SystemC and SDL Co-Modelling Methods

Alexander Stepanov, Irina Lavrovskaya, Valentin Olenev, Alexey Rabin

St. Petersburg State University of Aerospace Instrumentation
190000, 67, Bolshaya Morskaya, St. Petersburg, Russia
Alexander.Stepanov@guap.ru, Irina.Lavrovskaya@guap.ru, Valentin.Olenev@guap.ru,
Alexey.Rabin@guap.ru

Abstract

This paper gives an overview of three possible, from authors' viewpoint, co-modelling methods of SDL and SystemC. The first method assumes to insert SystemC into a SDL model by including of the C header files into the SDL model. The second method is an insertion of SDL into SystemC by "teaching" of the SDL model how to process requests and commands of the SystemC model. The third method assumes to run SDL and SystemC independently in operating environments using a specific tool with SDL and SystemC interfaces. Also in this paper we compare these approaches marking out their advantages and disadvantages.

Index Terms: SystemC, SDL, Co-Modelling.

I. INTRODUCTION

There are two widely used languages applicable for modelling, they are SDL and SystemC. These languages are much different and also they give different abilities for developers and have various benefits. SDL language is adopted for visible demonstration of interactions between modules. As for SystemC, it is more suitable to trace internal functions of the modules [1].

SDL and SystemC are separately used but is there any chance to use these two languages together and what benefits it will give? If a joint use is possible, will it give an ability to use both modules interactions and internal operations simulation at the same time? In present paper answers on these and similar questions are given.

According to our point of view, there are three ways of SystemC and SDL co-modelling in one communication system:

insertion of SystemC into SDL (into the SDL Tool);

insertion of SDL into SystemC (by compiling SDL to C code);

independent use of SDL and SystemC by the instrumentality of a specific tool with SDL and SystemC interfaces.

II. MAIN PART

A. Insertion of SystemC into SDL

Let's consider the first method of SystemC and SDL co-modelling. It is a question about SystemC insertion into SDL. SDL can understand C code by means of the Telelogic SDL Tool [2]. So an SDL/SystemC models integration could be broken into a number of stages:

- writing a SDL model;
- writing a SystemC model;

- writing a SystemC channel;
- writing an SDL interlayer;
- writing a patch to convert SDL data types to C types;
- writing a pure C code to a *.h file which will be an implementation of interface between interlayer and channel;
- include this *.h file in the SDL interlayer;
- include this *.h file in the SystemC channel.

Such kind of route is depicted on the Figure 1:

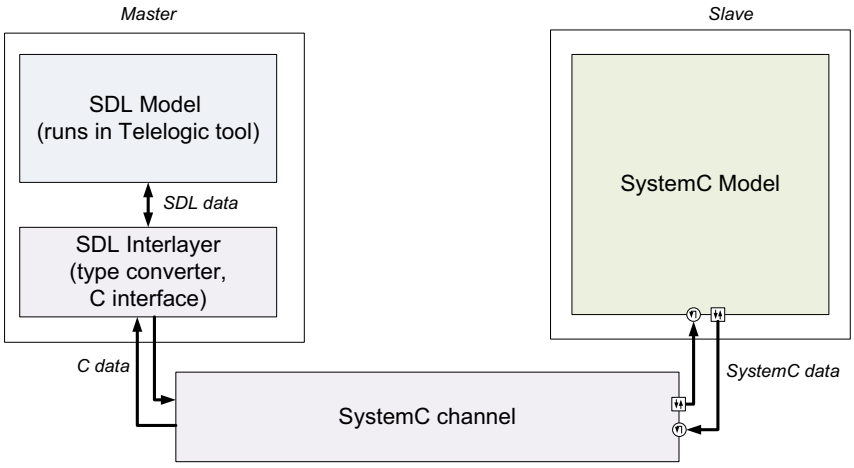

Figure 1. Insertion of SystemC into SDL route (SystemC→SDL)

This method can have some implementation restrictions. First, SDL could execute a *.h file written only in pure C, so use of classes, inheritance and other features of C++ and SystemC imposes difficulties. But even in this case SDL data types can not be easily converted to C types. For joint use of two languages kind of “type converter” should be implemented. This converter should be able to convert arrays, structures and signals into corresponding C types. These points make use of additional C and C++ libraries (e.g. SystemC) for SDL difficult. But it is still possible to implement such interlayer which would give an ability to convert data types and include SystemC.

