SST-SystemC Interoperability Toolkit

Sabbir Ahmed

April 22, 2019

Abstract

SST-SystemC Interoperability Toolkit (SSTSCIT) is a collection of header files developed to provide interoperability between Structural Simulation Toolkit (SST) and SystemC. SST is a parallel event based simulation framework that allows custom and vendor models to be interconnected to create a simulation environment. SystemC is a system-level modeling language composed of C++ classes and macros. SSTSCIT aims to provide the capability to interoperate the two systems without interfering with any of the kernels by concealing the communication protocols in black box interfaces. This project provides a demonstration of the interoperability by simulating a traffic intersection, where the traffic lights are determined by SystemC processes.

1 Introduction

This collection of header files provides methods to transmit and receive signals between SST components and SystemC modules. The toolkit provides a black box interface that can be interfaced with both SST and SystemC via their internal communication transports.

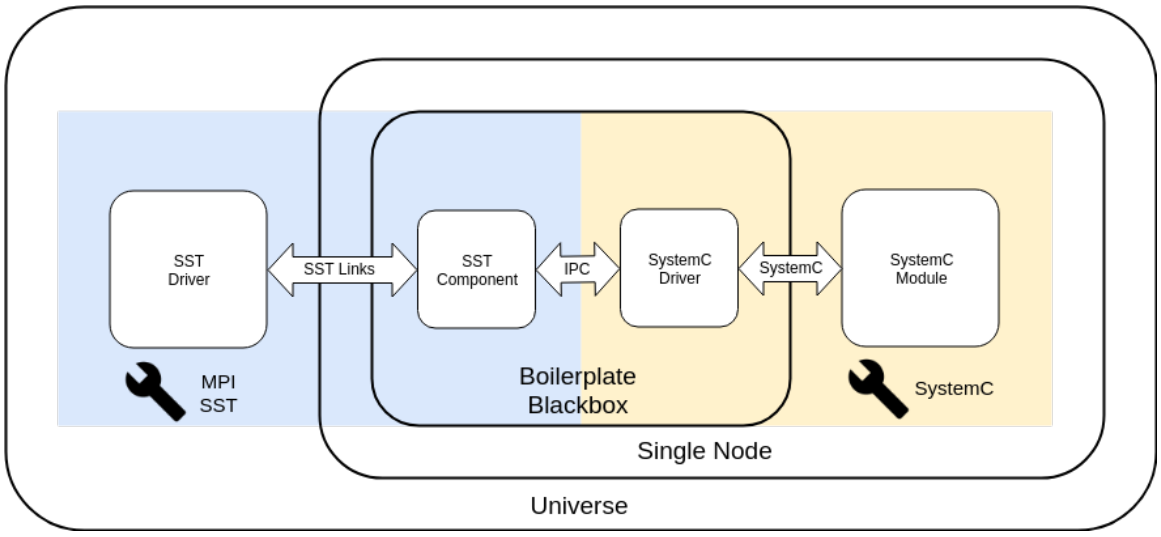

Figure 1: Components of SSTSCIT

2 Components

2.1 Black Box Interface

The black box interface consists of:

1. A SystemC driver
2. An SST component

Each SystemC modules must have their corresponding driver file to interoperate within the black box interface. It is possible to interoperate multiple SystemC modules with a single driver file. However, the additional communication lines must be accounted for in the corresponding black box SST component.

The toolkit includes a Python class that generates the boilerplate code required for the black box interface.

2.2 UML

3 Communication

3.1 Inter-Black Box Communication

The two components inside the black box interface are spawned in the same node and therefore communicate via interprocess communication (IPC) transports. The following is a list of supported IPC transports:

1. Unix domain sockets
2. ZeroMQ

It is possible to add custom IPC protocols to the interface by implementing a derived class of `sigutils::SignalIO` with customized sending and receiving methods.

3.2 SST-Black Box Communication

An SST model can interface the black box via standard SST links.

The following snippets demonstrate an SST link transmitting a unidirectional signal from the SST environment to the black box interface.

```
// parent_sst.cpp

// register a string event in the class declaration
SST_ELI_DOCUMENT_PORTS(
    { "demo_din", "Demo model data in", { "sst.Interfaces.StringEvent" }},
    ...
)

// initialize the link in the class declaration
SST::Link *demo_din;

// set up the SST link in the constructor
demo_din = configureLink("demo_din");

// trigger the event in the clocked function
demo_din->send(new SST::Interfaces::StringEvent(...));
```

```
// blackboxes/demo.cpp

// register the same string event in the class declaration
SST_ELI_DOCUMENT_PORTS(
    { "demo_din", "Demo model data in", { "sst.Interfaces.StringEvent" }},
    ...
)

// initialize the same link in the class declaration
SST::Link *demo_din;

// set up the SST link in the constructor with an event handler
demo_din = configureLink(
```

```
    "demo_din",
    new SST::Event::Handler<demo>(this, &demo::handle_event)
);

// receive and parse the event in the event handler
void demo::handle_event(SST::Event *ev) {
    auto *se = dynamic_cast<SST::Interfaces::StringEvent *>(ev);
    if (se) {
        std::string _data_in = se->getString();
        ...
    }
    delete ev;
}
```

3.3 SystemC-Black Box Communication

A SystemC module can be interfaced by a standard source file inclusion.

4 Proof of Concept

4.1 Intersection Simulation

A simulation prototype has been developed to test the project.

4.2 Continuous Integration