Using Timed Petri Net to Model Instruction-Level Loop Scheduling with Resource Constraints

Wang Jian, Christine Eisenbeis and Su Bogong

INRIA Rocquencourt, Domaine de Voluceau, BP 105, 78153 Le Chesnay Cedex, France

Received June 8, 1993; revised November 1, 1993.

Abstract

This paper uses timed Petri net to model and analyze the problem of instruction-level loop scheduling with resource constraints, which has been proven to be an NP complete problem. First, we present a new timed Petri net model to integrate functional unit allocation, register allocation and spilling into a unified theoretical framework. Then we develop a state subgraph, called Register Allocation Solution Graph, which can effectively describe the major behavior of our new model. The main property of this state subgraph is that the number of all its nodes is polynomial. Finally we present and prove that the optimum loop schedules can be found with polynomial computation complexity, for almost all practical loop programs. Our work lightens a new idea of finding the optimum loop schedules.

Keywords: Instruction level parallelism, loop scheduling, register allocation and spilling, Petri net, timed Petri net.

1 Introduction

Since loop execution dominates total execution time of almost all practical programs, the exploitation of instruction level parallelism (ILP) for loops is a major challenge in the design of optimizing compilers for high-performance computers such as Very Long Instruction Word (VLIW) machines, superscalars and pipelined processors. The exploitation of ILP for loops has been commonly modelled as the instruction level loop scheduling problem with resource constraint (for short, loop scheduling problem) [1,2,3,15,16], which has been proven to be an NP complete problem [14].

In fact, the loop scheduling problem can be viewed as a resource allocation problem where the functional unit allocation and the register allocation should be considered. Moreover, register spilling [4] should be also considered when there are more simultaneously alive variables than the available physical registers. Up to now, however, almost all research work on the loop scheduling problem only consider the functional unit allocation (called simplified loop scheduling problem), and handle the register allocation and spilling as an independent problem.

The aim of this paper is to establish a unified theoretical framework in which the functional unit allocation and the register allocation and spilling can be integrated together. We
use timed Petri net as a basis. Timed Petri net (TPN) has been used to model the (simplified) loop scheduling problem\cite{11,13}. However, paper [11] only models the simplified loop scheduling problem; paper [13] models the functional unit allocation and register allocation without spilling, but the optimum loop scheduling approach presented in [13] suffers from the exponential computation complexity.

The new model presented in this paper is more general than the existing ones because we consider register allocation and spilling as well as functional unit allocation. Furthermore, we develop a theoretical tool, called Register Allocation Solution Graph, by which we can efficiently analyze our model and deduce an important result — that is, for almost all practical loop programs, the loop scheduling problem can be solved with polynomial computation complexity. Our work lightens a new idea of development of the optimum loop scheduling approaches.

Our paper is organized as follows. In Section 2, we review previous work on the TPN models of the loop scheduling problem. Section 3 describes the definition and the construction of our timed Petri net model. In Section 4 we present the Register Allocation State Graph, and describe how to use it for finding an optimum loop schedule with polynomial computation complexity. In Section 5 the conditions on loops for polynomiality and the application of our model are discussed. We conclude this paper in Section 6.

2 Background

2.1 The Simplified Loop Scheduling Problem

We model a loop as a doubly weighted data dependence graph, $G = (O, E, \lambda, \delta)$, called Loop Data Dependence Graph (LDDG), where $O$ is the set of operations in the loop, $E$ is the set of dependence edges, $\lambda$ and $\delta$ are two non-negative integers associated to each edge, for instance, $e = (op_i, op_j) \in E$, $(\lambda(e), \delta(e))$ denotes that $op_j$ can only be executed $\delta(e)$ cycles after $op_i$ of the $\lambda(e)$-th previous iteration has started executing. Not more formally, the simplified loop scheduling problem can be described as follows:

Construct a loop schedule $\sigma$, a mapping function from $O \times N$ to $N$ (non-negative integer set), $\sigma(op, i)$ denotes the execution cycle where the instance of operation $op$ of $i$-th iteration is issued. If the following constraints are satisfied:

1. Resource constraint: In each cycle, the same functional unit (or stage for pipelined unit) cannot be used more than one time.
2. Dependence constraint:
   \[
   \forall e = (op_i, op_j) \in E, \forall k \in N, \sigma(op_i, k) + \delta(e) \leq \sigma(op_j, k + \lambda(e)).
   \]

then we say that $\sigma$ is a valid loop schedule for the given loop. The goal of loop scheduling is to find a valid loop schedule with the minimum execution time.

It has be proven that the simplified loop scheduling problem is still NP complete\cite{14}.

2.2 The Existing TPN Models for Loop Scheduling

The previous work have already considered to use timed Petri net to model the (simplified) loop scheduling problem \cite{11,13}. In their models, the operations are modelled by transitions, whereas resources are modelled by places. The edges represent either dependence constraints or use/releasing of resources.

The main contribution of paper [13] is a very precise model of functional units, by considering in details their working states. For instance, every stage of a pipeline unit is