Abstract

Several recent works have shown the interest of formalizing various arithmetic optimization problems as Integer Linear Programming (ILP). This Master-2 internship will explore mathematical modelling approaches to a range of arithmetic problems to improve the quality of the operators, starting with the problem of bit heap compression. Successful approaches will be integrated in the FloPoCo arithmetic core generator and validated on practical operators, from elementary multipliers to elementary function evaluators and machine-learning applications.

Supervision and funding

This 5-month internship is funded by the Agence Nationale de la Recherche in the framework of the PEPR IA Holigrail project.

It will be supervised by Florent de Dinechin, professor at INSA-Lyon.

The post-doctoral researcher will work in the CITI Laboratory located in the Lyon-La Doua campus.

Work context

FloPoCo\(^1\) is a generator of application-specific hardware arithmetic cores. It represents the state of the art for many specialised arithmetic operators, such as multiplication and division by constants, numerical function evaluators, and many others [1, 2]. Recently, Integer Linear programming (ILP) has been used to optimize some of the underlying problems. An illustrative example is bit heap compression.

A bit heap is a data structure that occurs in many operators in FloPoCo. It is an unevaluated sum of bits, each bit being weighted by a power of two. The simplest bit heap is a fixed-point number written in binary: \(X = \sum_{i=0}^{\text{msb}} x_i 2^i\). The product of two binary number is a bit heap where each bit is computed as the boolean AND of two input bits: \(XY = \sum_{i,j} x_i y_j 2^{i+j}\). Generalizing this, expressions such as \(A \times B + C\), \(X^2 + Y^2\), and indeed any multivariate polynomial can be represented as a bit heap. Further, many numerical

\(^1\text{http://www.flopoco.org/}\)

![Figure 1: Bit heaps obtained in FloPoCo have various shapes and sizes](image-url)
evaluation techniques consist in summing many terms from various sources (precomputed tables, other operators). Figure 1 shows a few examples of bit heaps generated in FloPoCo.

The advantage of this point of view is that this bit-level point of view exposes more parallelism than a word-level point of view. A practical application is that it is possible to derive, for any bit heap, efficient compressor tree architectures that compute the value of a bit heap. For instance, a full adder cell (Fig. 2) can be viewed as an elementary bit heap compressor that removes 3 bits from one column and replaces them with 2 bits on two adjacent columns. Many other compressors have been studied, especially on FPGAs [2]. Finding an architecture that computes the value of a bit heap is equivalent to finding a sequence of bit heap transformations (such as Fig. 2(b)) that reduce the initial bit heap to one or two lines at minimal cost.

Tentative workplan

The state of the art (implemented in FloPoCo) is to use ILP to derive optimal or near-optimal compressor trees [4]. It could be extended in many directions: experimenting with other solver families (constraint programming and SAT solvers) with the hope to scale to larger sizes; trying different modelling approaches, for instance inspired by tiling or scheduling problems; refining the timing model to better capture bit heaps where bits can arrive at different times, and possibly subcycle timing; studying the integration of other optimization problems that involve bit heap compression but also bit heap generation, such as large multipliers [5] or multipartite tables [3]; attempting to integrate frequency-directed pipelining in the model.

This internship will explore some of these directions, with small-scale experiments conducted in Python, Julia, or Minizinc. Successful models will then be integrated in FloPoCo using open-source libraries such as OR-Tools or ScaLP, and evaluated on actual operators.

References