Also such implementation would have a disadvantage related to usability. The SDL model could be changed, but any change in a “bottom” part of the model would cause updates in the interlayer. Also a big percent of changes in the SystemC model will cause a change in the channel which makes method calls.

Concerned method has another difficulty. Such joint usability would give an opportunity to see the results of the SDL operation part only. But all the SystemC side would be hidden “from the eyes of the developer”. So the SystemC and C parts debug becomes very difficult.

In addition, this method has a number of benefits. It is convenient to organize the SystemC and SDL interaction and it is the easiest way to implement the joint use of these languages. Also here the SystemC model is used as a library, so all the function calls are performed by the SDL model. This way simplifies SDL/SystemC synchronization problems. But in that case the SystemC model should be event oriented.

Also this method gives an ability to use the SDL Simulator, because the SDL model will be a master. Hereby we can view all interactions in the SDL model.

B. Insertion of SDL into SystemC

Let's turn to the second method of SDL and SystemC joint use. It is insertion of SDL into SystemC. There is an ability to create pure C code on the basis of a SDL model in the Telelogic SDL Tool. So the integration of the SDL model into the SystemC model is divided into following stages:

- writing a SDL model;
- generation of pure C code from the SDL model;
- writing a SystemC model;
- writing a SystemC channel;
- writing a C++ interface between C analogue of the SDL model and the SystemC channel.

Such kind of route is shown on the Figure 2:

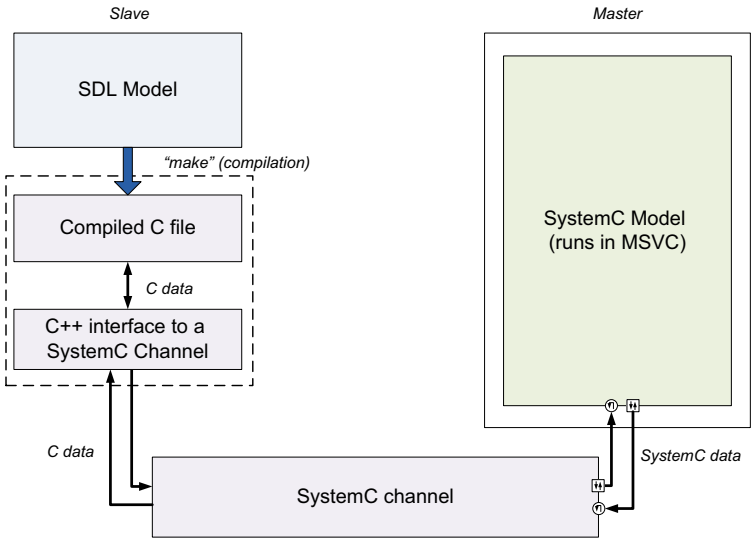

Figure 2. Insertion of SDL into SystemC route (SDL→SystemC)

In this method the SystemC model is a master and the SDL model is a slave. So the SystemC model would run on C++ compiler and this model will interact with C analogue of the SDL model by means of the SystemC channel.

This method has several disadvantages. First thing to mention is that by use of the given method it is impossible to work with the SDL Simulator. As a result it is impossible to make changes in the SDL model itself and to trace interactions between modules. Another point is that it is necessary to create a C++ interface for connection of the C analogue of the SDL model and the SystemC channel. Also such implementation causes changing of the SDL model and generation of a new C code each time when it is needed to change a testing model.

In addition, this way of co-modelling has a number of advantages. As the SystemC model is a master, so it is possible to work with code of the SystemC model. It means that we can perform a step by step model debugging. Moreover, such structure of co-modelling gives a possibility to use a SystemC clocking. And finally, it should be pointed out that this method allows working with C code only on the "SDL side".

