Research on Integrating Simulink Model in KD-DRT Platform Based on TLC

Yao Xinyu
The State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System
Luoyang, 471003, China
e-mail: sim001@139.com
.College of Information Systems and Management
National University Of Defense Technology
Changsha, 410073, China

Wu Wenbo[2]
The State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System
Luoyang, 471003, China
e-mail: sim001@139.com
College of Information Systems and Management
National University Of Defense Technology
Changsha, 410073, China

Li lin[1],
The State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System
e-mail: sim001@139.com
College of Information Systems and Management
National University Of Defense Technology
Changsha, 410073, China

Abstract—KD-DRT simulation platform tries to manage MATLAB/Simulink models, and make them compatible and cooperating with other C/C++ models in distributed real-time simulation. MATLAB/Simulink together with KD-DRT are compared and analyzed in system structure and properties. The shortages of MATLAB/Simulink in distributed real-time simulation are proposed, and KD-DRT is chosen as the support platform instead. KD-DRT chooses S-Function mode to create simulation models, and chooses Target Language Compiler(TLC) to inline S-Function and collect information of relationship, parameters and ports from Simulink models. Correlated RTW file is created and the structure is analyzed to extract these informations through TLC. Tests are carried out in the end to prove that KD-DRT can control parameters of Simulink models online, and transfer signals between multiple models outside the MATLAB environment. KD-DRT proves to be more suitable to serve as the support platform for distributed real-time simulation ,and is able to combine Simulink models with C/C++ ones in simulation system.

Keywords- TLC; Simulink; KD-DRT; RTW;S-Function

I. INTRODUCTION

Hardware In the Loop Simulation(HILS) is broadly used in various engineer fields for system analyzing, designing and optimization. Traditionally, single one simulation computer is used in HILS system, and all the real-time simulation tasks, including model calculation and IO interaction, are carried out by the sole computer. Under this architecture, simulation computer turns to be the key and the bottle neck of the simulation system[1]. With the developments of computer technique, multiple distributed computers are adopted to serve as the real-time simulation computers in HILS system.

KD-DRT is a platform to support this kind of distributed real-time HILS system[2]. Fig.1 illustrates that multiple distributed simulation computers are enrolled in KD-DRT system. Since Simulink is widely used to construct simulation models, How to integrate Simulink models into the simulation system is the key point for KD-DRT. Some works have to be done to fulfill this task.

At first, MATLAB/Simulink is analyzed for it’s usage in HILS, and the shortages are listed, and then KD-DRT is introduced to cope with these shortages. Secondly, S-Function, Target Language Compiler (TLC) and data structure of Simulink models are studied to integrate Simulink models. At last, a test is carried out to verify the method.

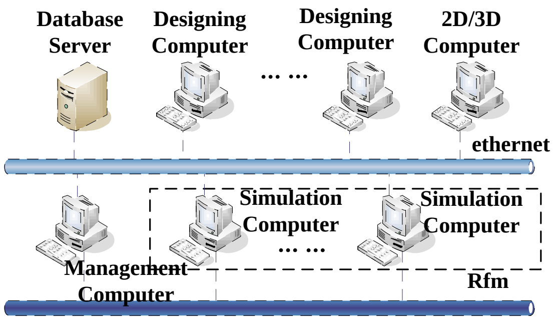


Figure 1. Hierarchical blocks in RTW file

II. ANALYZING OF MATLAB/SIMULINK

Simulink are widely used to constructed simulation models for its convenient[3-5]. As a powerful simulation platform, MATLAB/Simulink is used mainly in the field of mathematic analyzing simulation. It also provides a real-time simulation developing module, called as Real-Time Workshop (RTW), with which Simulink models can be turned to C-code for specific Operation System(OS), and then C-code can be compiled with the selected compiler and then run at real-time under the designed target computer[6]. So real-time simulation can be achieved for Simulink models, lots of related simulation supported platforms, such as xPC, dSPACE, RT-LAB and HRT1000, take this method to construct real-time simulation system to support HILS[7-10]. But There are two obvious problems for this kind of method:

Firstly, distributed real-time simulation under multiple computers can not be realized efficiently based on Simulink/RTW. Because Simulink simulation engine is designed for analytic simulation, which runs typically under sole computer. That is why the majority of supported platforms based on Simulink/RTW always take only one target computer for real-time simulation. RT-LAB however, is the exception, it can supports more than one target computer on the other hand[11]. But the target’s number is also limited. In fact, the number seldom exceeds four. That is because of the limitation of the simulation methods of Simulink, especially the integration methods used to calculate continuous models, which is not suitable for distributed calculation. The speeds-up of real-time simulation is hard to exceed 1 for those tightly coupled continuous simulation models, no matter how many targets are adopted. What is worse, the more targets are used, the slower simulation speed it would achieve.

