Available online at www.sciencedirect.com
SCIENCE DIRECT
J. Parallel Distrib. Comput. 66 (2006) 19–31
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

Rapid performance re-engineering of distributed embedded systems via latency analysis and k-level diagonal search☆

Jungkeun Park[a], Minsoo Ryu[b,*], Seongsoo Hong[a], Lucia Lo Bello[c]

[a] *School of Electrical Engineering and Computer Science, Seoul National University, Seoul 151-742, Korea*
[b] *College of Information and Communications, Hanyang University, Seoul 133-791, Korea*
[c] *Department of Computer Engineering and Telecommunications, University of Catania, Catania, Italy*

Received 7 April 2004; received in revised form 5 January 2005; accepted 20 June 2005
Available online 22 August 2005

Abstract

This paper presents a systematic methodology aimed at rapid and cost-effective re-engineering of distributed embedded systems. We define embedded system re-engineering as an analysis and alteration of a legacy system to guarantee newly imposed performance requirements such as throughput and input-to-output latency. Our methodology pinpoints performance bottlenecks of a system and selectively upgrades processing elements at the least cost. Inputs for our methodology include a system design specified by a process network over a set of processing elements and a new throughput requirement. The output is a set of scaling factors that represent the ratios of the performance upgrades for processing elements.

Our methodology works in two steps. First, it estimates the latency of each process and identifies bottleneck processes. Second, it derives a system of constraints with scaling factors being free variables and formulates an optimization problem. Then, it solves the optimization problem for scaling factors with an objective of minimizing upgrade cost. For this methodology, we propose an accurate latency analysis technique for precedence-constrained tasks under preemptive fixed priority scheduling. We also propose a k-level diagonal search algorithm that allows us to trade optimality for search time. Our experimental results show the effectiveness of the proposed re-engineering approach.

Keywords: Distributed embedded systems; Re-engineering; Latency analysis; Performance/cost optimization

1. Introduction

Due to the diversity and complexity of embedded systems and due to increased competition in the associated industry, developers are under very stringent requirements for increasing production speed. Many techniques and methodologies have been proposed to assist them in designing, analyzing, and testing embedded systems [11,18,21,23,24,28]. Recently, component-based software design approaches have been widely adopted for the rapid development of application-specific embedded systems.

In these approaches, an embedded system can be prototyped by composing reusable components. Such a development prototype is often subject to design modification when it fails to meet a given performance specification. In that case, the developer should locate performance bottlenecks in the prototype system, explore design alternatives using component libraries, and replace the bottleneck components with new ones at the least cost. Similar problems are encountered in industry during the re-engineering of a legacy system, when a product with additional features and enhanced performance is developed by modifying an old design.

☆ The work reported in this paper was supported in part by MOST under the National Research Laboratory (NRL) Grant M1-9911-00-0120, by the Institute of Computer Technology (ICT), by the Automation and Systems Research Institute (ASRI), and by the research fund of Hanyang University (HY-2003-T). An earlier version of this paper appeared in preliminary form in the *Proceedings of IEEE Real-Time Systems Symposium* (December 2001).

* Corresponding author.
E-mail addresses: jkpark@redwood.snu.ac.kr (J. Park), msryu@hanyang.ac.kr (M. Ryu), sshong@redwood.snu.ac.kr (S. Hong), llobello@diit.unict.it (L. Lo Bello).

0743-7315/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2005.06.004

Generally, the re-engineering problem is defined as a sequence of activities involving reverse engineering, system alteration, and forward engineering [11]. During a re-engineering process, the reverse engineering captures an understanding of the behavior and structure of the system, the system alteration modifies the structure and components of the system for enhanced performance, and the forward engineering creates new functionalities. Using this terminology, we define the performance re-engineering of an embedded system as a specific instance of the re-engineering problem, such that the reverse engineering corresponds to the bottleneck process analysis within the system, and the system alteration performs the latency reduction of the system. Such a performance re-engineering problem is of the utmost practical importance during the production of embedded systems, since it can lead to significant reduction in development time and cost.

