An Automatic Parallelization Scheme for Simulink-based Real-Time Multicore Systems*

Minji Cha, Seong Kyun Kim, and Kyong Hoon Kim

Department of Informatics, Gyeongsang National University
Gajwadong 900, Jinju 660-701, South Korea
{cha7355,buti,khkim}@gnu.ac.kr

Abstract. Matlab/Simulink provides developer with model-based development environments for various applications. Real-Time Workshop in Simulink toolkits automatically generates C/C++ programs, which enables user to build real-time systems easily. However, the generated program code is only for single process so that it is difficult to build high-performance real-time systems. In this paper, we propose an optimization scheme of parallelizing Simulink blocks for building multi-threaded real-time applications on multicore systems. The proposed the scheme extracts the dependency graph of Simulink blocks and estimates their execution times on target platforms. Based on the dependency graph with estimated execution times, multi-threaded real-time applications are automatically generated on RTAI Linux systems.

Keywords: Optimization, Parallelization, Simulink, Real-Time

1 Introduction

Matlab/Simulink [1] provides users with graphical user interface (GUI) for block model-based design. One of key functions is the automatic code generation of Real-Time Workshop toolkit that generates C/C++ program for various target platforms. Thus, users easily develop real-time programs and generate codes for their platforms.

Although the automatic code generation function of Simulink reduces cost, the generated program is only in the form of single process. This makes it difficult to utilize high performance of multi-core systems. Our earlier work [4] provided user-defined Simulink blocks in order to generate parallel codes for real-time multicore systems. However, the work has the drawback that users specify the parallel part using user-defined block manually. Thus, in this paper, we enhance the framework by adding a new automatic parallelization scheme which extracts the dependency graph of Simulink blocks and produces multi-threaded codes.

* This research was supported in part by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2012-H0301-12-3003) supervised by the NIPA (National IT Industry Promotion Agency).

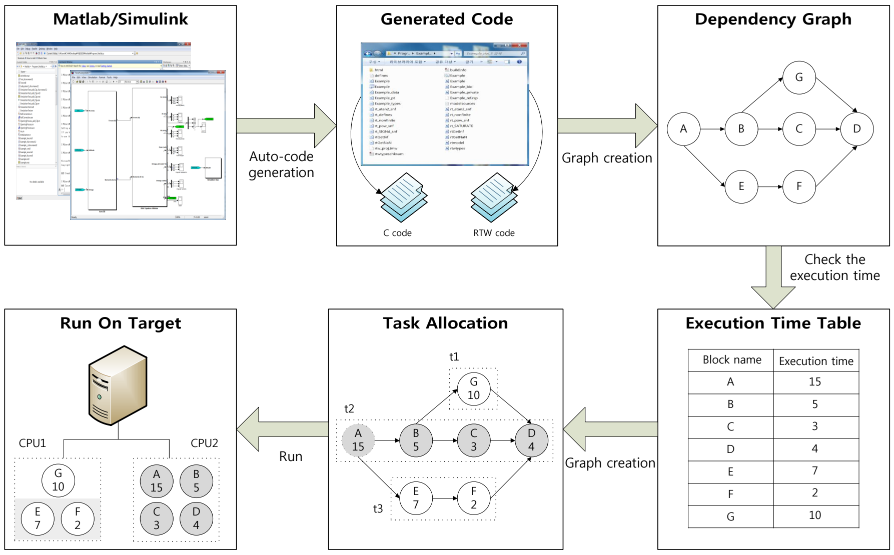
2 The Proposed Framework

In this section, we present how to run a Simulink application on real-time multicore systems with automatic parallelized threads. Fig. 1 shows the proposed framework. The followings explain each steps of the framework.

Step 1. *Programming a Simulink application:* First, a user develops a Simulink application based on block diagrams. The blocks which are to be parallelized should be specified as subsystems and functions. This program is generated automatically to C code using Real-Time workshop tool. RTW codes and the main C codes among the generated source codes are used to generate the dependency graph.

Step 2. *Creating the dependency gragh:* RTW code includes all of block information such as position, state, dependency, and so on. We can obtain the dependency graph of main blocks which are specified as subsystems and functions from RTW code. However this graph does not include the execution time information, so that we need to estimate the block execution time.

Step 3. *Estimating the execution time of each block:* Dependency graph needs execution time of main blocks in order to parallelize subtasks appropriately. So, We insert the codes for profiling execution times of blocks around the block codes. When the modified program is executed in the target platform, we



Block name	Execution time
A	15
B	5
C	3
D	4
E	7
F	2
G	10

Fig. 1. Steps of the proposed framework

can measure the execution time.

Step 4. *Generating the parallel C code:* The dependency graph with the execution time information is used to analyze an optimized parallelization of Simulink blocks. For example, the dependency graph with seven nodes in Fig. 1, it can be parallelized into three subtasks.

Step 5. *Executing program on the target plaform:* The compiled program runs in the real-time target system. These tasks are allocated and executed parallelly in multi-core systems. We implemented the proposed framework in RTAI Linux.

3 Discussion and Summary

Current Simulink provides auto-generated code in the form of single structure. In this paper, we proposed the automatic parallel code generation scheme for multicore real-time systems. Auto-generated code can be optimized using the block dependency graph that includes the dependency relation and execution time of each block. Based on the dependency graph, an optmized multi-threaded codes are generated automatically on the target platform. The proposed scheme is useful for building high-performance real-time systems based on Simulink applications.

References

1. Mathworks: Simulink. `http://www.mathworks.com/products/simulink/`
2. Mathworks: Simulink Model-Based and System-Based Design (Writing S-Function) (1998–2001)
3. Lineo, Inc.: DIAPM RTAI programming guide. `http://www.rtai.org`. (2000)
4. Cha, M., Kim, K., Lee, C., Ha, D., Kim, B.: Deriving High-Performance Real-Time Multicore Systems based on Simulink Applications. in: The 9th IEEE International Symposium on Embedded Computing (EmbeddedCom 2011), pp. 267-274. (2011)
5. Bucher, R., Balemi, S.: Rapid Controller Prototype with Matlab/Simulink and Linux. Control Engineering Pracrice, vol.14, pp. 185–192. (2003)
6. Bucher, R., Dozio, L.: CACSD under RTAI Linux with RTAI-LAB. In: Real-Time Linux Workshop, vol. 5. (2003)
7. Canedo, A., Yoshizawa, T., Komatsu. H.: Automatic Parallelization of Simulink Applications. In: IEEE/ACM International Symposium on Code Generation and Optimization (CGO), vol. 8, pp. 151–159. (2010)
8. Sarolahti, P.: Real-time Application Interface. `www.cs.helsinki.fi/u/sarolaht/papers/rtai.pdf`
9. Cho, S., Choi, K.: A Study on Validation of OFP for UAV using Auto Code Generation. The Kerean Society for Aeronautical & Space Sciences, vol. 37, no.4, pp. 359–366. (2009)
10. Cho, S., Park, J., Kim, S., Choi, K., Park, C.: Development of Software for UAV using Automatic Code Generation of MATLAB. The Korean Society for Aeronautical & Space Sciences, vol. 4. pp. 571–574. (2007)