A Technique for Avoiding Isomorphic Netlists in Architectural Synthesis*

Peter Marwedel, Steven Bashford,
Rainer Dömer, Birger Landwehr, Ingolf Markhof
Universität Dortmund, Germany
e-mail: marwedel@ls12.informatik.uni-dortmund.de

Introduction: If interconnections have to be taken into account during architectural synthesis, functional units must be uniquely labelled in order to identify the end points of interconnections. Units can be labelled in a number of ways. Fig. 1 shows two RT-structures with labelled units.

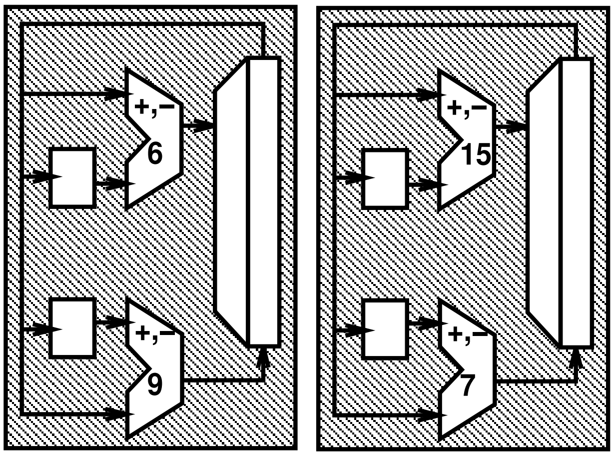

Figure 1: Isomorphic RT-Structures

Assuming that components with equal functionality are actually instances of the same component library element, these structures are equivalent. Structures differing only by their labels are called *isomorphic*. One would assume that tools synthesizing netlists consider only a single representative of each set of equivalent netlists. This, however, is not true. Hence, there are cases, in which exploiting netlist isomorphism would reduce the search space. This reduction has been exploited in our OSCAR synthesis system.

Approach: OSCAR uses an integer programming (IP) approach to architectural synthesis. Binary variables b_k are of central importance for this model. If $b_k = 1$, functional unit k will be part of the generated architecture. Due to the integrated scheduling, allocation and binding approach of OSCAR, units k' and k'' ($k' \neq k''$) may be instances of the same library component (e.g. both may be adders of the same type).

*Work Supported by the CEC under contract ESPRIT 6855 (LINK) and by NATO grant # CRG 950910.

Allocations $b_{k'} = 0, b_{k''} = 1$ and $b_{k'} = 1, b_{k''} = 0$ are equivalent, except for a renaming of units. Hence, OSCAR reduces the complexity of its search algorithm by "using lower indices first". OSCAR restricts itself, without loss of optimality, to solutions for which $b_{k'} \geq b_{k''}$ if $k' > k''$ and k' as well as k'' are instances of the same library element. Also, OSCAR uses canonical labels for units of the same type. Details can be found in a report [1].

Results: The resulting speed-up has been analysed for two different synthesis examples and for four different IP algorithms. The results for small libraries (2 resp. 3 components) can be found in table 1.

	EWF	determ
lp_solve (autmatic mode)	2.35	5.5
lp_solve (optimized decision seq.)	2.48	3.04
osl	8.04	>42.2
opbpd	51.77	4

Table 1: Speed-up of search (average) due to excluding equivalent solutions

This speedup is expected to grow with the size of the component libraries. We believe that the concept of isomorphm can be exploited in other synthesis tools as well and that this exploitation is a promising area for research reaching beyond the current application.

References: [1] Peter Marwedel, Steven Bashford, Rainer Dömer, Birger Landwehr, Ingolf Markhof: A Technique for Avoiding Isomorphic Netlists in Architectural Synthesis, *Report #95-28 of the University of California at Irvine, 1995*, `http://ls12-www.informatik.uni-dortmund.de`