C. SDL and SystemC independent use

And the last method to discuss in this paper is use of SDL and SystemC independently by passing the results through a file. Writing results to a file and reading them depend on clocks. This method consists of several steps:

- writing a SDL model;
- writing a SystemC model;
- creation of a file for connection of two models;
- writing a tool with interface for managing of both models.

Such kind of route is presented on the Figure 3:

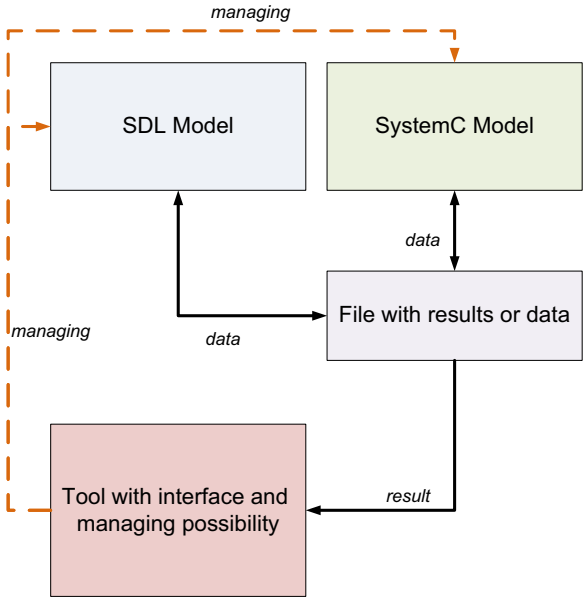

Figure 3. Use of SDL and SystemC independently (SDL / SystemC)

A special tool for managing SDL and SystemC models would be a master. This tool would handle point-to-point work of two SDL/SystemC systems and would give all information about interactions between both models. It should be mentioned that each model would have its own clocks.

This method has several advantages in comparison with ways discussed above. First of all, both models functionality could be fully observed. So there is a possibility to combine benefits of SystemC and SDL languages. Another point is that both models would work independently from each other, so that changes in one model wouldn't have an influence on the work of the whole system. Monitoring of transmitted through the channel data is simplified by an implementation of the management tool.

However this method has some restrictions. Multiple accesses to the file from both models and management tool can cause implementation difficulties. The management of models functionality by the tool is still a good question for research because SystemC and SDL models have various access and management mechanisms.

III. CONCLUSION

We considered three methods of SystemC/SDL co-modelling. Each of them has distinct features as it is described in Table 1.

Table 1. Specific features of SystemC and SDL co-modelling methods

Feature	SystemC → SDL	SDL → SystemC	SDL / SystemC
Master/slave	SDL is master, SystemC is slave	SystemC is master, SDL is slave	No explicit master
Clocking	SDL clocking	SystemC clocking	Independent clocks
Resource costs	SDL Tool is running	C++ compiler is running	SDL Tool, C++ compiler and a management tool are running at the same time
Visibility	Only SDL interactions are visible	Only SystemC interactions are visible	Both models functionality processes are visible

Examined approaches are pretty various, but applicability of each of them depends on compatibility of features to developers requirements.

ACKNOWLEDGMENT

The authors would like to thank Finnish-Russian University Cooperation in Telecommunications (FRUCT) program [3] for support of the related R&D activities, Nokia's and NSN university collaboration programs in Russia for supporting FRUCT community and all FRUCT experts for commenting and reviewing the paper.

REFERENCES

[1] A. Stepanov, V. Olenev, A. Rabin. Comparison of SDL and SystemC Languages applicability for the protocol stack modelling // The annual scientific conference of SUAI students. *Proceedings / St. Petersburg: SUAI*, 2009 (in Russian).

[2] SDL and TTCN Suite 6.1. SDL Suite and TTCN Suite Help, *http://www-01.ibm.com/software/awdtools/sdlsuite/*

[3] Official webpage of Finnish-Russian University Cooperation in Telecommunications (FRUCT) program, *http://www.fruct.org.*