Unfortunately, the performance re-engineering problem for an embedded system poses serious challenges to developers. First, it is quite difficult to accurately estimate the latency of an embedded system, since this requires extensive static timing analysis of the system. Because embedded systems often consist of a network of processes that run on a heterogeneous distributed multiprocessor platform composed of microprocessors, microcontrollers, digital signal processors, and application-specific instruction set processors, the complexity of this task is considerable. Second, it is fairly difficult to eliminate performance bottlenecks in the system since system resources are shared in a complicated manner, thus minor changes in a single processor may affect the synchronization and timing behavior of the entire system.

While there exist plenty of design techniques and software tools for embedded systems which are based on real-time scheduling theory and formal methods [2,5,6,8,9,20, 22,24,27], relatively few approaches address the performance re-engineering aspect of embedded systems. Without the help of systematic re-engineering methodologies, developers often resort to the ad hoc iteration of system analysis and re-design that often leads to over-optimization of the system. It is fairly obvious that this approach will fail when the system to be re-engineered becomes complex.

In this paper, we present a systematic methodology that allows rapid and cost-effective re-engineering of distributed embedded systems. A distributed embedded system is modeled as a process network and task graphs, where tasks are executed by a priority-based preemptive scheduler, as in many embedded systems. A performance requirement is given as the throughput of the system. For rapid performance re-engineering, our approach attempts to upgrade only the processing elements that execute bottleneck processes, while leaving the architecture and implementation intact. Inputs to our re-engineering problem are as follows:

(1) A process network and task graphs representing the underlying system.

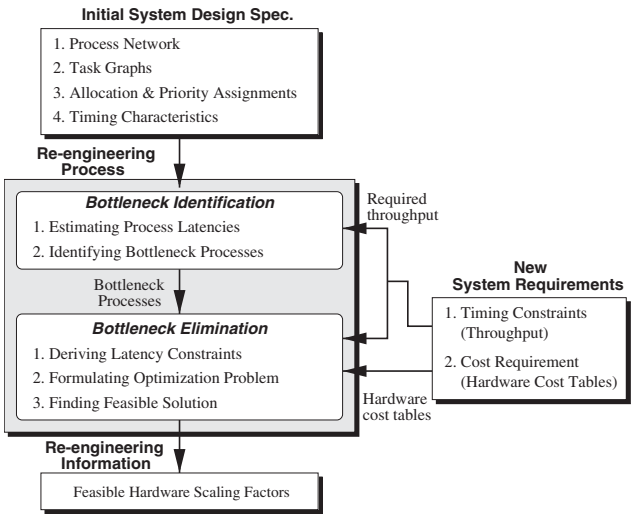

Fig. 1. Overview of the approach.

(2) Task allocation and priority assignment.
(3) A desired throughput requirement.
(4) Hardware upgrade cost tables at various performance profiles.

With the above inputs, the objective of our approach is to find speedup ratios of processing elements that satisfy the new throughput requirement with minimal hardware upgrade costs. Our approach is based on latency analysis and a cost–benefit optimization. We identify performance bottlenecks of the system by estimating the latency of each process and eliminate such bottlenecks by formulating and solving an optimization problem. To do so, our approach employs the following two techniques:

(1) An accurate latency analysis technique that estimates the latency of each process.
(2) An effective heuristic search algorithm that solves the optimization problem.

The proposed re-engineering method works in two steps, *bottleneck identification*, and *bottleneck elimination*, as shown in Fig. 1. First, it estimates the latency of each process and identifies bottleneck processes. Second, it derives a set of latency constraints for each bottleneck process and formulates an optimization problem with an objective of minimizing the re-engineering cost. Then, it finds optimal speedup ratios for processing elements that need speedups to improve the performance of bottleneck processes.

1.1. Related work

Existing re-engineering methods primarily deal with functional and structural analysis and modification of software systems [11,13,21] and hardware systems [18,28]. Madisetti et al. [18] propose a systematic technique for rapidly upgrading electronic systems. They propose using virtual prototyping accompanied by their tools and libraries