Analysis of hierarchical parameter naming in the SystemC CCI Standard

Arunkumar Vaidyanathan

15.08.2025

This discussion pertains to the following issue: `https://github.com/accellera-official/cci/issues/277`.

1 Background

1.1 Current status from standards and implementations

1.1.1 SystemC LRM (IEEE Std 1666-2023)

§5.17.1 states that:

> A hierarchical name shall be composed of a set of string names separated by the period character '.', starting with the string name of a top-level `sc_object` instance and including the string name of each module instance or process instance descending down through the object hierarchy until the current `sc_object` or `sc_event` is reached. The hierarchical name shall end with the string name of the `sc_object` or `sc_event` itself.
>
> Hierarchical names are case-sensitive. It shall be an error if a string name includes the period character (.) or any white-space characters. It is strongly recommended that an application limit the character set of a string name to the following: a) Lowercase letters a–z b) Uppercase letters A–Z c) Decimal digits 0–9 d) Underscore character _ An implementation may generate a warning if a string name contains characters outside this set but is not obliged to do so.

From the SystemC reference implementation (link):

```
const char SC_HIERARCHY_CHAR = '.';
```

1.1.2 SystemC CCI LRM v1.0

§5.9 states that:

> Both parameters and brokers are required to have unique names relative to each other; this extends to include all named SystemC objects for SystemC version 2.3.2 and later by using `sc_core::sc_register_hierarchical_name`.

From the SystemC CCI reference implementation, in the `cci_param_untyped` constructor (link):

```
if(name_type == CCI_ABSOLUTE_NAME) {
    m_name = name;
} else {
    sc_core::sc_object* current_obj = sc_core::sc_get_current_object();
    for (sc_core::sc_process_handle current_proc(current_obj);
          current_proc.valid();
          current_proc = sc_core::sc_process_handle(current_obj)) {
        current_obj = current_proc.get_parent_object();
    }
    if(current_obj) {
        m_name = std::string(current_obj->name()) +
            sc_core::SC_HIERARCHY_CHAR + name;
    } else {
        m_name = name;
    }
}
```

2 Supporting Examples

2.1 Scenario 1

Module `A` has a parameter `B.c` with relative naming. Module `B` has no parameters.

2.2 Scenario 2

Module `A` has a parameter `B.c` with relative naming. Module `B` has a parameter `c` with relative naming.

2.3 Scenario 3

Module `A` has a parameter `c` with relative naming. Module `B` has a parameter `A.c` with absolute naming.

2.4 Scenario 4

Module `A` has no parameters. Module `B` has a parameter `A.c` with absolute naming.