Secondly, complex customized simulation system is hard to be constructed by Simulink alone. Based on the various block modules and toolboxes embedded in Simulink, it is convenient for user to build a simple simulation model. But, there are usually many specific arithmetic or logic methods proposed by experts, which can not be realized by Simulink blocks efficiently or conveniently. C/C++ is much more flexible than Simulink to realize these method, and is much more acceptable to be adopted as the developing tool in common. Although C-mex S-Functions are introduced by Simulink to solve these problems, it would cost much more for experts to transfer the C-code method to S-Function, and efficiency would be lost more or less during the translating from C-code to C-mex S-Function and then translating back to C-code again by RTW builder[12].

So we can see that Simulink is good at constructing of some kinds of simulation models, but it is not suitable to be used to build complex simulation system alone, especially for those distributed real-time HILS system. KD-DRT, on other hand, is a valuable platform to construct complex distributed real-time simulation system. C/C++ is adopted by KD-DRT to build models for common, Simulink is also introduced to build related models. KD-DRT proposes correlated simulation engine to support both kinds of models.

In order to manager the models in KD-DRT simulation engine, some information including parameters and signals must be collected from models. There is no problem for C/C++ models to fulfill the task, since C/C++ are flexible and open. But, Simulink models are wrapped into the simulation engine, some extra works have to be done to extract the correlated information from the Simulink models. TLC is the tool chosen to do this work by KD-DRT.

III. USING TLC AND INLINING S-FUNCTIONS

A. *The functions of TLC & S-Function*

TLC used in this scenario is an integral part of the Simulink/RTW. TLC is designed to convert the model description file, usually named as model.rtw, into target-specific code. To get high efficient code, it can be used to customize generating code. Through customization, platform-specific code can be produced, or we can incorporate our own algorithmic changes for performance, code size, or compatibility also, with existing methods that we prefer to maintain. It is the advantage that we make use of to inline S-Functions with code we write to do further work. With this method, the efficiency is ensured. The overriding purpose to inline code of S-Functions with this method is that we must get the information of targets. Without the information, we can not find the way to control models running at targets. The information we need to get is listed as below:

- The relationship between models.
- Information of parameters of models
- Information of signals(ports) of models.

To get the information above, we have a few methods, but the efficiency is not good, since the callback functions are invoked from time to time during the procedure. Because the full information relative to targets is stored in the file of model.rtw, and the method to read file is direct and easy with high efficiency. The structure of RTW file needs to be studied.

Among various methods to construct simulation models, provided by Simulink, KD-DRT chooses S-Function method to integrate Simulink models. Because S-Function can wrap model’s detailed information, and provide with object-orientation (OO) operation to the end user, and thus make the integration much more effective. The Real-Time Workshop code generator will produce the inline code of S-Function when a corresponding TLC file exists for the S-Function. As to the reference from MATLAB, inline code of S-Function can generally produce more efficient code[13]. The content of TLC file maybe arithmetic method wrote by developers or an API called to arithmetic method from the outside of MATLAB system. According to the different kinds of TLC files, the methods of inline S-Function can be divided to two categories:

1) Fully inline S-Function. With this method, the developer can write his own arithmetic in TLC files to replace the call to S-Function methods.

2) Wrapper inline S-Function. With this method, the developer can write a small TLC file to call maturity arithmetic and TLC will invoke the arithmetic when compiling. This method meet the need to developers to produce a large project based on complex arithmetic codes.

In this paper, fully inline S-Function will be the key-point. To know how to inline, the structure of S-Function and TLC files must be studied at first.

B. Callback functions of S-Function and TLC

The comment below is a S-Function example provided by MATLAB:

```
#define S_FUNCTION_NAME s_function_name
#define S_FUNCTION_LEVEL 2
#include"simstruc.h"
staticvoid mdlInitializeSizes(SimStruct *S)
{
……
}
<additional S-function routines/code>
#ifdef MATLAB_MEX_FILE
#include"simulink.c"
#else
#include"cg_sfun.h"
#endif
```

The example shows that S-Function may contain three parts: definition and includes, callback functions, and external API. Obviously, callback functions including their internal functions are the main parts of the file. Each S-Function have a set of callback functions to execute when simulating[14]. TLC file is necessary to inline S-Function. Similar to the S-Function, TLC file also contains a set of callback functions, the functions are defined in the format illustrated as below:

```
%implements "s_function_name" "C"
%function Outputs(block, system) Output
……
%endfunction
```

The callback functions of TLC that can produce codes maybe corresponding to callback functions of S-Function. Others cannot, however. If an S-Function and the corresponding TLC file have the same callback function, the callback function of TLC file will replace the callback function of S-Function at the procedure of compilation. To inline an S-Function, the corresponding TLC file which named the same as it must be placed in the work directory. The table below lists part of corresponding callback functions.

