Improving the Schedule Quality of Static-List Time-Constrained Scheduling *

Sriram Govindarajan and Ranga Vemuri

Digital Design Environments Laboratory, Department of ECECS, ML. 30
University of Cincinnati, Cincinnati, OH 45221-0030
E-mail: sriram@ececs.uc.edu, ranga.vemuri@uc.edu Web: http://www.ececs.uc.edu/∼ddel/

The most compelling reason for High-Level Synthesis (HLS) to be accepted in the state-of-the-art CAD flow is its ability to perform design space exploration. Design space exploration requires efficient scheduling techniques that have a low complexity and yet produce good quality schedules. The Time-Constrained Scheduling (TCS) problem minimizes the number of functional units required to schedule a particular Data Flow Graph (DFG) within a specified number of time steps. Over the past few years a number of techniques [1, 2] have been proposed to solve the TCS problem. Heuristic *list scheduling* algorithms have been widely used for their low-complexity and good performance. The complexity of a dynamic-list scheduling algorithm, such as the *Force Directed Scheduling* (FDS), is $\Theta(T * N^2)$, where T is the time constraint and N is the number of operations. Static-list scheduling [1, 2] algorithms are the least complex among the known class of scheduling techniques with a linear time complexity of $\Theta(T * N)$. Typically, static-list scheduling algorithms, in order to maintain low-complexity, do not perform any look-ahead like that of FDS. The drawback is that, static-list scheduling algorithms may not generate high-quality schedules.

However, the proposed static-list algorithm incorporates a novel *topological clustering technique* which acts as the look-ahead mechanism without any computational overhead. Prior to scheduling, the entire DFG is topologically ordered into levels of clusters as shown in Figure 1. The intuition behind the technique is that, clusters expose the inherent parallelism specific to a DFG structure and exploiting this parallelism should lead to faster schedules. These clusters act as *rubber bands that try to keep the enclosed operations as close as possible in a schedule*. During scheduling, the clustering information provides a clear idea of the current schedule quality with respect to a possibly optimal schedule. Based on this clustering technique we have derived cost functions that guide the static-list scheduling algorithm towards an optimal schedule. The algorithm employs a simple operation selection function and two cost functions that collectively determine the time step where the operation is scheduled. This enables the proposed algorithm to generate high quality schedules, when compared to another recently proposed static-list scheduling algorithm.

*This work is part of the SPARCS [5] research, supported in part by the USAF, Wright Labs, WPAFB, under contract number F33615-97-C-1043.

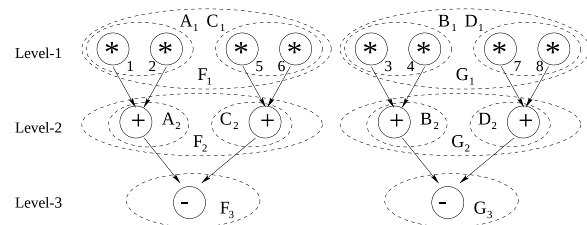

Figure 1. Topological Clustering

We introduced the preliminary idea of cones and clusters in [3] and used it for resource-constrained dynamic-list scheduling algorithms. The topological clustering technique proposed here has an enhanced way of forming cones and clusters. More importantly, the idea of multiple levels of clusters acting as rubber bands and the associated costs for operation selection and assignment priority are unique features of the proposed static-list TCS algorithm.

We have compared our work with a recent static-list scheduling algorithm [4] proposed by Kollig et al. that is shown to outperform other popular scheduling techniques. For a number of benchmarks, we showed that our algorithm generates higher quality schedules as compared to Kollig's technique. The results also conclude that the our algorithm, although having a low-complexity, is very effective in moving towards an optimal schedule. The table below shows some optimal schedules produced at different time constraints (TC), for the example shown in Figure 1.

TC	Proposed $\{*, +, -\}$	Kollig's $\{*, +, -\}$	Optimal $\{*, +, -\}$	% Area Saved
3	{8, 4, 2}	{8, 4, 2}	{8, 4, 2}	0
4	{4, 2, 1}	{5, 2, 1}	{4, 2, 1}	20
5	{3, 2, 1}	{4, 2, 1}	{3, 2, 1}	24
6	{2, 1, 1}	{3, 2, 1}	{2, 1, 1}	34

References

[1] D. D. Gajski et.al. *"High-Level Synthesis: Introduction to Chip and System Design"*. Kluwer Academic Publishers, 1992.

[2] J. Vanhoof, G. Goosens et al. . *"High-Level Synthesis for Real-Time Digital Signal Processing"*. Kluwer Academic Publishers, 1993.

[3] S. Govindarajan and R. Vemuri. "Cone-Based Clustering Heuristic for List Scheduling Algorithms". In *Proc. of European Design & Test Conference (ED&TC)*, pages 456–462, March 1997.

[4] P. Kollig, B. M. Al-Hashimi, and K. M. Abbott. "Efficient scheduling of behavioral descriptions in high-level synthesis". In *IEE Proc. of Computers and Digital Tech.*, vol. 144(2), pages 75–82, March 1997.

[5] I.Ouaiss, et al.. "An Integrated Partitioning and Synthesis System for Dynamically Reconfigurable Multi-FPGA Architectures". In *Proc. of the Reconfig. Arch. Workshop 1998, LNCS 1388*, pages 31–36.