Figure 12-2 is a list of the hierarchical forms of the names of all the objects defined in the code.

```
wave                        wave.a.bmod
wave.stim1                  wave.a.bmod.in
wave.stim2                  wave.a.bmod.keep
wave.a                      wave.a.bmod.keep.hold
wave.a.stim1                wave.wave1
wave.a.stim2                wave.wave1.innerwave
wave.a.amod                 wave.wave1.innerwave.hold
wave.a.amod.in
wave.a.amod.keep
wave.a.amod.keep.hold
```

Figure 12-2—Hierarchical path names in a model

Hierarchical name referencing allows free data access to any object from any level in the hierarchy. If the unique hierarchical path name of an item is known, its value can be sampled or changed from anywhere within the description.

Example 2—The next example shows how a pair of named blocks can refer to items declared within each other.

```
begin
    fork :mod_1
        reg x;
        mod_2.x = 1;
    join
    fork :mod_2
        reg x;
        mod_1.x = 0;
    join
end
```

12.5 Upwards name referencing

The name of a module or module instance is sufficient to identify the module and its location in the hierarchy. A lower-level module can reference items in a module above it in the hierarchy.Variables can be referenced if the name of the higher-level module or its instance name is known. For tasks, functions, and named blocks, Verilog shall look in the enclosing module for the name until it is found or until the root of the hierarchy is reached. It shall only search in higher enclosing modules for the name, not instances. The syntax for an upward reference is given in Syntax 12-8.

```
upward_name_reference ::= (Not in the Annex A BNF)
      module_identifier.item_name
item_name ::=
       function_identifier
     | block_identifier
     | net_identifier
     | parameter_identifier
     | port_identifier
     | task_identifier
     | variable_identifier
```

Syntax 12-8—Syntax for upward name referencing

Upwards name references can also be done with names of the form

```
module_instance_name.item_name
```

A name of this form shall be resolved as follows:

a) Look in the current module for a module instance named `module_instance_name`. If found, this name reference shall be treated as a downward reference, and the item name shall be resolved in the corresponding module.
b) Look in the parent module for a module instance named `module_instance_name`. If found, the item name shall be resolved from that instance, which is the sibling of the module containing the reference.
c) Repeat step b), going up the hierarchy.

There shall be no spaces within the hierarchical name reference, except for escaped identifiers embedded in the hierarchical name reference, which are followed by separators composed of white space and a period-character.

Example:

In this example, there are four modules, `a, b, c,` and `d`. Each module contains an integer `i`. The highest-level modules in this segment of a model hierarchy are `a` and `d`. There are two copies of module `b` because module `a` and `d` instantiate `b`. There are four copies of `c.i` because each of the two copies of `b` instantiates `c` twice.

```
module a;
integer i;
b  a_b1();
endmodule

module b;
integer i;
c  b_c1(), b_c2();
initial               // downward path references two copies of i:
   #10 b_c1.i = 2;// a.a_b1.b_c1.i, d.d_b1.b_c1.i
endmodule

module c;
integer i;
initial begin         // local name references four copies of i:
   i = 1;             // a.a_b1.b_c1.i, a.a_b1.b_c2.i,
                      // d.d_b1.b_c1.i, d.d_b1.b_c2.i
   b.i = 1;           // upward path references two copies of i:
                      // a.a_b1.i, d.d_b1.i
   end
endmodule

module d;
integer i;
b d_b1();
initial begin         // full path name references each copy of i
   a.i = 1;                    d.i = 5;
   a.a_b1.i = 2;               d.d_b1.i = 6;
   a.a_b1.b_c1.i = 3;          d.d_b1.b_c1.i = 7;
   a.a_b1.b_c2.i = 4;          d.d_b1.b_c2.i = 8;
end
endmodule
```