TABLE I. CALLBACK FUNCTIONS

TLC	S-Function
Enable(block, system)	mdlEnable
Disable(block, system)	mdlDisable
Start(block, system)	mdlStart
InitializeConditions(block, system)	mdlInitializeConditions
Outputs(block, system)	mdlOutputs
Update(block, system)	mdlUpdate
Derivatives(block, system)	mdlDerivatives
Terminate(block, system)	mdlTerminate

Based on the study ahead, we know what is inline S-Function. Inline S-Function means that TLC will invoke the callback function corresponding to S-Function's, and replace it and produce target code while compiling the S-Function.

For an example, callback function of Outputs(block, system) from TLC file that output the signal as 2*u[i], will replace the callback function of mdlOutputs(SimStruct *S, int_T tid) from S-Function that output the signal as 3*u[i] at the procedure of compilation. With this method, the code produced is determined by the arithmetic in TLC files. Then inlining of S-Function is realized.

IV. PROBING RTW FILE BY TLC

The control of the models is often from the supporting platform separated from MATLAB/Simulink environment in real test. It is the base to get the information of relationships among models, parameters of a model, and ports for KD-DRT platform. TLC and RTW file are used to fulfill this task.

A. The structure of RTW file

RTW file, usually named as model_name.rtw, can be set to be created during the code generation from Simulink model, which contains all the information of the model. In the RTW file, the information are stored as hierarchical information blocks. Blocks with different ranks have relationship like a father with sons[13].

The content of Fig. 2 illustrates part of a RTW file. In the file, we can see that a block called "CompileModel" is the root block with all the other blocks following it.

The father block and son block are connected with symbol of ".". So in order to get son block of "Name" from the father block of "CompileModel", notation of "CompiledModel.Name" is used.

Blocks with the same name are arranged in order under their common father block. Because of that, you can get the 3rd "block" in 1st "Subsystem" using the command below: "CompiledModel.BlockHierarchyMap.System[0].Block[2]". The son blocks from the a father block with the same name are indexed from zero.

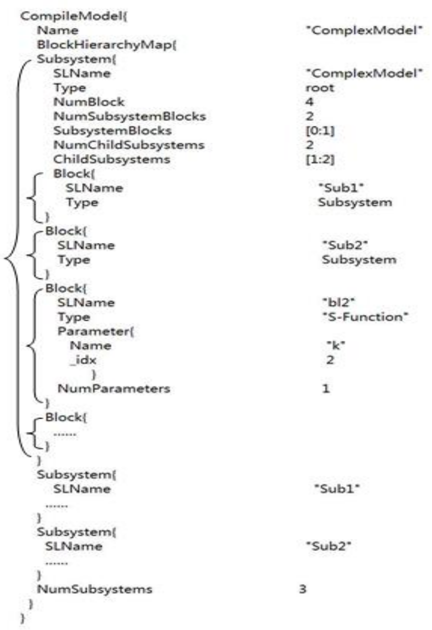
B. The method to get relationship between models

All the models forming a Simulink system have their own information blocks. The information blocks are independent, orderly stored in a block named with "BlockHierarchyMap". Fig.3 shows the components of it.

To get the relationship, two points must be got: One is the number of its brothers, and another is the model name.

For example, if a information block's name is "SonBlock", and its father block is "FatherSubsystem[i]", the number of it's brothers can be got through TLC notation as "Subsystem[i]. NumBlocks". And it's model name can be got as “SonBlock.SLName”.

If multiple blocks with the same name are existed, index would be used. For example, the model name of the 2nd block in the 1st Subsystem is got with TLC notation of "Subsystem[0].Block[1].SLName".



```
CompileModel{
  Name                              "ComplexModel"
  BlockHierarchyMap{
  Subsystem{
    SLName                          "ComplexModel"
    Type                            root
    NumBlock                        4
    NumSubsystemBlocks              2
    SubsystemBlocks                 [0:1]
    NumChildSubsystems              2
    ChildSubsystems                 [1:2]
    Block{
      SLName                          "Sub1"
      Type                            Subsystem
   }
   Block{
     SLName                           "Sub2"
     Type                             Subsystem
   }
   Block{
     SLName                           "bl2"
     Type                             "S-Function"
     Parameter{
       Name                            "k"
       _idx                           2
          }
     NumParameters                    1
   }
   Block{
     ......
   }
  }
  Subsystem{
    SLName                           "Sub1"
   ......
  }
  Subsystem{
   SLName                           "Sub2"
   ......
  }
  NumSubsystems                     3
 }
}
```

