Abstract

UTAVP (University of Tehran TLM Assertion based Verification Package) is a package to aid system level designers debug and verify their SystemC RTL and TLM designs. This package provides the designers with an OVL-equivalent assertion package ported to SystemC, a package of Transaction level assertions, and a library of enhanced HDL OVL assertions to verify the synthesized RT level TLM channels.

1. Introduction

Increasing complexity of digital circuits has led designers to use higher levels of abstraction in their digital designs. This transition from gate level to higher abstraction levels resulted in the emergence of Transaction Level Modeling (TLM) methods. In this methodology, TLM designers can deal with the design functionality while neglecting low level details. Moreover, Due to its higher simulation and prototyping speed, TLM has better performance compared to RTL methods. In today’s digital designs, the IEEE standard SystemC language and its TLM 2.0 derivative are the most commonly used design languages for ESL (Electronic System Level). However, as SystemC and TLM 2.0 are really not defined as stand-alone design languages, and are basically a set of libraries in C++, debugging SystemC designs has always been a problem.

In the absence of proper debugging tools, which would help designers cope with common but hard to detect design and programming mistakes, the necessity of special verification methods for TLM becomes apparent. Introducing methods to improve design verification and code debugging will enable designers to take a better advantage of the high simulation speed in TLM design methodology.

Our TLM Assertion based Verification package provides a good solution to code debugging and TLM design verification. This package utilizes SystemC equivalent of RT level OVL assertions and introduces new Transaction Level assertions to be used in mixed RTL / TLM designs.

This package is composed of three components, each of which helps the designer in a different abstraction level and for different purposes. Below, these three components are briefly described:

- **OVL assertions for RT SystemC**: The development of the first component of our collection is justified for the situations that a SystemC RT level code is used as a design module in a TLM design. The provided assertions could be used to verify the functionality of such a module and its interface and interactions with adjacent modules, that may be at RTL or TLM level.
- **TLM channel assertions**: This component provides designers with a set of assertions to check the proper use of TLM channels. Design errors such as port connections, or getting from an empty channel are not detected by C++ errors. Furthermore, other code errors in the TLM code result in C++ syntax and semantics errors that are much distanced from the source of the problem, and do not help the designer find the error. Using our TLM assertions, designers are issued assertion violation errors and become aware of their errors, and will not have to depend on misleading C++ errors or no errors at all.
- **Synthesized TLM channel assertions**: The third component of this TLM assertion package supports the RT level verification of TLM channels. These channels were synthesized into RTL codes in order to get TLM based descriptions one step closer to the actual hardware implementation. RT level verification of these channels requires some specific RTL, and in our case, OVL like assertions. This way, the RTL HDL codes are enhanced to provide users with checks for proper use of the synthesized channels.

Figure 1: UTAVP Structure

Figure 1 shows the overall structure of our UTAVP Package. In the following sections, the three parts of this verification package will be described. These parts together, enable the designer to verify their channels by assertions put in TLM channels, around channels and in their synthesized RTL code.

2. Mixed TLM-RTL designs assertions

The first component of our package consists of OVL equivalent SystemC assertions. At the RT Level, Open Verification Library has been used to define proper assertions to verify RTL HDL codes. In this part, these assertions are ported to the SystemC language so that they can be used to verify the SystemC definition of RTL designs. This is suitable for the cases in which a TLM module is substituted by a Systemc RTL module, or perhaps for SystemC codes developed as parts of a design to interact with the other design components in TLM or SystemC. Figure 2 shows an instance of such mixed design. In cases of mixed TLM and RTL SystemC designs, these provided SystemC RTL assertions, could verify both the functionality and the interfaces of such SystemC RTL code.
Assert\_always, assert\_delta, assert\_quiescent\_state, and assert\_one\_hot are some of these implemented assertions. The SystemC implementation of assert\_always and an example of its use is shown in Figure 3.

```cpp
// inclusions are omitted
template <int severity_level = 0, int options = 1, 
    class assert\_msg = &msg
    class assert\_always : public sc\_assertion
public: sc\_inbool() test\_expr;
public: void do\_assert(void)
    while(1)
        unit\_check\_message
        if (ASSERT\_reset\_signal\_en(1) == true) 
            print\_msg(;message());
        sc\_assert\_always(argc);
    
public: sc\_msg\_args(argc, class* argv[])
    // signal definitions are omitted, 1 is the test expr
    Test\_vec[1];
    assert\_always assert\_always\_always("assert");
    assert\_always\_always\_write();
    assert\_always\_always\_write\_expr(1);
    assert\_always\_reset\_first();
    sc\_assert\_always(argc);
    sc\_assert\_always(argc);
    sc\_assert\_always(argc);
    sc\_assert\_always(argc);
    return 0;
```

Figure 3: SystemC implementation and usage of assert\_always

3. Transaction level assertions

The second part of our package is used for verification of TLM channels. Several assertions are provided in this category. These assertions could be used for primitive TLM channels, including FIFO channels, request-response channels, and transport channels. Our TLM channel assertions could be categorized as below:

a. **Channel assertions**: these assertions are used to verify the functionality of channel independent of its interfaces.

b. **Reader side assertions**: reader side assertions are used to verify the functionality of a channel when interfacing to a reader module. In this category, assertions check the correctness of data transmission in addition to some more properties.

c. **Writer side assertions**: these assertions are fundamentally like the reader side assertions, except that these will check the functionality of the channel when interfacing with a writer module. Different modes of operation (blocking or non-blocking) have their own set of assertions.

d. **Whole system assertions**: there are some more assertions provided to verify the channel included in a complete system of writer, channel, and reader.

 Furthermore, all of these assertions are carefully designed in a way that they will flag the malfunctioning part and the error accurately.

4. RT level TLM channels assertions

The last component of this package is dedicated to defining a new set of RT level assertions to verify the synthesized TLM channels. In this part, all the primitive TLM channels are synthesized into HDL code. In the next step OVL assertions are inserted into the synthesized RTL codes to verify the RT level operations of the channels. Examples of assertions for RTL verification of the channels are an assertion for FIFO overflow, an assertion for not reading from an empty FIFO, and proper handshaking assertions in case of blocking and non-blocking read and write.

Figure 4: utilizing new assertions in verification of a synthesized TLM FIFO channel

Figure 4 illustrates the method used here. The synthesis procedure provides HDL equivalent of the TLM channel, and then this equivalent model is verified by use of assertions provided in the third component of UTAVP.

5. Conclusions

In this project, a useful package for system level designers was developed which will be found of great help for debugging and verification of SystemC and OSCI TLM library based implementations. It provides a library of SystemC and OSCI TLM assertions which can be used for faster and more accurate debugging of system level designs.

6. References


Acknowledgement

The authors would like to thank the former member of CAD group, Mrs. Naghilou who has contributed willingly.