```
package types_package;
typedef struct packed
{
    logic a;
    logic b;
} t_ab;
endpackage
```

```
import types_package::*;
module foo (
    input  sys_clk,
    input  t_ab  test_ab,
    output foo_out
    );
reg foo_out_r;
assign foo_out = foo_out_r;
always @(posedge sys_clk)
begin
    foo_out_r <= test_ab.a & test_ab.b;
end
endmodule
```

Toplevel:

```
// wire [1:0] test; Either struct or wire will produce error!
t_ab test;
wire test_out;
foo foo1 (
    .sys_clk(sys_clk),
    .test_ab(test),
    .foo_out(test_out)
);
```

Error (12002): Port "test_ab" does not exist in macrofunction "foo1"

Workaround is to modify foo module parameter to pass as a **wire array** and then reassign back to struct in module.

```
import types_package::*;
module foo (
    input  sys_clk,
```

```
    // input  t_ab  test_ab, ERROR
    input [$bits(t_ab)-1:0] test_ab, // <<=THIS WORKS???
    output foo_out
    );
reg foo_out_r;
assign foo_out = foo_out_r;
t_ab  test_ab_s;
assign test_ab_s = test_ab;
always @(posedge sys_clk)
begin
    // foo_out_r <= test_ab.a & test_ab.b;
    foo_out_r <= test_ab_s.a & test_ab_s.b;
end
endmodule
```