Figure 2. Hierarchical blocks in RTW file

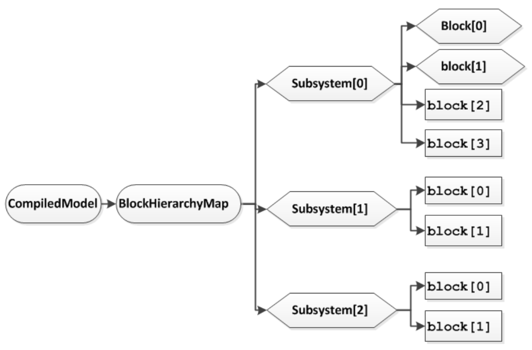


Figure 3. Model relationship in RTW file

C. *The method to get parameters of a model*

For KD-DRT, initial values of parameters are needed. In order to get the value of each parameter, the number of parameters must be obtained firstly. In a system, all the parameters are attached to models. For this reason, we must get the information block of correlated model.

From RTW file, the number of parameters of a block can be got with the TLC notation like "Block[i]. NumParameters". Then, for the 2nd parameter, we can got it's name with "Block[i].NumParameters.Name" and it's idx with "Block[i].NumParameters._idx".

The idx is a sole identifier of this parameter in the system. With the idx, fine-detail information of it can be got in "ModelParameters", which stores all the parameters of this system in order. So the initial value of the parameter is obtained by TLC notation of "ModelParameters. Parameter[2].Value" from RTW file.

So, the methods are clear, with which, the initial values of all parameters of any model in the simulation system can be accessed from the RTW file through steps of "number of parameter->name and idx of parameter->initial value".

D. *The method to get ports of a model.*

Among various information of ports, only the numbers of the ports for KD-DRT, including input ports and the output ports, are needed. The method to get number of ports is much easy than that to get parameters. In RTW file, TLC notation of "Block[i] .NumDataInputPorts" is used to get number of input ports of a model, while "Block[i] .NumDataOutputPorts" is used to get that of output ports.

With the information of parameters and ports of all the models, KD-DRT can control these Simulink models outside MATLAB environments, just like the other C/C++ models built under the KD-DRT standard.

V. TEST OF METHOD

As studied in previous sections, Simulink models are built as S-Functions, and correspond RTW files are generated, form which information of parameters and pots are obtained using TLC. In this section, a test is carried out to verify the efficiency of the method. Fig. 4 shows a simulation system with four Simulink models, built from S-Functions. Each model has some parameters and ports.

MATLAB/Simulink .mdl files of the model are built by RTW Target system first, which is proposed by KD-DRT, and the RTW file is accessed by KD-DRT, and then relevant information of parameter and ports are shown.

In this example, output port of "out1" from model "100" and input port of "in8" from model "103" are connected. Because connections among models are done by KD-DRT, Simulink lines are not needed to connect ports here. As the figure shows, model "100" output a sinusoid wave signal through "out1", and model "103" multiply input port of "in8" by the first parameter of the model, and send out to output port of "out10":

$$in8 = out1= \sin(t) \quad (1)$$

$$out10 = in8 * Par1 = Par1*\sin(t) \quad (2)$$

If the amplitude of sinusoid wave from out1 is 1, and the first parameter of model "103" is 5, the amplitude of signal of out10 is 5(Fig. 5). This Figure prove that the connection between ports from different model is correct.

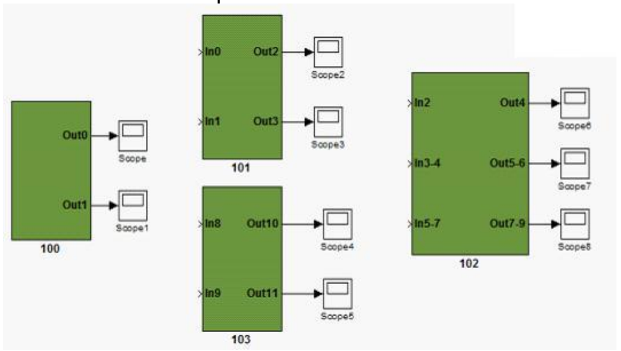


Figure 4. Simulink model

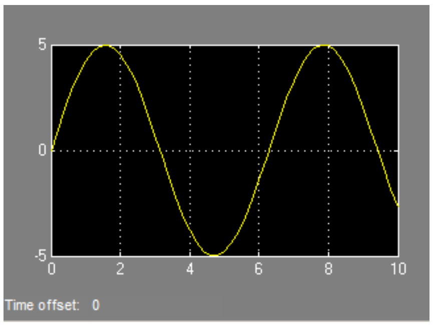


Figure 5. Output signal of "out10"

