Bounding Algorithms for Design Space Exploration[*]

Samit Chaudhuri
Magma Design Automation
Palo Alto, CA 94303

Robert A. Walker
Kent State University
Kent, OH 44242

Abstract

This paper describes several new algorithms for computing lower bounds on the length of the schedule and the number of functional units in high-level synthesis.

1 Introduction

A high-level synthesis system can explore the design space, finding the optimal tradeoff curve between area and time (schedule length, or latency) as illustrated in Figure 1. Even when the scheduling problem is combined with clock length determination and module selection, this exploration can be done efficiently [2]. One key to efficient exploration is the use of fast algorithms to compute lower and upper bounds on the optimal curve. As Figure 1 shows, when those bounds are the same, they represent the optimal solution. For the remaining points, more computationally intensive algorithms are required.

1.1 Basic Notation

Given a data flow graph (DFG), let the set of all operations be denoted as $\{o_i \mid i \in I\}$, where I is the index set of all operations. Let $asap_i$ (resp. $alap_i$) denote the as-soon-as-possible (resp. as-late-as-possible) control step (cstep) into which operation o_i can start execution. The cstep interval $S_i = [asap_i, alap_i]$ is then referred to as the *schedule interval* of operation o_i. Let the number of functional units (FUs) of type k be denoted as m_k, and the area as a_k, where k can be any type from a set of types K. Let the set I_k denote the index set of operations that are executed on a type-k FU.

2 Lower-Bounding Problems

This paper examines two lower-bounding problems. The Schedule Length Lower-Bounding Problem (SL-LB) is that of computing a lower bound on the minimum number of csteps required to schedule a DFG, while using no more than a specified number of FUs. The Functional Unit Lower-Bounding Problem (FU-LB) is that of computing lower bounds on the minimum number of FUs, while using no more than a specified number of csteps.

2.1 Formulation of Lower-Bounding Problems

In general, a lower-bounding problem can be viewed as an optimization problem, one that minimizes some objective function while satisfying a set of constraints. These constraints are obtained by *relaxing* or dropping some of the constraints on the original scheduling problem, leading to an easier problem to solve.

A solution to the scheduling problem satisfies three sets of constraints: (a) *resource constraints*, to ensure that no more

[*] Portions of this work were supported by the National Science Foundation under Grants MIP-9423953 and MIP-9796085.

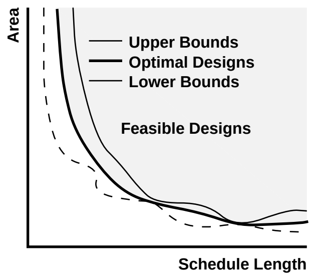

Figure 1: The 2-Dimensional Design Space

than m_k, $k \in K$ FUs are used in any cstep, (b) *precedence constraints*, to ensure that each operation finishes execution before all its successors, and (c) *interval constraints*, to ensure that each operation o_i is scheduled within its schedule interval $[asap_i, alap_i]$. Taken together, these three sets of constraints are denoted as C1.

To formulate lower-bounding problems, these constraints C1 are relaxed in two ways. First, the precedence constraints of C1 can be relaxed, forming constraints C2. Then the multi-cycle operations are broken up into chains of unicycle operations, forming constraints C3. In general, problems formulated with constraints C2 produce more accurate bounds than those formulated with constraints C3.

2.2 Lower Bound on Schedule Length

Given a limit m_k on the number of FUs of each type $k \in K$, a schedule length lower-bounding (SL-LB) problem can be formulated as follows. First, the $alap_i$ values are computed for each operation o_i, $i \in I$, based on a schedule length equal to the critical path length z of the DFG. Then an objective function is formulated to minimize a quantity called *max-tardiness*, which we denote as y and define as the maximum number of csteps that an operation o_i is delayed beyond $alap_i$. The lower bound $\underline{c}$ on the schedule length is then computed as $z + y$.

