Design Refinement for Efficient Clustering of Objects in Embedded Systems

Waseem Ahmed[1], Doug Myers[2]
[1]Curtin University of Technology, Sarawak Campus, Malaysia
[2]Curtin University of Technology, Bentley Campus, Western Australia

Abstract

Hardware software co-design seeks to meet performance objectives via a combination of hardware and software modules. One difficulty in reaching these objectives lies in lack of cohesion and increased coupling amongst the implemented modules that results in an increased inter module communication cost. While most of the traditional partitioning approaches are initiated in the post-coding phase, we suggest the design stage may be a better focus of attention in addressing this problem.

In this paper, we propose a novel approach that uses information from sequence diagrams in UML designs to help ease the partitioning problem.

1. Introduction

A key phase in the design of an embedded system is hardware/software partitioning that refers to the partitioning of the application into separate hardware and software modules. Traditional approaches to this problem as highlighted in [1][2] have been to initiate the process after the system specifications have been translated into code. The input to such partitioning approaches is thus the source code of the application, a binary implementation, or an internal format generated from the source code during analysis as seen in Figure 1.

An exception to the above is a work based on UML design specification [3] that uses function point analysis and COCOMO to compare different design alternatives at an early stage of analysis.

A major assumption in most of these approaches is that the source code reflects the best possible design, which may not be always true, as the designer of the code might not have taken into consideration the mixed nature of the final implementation. The limitations of the design in terms of mixed implementation are thus carried unchanged into the implementation phase.

In this paper we propose to analyze the design of an application with a mixed hardware software implementation, prior to subjecting it to the partitioning process.

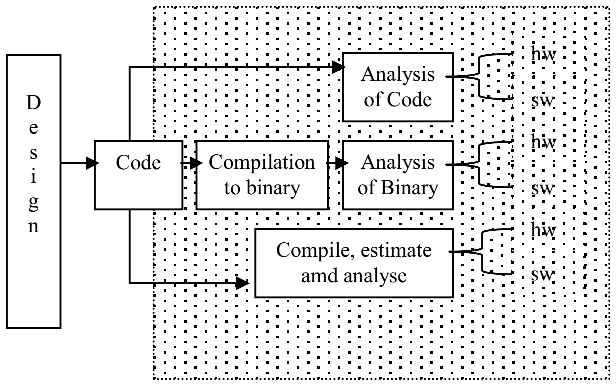

Figure 1. Traditional Partitioning Approaches

2. Cohesion, Coupling and Design Shuffle

An effective hardware software co-design is one that maximizes *cohesion (*the degree to which communication takes place among the module's elements) while minimizing *coupling* (the degree of inter-modular communication) amongst modules [ ]. Modules that have been designed by not taking into account the mixed nature of implementation may have high coupling, resulting in a high inter module communication cost (IMCC), that cannot be wholly rectified by the current partitioning approaches.

Reducing coupling between components may involve either minimizing the interaction between them by shifting the onus of communication to another component or by shifting the entire function (and/or any interaction between them) to another object(s) if possible. We choose to refer to this heuristic as the *design shuffle* or just *shuffle* in this document.

3. Sequence diagram analysis

Sequence diagrams in UML are used for depicting the scenarios of typical interactions and message passing between objects that constitute the system. For a single