Then, in the KD-DRT platform, we can click the first parameter of model "103", then the content of it is changed from 5 to 10. After that, the download button is clicked and the parameter in target is modified on line. The amplitude of sinusoid signal of out10 changes to 10 from 5 just as Fig. 6 shows. This test proves that that the parameters of Simulink model is controlled by KD-DRT.

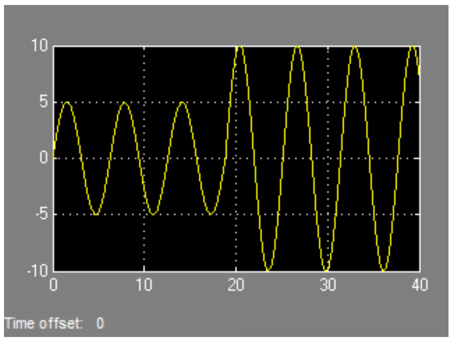


Figure 6. Parameter is changed on line

VI. CONCLUSION

Simulink model is built through S-Function mode in KD-DRT. TLC is used to inline S-Function of model, and RTW file is created during c-code generation procedure by Simlunik/RTW. The informations of relationship, parameters and ports are collected through the RTW file by TLC. With these informations, KD-DRT can control the Simulink models as the standard C/C++ models in the simulation system outside the MATLAB/Simulink environment, and merge these two kinds of models during simulation process. Signals can be transferred from one model to the other, and parameters can be changed on line. So, the distributed real-time simulation can be realized.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China under Grant 61374185, and the State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System under CEMEE2014K0303B.

REFERENCES

[1] Yao Xinyu, Huang Kedi. Distributed real-time simulation system based on high speed network[C] Proc. of the 2004 Simulation Computer, Software, Method and Modeling Conference, XiaMen, [s.n.], 2004: 33-41.

[2] Yao Xinyu, Huang Kedi. Application and research of architecture of distributed real-time simulation system[C] Proc. of the 2009 China Simulation Technology Conference, Jiu Jiang, [s.n.], 2009: 30-35.

[3] Tian Xianzhao, Zhou Fanli. Co-simulation based on MWorks and Simulink[J], Computer Aided Engineering, 2013, 22(1):54-63

[4] Liu Jie, Yi Shengyong. Study on flight simulation of an UAV Based on MALAB/Simulink[J], Automation Information, 2012, 136(8): 50-51.

[5] Zhang Xinyan, Jin Ping, Chen Huaimin, Duan Xiaojun. Driver under RTX environment and Simulink module encapsulation technique[J], Technique of Measurement and Control, 2013, 32(4): 96-99.

[6] Yang Di, Li Litao, Yang Xu, and Zhu Chenyuan. Development platform and application of real-time simulation system[M], Beijing: Tsinghua University Press, 2002.

[7] Hu Jiangfeng，He Hao，Zhang Long, Zhai Erning, and Li Juan. Design of the xPC real-time simulation system using wavelet denoising method[J], Journal of Gun Launch & Control, 2013, 3(9): 54-57.

[8] Zhu Yuguan. Design of missile seeker in the-loop simulation based on RT-LAB environment[J], Journal of Ship Electronic Engineering, 2012, 32(3): 81-83.

[9] Ma Yujing, Li Xuejun, and Liu Wei. A two dimentional freedom control system based on dSPACE[J], Journal of Changchun University, 2013, 23(4): 397-400.

[10] dSPACE Inc. Real-time interface (RTI and RTIMP) implementation guide[Z], dSPACE GmbH, 2005.

[11] OPAL-RT Inc. RT-LAB Version 10.4 user guide[Z]. OPAL-RT Inc. ,2007.

[12] Fen Lei, Yao Xinyu. Adjusting parameter online with C-API for real-time simulation systems based on RTW[J], Ordnance industry automation, 2006, 25(1): 87-90.

[13] Shi Renxing. Key technical research of distributed real-time simulation based on Simulink RTW[D], Hunan, Changsha: National University of Defense Technology, 2011.

[14] Wang Ziyu, Deng Fujun, and Liu Tingting. Simulation of direct torque control system based on the Simulinkl S-Function[J], The World of Inverters, 2013, (3):55-68.