2.2.1 Schedule Length Lower-Bounding Problem SL-LB1

The most common SL-LB problem is the problem of minimizing y subject to constraints C3; we denote this problem as SL-LB1. Problem SL-LB1 is the same as the well-known problem of multiprocessor scheduling of independent, unit-time tasks with integer release times and deadlines, which can be solved by Jackson's Earliest Deadline Rule (ED-Rule) [1]. In high-level synthesis, the ED-Rule has been used to solve problem SL-LB1 by Rim and Jain [10], and by Rabaey and Potkonjak [9]. The complexity of this algorithm is $O(n \log n)$.

Another technique for solving problem SL-LB1 is based on a theorem originally given by Fernández and Bussell in [4, Theorem 2]. In high-level synthesis, this technique has been used to compute lower bounds on the schedule length by Sharma and Jain [11], and by Hu *et al.* [6]. We have proven in [3] that these bounds are exactly the same as those obtained by solving problem SL-LB1, so using this theorem is an alternative technique for solving SL-LB1. However, this technique is slower than the ED-Rule algorithm (see [10]) because its complexity, $O(nz^2)$, is higher than the ED-Rule since $z^2 >> \log n$. Unfortunately, the bounds produced by solving SL-LB1 are not always as tight as we might like.

2.2.2 Schedule Lower-Bounding Problem SL-LB2

One approach to finding tighter lower bounds is that of applying one of the basic SL-LB1 algorithms iteratively, thus gradually shrinking the schedule interval; we denote these approaches as problem SL-LB2. Since this new bound is produced with shorter schedule intervals than the original SL-LB1 problem, it is often tighter (but never looser) than the SL-LB1 lower bound. Techniques by Langevin [7], and by Hu and Carlson [5], fall into this category.

Our algorithm for solving problem SL-LB2 is also iterative, and in each iteration solves one SL-LB1 problem using Jackson's ED-Rule. The complexity of this algorithm is $O(n^2 \log n)$.

2.2.3 Schedule Lower-Bounding Problem SL-LB4

Although the bounds produced by SL-LB2 are tighter that SL-LB1 bounds, they can still be less than satisfactory, particularly for DFGs with multi-cycle operations. This motivates us to solve a relaxation problem based on constraints C2 instead of constraints C3. To the best of our knowledge, this new problem, denoted as SL-LB4, has not been explored previously in high-level synthesis.

Our algorithm for solving problem SL-LB4 starts with the SL-LB1 lower bound $\underline{c}$. For each type $k \in K$ of operations, it solves a feasibility problem to determine whether or not a feasible schedule exists, which uses no more than m_k FUs and $\underline{c}$ csteps, and which satisfies constraints C2. This feasibility problem is solved using the Barriers Algorithm originally proposed by Simons in [12]. If no such feasible solution exists, then $\underline{c}$ is increased and the procedure is repeated.

The complexity of our algorithm is $O(mn^2 \log n)$, where $m = \max_{k \in K} m_k$; however, if priority queues are implemented on stratified binary trees then the complexity reduces to $O(mn^2 \log \log n)$. In practice, this algorithm performs much faster than our SL-LB2 algorithm because the constant factor is much smaller, since it does not break each multi-cycle operation into numerous uni-cycle operations.

2.3 Lower Bounds on Functional Units (FUs)

Given a limit c on the schedule length, (where $c \geq$ critical path length z), a functional (FU) unit lower-bounding (FU-LB) problem computes lower bounds on the number of functional units of type k, $k \in K$, needed for any schedule of length $= c$, and can be formulated as follows. First, the $alap_i$ values are computed for each operation o_i, $i \in I$, based on a schedule length equal to c. Then an objective function is formulated to minimize the number m_k of functional units for each FU-type $k \in K$.

2.3.1 FU Lower-Bounding Problem FU-LB1

The most common FU-LB problem is the problem of minimizing m_k subject to constraints C3; we denote this as problem FU-LB1. One technique for solving problem FU-LB1 is based on a theorem originally given by Fernández and Bussell in [4, Theorem 1]. In high-level synthesis, this technique has been used by Sharma and Jain [11], by Ohm *et al.* [8], and by Hu *et al.* [6, 5] to compute lower bounds on the number of FUs. As we proved in [3], the bounds obtained by this algorithm are exactly the same as those obtained by solving problem FU-LB1. Rabaey and Potkonjak [9] also solve problem FU-LB1, although using a different method that iteratively solves SL-LB1.

