Scheduling using mixed arithmetic: an ILP formulation

A. Mignotte, O. Peyran

Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon
46, Allée d'Italie
69364 LYON Cedex 7, FRANCE

Abstract

We present a way to automatically select, within an architectural synthesis tool, the best operand and operator number systems, in order to find the best speed/area tradeoff. This implies the use of different number systems (redundant and non-redundant) for the same design: this is what we call mixed arithmetics. We present an ILP formulation to solve a scheduling problem.

1 Introduction

Some number systems may allow faster arithmetic operations than our conventional (binary or decimal) number systems. Redundant number systems are rather commonly used into arithmetic operators such as multipliers and dividers. The result, before being output from the corresponding operator, is converted from a redundant (R) number system to a non-redundant (NR), which costs a conventional addition.

In a special-purpose design, where several consecutive arithmetic operations are to be performed before outputting a result, this may lead to useless waste of computation time and silicon area. Using instead operators with redundant inputs would allow to skip this costfull conversion. However, area is increased with redundant operands. To sum-up this section, we can draw the following conclusions:

- operators with **redundant** *outputs* are faster and smaller than the corresponding operators with NR *outputs*; we call them redundant operators.
- redundant operators with **non-redundant** *inputs* are faster and smaller than the corresponding operators with redundant *inputs*.

2 Synthesis with mixed arithmetic

We looked for an integer linear programming formulation to solve a scheduling problem using mixed operators. Hwang et al [1] proposed different ILP formulations for different classical scheduling problems. An important difference with these problems is that we had to perform operator type selection **while** scheduling. Indeed, as this selection may imply the insertion of a converter, the choice, *a priori* of a redundant operator to speed up the design may lead to a drastic increase of the number of cycles.

Our formulation implicitly handles this selection. The idea is that a $R + R- > R$ adder is made up with two $R + NR- > R$ adders, and that a $NR + NR- > NR$ adder is a $NR + NR- > R$ adder (reduced to nothing) followed by a conversion. Thus, we consider that an adder is composed by 0, 1 or 2 instances of a $R + NR- > R$ adder and by 0 or 1 instance of a converter. The main difficulty is to handle the "zero instance" case, as it leads to a non-scheduled operation, which modifies the dependency constraints. For example, with the redundant addition of two NR numbers, there is 0 instance of adder (two NR numbers *are* a R number), allowing a following operation to be scheduled into the same step.

3 Conclusion

The formulation [2] is too large to be presented here. However the table below shows some results with different benchmarks. The formulation should be simplified, as large examples can not be resolved because of numerical instability.

benchmarks	# eq.	# var.	CPU (s)
2 way method	51	70	6.2
FFT	56	113	7.6
Diff. equation	97	173	15.0
5^{th} order filter	449	1038	

References

[1] C-T. Hwang, J-H. Lee, and Y-C. Hsu. A formal approach to the scheduling problem in high level synthesis. *IEEE Transaction on Computer Aided Design.*

[2] A. Mignotte and O. Peyran. Scheduling using mixed arithmetic: an ILP formulation. Research report, LIP.

ED&TC '97 on CD-ROM