Lower Bound Estimation of Hardware Resources for Scheduling in High-Level Synthesis

Shen Zhaoxuan and Jong Ching Chuen
School of Electrical and Electronic Engineering
Nanyang Technological University, Nanyang Avenue, Singapore 639798
E-mail: eccjong@ntu.edu.sg
Received January 28, 2002; revised April 1, 2002.

Abstract In high-level synthesis of VLSI circuits, good lower bound prediction can efficiently narrow down the large space of possible designs. Previous approaches predict the lower bound by relaxing or even ignoring the precedence constraints of the data flow graph (DFG), and result in inaccuracy of the lower bound. The loop folding and conditional branch were also not considered. In this paper, a new stepwise refinement algorithm is proposed, which takes consideration of precedence constraints of the DFG to estimate the lower bound of hardware resources under time constraints. Processing techniques to handle multi-cycle, chaining, pipelining, as well as loop folding and mutual exclusion among conditional branches are also incorporated in the algorithm. Experimental results show that the algorithm can produce a very tight and close to optimal lower bound in reasonable computation time.

Keywords lower bound estimation, chaining, pipelining, mutual exclusion, high-level synthesis

1 Introduction

High-level synthesis is the realization of register transfer level (RTL) structure from the system functional specification. It consists of two major tasks, scheduling and allocation, the former determines the assignment of operations to control steps while the latter binds operations to hardware resources. The main difficulty in high-level synthesis is how to select the best design from the large number of possible designs. Trade-offs on design space exploration and optimization goals become the crucial problem in high-level synthesis.

In synthesis of large systems, lower bound prediction not only can narrow down the design space by pruning lots of inferior designs but also enables the synthesis system to explore the large design space efficiently. It is extremely time consuming for the synthesis system to locate directly at several "good" points in the design space without actually synthesizing all the possible designs. Furthermore, the accurate lower bound estimation results can be used to evaluate the quality of designs synthesized by heuristic algorithms.

There were some previous studies on lower bound predictions before actual scheduling. Jain et al. formulated a mathematical model for area-delay prediction for high-level synthesis. Timmer et al. generated the area-delay curves by relaxing some of the precedence constraints to select the minimal cost module set. Nourani and Papachristou gave a layout area cost estimation algorithm for a given RTL datapath with standard-cell and full-custom layout methodologies. Execution interval analysis approach is widely used for scheduling and estimation. Timmer and Jess adopted the bipartite graph matching algorithm for resource constraints scheduling and estimation while Sharma and Jain estimated architecture resource and performance by calculating the minimal overlap among different execution intervals of operations. Wehn et al. obtained the hardware lower bounds with a simple mobility matrix calculation. Ohm et al. proposed a fanout reduction and a variable merging approach to refine the lower bound on registers and an integrated approach for lower bound estimation with an area cost model covering register, buses as well as functional units. Hu et al. extended their interval estimation method to functional pipeline. Another popular model in high-level synthesis is the integer linear programming (ILP) formulation. Rim and Jain derived the lower bound using a relaxed ILP formulation and a greedy algorithm. Chaudhuri and Walker gave another
ILP formulation to compute the lower bound on functional units by considering the interdependence of the bound on different functional unit (FU) types while Lantgevin and German[20] tried to improve Rim and Jain's relaxation algorithm by adopting a recursive greedy technique to compute the ASAPUC (as soon as possible under constraint) value. Shen and Jong[21] proposed an integer programming model with a surrogate relaxation technique for the lower bound and upper bound when scheduling and multicomponent selection were considered simultaneously. Other models and techniques such as parameterized component estimation[22], timing and re-timing analysis and estimation[23,24] and power consumption estimation[26] have been proposed to aid the high-level synthesis.

The quality of a lower bound prediction depends on the accuracy of the lower bound and the efficiency of the estimator. The previous approaches estimated the lower bound by relaxing or even ignoring the precedence constraints of the Data Flow Graph (DFG), thus resulted in inaccuracy of the lower bound. Neither the loop folding nor conditional branch was considered. In this paper, a stepwise refinement algorithm is proposed to predict the lower bound on the number of hardware resources under the time constraints, taking the precedence constraints of the DFG into account. Experimental results show that the proposed algorithm can provide a very tight lower bound in a reasonable time. It can also handle multi-cycle, chaining, pipelining, as well as loop folding. A new condition tree naming and matching algorithm is also proposed to handle the mutual exclusion among conditional branches.

The rest of the paper is organized as follows. Section 2 describes the estimation algorithm. Section 3 discusses the variations on the active range graph due to the DFG precedence constraints and presents the stepwise refinement estimation algorithm. Section 4 extends the estimation algorithm to handle chaining, pipelining and loop folding. A new condition tree naming and matching algorithm for estimating the architectural resources with mutual exclusion among conditional branches is presented in Section 5. Experimental results are shown in Section 6 and Section 7 concludes the paper.

2 Estimating the Lower Bound on the AR Graph

2.1 Definitions

The following definitions are used to describe the estimation and stepwise refinement algorithms.

**Definition 1 (ASAP and ALAP).** ASAP time of an operation node in the DFG is the earliest time step in which the node can be executed subject to the precedence constraints of the DFG and the amount of the available resources. ALAP time of an operation node is the latest time step in which the operation must be completed so that all the operations in the DFG can be finished by a given time constraint \( T \).

Let the starting time step of the DFG be 0. The initial ASAP and ALAP time can be obtained under the assumption, where the amount of available resource is unlimited.

**Definition 2 (Active range (AR)).** The active range of an operation node is the time interval between its ASAP time and ALAP time, denoted by \([\text{ASAP}, \text{ALAP}]\).

**Definition 3 (Active range graph (AR graph)).** The active range graph of an operation type is the graph that depicts all the active ranges of this type of operations in the DFG.