow It?
  - The types match
  - It's a natural thing to do

```
typedef struct {
 logic a1;
 logic a2;
} ups_t;
typedef struct {
  ups_t a3[1:0];
  logic [1:0] a4;
} root_t;
root_t ur[1:0],
       us[1:0],ut[1:0];
```