Maximizing Conditional Reuse by Pre-synthesis Transformations[*]

O. Peñalba. J. M. Mendías, R. Hermida

Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid
{olgape, mendias, rhermida}@dacya.ucm.es

Abstract

The property called mutual exclusiveness, responsible for the degree of conditional reuse achievable after a high-level synthesis (HLS) process, is intrinsic to the systems behavior. But sometimes it is only partially reflected in the actual description written by a designer. Our algorithm performs a transformation of the input description that exploits the maximum conditional reuse of the behavior, independently of description style, allowing the HLS tools to obtain circuits with less area.

1 Introduction

The possibility of conditional reuse depends not only on the number of mutex operations pairs detected by an algorithm, but also on the way in which specifications are written by designers. Most of the previous methods [1] perform exhaustive control-flow and/or data-flow analyses, achieving independence of the sequence of the operations and conditional sentences. Only the most recent methods [2] achieve some independence of the particular set of operations used to describe the behavior.

Our approach manipulates not only expressions and conditional sentences, but also operations. In the transformed description the conditional reuse has been totally exploited, simplifying the tasks of the following HLS tools, that only have to cope with the classical type of reuse: the unconditional one. This way the resulting description can be directly processed by a commercial HLS tool to complete the synthesis process, like for instance the SYNOPSYS Behavioral Compiler.

2 Problem formulation and solution

There are three types of functional units (FU) reuse:
- *Unconditional reuse means* using the same FU to execute several operations scheduled in different cycles.
- *Conditional reuse* means using the same FU to execute several operations that are mutually exclusive (mutex).
- *Implicit reuse means* using the same module to execute several computations that consist of the same operator symbol and the same arguments

The first one is always applied during the HLS process, whereas the others can appear after or before synthesis. Normally, the conditional reuse is applied by the HLS tool, and the implicit reuse is applied by the designer. However they are so closely related, that we have to take both into account simultaneously, finding their optimal combination, to obtain the best implementation. With this combination we fix the minimum number of FU needed in an 1-cycle implementation. Afterwards, the HLS tool applies the unconditional reuse to achieve a compromise between the number of cycles and the number of FU, that satisfies the restrictions imposed by the designer.

Our algorithm starts applying the maximum implicit reuse. Then, the description is modified removing the implicit reuse when the conditional one can provide better results. During this process the operations are grouped in macronodes. Every macronode represents a collection of operations that are conditionally sharing a FU. These macronodes will be handle as unique unconditional operations by the following HLS tool.

Example.

Consider the following descriptions. In the left one, the maximum implicit reuse is applied, but no conditional reuse is possible. An 1-cycle implementation requires three FU. In the right one, some implicit reuse has been removed, allowing conditional reuse between operations. This requires only two FU in an 1-cycle implementation.
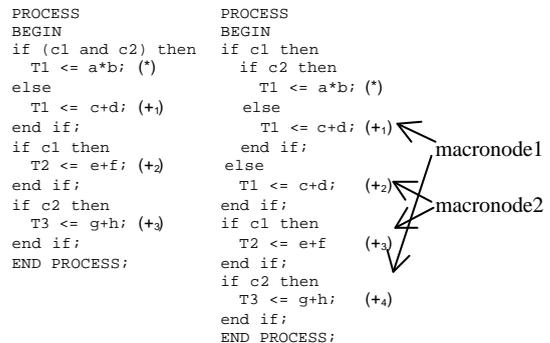
3 References

[1] J. Li, R. K. Gupta, "An Algorithm to Determine Mutually Exclusive Operations in Behavioral Descriptions", *Proceedings of the DATE Conference*, pp. 457-463, 1998.

[2] O.Penalba, J.M. Mendías, M.C. Molina, "Execution Condition Analysis in High-level Synthesis: A Unified Approach", *Proceedings of the 13th ISSS*, pp. 73-78 , 2000.