A Platform Independent Model for MPSoC Scheduling Using UML-RT

S. Ewins Pon Pushpa and D. Manamalli

Abstract—Integration of object modeling and real-time scheduling theory is a key to successful use of object technology for real-time software. Unified Modeling language (UML) is widely used modeling language in software engineering community which supports object-oriented design which in turn encourages component reuse. Recently, system modeling with a Unified Modeling language (UML) is an active research area for developing MPSoC real-time system. Our aim is to propose a unified modeling approach from model design, simulation, until code generation to analyze Rate Monotonic algorithm for global, partitioned and semi-partitioned scheduling for MPSoC using UML-Real Time. As a technical contribution success ratio and average processor utilization for the generated periodic tasks is compared with various scheduling principles, to validate our proposal.

Index Terms—Model based development, multi processor system-on-chip, rate monotonic scheduling, unified modeling language-real time.

I. INTRODUCTION

Modeling is an essential part for large software projects, and helpful to medium and small projects too. Modeling can assure that business functionality is complete and correct, end-user needs are met. Modeling is the only way to visualize our design and check it against requirements before actual coding [1]-[2]. The integration of object modeling and design methods, and real-time scheduling theory is the key to successful use of object technology for real-time software. Surprisingly many past approaches to integrate the two, either restrict the object models, or do not allow sophisticated schedulability analysis techniques [3].

Platform-based design has been envisioned to meet the design challenges of ever-increasing system complexity [4]. The design approach theoretically allows rapid construction of very large systems and their architecture exploration. However, the analysis of such systems is very demanding and slow with traditional methods, partly because many of the physical architecture evaluation tools operate on relatively low abstraction levels, and partly because of the typically vast design space. Hence, platform independent methods, in the form of a design flow, are needed to overcome the design and verification gaps. Unified Modeling Language (UML) has been utilized in novel design methods proposing a solution for the challenge [5].

UML is widely used modeling language in software engineering community, to specify the requirement and analyze the target system. Recently, one of the most popular modeling language is UML, which is widely used in software design to specify the static and dynamic aspects of an object oriented system [6]. UML is becoming a de-facto standard for object-oriented modeling [7]-[8]. Rich graphical notation and its modeling capabilities allow to capture and visualization of the system structure and behavior at multiple level of abstraction. The tool, named UML-RT for real-time, developed by the Rational Corporation, uses UML to express the original ROOM (Real-Time Object-Oriented Modeling) concepts and their extensions. ROOM (Real-Time Object-Oriented Modeling) is an architecture description language widely used in the telecommunication industry to develop embedded software. The concepts of ROOM have been incorporated into the CASE tool Rational Rose Real-Time (RoseRT) in the form of a UML profile, commonly called UML-RT [9].

In recent years, model based system level design has gained considerable attention in Multiprocessor system-on-chip, since it simplifies the application behaviour and reveals the top level structure of the behaviour, abstracting out the low-level details [10]. Real-time scheduling techniques for multiprocessors are mainly classified into global scheduling and partitioned scheduling. In partitioned scheduling, tasks are first assigned statically to specific processors and then executed without migrations. In global scheduling, all tasks are stored in a global queue, and the tasks with highest priority are assigned dynamically to available processors [11]. Recently semi-partitioned scheduling posses the advantage of global scheduling having improved schedulability and partitioned scheduling having less runtime overhead [12]. And non-preemptive scheduling is pertinent to high performance multi-core platforms since it exhibits low run-time overhead as well as easier implementation [13].

The aim of the proposed methodology is to propose a platform independent model for MPSoC using UML-RT and analyze Rate Monotonic algorithm for global, partitioned and semi-partitioned scheduling techniques. A comparative analysis is done on finding success ratio and average processor utilization for the above three mentioned scheduling techniques using the generated MPSoC model.

The paper is constructed as follows: Overview of UML-RT in section 2; MPSoC model using UML-RT is discussed in section 3; Rate monotonic algorithm (non-preemptive) for global, partitioned and semi-partitioned scheduling are analysed in section 4; Simulation and performance evaluation in section 5.
II. OVERVIEW OF UML-RT

In UML-RT, the three principle constructs for modeling are capsules, ports and connectors [7].

A. Capsules

Capsules are the fundamental modeling constructs of UML-RT. They are complex, physical, possibly distributed architectural objects that interact with their surroundings exclusively through one or more ports.

B. Connectors

Connectors really capture the key communication relationships between capsule roles. They interconnect capsule roles that have similar public interfaces, which are called ports. A key feature of connectors is that, they can only interconnect compatible ports.

C. Ports

Ports are objects whose purpose is to send and receive messages to and from capsule instances.

III. MPSOC MODEL USING UML-RT

Real-time scheduling of MPSOC is shown in Fig. 1. We assume that a multiprocessor platform consists of M identical processors. A periodic task set P consists of N periodic tasks. A periodic task is denoted by Task \(i = (C_i, D_i, T_i)\), where \(C_i\) is the computation time, \(D_i\) is the relative deadline and \(T_i\) is the minimum inter-release separation, also referred to as its period. In rate monotonic algorithm \(D_i = T_i\).

The utilization of task, \(U_i\) for Task \(i\) is defined as

\[ U_i = \frac{C_i}{T_i} \] (1)

Each task has a unique priority. Higher priorities are assigned to tasks with shorter periods.

