

12.8 Elaboration

Elaboration is the process that occurs between parsing and simulation. It binds modules to module instances, builds the model hierarchy, computes parameter values, resolves hierarchical names, establishes net connectivity and prepares all of this for simulation. With the addition of generate statements, the order in which these tasks occur becomes significant.

12.8.1 Order of elaboration

Because of generate constructs, the model hierarchy can depend on parameter values. Because defparam statements can alter parameter values from almost anywhere in the hierarchy, the result of elaboration can be ambiguous when generate constructs are involved. The final model hierarchy can depend on the order in which defparams and generate constructs are evaluated.

The following algorithm defines an order that produces the correct hierarchy.

1. A list of starting points is initialized with the list of top-level modules.
2. The hierarchy below each starting point is expanded as much as possible without elaborating generate constructs. All parameters encountered during this expansion are given their final values by applying initial values, parameter overrides and defparam statements. This means that any defparam statement whose target can be resolved within the hierarchy elaborated so far must have its target resolved and its value applied. Defparam statements whose target cannot be resolved are deferred until the next iteration of this step.
3. Each generate construct encountered in step 2 is revisited, and the generate scheme is evaluated. The resulting generate block instantiations make up the new list of starting points. If the new list of starting points is not empty, go to step 2.

Note that no defparam inside the hierarchy below a generate construct is allowed to refer to a parameter outside the generate construct. Therefore it is possible for parameters to get their final values in step 2.

12.8.2 Early resolution of hierarchical names

In order to comply with this algorithm, hierarchical names in some defparam statements will need to be resolved prior to the full elaboration of the hierarchy. It is possible that when elaboration is complete, rules for name resolution would dictate that a hierarchical name in a defparam statement would have resolved differently had early resolution not been required. This could result in a situation where an identical hierarchical name in some other statement in the same scope would resolve differently from the one in the defparam statement. Below is an example of a design that has this problem:

```

module m;
  mid1 n();
endmodule

module mid1;
  parameter p = 2;

  defparam m.n.p = 1;
  initial $display(m.n.p);

  generate
    if (p == 1) begin : m
      mid2 n();
    end
  endgenerate
endmodule

module mid2();
  parameter p = 3;
endmodule

```

In this example, the `defparam` must be evaluated before the conditional `generate` is elaborated. At this point in elaboration, the name resolves to parameter `p` in module `mid1`, and this parameter is used in the `generate` scheme. The result of the `defparam` is to set that parameter to 1, so the `generate` condition is true. After the hierarchy below the `generate` construct is elaborated, the rules for hierarchical name resolution would dictate that the name should have resolved to parameter `p` in module `mid2`. In fact, the identical name in the `$display` statement will resolve to that other parameter.

It shall be an error if a hierarchical name in a `defparam` is resolved before the hierarchy is completely elaborated and that name would resolve differently once the model is completely elaborated.

This situation will occur very rarely. In order to cause the error, there has to be a named `generate` block that has the same name as one of the scopes in its full hierarchical name. Furthermore, there have to be two instances with the same name, one in the `generate` block and one in the other scope with the same name as the `generate` block. Then, inside these instances there have to be parameters with the same name. If this problem occurs, it can be easily fixed by changing the name of the `generate` block.