Our algorithm for solving problem FU-LB1 first breaks multi-cycle operations into uni-cycle operations, and then computes a quantity $P^k_{s,t}$ that indicates the number of type-k uni-cycle operations whose ASAP and ALAP times are within the interval $[s,t]$. Then it computes the lower bound $\underline{m_k}$ as

$$\underline{m_k} = \max_{[s,t] \subseteq [1,c]} \{ \lceil P^k_{s,t} / (t - s + 1) \rceil \}.$$

We prove in [3] that $\underline{m_k}$ is an optimal solution of FU-LB1. The basic concepts behind our algorithm and the theorem of Fernández and Bussell [4] are analogous, but our algorithm's complexity, $O(n + c^2)$, is lower than the latter algorithm's $O(nc^2)$.

3 Upper-Bounding Problems

The previous section has discussed a variety of algorithms for computing *lower bounds* on the optimal schedule. However, knowing the *upper* bounds on the optimal schedule also helps to restrict the design space, as discussed in Section 1. In the interests of space, our upper-bounding algorithms will not be discussed here. For details, see [3].

References

[1] J. Błażewicz. Simple Algorithms for Multiprocessor Scheduling to Meet Deadlines. *Information Processing Letters*, 6(5):162 – 164, Oct. 1977.

[2] S. A. Blythe and R. A. Walker. Efficiently Searching the Optimal Design Space. In *Proc. of the 9th Great Lakes Symposium on VLSI*, pages ??–??, Ann Arbor, Michigan, Mar. 4-6 1999.

[3] S. Chaudhuri. *Scheduling and Design Space Exploration in High-Level Synthesis*. PhD thesis, Electrical, Computer, and Systems Engineering – Rensselaer Polytechnic Institute, 1995.

[4] E. B. Fernández and B. Bussell. Bounds on the number of Processors and Time for Multiprocessor Optimal Schedule. *IEEE Transactions on Computers*, C-22(8):745–751, Aug. 1973.

[5] Y. Hu and B. S. Carlson. Improved Lower Bounds for the Scheduling Optimization Problem. In *Proc. of 1994 IEEE International Symp. on Circuits and Systems.*, pages 295–298, London, England, May 30-June 2 1994. IEEE Computer Society Press.

[6] Y. Hu, A. Ghouse, and B. S. Carlson. Lower Bounds on the Iteration Time and the number of Resources for Functional Pipelined Data Flow Graphs. In [13], pages 21–24.

[7] M. Langevin and E. Cerny. A Recursive Technique for Computing Lower-Bound Performance of Schedules. In [13], pages 16–20.

[8] S. Y. Ohm, F. J. Kurdahi, and N. Dutt. Comprehensive Lower Bound Estimation from Behavioral Descriptions. In *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, pages 182–187, San Jose, California, Nov. 6-10 1994. IEEE Computer Society Press.

[9] J. M. Rabaey and M. Potkonjak. Estimating Implementation Bounds for Real Time DSP Application Specific Circuits. *IEEE Transactions on Computer-Aided Design*, 13(6):669–683, June 1994.

[10] M. Rim and R. Jain. Lower-Bound Performance Estimation for the High-Level Synthesis Scheduling Problem. *IEEE Transactions on Computer-Aided Design*, 13(4):451–458, Apr. 1994.

[11] A. Sharma and R. Jain. Estimating Architectural Resources and Performance for High-Level Synthesis Applications. *IEEE Transactions on VLSI Systems*, 1(2):175–190, June 1993.

[12] B. Simons. A Fast Algorithm for Multiprocessor Scheduling. In *Proc. of the 21st Annual Symposium on Foundations of Computer Science*, pages 50–53, Syracuse, New York, Oct. 13-15 1980. IEEE Computer Society Press.

[13] *Proc. of the IEEE International Conference on Computer Design*, Cambridge, Massachusetts, Oct. 3-6 1993. IEEE Computer Society Press.