A. Top capsule

Technological trends on high performance computational systems are moving towards execution platforms made up of multiple programmable and dedicated processing elements implemented on a single chip known as multiprocessor system-on-chip (MPSoC).

The structure diagram of the top capsule for the proposed MPSoC architecture is shown in Fig. 2. Two capsules are generated, one for generating tasks periodically which is named as gentask and the other capsule, scheduler which schedules the periodic tasks according to the scheduling algorithm. Using capsule instance, three identical processors are generated to execute the tasks according to the scheduling principle.

B. Protocol

A protocol named ‘signal’ is created and the various incoming and outgoing signals are generated. All capsules use ‘signal’ protocol to communicate with each other.

IV. RATE MONOTONIC ALGORITHM FOR GLOBAL, PARTITIONED AND SEMI-PARTITIONED TECHNIQUES.

In a single-processor system, a deadline of a task is missed when, a high-priority task is blocked by a low priority task for a long time due to non-preemptiveness. Therefore, when scheduling real-time systems, non-preemptive has been considered inferior because of its poor responsiveness. However in multi-processor systems, high-priority tasks will still have a chance to execute on available processors to meet deadlines. Moreover non-preemptive scheduling algorithms have lower runtime overhead and are easier to implement [14]. As per [14] experimental simulation surprisingly shows, that the global non-preemptive fixed priority scheduling (NP-FP) outperforms global preemptive fixed priority scheduling (P-NP).

The following is the Rate monotonic algorithm for different scheduling principles:

A. Global Scheduling

1) begin
2) for \(i = 1\) to \(P\) (periodic task set);
3) for Task1 = Task1 to TaskN (N periodic task);
4) check priority according to period \(T_i\)
5) find free processor 1 to M
6) assign Taski to free processor dynamically
7) endfor
8) endfor

B. Partitioned Scheduling

1) begin
2) for \(i = 1\) to \(P\) (periodic task set);
3) for Taski = Task1 to TaskN (N periodic task);
4) check priority according to period \(T_i\)
5) assign Taski to processor pre-assigned
6) endfor
C. Semi-Partitioned Scheduling

1) begin
2) for i = 1 to P (periodic task set);
3) task i = Task 1 to Task N (N periodic tasks);
4) check priority according to period T;
5) check Task i is global or partitioned;
6) if global assign Task i to processor dynamically;
7) else Task i to processor pre-assigned;
8) endfor
9) endfor

V. SIMULATION AND PERFORMANCE EVALUATION

The task parameters settings are as follows:
1) The number of processor is 3.
2) Four tasks, each task, its period T in seconds, is chosen from {2, 14}.
3) Its utilisation U is accordingly in {0.1, 0.6}.
4) Higher priorities are assigned to tasks with shorter periods.

Success ratio and effective processor utilisation are considered for performance measuring criteria. Effective processor utilization UT is given as

\[ UT = \frac{\sum C_j}{\Gamma} \]  

where, \( \kappa \) is number of successful tasks scheduled, \( \Gamma \) is total time of scheduling. Success ratio \( S \) is defined as

\[ S = \frac{\text{Number of successfully scheduled tasks}}{\text{Total number of tasks scheduled}} \]  

We use the same task generation strategy for global, partitioned and semi-partitioned scheduling and the results are compared and the performance of proposed algorithm is shown in Fig. 3 and Fig. 4.

From the Fig. 3, and Fig. 4, success ratio and average processor utilization of the tasks are improved for semi-partitioned scheduling when compared with global scheduling.

Task 3 and task 4 are statically scheduled in processor 3 for partitioned scheduling and dynamically scheduled for the same in semi-partitioned scheduling. From Fig. 5, the number of successful tasks for task 3 and task 4 in partitioned scheduling is more when compared with Fig. 6, for the same load. Runtime overhead for partitioned scheduling will be less because all tasks are statically pre-assigned to processors. Here, in semi-partitioned scheduling tasks 3 and task 4 are dynamically assigned to processor 3 during runtime depending on their priority.

Therefore runtime overhead for semi-partitioned scheduling will be more compared to partitioned scheduling and less compared to global scheduling.

Semi-partitioned scheduling posses the advantage of global scheduling having improved schedulability and partitioned scheduling having less runtime overhead. While the drawback is, the least priority task gets less chance to get executed in semi-partitioned scheduling.

REFERENCES


S. Ewins Pon Pushpa hails from Chennai, Tamilnadu, India. She received her B.E. degree in Electronics and Communication Engineering at Karunya Institute of Technology, Coimbatore in 2002. She obtained her M.E. in Applied Electronics from Government College of Engineering under Anna University, Salem in 2004. She is currently working as Teaching Fellow in the Department of Electronics, Madras Institute of Technology, Anna University, Chennai. She is currently pursuing her Ph.D under Anna University. Her area of interest are Embedded Software, Parallel Computing, Embedded system.

D. Manamalli hails from Chennai, Tamilnadu, India. She received her B.E. degree in Electrical and Electronics Engineering at Government College of Technology, Coimbatore in 1993 and M.E. in Instrumentation Engineering at Anna University, Chennai in 1999. She completed her Ph.D at Anna University, Chennai in 2006. She is currently working as Associate Professor in the Department of Instrumentation Engineering, Madras Institute of Technology, Anna University, Chennai. Her research areas are Process control and dynamics, Modeling and control of discrete event systems, Neural networks and Fuzzy logic control.